RM0391
Reference manual

SPC58 E/G Line - 32 bit Power Architecture automotive MCU Triple z4 cores 180 MHz, 6 MBytes Flash, HSM, ASIL-D

Overview

This document defines the functionality of the SPC58xEx/SPC58xGx microcontroller for use by software and hardware developers. The SPC58xEx/SPC58xGx is built on Power Architecture technology and integrates technologies that are important for today's and tomorrow's automotive applications.

SPC58xEx/SPC58xGx applications include

- automotive powertrain controllers for six to eight cylinder gasoline, diesel engines and advanced combustion systems as well as high-end hybrid and transmission control
- integrated chassis control, high-end steering and braking
- high-end central body controller, smart junction boxes, stand-alone and integrated gateway
- in general any safety critical application requiring a very high level of safety integrity

The information in this book is subject to change without notice, as described in the disclaimer. As with any technical documentation, it is the reader's responsibility to be sure he or she is using the most recent version of the documentation.

To locate any published errata or updates for this document, visit the ST Web site at http://www.st.com.

Audience

This manual is intended for system software and hardware developers and applications programmers who want to develop products with the SPC58xEx/SPC58xGx device. It is assumed that the reader understands operating systems, microprocessor system design, basic principles of software and hardware, and basic details of the Power Architecture.

Contents

1 Preface 80

1.1 Document organization 80

1.2 Register conventions 80

1.3 Acronyms and abbreviations 81

1.4 Reference documents 81

2 Introduction 82

2.1 SPC58xEx microcontroller 82

2.1.1 Target applications 82

2.1.2 Overall architecture 82

2.1.3 Core features 83

2.1.4 Core_2 83

2.1.5 Memory hierarchy 84

2.1.6 Features 85

2.1.7 Feature list 86

2.1.8 Packages 88

2.1.9 Software debug and calibration 89

2.1.10 Block diagram 89

2.2 SPC58xGx microcontroller 92

2.2.1 Target applications 92

2.2.2 Overall architecture 92

2.2.3 Core features 92

2.2.4 Core_2 93

2.2.5 Memory hierarchy 94

2.2.6 Features 94

2.2.7 Feature list 96

2.2.8 Packages 98

2.2.9 Software debug and calibration 98

2.2.10 Block diagram 98

3 Embedded memories 101

3.1 Overview 101

3.2 SRAM 101

3.2.1 System SRAM 101

3.2.2 Standby SRAM regulator . . . 101
3.2.3 Overlay SRAM . . . 101
3.2.4 Processor core local RAM . . . 102
3.3 Embedded Flash memory . . . 102
3.3.1 Flash memory controller . . . 103
3.3.2 Flash memory array . . . 103
3.4 End-to-end Error Correction Code (e2eECC) . . . 106
3.5 Security features . . . 108

4 Signal Description . . . 109
4.1 Production packages . . . 109
4.2 Emulation package . . . 109
4.3 Package pinouts and ballouts . . . 109

5 Memory Map . . . 110
5.1 Peripheral memory . . . 112
5.2 Flash memory . . . 120
5.3 Emulation and debug memory . . . 124
5.4 System memory . . . 125
5.5 Local memory . . . 125
5.6 UTest Flash memory map . . . 126
5.7 Security module memory . . . 127

6 Functional safety . . . 128
6.1 Introduction . . . 128
6.2 Safety overview . . . 128
6.3 Module categorization . . . 128
6.4 System implementation . . . 130
6.4.1 General concept . . . 130
6.4.2 Dual-core lockstep . . . 160
6.4.3 Common cause failure measures . . . 163
6.4.4 ECC . . . 163

7 Device configuration . . . 167
7.1 Introduction . . . 167

7.2 Core modules . . . 167
7.3 System modules . . . 173
7.3.1 SIUL2 configuration . . . 173
7.3.2 Semaphores2 (SEMA42) configuration . . . 173
7.3.3 Crossbar switch . . . 173
7.3.4 System Memory Protection Unit (SMPU) configuration . . . 180
7.3.5 Peripheral bridge configuration . . . 183
7.3.6 Interrupt controller (INTC) configuration . . . 184
7.3.7 DMA controller configuration . . . 221
7.3.8 DMACHMUX configuration . . . 225
7.3.9 Platform Configuration Module (PCM) configuration . . . 230
7.3.10 Wakeup unit (WKPU) configuration . . . 240
7.3.11 Crossbar integrity checker (XBIC) . . . 244
7.4 Clocking . . . 244
7.5 Memories and memory interfaces . . . 244
7.5.1 Flash memory controller (PFLASH) configuration . . . 244
7.5.2 Platform RAM Controller (PRAMC) configuration . . . 246
7.5.3 Decorated Storage Memory Controller (DSMC) . . . 246
7.6 Analog modules . . . 247
7.6.1 Temperature sensor configuration . . . 247
7.6.2 SARADC . . . 247
7.6.3 SDADC . . . 248
7.6.4 Body Cross Triggering Unit (BCTU) configuration . . . 248
7.7 Timers . . . 250
7.7.1 System Timer Module (STM) configuration . . . 250
7.7.2 Software Watchdog Timer (SWT) configuration . . . 250
7.7.3 PIT configuration . . . 252
7.7.4 GTM subsystem configuration . . . 254
7.8 Communication interfaces . . . 256
7.8.1 Ethernet configuration . . . 256
7.8.2 CAN subsystem configuration . . . 258
7.8.3 LFAST and SIPI . . . 258
7.8.4 DSPI configuration . . . 260
7.8.5 FlexRay configuration . . . 262
7.8.6 PSI5 controller configuration . . . 263
7.8.7 SENT Receiver (SRX) configuration . . . 263

7.8.8 LINFlexD . . . 267
7.9 Reset and Boot modules . . . 269
7.9.1 BAF configuration . . . 272
7.9.2 System Status and Configuration Module (SSCM) configuration . . . 273
7.10 Safety modules . . . 274
7.10.1 CRC . . . 274
7.10.2 MEMU configuration . . . 275
7.10.3 Indirect Memory Access (IMA) configuration . . . 289
7.10.4 FCCU configuration . . . 293
7.10.5 Register protection (REG_PROT) configuration . . . 295
7.10.6 STCU2 configuration . . . 305
7.11 Security modules . . . 313
7.11.1 Password and Device Security Module (PASS) configuration . . . 313
7.11.2 Tamper Detection Module (TDM) configuration . . . 318

8 Reset and Boot . . . 325
8.1 Introduction . . . 325
8.1.1 TEST flash memory block . . . 325
8.1.2 UTEST flash memory block . . . 325
8.1.3 Boot Assist Flash . . . 325
8.2 Modules used in reset sequence . . . 326
8.2.1 Power Management Controller . . . 326
8.2.2 Reset Generation Module . . . 326
8.2.3 Mode Entry module . . . 327
8.2.4 System Status and Configuration Module . . . 327
8.2.5 Self-Test Control Unit . . . 327
8.3 Reset sequence . . . 327
8.3.1 Power-on and the Reset Generation Module . . . 327
8.3.2 Power-up phase: power stabilization . . . 330
8.3.3 PHASE0 Phase: analog supply initial configuration . . . 331
8.3.4 PHASE1[DEST] Phase: temporization and monitoring setup . . . 331
8.3.5 PHASE2[DEST] Phase: flash initial configuration . . . 331
8.3.6 PHASE3[DEST] Phase: device configuration . . . 332
8.3.7 IDLE[DEST] Phase: self-test execution . . . 333
8.3.8 PHASE1[FUNC] Phase: temporization and monitoring setup . . . 334
8.3.9 PHASE2[FUNC] Phase: flash initial configuration . . . 334

8.3.10 PHASE3[FUNC] Phase: device configuration monitoring 334
8.3.11 IDLE[FUNC] Phase 335
8.3.12 System start-up 335
8.3.13 Waking-up other CPUs 343
8.3.14 BAF Bootloader 343

9 Device Configuration Format (DCF) Records 345
9.1 Introduction 345
9.2 DCF clients 346
9.3 DCF records 346
9.3.1 UTEST DCF records 350
9.4 DCF client table 350
9.4.1 DCF client list 351
9.4.2 BAF configuration DCF register 363
9.4.3 Miscellaneous DCF registers 364

10 Power management 372
10.1 Overview 372
10.1.1 Power management framework 373
10.1.2 Power management supply description 373
10.1.3 Power management controller overview 375
10.2 Low power mode support 377
10.2.1 Low power mode (HALT/STOP/STANDBY) 378
10.3 Flash power requirements 379
10.4 Device trimming 379
10.5 Supply monitoring (POR and LVDs) 379
10.5.1 Power-on reset (POR) 380
10.5.2 Behavior of device LVD / HVD 380
10.5.3 Voltage detections (MVDs, LVDs, HVDs, UVDs) 381
10.6 Power sequence 384
10.6.1 Power-up sequence 384
10.6.2 Power-down sequence 388
10.6.3 Brown-out management 388
10.6.4 Low voltage requirement during crank 389

11 Smart Stand-by Wake-up Unit (SSWU) 390

11.1 Overview . . . 390
11.2 Functional description . . . 390
11.3 Architecture . . . 391
11.4 Clocking . . . 393
11.5 Memory map and register descriptions . . . 394
11.5.1 Memory map . . . 394
11.5.2 Trigger generator subunit input selection register (TGSISR) . . . 396
11.5.3 Trigger generator subunit control register (TGSCR) . . . 396
11.5.4 Trigger compare register 0–7 (TCR0–TCR7) . . . 397
11.5.5 TGS counter compare register (TGSCCR) . . . 398
11.5.6 TGS counter reload register (TGSCRR) . . . 398
11.5.7 Commands list control register 1 (CLCR1) . . . 399
11.5.8 Commands list control register 2 (CLCR2) . . . 399
11.5.9 Trigger handler control register 1 (THCR1) . . . 400
11.5.10 Trigger handler control register 2 (THCR2) . . . 401
11.5.11 Commands list register 1–32 (CLR1–CLR32) . . . 402
11.5.12 Cross triggering unit error flag register (CTUEFR) . . . 406
11.5.13 Cross triggering unit interrupt flag register (CTUIFR) . . . 407
11.5.14 Cross triggering unit control register (CTUCR) . . . 408
11.5.15 Double buffered registers . . . 409
11.6 SSWU module descriptions . . . 410
11.6.1 Stand-by eCTU . . . 410
11.6.2 CD (Command Dispatcher) . . . 414
11.6.3 ADC . . . 417
11.6.4 OPC . . . 418
11.6.5 PDC . . . 419
11.6.6 TU . . . 420
11.7 SSWU Configuration example . . . 421
11.7.1 Description . . . 421
11.7.2 Registers configuration . . . 421

12 Security . . . 424
12.1 Basic security . . . 424
12.2 Advanced security . . . 424
12.3 Detailed security information . . . 425

13 Calibration and Debug 426
13.1 Core debug support 426
13.2 Run control and memory access 426
13.2.1 Debug and Calibration Interface (DCI) 427
13.2.2 JTAG Data Communication (JDC) 430
13.2.3 Sequence Processing Unit (SPU) 430
13.3 Calibration interface 442
13.3.1 JTAG Master (JTAGM) 443
13.3.2 Debug LFAST 443
13.3.3 Development Trigger Semaphore (DTS) 443
13.4 Debug over CAN 444
13.5 Nexus Trace Aurora interface 444
13.5.1 Nexus Aurora overview 447
13.5.2 Nexus Aurora Router (NAR) 448
13.5.3 Nexus Aurora Link (NAL) 451
13.5.4 Nexus Aurora PHY (NAP) 451
13.6 Nexus clients 452
13.6.1 e200z4 Nexus 3 452
13.6.2 GTM Development Interface (GTMDI) 453
13.6.3 Nexus Crossbar Multi-master Client (NXMC) 453
13.7 Data trace filtering 455

14 e200z4d Core Complex Overview 456
14.1 Overview of the e200z420n3, e200z419, and e200z425n3 cores 456
14.2 Features 458
14.3 Microarchitecture summary 459
14.3.1 Instruction unit features 461
14.3.2 Integer unit features 461
14.3.3 Load/Store unit features 461
14.3.4 MPU features 462
14.3.5 Cache features 462
14.3.6 Local memory features 463
14.3.7 e200z420n3 and e200z425n3 system bus features 463
14.3.8 e200z419 system features 463

15 Core (z420n3) Description 464

15.1 Overview of the Core (z420n3) 464
15.2 Register Model 464
15.3 Dual Issue Operation 468
15.4 Reservation Instructions and Cache Interactions 468
15.5 Signal Processing Extension / Embedded Floating-point Status and Control Register (SPEFSCR) 468
15.6 Cache 471
15.6.1 Cache Overview 471
15.6.2 L1 Cache Control and Status Register 0 (L1CSR0) 471
15.6.3 L1 Cache Control and Status Register 1 (L1CSR1) 474
15.6.4 L1 Cache Configuration Register 0 (L1CFG0) 476
15.6.5 L1 Cache Configuration Register 1 (L1CFG1) 477
15.6.6 Data Cache Software Coherency 478
15.6.7 Data Cache Hardware Coherency 478
15.6.8 Cache Invalidate by Set and Way 478
15.6.9 Cache EDC/ECC Protection 480
15.6.10 Cache Error Injection 481
15.7 Exceptions 481
15.7.1 Exception Syndrome Register 483
15.7.2 Machine State Register 484
15.7.3 Machine Check Syndrome Register (MCSR) 485
15.7.4 Interrupt Vector Prefix Registers (IVPR) 488
15.7.5 Interrupt Definitions 489
15.8 MPU 498
15.8.1 MPU Overview 498
15.8.2 Software Interface and MPU Instructions 498
15.8.3 MPU Read Entry Instruction (mpure) 498
15.8.4 MPU Write Entry Instruction (mpuwe) 499
15.8.5 MPU Synchronize Instruction (mpusync) 499
15.8.6 MMU/MPU Configuration Register (MMUCFG) 500
15.8.7 MPU0 Configuration Register (MPU0CFG) 500
15.8.8 MPU0 Control and Status Register 0 (MPU0CSR0) 501
15.8.9 MPU Assist Registers (MAS) 504
15.8.10 MAS Registers Summary 509
15.9 Local memories 509
15.9.1 Local instruction and data memory overview 509

15.9.2 Local memory control and configuration 509
15.10 End-to-End ECC support 522
15.10.1 e2eECC control and configuration 523

16 Core (z425n3) description 527
16.1 Overview of the Core (z425n3) 527
16.2 Register model 527
16.3 Dual issue operation 530
16.4 Reservation instructions and cache interactions 531
16.5 Signal Processing Extension / Embedded Floating-point Status and Control Register (SPEFSCR) 531
16.6 Cache 534
16.6.1 Cache overview 534
16.6.2 L1 Cache Control and Status Register 0 (L1CSR0) 534
16.6.3 L1 Cache Control and Status Register 1 (L1CSR1) 535
16.6.4 L1 Cache Configuration Register 0 (L1CFG0) 537
16.6.5 L1 Cache Configuration Register 1 (L1CFG1) 538
16.6.6 Cache Invalidate by Set and Way 539
16.6.7 Cache EDC/ECC protection 540
16.6.8 Cache error injection 541
16.7 Exceptions 541
16.7.1 Exception syndrome register 542
16.7.2 Machine State Register 543
16.7.3 Machine Check Syndrome Register (MCSR) 545
16.7.4 Interrupt Vector Prefix Registers (IVPR) 548
16.7.5 Interrupt definitions 548
16.8 MPU 558
16.8.1 MPU overview 558
16.8.2 Software interface and MPU instructions 558
16.8.3 MPU read entry instruction (mpure) 558
16.8.4 MPU Write Entry Instruction (mpuwe) 559
16.8.5 MPU Synchronize Instruction (mpusync) 559
16.8.6 MMU/MPU Configuration Register (MMUCFG) 559
16.8.7 MPU0 Configuration Register (MPU0CFG) 560
16.8.8 MPU0 Control and Status Register 0 (MPU0CSR0) 561
16.8.9 MPU Assist registers (MAS) 564

16.8.10 MAS registers summary 569
16.9 Local memories 569
16.9.1 Local instruction and data memory overview 569
16.9.2 Local memory control and configuration 569
16.10 End-to-End ECC support 582
16.10.1 e2eECC control and configuration 583

17 System Integration Unit Lite2 (SIUL2) 587
17.1 Introduction 587
17.1.1 Overview 587
17.1.2 Features 589
17.1.3 Register protection 589
17.2 Memory map and register description 590
17.2.1 Memory map 590
17.2.2 Register descriptions 591
17.2.3 Multiplexed Signal Configuration Registers 609
17.3 Functional description 610
17.3.1 General 610
17.3.2 Pad control 611
17.3.3 General purpose input or output pads (GPIO) 611
17.3.4 External interrupts/DMA requests (EIRQ pins) 612

18 Crossbar switch (XBAR) 616
18.1 Introduction 616
18.2 Features 616
18.3 Memory map and register definition 616
18.3.1 XBAR Priority Registers Slave (XBAR_PRSn) 617
18.3.2 XBAR Control Register (XBAR_CRSn) 620
18.3.3 Master General Purpose Control Register (XBAR_MGPCRn) 621
18.4 Functional description 622
18.4.1 General operation 622
18.4.2 Register coherency 623
18.4.3 Arbitration 623
18.5 Initialization/application information 624

19 Crossbar Integrity Checker (XBIC) 625

19.1 Overview . . . 625
19.2 Features . . . 625
19.3 Block diagram . . . 626
19.4 External signal description . . . 627
19.5 Memory map and register definition . . . 627
19.5.1 Memory map . . . 627
19.5.2 Register descriptions . . . 628
19.6 Functional description . . . 633

20 Peripheral Bridge (PBRIDGE) . . . 636
20.1 Introduction . . . 636
20.1.1 Features . . . 636
20.1.2 General operation . . . 636
20.2 Memory map and register definition . . . 636
20.2.1 Master Privilege Register A (MPRA) . . . 638
20.2.2 Master Privilege Register B (MPRB) . . . 639
20.2.3 Peripheral Access Control Register (PACRx) . . . 640
20.2.4 Off-Platform Peripheral Access Control Registers (OPACRx) . . . 642
20.3 Functional description . . . 646
20.3.1 Access support . . . 646

21 System Memory Protection Unit (SMPU) . . . 647
21.1 Overview . . . 647
21.2 Block diagram . . . 647
21.3 Features . . . 648
21.4 Memory map and register definition . . . 649
21.4.1 Memory map . . . 649
21.4.2 Register descriptions . . . 650
21.5 Functional description . . . 657
21.5.1 Access evaluation macro . . . 657
21.5.2 Putting it all together and error terminations . . . 658
21.6 Initialization information . . . 659
21.7 Application information . . . 659

22 Intelligent AHB Gasket (IAHBG) . . . 661

22.1 Introduction . . . 661
22.2 Timing modes . . . 661
22.2.1 1:1 . . . 661
22.2.2 2:1 . . . 661
22.2.3 1:2 . . . 662

23 Semaphores2 (SEMA42) . . . 663
23.1 Introduction . . . 663
23.1.1 Multi-core programming . . . 663
23.1.2 Features . . . 664
23.2 Memory map/register definition . . . 665
23.2.1 Semaphores2 Gate n Register (SEMA4_GATEn) . . . 666
23.2.2 Semaphores2 (Secure) Reset Gate (SEMA4_RSTGT) . . . 667
23.3 Functional description . . . 668

24 Interrupt Controller (INTC) . . . 671
24.1 Introduction . . . 671
24.2 Block diagram . . . 671
24.3 Features . . . 673
24.4 Modes of operation . . . 673
24.4.1 Software vector mode . . . 673
24.4.2 Hardware vector mode . . . 674
24.5 Memory map and register definition . . . 674
24.5.1 Memory map . . . 674
24.5.2 Register descriptions . . . 675
24.6 Functional description . . . 683
24.6.1 Interrupt request sources . . . 683
24.6.2 Priority management . . . 684
24.6.3 Handshaking with processor . . . 686
24.7 Initialization/application information . . . 688
24.7.1 Initialization flow . . . 688
24.7.2 Interrupt exception handler . . . 689
24.7.3 ISR, RTOS, and task hierarchy . . . 691
24.7.4 Order of execution . . . 691
24.7.5 Priority ceiling protocol . . . 692
24.7.6 Selecting priorities according to request rates and deadlines . . . 696

24.7.7 Software-settable interrupt requests 696
24.7.8 Lowering priority within an ISR 697
24.7.9 Negating an interrupt request outside of its ISR 697
24.7.10 Examining LIFO contents 698
24.8 Interrupt sources 698

25 Enhanced Direct Memory Access (eDMA) 699
25.1 Introduction 699
25.1.1 Features 699
25.2 Modes of operation 700
25.2.1 Normal mode 700
25.2.2 Debug mode 701
25.2.3 Wait mode 701
25.3 Memory map/register definition 701
25.3.1 Control Register (eDMA_CR) 703
25.3.2 Error Status Register (eDMA_ES) 705
25.3.3 Enable Request Register High (eDMA_ERQH) 706
25.3.4 Enable Request Register Low (eDMA_ERQL) 707
25.3.5 Enable Error Interrupt Register High (eDMA_EEIH) 708
25.3.6 Enable Error Interrupt Register Low (eDMA_EEIL) 709
25.3.7 Set Enable Request Register (eDMA_SERQ) 709
25.3.8 Clear Enable Request Register (eDMA_CERQ) 710
25.3.9 Set Enable Error Interrupt Register (eDMA_SEEI) 711
25.3.10 Clear Enable Error Interrupt Register (eDMA_CEEI) 711
25.3.11 Clear Interrupt Request Register (eDMA_CINT) 712
25.3.12 Clear Error Register (eDMA_CERR) 712
25.3.13 Set START Bit Register (eDMA_SSRT) 713
25.3.14 Clear DONE Status Bit Register (eDMA_CDNE) 714
25.3.15 Interrupt Request Register High (eDMA_INTH) 714
25.3.16 Interrupt Request Register Low (eDMA_INTL) 715
25.3.17 Error Register High (eDMA_ERRH) 716
25.3.18 Error Register Low (eDMA_ERRL) 717
25.3.19 Hardware Request Status Register High (eDMA_HRSH) 718
25.3.20 Hardware Request Status Register Low (eDMA_HRSL) 719
25.3.21 Channel n Priority Register (eDMA_DCHPRIn) 719
25.3.22 Channel n Master ID Register (eDMA_DCHMIDn) 720
25.3.23 TCD Source Address (eDMA_TCDn_SADDR) 721

25.3.24 TCD Transfer Attributes (eDMA_TCDn_ATTR) 722
25.3.25 TCD Signed Source Address Offset (eDMA_TCDn_SOFF) 722
25.3.26 TCD Minor Byte Count (Minor Loop Disabled) (eDMA_TCDn_NBYTES_MLNO) 723
25.3.27 TCD Signed Minor Loop Offset (Minor Loop Enabled and Offset Disabled) (eDMA_TCDn_NBYTES_MLOFFNO) 724
25.3.28 TCD Signed Minor Loop Offset (Minor Loop and Offset Enabled) (eDMA_TCDn_NBYTES_MLOFFYES) 725
25.3.29 TCD Last Source Address Adjustment (eDMA_TCDn_SLAST) 726
25.3.30 TCD Destination Address (eDMA_TCDn_DADDR) 726
25.3.31 TCD Current Minor Loop Link, Major Loop Count (Channel Linking Enabled) (eDMA_TCDn_CITER_ELINKYES) 727
25.3.32 TCD Current Minor Loop Link, Major Loop Count (Channel Linking Disabled) (eDMA_TCDn_CITER_ELINKNO) 728
25.3.33 TCD Signed Destination Address Offset (eDMA_TCDn_DOFF) 728
25.3.34 TCD Last Destination Address Adjustment/Scatter Gather Address (eDMA_TCDn_DLASTSGA) 729
25.3.35 TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Enabled) (eDMA_TCDn_BITER_ELINKYES) 730
25.3.36 TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Disabled) (eDMA_TCDn_BITER_ELINKNO) 731
25.3.37 TCD Control and Status (eDMA_TCDn_CSR) 731
25.4 Functional description 734
25.4.1 eDMA microarchitecture 734
25.4.2 eDMA basic data flow 735
25.4.3 Error reporting and handling 738
25.4.4 Channel preemption 740
25.4.5 eDMA performance 740
25.5 Initialization/application information 743
25.5.1 eDMA initialization 743
25.5.2 DMA programming errors 745
25.5.3 DMA Arbitration mode considerations 746
25.5.4 DMA transfer 747
25.5.5 eDMA TCDn status monitoring 750
25.5.6 Channel linking 751
25.5.7 Dynamic programming 752

26 DMA channel multiplexer (DMACHMUX) 756

26.1 Introduction 756

26.1.1 Overview . . . 756
26.1.2 Features . . . 757
26.1.3 Modes of operation . . . 757
26.2 External signal description . . . 757
26.2.1 Overview . . . 757
26.3 Memory map and register definition . . . 757
26.3.1 Register descriptions . . . 758
26.4 Functional description . . . 759
26.4.1 DMA channels with periodic triggering capability . . . 759
26.4.2 DMA channels with no triggering capability . . . 761
26.4.3 “Always Enabled” DMA sources . . . 761
26.5 Initialization/application information . . . 762
26.5.1 Reset . . . 762
26.5.2 Enabling and configuring sources . . . 762

27 Clocking . . . 767
27.1 Introduction . . . 767
27.2 Clock generation . . . 767
27.2.1 MC_CGM registers . . . 770
27.3 System clock frequency limitations . . . 772
27.3.1 JTAG frequencies . . . 775
27.4 Default clock configuration . . . 776
27.5 Clock sources . . . 776
27.5.1 PLL . . . 777
27.5.2 External oscillator (XOSC) . . . 780
27.5.3 16 MHz internal RC oscillator (IRCOSC) . . . 782
27.6 Peripheral clocks . . . 782
27.6.1 HSM clock divider . . . 784
27.6.2 PSI5 clock dividers . . . 784
27.6.3 LFAST clocking . . . 784
27.6.4 Ethernet clocking . . . 785
27.6.5 FlexRay clocking . . . 786
27.6.6 M_TTCAN/M_CAN clocking . . . 787
27.6.7 Sigma-Delta ADC clocking . . . 787
27.6.8 System Clock/GTM output clock . . . 788
27.7 Clock monitoring . . . 789

27.7.1 CMU configuration . . . 789
27.7.2 PLL0 monitor . . . 792
27.7.3 External oscillator (XOSC) monitor . . . 792
27.7.4 Internal RC oscillator (IRCOSC) monitor . . . 792
27.7.5 System clock monitors . . . 792
27.8 Loss of system clock behavior . . . 793
27.8.1 Loss of PLL/XOSC clock . . . 793
27.8.2 Loss of IRCOSC clock . . . 793
27.9 Progressive clock switching . . . 794

28 Dual PLL digital interface (PLLDIG) . . . 795
28.1 Introduction . . . 795
28.2 Block diagram . . . 795
28.3 Features . . . 796
28.4 Modes of operation . . . 796
28.4.1 Normal mode with reference, PLL0 or both PLLs enabled . . . 796
28.5 Memory map and register definition . . . 797
28.5.1 Memory map . . . 797
28.5.2 Register descriptions . . . 797
28.5.3 Register classification for safety requirements . . . 806
28.6 Functional description . . . 806
28.6.1 Input clock frequency . . . 807
28.6.2 Clock configuration . . . 807
28.6.3 Frequency modulation . . . 808
28.7 Initialization information . . . 810

29 Clock Monitor Unit (CMU) . . . 812
29.1 Introduction . . . 812
29.1.1 Main features . . . 812
29.2 Block diagram . . . 812
29.3 Signals . . . 812
29.4 Memory map and register definition . . . 813
29.4.1 Register descriptions . . . 814
29.5 Functional description . . . 819
29.5.1 Frequency meter . . . 819
29.5.2 CLKMN0_RMT supervisor . . . 819

29.5.3 CLKMN1 supervisor 819

30 Clock Generation Module (MC_CGM) 821

30.1 Introduction 821

30.1.1 Overview 821

30.1.2 Features 823

30.2 External signal description 823

30.3 Memory map and register definition 823

30.3.1 Register descriptions 825

30.4 Functional description 873

30.4.1 System clock generation 873

30.4.2 Auxiliary clock generation 877

30.4.3 Dividers functional description 879

31 OSC digital interface (XOSC) 882

31.1 Introduction 882

31.2 Functional description 882

31.2.1 Oscillator power-down control and status 882

31.2.2 Oscillator startup delay 882

31.2.3 Oscillator clock available interrupt 883

31.2.4 Oscillator bypass mode 883

31.3 Memory map and register definition 884

31.3.1 Register descriptions 884

32 IRCOSC digital interface 886

32.1 Introduction 886

32.2 Functional description 886

32.3 Memory map and register definition 886

32.3.1 Register descriptions 887

33 LPRC digital interface 890

33.1 Introduction 890

33.2 Low Power RC oscillator 890

33.3 Memory map and register definition 890

33.3.1 Register description 891

34 OSC32K digital interface (OSC32K) 892
34.1 Introduction 892
34.2 Memory map and register definition 892
34.2.1 Register descriptions 893
34.3 Functional description 894

35 Platform RAM controller AHB (PRAMC_AHB) 895
35.1 Introduction 895
35.2 SRAM controller memory map and register definitions 896
35.2.1 Platform RAM configuration register 1 (PRCR1) 896
35.3 Functional description 898
35.3.1 Read/Write introduction 898
35.3.2 Writes 899
35.4 Initialization/application information 899
35.5 Safety considerations 900
35.5.1 Hsiao ECC algorithm 900
35.5.2 Transaction monitor 903

36 Flash memory controller 905
36.1 Introduction 905
36.2 Features 905
36.3 Block diagrams 906
36.4 Flash memory controller memory map 906
36.4.1 Overview 906
36.5 Functional description 931
36.5.1 Basic interface protocol 931
36.5.2 Access protections 931
36.5.3 Read cycles – buffer miss 932
36.5.4 Read cycles – buffer hit 932
36.5.5 Error termination 932
36.5.6 Flash error response operation 932
36.5.7 Security module exclusive control 932
36.5.8 Access pipelining 933
36.5.9 Line read buffers and prefetch operation 934
36.5.10 Instruction/Data prefetch triggering 935
36.5.11 Per-Master prefetch triggering 935

36.5.12 Buffer allocation . . . 935
36.5.13 PFlash calibration remap support . . . 935
36.5.14 Safety considerations . . . 940
36.5.15 e2eECC on data Flash accesses . . . 944
36.5.16 Array integrity considerations . . . 944

37 Embedded Flash Memory . . . 945

37.1 Introduction . . . 945
37.1.1 Overview . . . 945
37.1.2 Features . . . 948
37.1.3 Modes of operation . . . 948
37.2 Flash memory map and description . . . 949
37.2.1 Flash array memory map . . . 949
37.3 Register memory maps and descriptions . . . 953
37.3.1 Register memory maps . . . 953
37.3.2 User register descriptions . . . 955
37.4 Functional description . . . 1014
37.4.1 Reset . . . 1015
37.4.2 Power-down mode (Disable mode) . . . 1015
37.4.3 Low-power mode (Sleep mode) . . . 1016
37.4.4 Read mode . . . 1016
37.4.5 Modify mode . . . 1020
37.4.6 Alternate program and erase interface . . . 1027
37.4.7 User Test mode . . . 1029
37.4.8 Protection strategy . . . 1034

38 Flash Memory Programming and Configuration . . . 1036

38.1 Selection of Flash memory blocks for erase . . . 1037
38.2 Non-secure write protection . . . 1038
38.3 Secure write protection . . . 1038
38.3.1 Implementing secure write protection . . . 1040
38.3.2 Overriding secure write protection . . . 1042
38.4 Secure read protection . . . 1044
38.4.1 Implementing secure read protection . . . 1045
38.4.2 Overriding secure read protection . . . 1047
38.5 Debug port enable/disable . . . 1048

38.6 Tamper detection 1048
38.6.1 Implementing tamper detection 1049
38.6.2 Creating the tamper detect diary 1049
38.6.3 Assigning blocks to Tamper Detection Regions (TDRs) 1050
38.6.4 Overriding tamper detection 1051
38.7 Implementing OTP 1052
38.8 Implementing test mode disable 1052
38.8.1 Unconditional test mode disable seal 1052
38.8.2 Passcode-protected test mode disable seal 1053
38.8.3 Selecting Flash memory blocks for test mode disable seal 1053
38.9 Security configuration planning 1054
38.9.1 Hardware Security Module (HSM) 1054
38.9.2 Creating password groups 1054
38.9.3 Planning secure write protection 1055
38.9.4 Planning secure read protection 1055
38.9.5 Planning debug port enable/disable 1055
38.9.6 Planning OTP Flash memory block assignment 1055
38.9.7 Planning factory test mode disable 1055

39 Decorated Storage Memory Controller (DSMC) 1056
39.1 Introduction 1056
39.2 Decorated Stores: st[b,h,w]d{cb}x rS,rB,rA 1058
39.2.1 Bit Field Insert (BFINS) into an 8, 16 or 32-bit Memory Container . . 1058
39.2.2 Compare-and-Store (CAST) 1060
39.2.3 Logical AND (AND) 1060
39.2.4 Logical OR (OR) 1061
39.2.5 Logical Exclusive-OR (XOR) 1061
39.3 Decorated Loads: l[b,h,w]d{cb}x rT,rB,rA 1061
39.3.1 Simple Load (SLD) 1062
39.3.2 Registers-and-Memory Exchange (SWAP) 1062
39.3.3 Load-and-Set-1(Bit) (LAS1) 1063
39.4 DSMC Instantiations 1063

40 Analog-to-Digital Converters (ADC) Configuration 1064
40.1 ADC overview 1064
40.1.1 ADC subsystem block diagram 1066

40.1.2 Analog input pin multiplexing 1066
40.2 Configuration of ADC modules 1067
40.2.1 Sigma-Delta Analog-to-Digital Converter (SDADC) 1067
40.2.2 Successive Approximation Register Analog-to-Digital Converter (SAR ADC) 1079

41 Sigma-Delta Analog-to-Digital Converter (SDADC) Digital Interface 1091
41.1 Introduction 1091
41.2 Overview 1091
41.3 Features 1093
41.4 Modes of operation 1093
41.4.1 Differential input mode 1093
41.4.2 Single-ended input mode 1093
41.4.3 External modulator mode 1094
41.5 External signal description 1094
41.5.1 Detailed signal descriptions 1094
41.6 Memory map and register descriptions 1095
41.6.1 Memory map 1095
41.6.2 Module Configuration Register (SDADC_MCR) 1096
41.6.3 Channel Selection Register (SDADC_CSR) 1100
41.6.4 Reset Key Register (SDADC_RKR) 1102
41.6.5 Status Flag Register (SDADC_SFR) 1103
41.6.6 Request Select and Enable Register (SDADC_RSER) 1105
41.6.7 Output Settling Delay Register (SDADC_OSDR) 1107
41.6.8 FIFO Control Register (SDADC_FCR) 1108
41.6.9 Software Trigger Key Register (SDADC_STKR) 1109
41.6.10 Converted Data Register (SDADC_CDR) 1110
41.6.11 WDG Threshold Register (SDADC_WTHHLR) 1111
41.7 Functional description 1111
41.7.1 Differential input mode 1112
41.7.2 Single-ended input mode 1112
41.7.3 External modulator mode 1112
41.7.4 Low consumption mode 1112
41.7.5 Analog input multiplexing and bias support 1112
41.7.6 Programmable gain and decimation rate 1113
41.7.7 High-pass filter support 1113

41.7.8 Data conversion 1113
41.7.9 Hardware triggering 1114
41.7.10 Interrupt/DMA request support 1115
41.7.11 Gain calibration support 1115
41.7.12 Offset calibration support 1116
41.8 Initialization information 1116
41.8.1 Data conversion step 1117

42 Successive Approximation Register Analog-to-Digital Converter (SARADC) 1118
42.1 Introduction 1118
42.2 Overview 1118
42.3 Feature description 1119
42.3.1 Main features 1120
42.4 Functional description 1120
42.4.1 Normal channel conversion 1121
42.4.2 Injected channel conversion 1122
42.4.3 Abort conversion 1124
42.4.4 Analog conversion timings 1124
42.4.5 Cross Triggering Unit (CTU) interface 1125
42.4.6 Test channel connection with internal analog channel 1128
42.4.7 External channel mapping to internal analog channel 1128
42.4.8 Programmable analog watchdog 1129
42.4.9 DMA functionality 1130
42.4.10 Interrupts 1130
42.4.11 External decode signals selection and delay 1131
42.4.12 Power down mode 1131
42.4.13 Stop mode 1132
42.4.14 Fast comparator mode conversion 1132
42.4.15 Smart standby wake up support 1133
42.5 Memory map and registers 1134
42.5.1 Register descriptions 1136
42.6 Start of conversion pulse delay 1180
42.7 Initialization information 1182

43 Temperature Sensor 1184

43.1 Introduction 1184
43.1.1 Signal Descriptions 1184
43.2 Functional description 1184
43.2.1 Linear temperature sensor (analog output generation) 1184
43.3 Temperature formula 1185
43.3.1 Equations for converting TSENS voltages to junction temperature . . 1185
43.3.2 Equations for converting TSENS voltages into constant reference (Digital Bandgap Voltage) 1185

44 Body Cross Triggering Unit (BCTU) 1187
44.1 Introduction 1187
44.2 Main features 1187
44.3 Block diagram 1187
44.4 Memory map and register descriptions 1187
44.4.1 Control Status Register (BCTU_CSR) 1188
44.4.2 Event Configuration Register n (BCTU_EVTCFGRn) 1189
44.5 Functional description 1190

45 System Timer Module (STM) 1191
45.1 Introduction 1191
45.1.1 Overview 1191
45.1.2 Features 1191
45.1.3 Modes of operation 1191
45.2 External signal description 1191
45.3 Memory map and register definition 1191
45.3.1 Memory map 1191
45.3.2 Register descriptions 1192
45.4 Functional description 1195

46 Software Watchdog Timer (SWT) 1196
46.1 Introduction 1196
46.1.1 Overview 1196
46.1.2 Features 1196
46.1.3 Modes of operation 1196
46.2 External signal description 1196
46.3 Memory map and register definition 1197

46.3.1 Register descriptions 1197
46.4 Functional description 1202
46.4.1 Introduction 1202
46.4.2 Configuration locking 1203
46.4.3 Unlock sequence 1203
46.4.4 Servicing operations 1203
46.4.5 Time-out 1204
46.4.6 Initialization 1205

47 GTM343 Integration (GTMINT) Module 1206
47.1 Introduction 1206
47.1.1 Features 1207
47.1.2 Modes of Operation 1208
47.1.3 Block Diagram 1209
47.2 Memory Map and Register Definition 1209
47.2.1 Module Memory Map 1210
47.2.2 Register Descriptions 1239
47.3 Functional Description 1242
47.3.1 GTM-IP Timer 1243
47.3.2 GTMDI Debug Interface 1244
47.3.3 Error Detection on GTM Memories Accesses 1244
47.3.4 Stop Mode Description 1248
47.3.5 Debug / Halt Mode Description 1249
47.3.6 GTM-IP Timer Software Reset Description 1249
47.3.7 AEI Interface Software Reset Description 1249
47.3.8 Interrupt and DMA Support Logic 1249
47.3.9 Clock Enables 1251
47.3.10 ADC Interface 1251
47.4 Initialization/Application Information 1251

48 Generic Timer Module (GTM343) 1252
48.1 About this module 1252
48.2 For more information 1252

49 GTM Development Interface (GTMDI) 1253
49.1 Introduction 1253

49.2 Overview . . . 1253
49.2.1 Features . . . 1257
49.2.2 Modes of operation . . . 1258
49.2.3 DATA port . . . 1258
49.3 External signal description . . . 1258
49.3.1 Event in (EVTI) . . . 1259
49.3.2 Event out EVTO[1:0] . . . 1259
49.3.3 Test clock input (TCK) . . . 1259
49.3.4 Test data input (TDI) . . . 1259
49.3.5 Test data output (TDO) . . . 1259
49.3.6 Test Mode Select (TMS) . . . 1260
49.4 Register definition . . . 1260
49.4.1 Register descriptions . . . 1262
49.5 Functional description . . . 1334
49.5.1 Debug enable logic . . . 1334
49.5.2 TIM-TOM-ATOM selection and control logic . . . 1335
49.5.3 GTMDI reset configuration . . . 1336
49.5.4 Message Data Bus —interface with NAR module . . . 1337
49.5.5 IEEE 1149.1 (JTAG) input port . . . 1341
49.5.6 Nexus Class 1 development support . . . 1346
49.5.7 Debug status . . . 1346
49.5.8 Data trace . . . 1346
49.5.9 Fetch trace . . . 1349
49.5.10 Watchpoint trace . . . 1351
49.6 Initialization/application information . . . 1352
49.6.1 Accessing GTMDI tool-mapped registers . . . 1352

50 Periodic Interrupt Timer (PIT) . . . 1353
50.1 Introduction . . . 1353
50.1.1 Block diagram . . . 1353
50.1.2 Features . . . 1354
50.2 Signal description . . . 1354
50.2.1 Memory map/register description . . . 1354
50.3 Functional description . . . 1362
50.3.1 General . . . 1362
50.3.2 Interrupts . . . 1363

50.3.3 Chained timers . . . 1363
50.4 Initialization and application information . . . 1364
50.5 Example Configuration for the Lifetime Timer . . . 1365

51 Enhanced Modular IO Subsystem (eMIOS) . . . 1366
51.1 Introduction . . . 1366
51.1.1 Overview of the eMIOS module . . . 1366
51.1.2 Features of the eMIOS module . . . 1366
51.1.3 Modes of operation . . . 1367
51.1.4 Channel implementation . . . 1367
51.2 External signal description . . . 1369
51.3 Memory map and register description . . . 1369
51.3.1 Memory maps . . . 1369
51.3.2 Register description . . . 1370
51.4 Functional description . . . 1382
51.4.1 Unified Channel (UC) . . . 1383
51.4.2 IP Bus Interface Unit (BIU) . . . 1411
51.4.3 Global Clock Prescaler Submodule (GCP) . . . 1412
51.5 Initialization/Application information . . . 1412
51.5.1 Considerations . . . 1412
51.5.2 Application information . . . 1412

52 CAN Subsystem . . . 1416
52.1 Introduction . . . 1416
52.2 Features . . . 1417
52.3 Modular CAN (M_CAN) cores . . . 1418
52.3.1 Features . . . 1419
52.3.2 Block diagram . . . 1420
52.3.3 Dual clock sources . . . 1421
52.3.4 Dual interrupt lines . . . 1421
52.3.5 Memory map and register descriptions . . . 1421
52.3.6 Message RAM . . . 1469
52.3.7 M_CAN functional description . . . 1479
52.3.8 Timestamp generation . . . 1489
52.3.9 Timeout counter . . . 1489
52.3.10 Rx handling . . . 1490

52.3.11 Dedicated Rx Buffers . . . 1496
52.3.12 Debug on CAN Support . . . 1497
52.3.13 Tx handling . . . 1499
52.3.14 FIFO acknowledge handling . . . 1503
52.3.15 Clock Calibration on CAN Unit (CCCU) . . . 1504
52.4 Time Triggered Modular CAN (M_TTCAN) core . . . 1516
52.4.1 Features . . . 1518
52.4.2 Block Diagram . . . 1519
52.4.3 Dual clock sources . . . 1520
52.4.4 Dual interrupt lines . . . 1520
52.4.5 Memory map and registers description . . . 1521
52.4.6 Message RAM . . . 1591
52.4.7 M_TTCAN functional description . . . 1601
52.4.8 FIFO acknowledge handling . . . 1623
52.4.9 M_TTCAN operations . . . 1624
52.4.10 M_TTCAN configuration . . . 1625
52.4.11 Message scheduling . . . 1627
52.4.12 M_TTCAN gap control . . . 1634
52.4.13 Stop watch . . . 1635
52.4.14 Local time, cycle time, global time, and external clock synchronization . . . 1635
52.4.15 M_TTCAN error level . . . 1638
52.4.16 M_TTCAN message handling . . . 1639
52.4.17 M_TTCAN interrupt and error handling . . . 1642
52.4.18 Level 0 . . . 1642
52.4.19 Synchronization to external time schedule . . . 1645
52.5 CAN RAM arbiter . . . 1646
52.5.1 Features . . . 1646
52.5.2 Functional overview using examples . . . 1647
52.6 SRAM interface and memory organization . . . 1647
52.6.1 ECC controller . . . 1647
52.7 Time Triggered CAN matrix and basic cycles management . . . 1648
52.8 CAN nodes 1 and 2 I/Os sharing . . . 1649
52.9 External Time Stamp . . . 1649
52.10 External signal description . . . 1650
52.11 Shared memory map . . . 1650

53 Serial Interprocessor Interface (SIPI) 1653

53.1 Introduction 1653

53.1.1 Scalability 1653

53.2 Overview 1653

53.3 SIPI block diagram 1655

53.4 Feature description 1655

53.4.1 Main features 1655

53.4.2 Standard features 1656

53.5 SIPI operation from reset 1656

53.6 Functional description 1656

53.6.1 External signals 1656

53.6.2 Frame format 1656

53.7 Transfer types 1662

53.7.1 Read transfer 1662

53.7.2 Register read answer transfer 1663

53.7.3 Register Write transfer 1665

53.7.4 Write acknowledge transfer 1667

53.7.5 ID request response 1668

53.8 Transfer API and flow charts 1669

53.9 DMA handling in transfer initiator mode 1675

53.10 Modes of operation 1676

53.10.1 Initialization mode (INIT) 1676

53.10.2 Normal Mode (NM) 1676

53.10.3 Module Disable (MD) 1677

53.11 Errors 1677

53.11.1 Timeout error 1677

53.11.2 CRC error 1677

53.11.3 Maximum count reached error 1677

53.11.4 Transaction ID error 1678

53.11.5 Acknowledge error 1678

53.12 CRC calculation 1678

53.13 Interrupt logic 1678

53.14 SIPI control and status overview 1679

53.15 Memory map and register description 1680

53.15.1 Register descriptions 1682

54 Zipwire 1696
54.1 Overview 1696
54.2 Introduction 1696
54.3 Zipwire block diagram 1696
54.4 Architecture 1697
54.5 LFAST clocking 1698
54.6 Zipwire interconnections 1698

55 LVDS Fast Asynchronous Serial Transmission – Interprocessor Communications 1700
55.1 Introduction 1700
55.2 Block diagram 1700
55.3 External signals 1700
55.3.1 LFAST operating data rates 1701
55.4 LFAST frame structure 1701
55.5 Features 1704
55.6 Memory map and register definition 1704
55.6.1 Memory map 1704
55.6.2 Register descriptions 1706
55.6.3 Register safety classification requirements 1736
55.7 Functional description 1737
55.7.1 Startup procedure 1737
55.7.2 Line Receiver (LR) 1739
55.7.3 Transmit controller 1749
55.7.4 CTS mode support 1751
55.7.5 Frames supported 1752
55.7.6 Frame flow 1753
55.7.7 Test and debug support 1759
55.7.8 Interrupts 1766
55.8 Packet memory 1768
55.9 Resets 1769
55.10 Clocks 1770
55.10.1 Clocking strategy 1770
55.10.2 Slow speed clock 1771
55.10.3 Rx controller clocks 1773

55.10.4 Tx controller clocks 1774

56 Ethernet 1776

56.1 Overview 1776

56.1.1 Ethernet module Features 1777

56.2 Memory map and register description 1783

56.2.1 MAC Configuration Register (MAC_CONFIGURATION) 1793

56.2.2 MAC Extended Configuration Register (MAC_EXT_CONFIGURATION) 1798

56.2.3 MAC Packet Filter Register (MAC_PACKET_FILTER) 1799

56.2.4 Watchdog Timeout Register (MAC_WATCHDOG_TIMEOUT) 1802

56.2.5 Hash Table Register 0 (MAC_HASH_TABLE_REG0) 1803

56.2.6 Hash Table Register 1 (MAC_HASH_TABLE_REG1) 1804

56.2.7 VLAN Tag Register (MAC_VLAN_TAG) 1805

56.2.8 VLAN Hash Table Register (MAC_VLAN_HASH_TABLE) 1807

56.2.9 VLAN Tag Inclusion Register (MAC_VLAN_INCL) 1808

56.2.10 Inner VLAN Tag Inclusion Register (MAC_INNER_VLAN_INCL) . . . 1809

56.2.11 MAC Q0 Flow Control Register (MAC_Q0_TX_FLOW_CTRL) 1811

56.2.12 MAC Q1 Flow Control Register (MAC_Q1_TX_FLOW_CTRL) 1812

56.2.13 MAC Receive Flow Control Register (MAC_RX_FLOW_CTRL) 1814

56.2.14 Transmit Queue Priority Mapping 0 Register (MAC_TXQ_PRTY_MAP0) 1815

56.2.15 Receive Queue Control 0 Register (MAC_RXQ_CTRL0) 1816

56.2.16 Receive Queue Control 1 Register (MAC_RXQ_CTRL1) 1817

56.2.17 Receive Queue Control 2 Register (MAC_RXQ_CTRL2) 1818

56.2.18 Interrupt Status Register (MAC_INTERRUPT_STATUS) 1819

56.2.19 Interrupt Enable Register (MAC_INTERRUPT_ENABLE) 1821

56.2.20 Receive Transmit Status Register (MAC_RX_TX_STATUS) 1822

56.2.21 PMT Control and Status Register (MAC_PMT_CONTROL_STATUS) 1823

56.2.22 Wake-up Packet Filter Registers 1825

56.2.23 LPI Control and Status Register (MAC_LPI_CONTROL_STATUS) . 1832

56.2.24 LPI Timers Control Register (MAC_LPI_TIMERS_CONTROL) 1834

56.2.25 LPI Entry Timer Register (MAC_LPI_ENTRY_TIMER) 1835

56.2.26 1US Tic Counter Register (MAC_1US_TIC_COUNTER) 1835

56.2.27 Version Register (MAC_VERSION) 1836

56.2.28 Debug Register (MAC_DEBUG) 1837

56.2.29 Hardware Feature0 Register (MAC_HW_FEATURE0) 1837

56.2.30 Hardware Feature1 Register (MAC_HW_FEATURE1) 1839
56.2.31 Hardware Feature2 Register (MAC_HW_FEATURE2) 1841
56.2.32 MDIO Address Register (MAC_MDIO_ADDRESS) 1842
56.2.33 MDIO Data Register (MAC_MDIO_DATA) 1844
56.2.34 ARP Address Register (MAC_ARP_ADDRESS) 1845
56.2.35 Address0 High Register (MAC_ADDRESS0_HIGH) 1845
56.2.36 Address0 Low Register (MAC_ADDRESS0_LOW) 1846
56.2.37 MAC Address High register 1 to 31 (MAC_ADDRESSn_HIGH31_1) 1846
56.2.38 MAC Address Low register 1 to 31 (MAC_ADDRESSn_LOW31_1) 1848
56.2.39 MAC Address High register 32 to 63 (MAC_ADDRESSn_HIGH63_32) 1848
56.2.40 MAC Address Low register 32 to 63 (MAC_ADDRESSn_LOW63_32) 1849
56.2.41 MAC Address High register 64 to 127 (MAC_ADDRESSn_HIGH127_64) 1849
56.2.42 MAC Address Low register 64 to 127 (MAC_ADDRESSn_LOW127_64) 1850
56.2.43 MMC Control Register (MMC_CONTROL) 1851
56.2.44 MMC Receive Interrupt Register (MMC_RX_INTERRUPT) 1852
56.2.45 MMC Tx Interrupt Register (MMC_TX_INTERRUPT) 1856
56.2.46 MMC Rx Interrupt mask Register (MMC_RX_INTERRUPT_MASK) 1861
56.2.47 MMC Tx Interrupt Mask Register (MMC_TX_INTERRUPT_MASK) 1863
56.2.48 Transmit Octet Count Good Bad Register (TX_OCTET_COUNT_GOOD_BAD) 1866
56.2.49 Transmit Packet Count Good Bad Register (TX_PACKET_COUNT_GOOD_BAD) 1867
56.2.50 Transmit Broadcast Packets Good Register (TX_BROADCAST_PACKETS_GOOD) 1868
56.2.51 Transmit Multicast Packets Good Register (TX_MULTICAST_PACKETS_GOOD) 1868
56.2.52 Transmit 64Octets Packets Good Bad Register (TX_64OCTETS_PACKETS_GOOD_BAD) 1869
56.2.53 Transmit 65to127Octets Packets Good Bad Register (TX_65TO127OCTETS_PACKETS_GOOD_BAD) 1869
56.2.54 Transmit 128To255Octets Packets Good Bad Register (TX_128TO255OCTETS_PACKETS_GOOD_BAD) 1870
56.2.55 Transmit 256To511Octets Packets Good Bad Register (TX_256TO511OCTETS_PACKETS_GOOD_BAD) 1871

56.2.56 Transmit 512To1023Octets Packets Good Bad Register (TX_512TO1023OCTETS_PACKETS_GOOD_BAD) 1871
56.2.57 Transmit 1024ToMaxOctets Packets Good Bad Register (TX_1024TOMAXOCTETS_PACKETS_GOOD_BAD) 1872
56.2.58 Transmit Unicast Packets Good Bad Register (TX_UNICAST_PACKETS_GOOD_BAD) 1872
56.2.59 Transmit Multicast Packets Good Bad Register (TX_MULTICAST_PACKETS_GOOD_BAD) 1873
56.2.60 Transmit Broadcast Packets Good Bad Register (TX_BROADCAST_PACKETS_GOOD_BAD) 1874
56.2.61 Transmit Underflow Error Packets Register (TX_UNDERFLOW_ERROR_PACKETS) 1874
56.2.62 Transmit Single Collision Good Packets Register (TX_SINGLE_COLLISION_GOOD_PACKETS) 1875
56.2.63 Transmit Multiple Collision Good Packets Register (TX_MULTIPLE_COLLISION_GOOD_PACKETS) 1875
56.2.64 Transmit Deferred Packets Register (TX_DEFERRED_PACKETS) 1876
56.2.65 Transmit Late Collision Packets Register (TX_LATE_COLLISION_PACKETS) 1877
56.2.66 Transmit Excessive Collision Packets Register (TX_EXCESSIVE_COLLISION_PACKETS) 1877
56.2.67 Transmit Carrier Error Packets Register (TX_CARRIER_ERROR_PACKETS) 1878
56.2.68 Transmit Octet Count Good Register (TX_OCTET_COUNT_GOOD) 1878
56.2.69 Transmit Packet Count Good Register (TX_PACKET_COUNT_GOOD) 1879
56.2.70 Transmit Excessive Deferral Error Register (TX_EXCESSIVE_DEFERRAL_ERROR) 1880
56.2.71 Transmit Pause Packets Register (TX_PAUSE_PACKETS) 1880
56.2.72 Transmit VLAN Packets Good Register (TX_VLAN_PACKETS_GOOD) 1881
56.2.73 Transmit OSize Packets Good Register (TX_OSIZE_PACKETS_GOOD) 1881
56.2.74 Receive Packets Count Good Bad Register (RX_PACKETS_COUNT_GOOD_BAD) 1882
56.2.75 Receive Octet Count Good Bad Register (RX_OCTET_COUNT_GOOD_BAD) 1882
56.2.76 Receive Octet Count Good Register (RX_OCTET_COUNT_GOOD) 1883
56.2.77 Receive Broadcast Packets Good Register (RX_BROADCAST_PACKETS_GOOD) 1884

56.2.78 Receive Multicast Packets Good Register (RX_MULTICAST_PACKETS_GOOD) 1884
56.2.79 Receive CRC Error Packets Register (RX_CRC_ERROR_PACKETS) 1885
56.2.80 Receive Alignment Error Packets Register (RX_ALIGNMENT_ERROR_PACKETS) 1885
56.2.81 Receive Runt Error Packets Register (RX_RUNT_ERROR_PACKETS) 1886
56.2.82 Receive Jabber Error Packets Register (RX_JABBER_ERROR_PACKETS) 1887
56.2.83 Receive Undersize Packets Good Register (RX_UNDERSIZE_PACKETS_GOOD) 1887
56.2.84 Receive Oversize Packets Good Register (RX_OVERSIZE_PACKETS_GOOD) 1888
56.2.85 Receive 64Octets Packets Good Bad Register (RX_64OCTETS_PACKETS_GOOD_BAD) 1888
56.2.86 Receive 65To127Octets Packets Good Bad Register (RX_65TO127OCTETS_PACKETS_GOOD_BAD) 1889
56.2.87 Receive 128To255Octets Packets Good Bad Register (RX_128TO255OCTETS_PACKETS_GOOD_BAD) 1890
56.2.88 Receive 256To511Octets Packets Good Bad Register (RX_256TO511OCTETS_PACKETS_GOOD_BAD) 1890
56.2.89 Receive 512To1023Octets Packets Good Bad Register (RX_512TO1023OCTETS_PACKETS_GOOD_BAD) 1891
56.2.90 Receive 1024ToMax Octets Packets Good bad Register (RX_1024TOMAXOCTETS_PACKETS_GOOD_BAD) 1891
56.2.91 Receive Unicast Packets Good Register (RX_UNICAST_PACKETS_GOOD) 1892
56.2.92 Receive Length Error Packets (RX_LENGTH_ERROR_PACKETS) 1893
56.2.93 Receive Out Of Range Type Packets (RX_OUT_OF_RANGE_TYPE_PACKETS) 1893
56.2.94 Receive Pause Packets Register (RX_PAUSE_PACKETS) 1894
56.2.95 Receive FIFO Overflow Packets Register (RX_FIFO_OVERFLOW_PACKETS) 1894
56.2.96 Receive VLAN Packets Good Bad Register (RX_VLAN_PACKETS_GOOD_BAD) 1895
56.2.97 Receive Watchdog Error Packets Register (RX_WATCHDOG_ERROR_PACKETS) 1895
56.2.98 Receive Error Packets Register (RX_RECEIVE_ERROR_PACKETS) 1896
56.2.99 Receive Control Packets Good Register (RX_CONTROL_PACKETS_GOOD) 1897

56.2.100 Transmit LPI USEC Counter (TX_LPI_USEC_CNTR) 1897
56.2.101 Transmit LPI Transaction Counter Register (TX_LPI_TRAN_CNTR) 1898
56.2.102 Receive LPI USEC Counter (RX_LPI_USEC_CNTR) 1898
56.2.103 Receive LPI Transaction Counter Register (RX_LPI_TRAN_CNTR) 1899
56.2.104 MMC IPC Receive Interrupt Mask Register (MMC_IPC_RX_INTERRUPT_MASK) 1899
56.2.105 MMC IPC Receive Interrupt Register (MMC_IPC_RX_INTERRUPT) 1902
56.2.106 Receive IPv4 Good Packets Register (RXIPV4_GOOD_PACKETS) 1906
56.2.107 Receive IPv4 Header Error Packets (RXIPV4_HEADER_ERROR_PACKETS) 1906
56.2.108 Receive IPv4 No Payload Packets Register (RXIPV4_NO_PAYLOAD_PACKETS) 1907
56.2.109 Receive IPv4 Fragmented Packets Register (RXIPV4_FRAGMENTED_PACKETS) 1908
56.2.110 Receive IPv4 UDP Checksum Disabled Packets Register (RXIPV4_UDP_CHECKSUM_DISABLED_PACKETS) 1908
56.2.111 Receive IPv6 Good Packets Register (RXIPV6_GOOD_PACKETS) 1909
56.2.112 Receive IPv6 Header Error Packets (RXIPV6_HEADER_ERROR_PACKETS) 1909
56.2.113 Receive IPv6 Payload Packets Register (RXIPV6_NO_PAYLOAD_PACKETS) 1910
56.2.114 Receive UDP Good Packets Register (RXUDP_GOOD_PACKETS) 1911
56.2.115 Receive UDP Error Packets Register (RXUDP_ERROR_PACKETS) 1911
56.2.116 Receive TCP Good Packets Register (RXTCP_GOOD_PACKETS) 1912
56.2.117 Receive TCP Error Packets Register (RXTCP_ERROR_PACKETS) 1912
56.2.118 Receive ICMP Good Packets Register (RXICMP_GOOD_PACKETS) 1913
56.2.119 Receive ICMP Error Packets Register (RXICMP_ERROR_PACKETS) 1914
56.2.120 Receive IPv4 Good Octets Register (RXIPV4_GOOD_OCTETS) . . 1914
56.2.121 Receive IPv4 Header Error Octets Register (RXIPV4_HEADER_ERROR_OCTETS) 1915
56.2.122 Receive IPv4 No Payload Octets Register (RXIPV4_NO_PAYLOAD_OCTETS) 1915

56.2.123 Receive IPv4 Fragmented Octets Register (RXIPV4_FRAGMENTED_OCTETS) 1916
56.2.124 Receive IPv4 UDP Checksum Disable Octets Register (RXIPV4_UDP_CHECKSUM_DISABLE_OCTETS) 1917
56.2.125 Receive IPv6 Good Octets Register (RXIPV6_GOOD_OCTETS) . . 1917
56.2.126 Receive Header Error Octets Register (RXIPV6_HEADER_ERROR_OCTETS) 1918
56.2.127 Receive IPv6 No Payload Octets Register (RXIPV6_NO_PAYLOAD_OCTETS) 1918
56.2.128 Receive UDP Good Octets Register (RXUDP_GOOD_OCTETS) . . 1919
56.2.129 Receive UDP Error Octets Register (RXUDP_ERROR_OCTETS) . . 1920
56.2.130 Receive TCP Good Octets Register (RXTCP_GOOD_OCTETS) . . . 1920
56.2.131 Receive TCP Error Octets Register (RXTCP_ERROR_OCTETS) . . 1921
56.2.132 Receive ICMP Good Octets Register (RXICMP_GOOD_OCTETS) . 1921
56.2.133 Receive ICMP Error Octets Register (RXICMP_ERROR_OCTETS) 1922
56.2.134 Layer 3 and Layer 4 Control 0 Register (MAC_L3_L4_CONTROL0) 1922
56.2.135 Layer 4 Address 0 Register (MAC_LAYER4_ADDRESS0) 1925
56.2.136 Layer 3 Address 0 Register 0 Register (MAC_LAYER3_ADDRESS0_REG0) 1926
56.2.137 Layer3 Address1 Register 0 Register (MAC_LAYER3_ADDRESS1_REG0) 1927
56.2.138 Layer3 Address2 Register 0 Register (MAC_LAYER3_ADDRESS2_REG0) 1928
56.2.139 Layer3 Address3 Register 0 Register (MAC_LAYER3_ADDRESS3_REG0) 1928
56.2.140 Timestamp Control Register (MAC_TIMESTAMP_CONTROL) 1929
56.2.141 Sub Second Increment Register (MAC_SUB_SECOND_INCREMENT) 1932
56.2.142 System Time Seconds Register (MAC_SYSTEM_TIME_SECONDS) 1933
56.2.143 System Time Nanoseconds Register (MAC_SYSTEM_TIME_NANOSECONDS) 1934
56.2.144 System Time Seconds Update Register (MAC_SYSTEM_TIME_SECONDS_UPDATE) 1934
56.2.145 System Time Nanoseconds Update Register (MAC_SYSTEM_TIME_NANOSECONDS_UPDATE) 1935
56.2.146 Timestamp Addend Register (MAC_TIMESTAMP_ADDEND) 1936
56.2.147 System Time Higher Word Seconds Register (MAC_SYSTEM_TIME_HIGHER_WORD_SECONDS) 1936
56.2.148 Timestamp Status Register (MAC_TIMESTAMP_STATUS) 1937

56.2.149 Transmit Timestamp Status Nanoseconds Register (MAC_TX_TIMESTAMP_STATUS_NANOSECONDS) 1938
56.2.150 Transmit Timestamp Status Seconds Register (MAC_TX_TIMESTAMP_STATUS_SECONDS) 1939
56.2.151 Auxiliary Control Register (MAC_AUXILIARY_CONTROL) 1940
56.2.152 Auxiliary Timestamp Nanoseconds Register (MAC_AUXILIARY_TIMESTAMP_NANOSECONDS) 1941
56.2.153 Auxiliary Timestamp Seconds Register (MAC_AUXILIARY_TIMESTAMP_SECONDS) 1942
56.2.154 Timestamp Ingress Asymmetry Correction Register (MAC_TIMESTAMP_INGRESS_ASYM_CORR) 1942
56.2.155 Timestamp Egress Asymmetry Correction Register (MAC_TIMESTAMP_EGRESS_ASYM_CORR) 1943
56.2.156 Timestamp Ingress Correction Nanosecond Register (MAC_TIMESTAMP_INGRESS_CORR_NANOSECOND) 1943
56.2.157 Timestamp Egress Correction Nanosecond Register (MAC_TIMESTAMP_EGRESS_CORR_NANOSECOND) 1944
56.2.158 PPS Control Register (MAC_PPS_CONTROL) 1945
56.2.159 PPS0 Target Time Seconds Register (MAC_PPS0_TARGET_TIME_SECONDS) 1947
56.2.160 PPS0 Target Time Nanoseconds Register (MAC_PPS0_TARGET_TIME_NANOSECONDS) 1948
56.2.161 PPS0_Interval Register (MAC_PPS0_INTERVAL) 1948
56.2.162 PPS0 Width Register (MAC_PPS0_WIDTH) 1949
56.2.163 Operation Mode Register (MTL_OPERATION_MODE) 1950
56.2.164 Debug Access Control Register (MTL_DBG_CTL) 1951
56.2.165 Debug Status Register (MTL_DBG_STS) 1953
56.2.166 FIFO Debug Data register (MTL_FIFO_DEBUG_DATA) 1954
56.2.167 Interrupt Status Register (MTL_INTERRUPT_STATUS) 1955
56.2.168 Receive Queue and DMA Channel Mapping 0 Register (MTL_RXQ_DMA_MAP0) 1956
56.2.169 Queue 0 Transmit Operation Mode Register (MTL_TXQ0_OPERATION_MODE) 1957
56.2.170 Queue 0 Underflow Counter Register (MTL_TXQ0_UNDERFLOW) 1958
56.2.171 Queue 0 Transmit Debug Register (MTL_TXQ0_DEBUG) 1959
56.2.172 Queue 0 ETS Status Register (MTL_TXQ0_ETS_STATUS) 1961
56.2.173 Queue 0 Quantum or Weights Register (MTL_TXQ0_QUANTUM_WEIGHT) 1961
56.2.174 Interrupt Control Status Register (MTL_Q0_INTERRUPT_CONTROL_STATUS) 1962

56.2.175 Queue 0 Receive Operation Mode Register (MTL_RXQ0_OPERATION_MODE) 1963
56.2.176 Queue 0 Missed Packet and Overflow Counter Register (MTL_RXQ0_MISSED_PACKET_OVERFLOW_CNT) 1966
56.2.177 Queue 0 Receive Debug Register (MTL_RXQ0_DEBUG) 1967
56.2.178 Queue Receive Control Register (MTL_RXQ0_CONTROL) 1968
56.2.179 Transmit Qn Operation Mode register (MTL_TXQn_OPERATION_MODE) 1969
56.2.180 Transmit Qn Underflow register (MTL_TXQn_UNDERFLOW) 1970
56.2.181 Transmit Qn Debug register (MTL_TXQn_DEBUG) 1971
56.2.182 Transmit Qn ETS Control register (MTL_TXQn_ETS_CONTROL) . . 1973
56.2.183 Transmit Qn ETS Status register (MTL_TXQn_ETS_STATUS) 1974
56.2.184 Transmit Qn Quantum Weight register (MTL_TXQn_QUANTUM_WEIGHT) 1975
56.2.185 Transmit Qn Send Slope Credit register (MTL_TXQn_SENDSPLOPECREDIT) 1976
56.2.186 Transmit Qn High Credit register (MTL_TXQn_HICREDIT) 1976
56.2.187 Transmit Qn Low Credit register (MTL_TXQn_LOCREDIT) 1977
56.2.188 Qn Interrupt Control Status register (MTL_Qn_INTERRUPT_CONTROL_STATUS) 1977
56.2.189 Receive Qn Operation Mode register (MTL_RXQn_OPERATION_MODE) 1979
56.2.190 Receive Qn Missed Packet and Overflow Counter register (MTL_RXQn_MISSED_PACKET_OVERFLOW_CNT) 1981
56.2.191 Receive Qn Debug register (MTL_RXQn_DEBUG) 1983
56.2.192 Receive Qn Control register (MTL_RXQn_CONTROL) 1983
56.2.193 DMA_Mode Register (DMA_MODE) 1984
56.2.194 System Bus Mode Register (DMA_SYSBUS_MODE) 1986
56.2.195 Interrupt Status Register (DMA_INTERRUPT_STATUS) 1987
56.2.196 Debug Status 0 Register (DMA_DEBUG_STATUS0) 1988
56.2.197 DMA Channel n Control register (DMA_CHn_CONTROL) 1989
56.2.198 DMA Channel n Transmit Control register (DMA_CHn_TX_CONTROL) 1990
56.2.199 DMA Channel n Receive Control register (DMA_CHn_RX_CONTROL) 1991
56.2.200 DMA Channel n Transmit Descriptor List Address register (DMA_CHn_TXDESC_LIST_ADDRESS) 1993
56.2.201 DMA Channel n Receive Descriptor List Address register (DMA_CHn_RXDESC_LIST_ADDRESS) 1994
56.2.202 DMA Channel n Transmit Descriptor Tail Pointer register (DMA_CHn_TXDESC_TAIL_POINTER) 1995

56.2.203 DMA Channel n Receive Descriptor Tail Pointer (DMA_CHn_RXDESC_TAIL_POINTER) 1996
56.2.204 DMA Channel n Transmit Descriptor Ring Length register (DMA_CHn_TXDESC_RING_LENGTH) 1996
56.2.205 DMA Channel n Receive Descriptor Ring Length register (DMA_CHn_RXDESC_RING_LENGTH) 1997
56.2.206 DMA Channel n Interrupt Enable register (DMA_CHn_INTERRUPT_ENABLE) 1998
56.2.207 DMA Channel n Receive Interrupt Watchdog Timer register (DMA_CHn_Rx_INTERRUPT_WATCHDOG_TIMER) 2000
56.2.208 DMA Channel n Slot Function Control Status register (DMA_CHn_SLOT_FUNCTION_CONTROL_STATUS) 2001
56.2.209 DMA Channel n Current Application Transmit Descriptor register (DMA_CHn_CURRENT_APP_TXDESC) 2002
56.2.210 DMA Channel n Current Application Transmit Descriptor register (DMA_CHn_CURRENT_APP_RXDESC) 2003
56.2.211 DMA Channel n Current Application Transmit Buffer register (DMA_CHn_CURRENT_APP_TXBUFFER) 2003
56.2.212 DMA Channel n Current Application Receive Buffer register (DMA_CHn_CURRENT_APP_RXBUFFER) 2004
56.2.213 DMA Channel n Status register (DMA_CHn_STATUS) 2005
56.2.214 DMA Channel n Miss Frame Counter register (DMA_CHn_MISS_FRAME_CNT) 2008
56.3 Functional Description 2009
56.3.1 AHB Master Interface 2009
56.3.2 DMA Controller 2010
56.3.3 MAC Transaction Layer (MTL) 2023
56.3.4 MAC 2028
56.3.5 Multiple Channels and Queues Support 2032
56.3.6 Double VLAN Processing 2042
56.3.7 Source Address Insertion, Replacement, or Deletion 2044
56.3.8 Packet Filtering 2044
56.3.9 IEEE 1588 Timestamps 2051
56.3.10 IPv4 ARP Offload 2069
56.3.11 Memory Access in Debug Mode 2069
56.3.12 Loopback 2073
56.3.13 MAC Flow Control 2074
56.3.14 Checksum Offload Engine 2076
56.3.15 MAC Management Counters 2081
56.3.16 Low Power Modes 2082

56.3.17 Interrupts . . . 2089
56.4 Descriptors . . . 2093
56.4.1 Overview . . . 2093
56.5 Programming Guidelines . . . 2117
56.5.1 Initializing DMA . . . 2117
56.5.2 Initializing MTL Registers . . . 2118
56.5.3 Initializing MAC . . . 2118
56.5.4 Programming Guidelines for Multi-Channel Multi-Queuing . . . 2120
56.5.5 Programming Guidelines for MII Link State Transitions . . . 2121
56.5.6 Programming Guidelines for IEEE 1588 Timestamping . . . 2122
56.5.7 Programming Guidelines for AV Feature . . . 2123
56.5.8 Programming Guidelines for Energy Efficient Ethernet . . . 2125
56.5.9 Programming Guidelines for Flexible Pulse-Per-Second (PPS) Output . . . 2126
56.5.10 Programming Guidelines for VLAN filtering on Receive . . . 2128

57 FlexRay communication controller (FlexRay) . . . 2129
57.1 Introduction . . . 2129
57.1.1 Reference . . . 2129
57.1.2 Color coding . . . 2129
57.1.3 Overview . . . 2129
57.1.4 Features . . . 2131
57.1.5 Modes of operation . . . 2132
57.2 External signal description . . . 2133
57.2.1 Detailed signal description . . . 2133
57.3 Controller host interface clocking . . . 2134
57.4 Protocol engine clocking . . . 2134
57.4.1 Oscillator clocking . . . 2134
57.4.2 PLL clocking . . . 2134
57.5 Memory map and register description . . . 2135
57.5.1 Memory map . . . 2135
57.5.2 Register descriptions . . . 2140
57.6 Functional description . . . 2232
57.6.1 Message buffer concept . . . 2232
57.6.2 Physical message buffer . . . 2232
57.6.3 Message buffer types . . . 2233

57.6.4 FlexRay memory area layout . . . 2240
57.6.5 Physical message buffer description . . . 2243
57.6.6 Individual message buffer functional description . . . 2253
57.6.7 Individual message buffer search . . . 2271
57.6.8 Individual message buffer reconfiguration . . . 2274
57.6.9 Receive FIFOs . . . 2274
57.6.10 Channel device modes . . . 2280
57.6.11 External clock synchronization . . . 2282
57.6.12 Sync frame ID and sync frame deviation tables . . . 2283
57.6.13 MTS generation . . . 2286
57.6.14 Key slot transmission . . . 2287
57.6.15 Sync frame filtering . . . 2287
57.6.16 Strobe signal support . . . 2288
57.6.17 Timer support . . . 2289
57.6.18 Slot status monitoring . . . 2290
57.6.19 System bus access . . . 2294
57.6.20 Interrupt support . . . 2295
57.6.21 Lower bit rate support . . . 2299
57.6.22 PE data memory (PE DRAM) . . . 2299
57.6.23 CHI lookup-table memory (CHI LRAM) . . . 2300
57.6.24 Memory content error detection . . . 2301
57.6.25 Memory error injection . . . 2305
57.6.26 StopWatch function . . . 2307
57.7 Application information . . . 2309
57.7.1 Module configuration . . . 2309
57.7.2 Initialization sequence . . . 2310
57.7.3 Memory error injection out of POC:default config . . . 2312
57.7.4 Shutdown sequence . . . 2312
57.7.5 Number of usable message buffers . . . 2312
57.7.6 Protocol control command execution . . . 2314
57.7.7 Message buffer search on simple message buffer configuration . . . 2315

58 Inter-Integrated Circuit (I2C) . . . 2318
58.1 Overview . . . 2318
58.2 Introduction to I2C . . . 2318
58.2.1 Definition: I^2C module . . . 2318
58.2.2 Advantages of the IC bus . . . 2318

58.2.3 Module block diagram . . . 2318
58.2.4 Features . . . 2320
58.2.5 Modes of operation . . . 2320
58.2.6 Definition: I2C conditions . . . 2321
58.3 External signal descriptions . . . 2322
58.3.1 Signal overview . . . 2322
58.3.2 Detailed external signal descriptions . . . 2322
58.4 Memory map and register definition . . . 2322
58.4.1 I2C memory map . . . 2322
58.4.2 Register accessibility . . . 2323
58.4.3 I2C Bus Address register (IBAD) . . . 2323
58.4.4 I2C Bus Frequency Divider register (IBFD) . . . 2324
58.4.5 I2C Bus Control Register (IBCR) . . . 2324
58.4.6 I2C Bus Status Register (IBSR) . . . 2326
58.4.7 I2C Bus Data I/O Register (IBDR) . . . 2328
58.4.8 I2C Bus Interrupt Config register (IBIC) . . . 2329
58.4.9 I2C Bus Debug register (IBDBG) . . . 2330
58.5 Functional description . . . 2331
58.5.1 Notes about module operation . . . 2331
58.5.2 Transactions . . . 2331
58.5.3 Arbitration procedure . . . 2334
58.5.4 Clock behavior . . . 2334
58.5.5 Interrupts . . . 2343
58.5.6 IPG STOP mode . . . 2344
58.5.7 IPG DEBUG mode . . . 2345
58.5.8 DMA interface . . . 2347
58.6 Initialization and application information . . . 2347
58.6.1 Recommended interrupt service flow . . . 2347
58.6.2 General programming guidelines (for both master and slave mode) . 2349
58.6.3 Programming guidelines specific to master mode . . . 2350
58.6.4 Programming guidelines specific to slave mode . . . 2353
58.6.5 DMA application information . . . 2353

59 SENT Receiver (SRX) . . . 2359

59.1 Introduction . . . 2359
59.1.1 Features . . . 2360
59.1.2 Modes of operation . . . 2361

59.1.3 Block diagram . . . 2363
59.1.4 Design overview . . . 2363
59.2 External signal description . . . 2365
59.2.1 SENT_RX [(CH-1):0] . . . 2365
59.3 Memory map and register definition . . . 2365
59.3.1 Memory map . . . 2365
59.3.2 Register descriptions . . . 2367
59.3.3 Register classification for safety . . . 2398
59.4 Functional description . . . 2399
59.4.1 Initialization sequence . . . 2399
59.4.2 DMA read logic . . . 2400
59.4.3 Message reading via interrupts . . . 2406
59.4.4 Overflow behavior . . . 2407
59.4.5 Adjustment for variation in sensor (Tx) clock . . . 2407
59.4.6 Input programmable filter . . . 2408
59.4.7 Receiver diagnostics . . . 2409
59.4.8 Time stamp logic . . . 2413
59.4.9 Bus-idle diagnostic . . . 2414
59.5 Clocks and resets . . . 2414
59.5.1 Clocking strategy . . . 2414
59.5.2 System bus clock requirements . . . 2415
59.5.3 High frequency receiver clock (protocol clock) requirements . . . 2416

60 Peripheral Sensor Interface (PSI5) . . . 2417
60.1 Introduction . . . 2417
60.1.1 Overview . . . 2417
60.1.2 Features . . . 2419
60.1.3 Modes of operation . . . 2420
60.2 External signal description . . . 2425
60.3 Memory map and register description . . . 2425
60.3.1 Memory map . . . 2425
60.3.2 Register description . . . 2434
60.4 Functional description . . . 2486
60.4.1 Sensor-to-ECU communication . . . 2487
60.4.2 Data Link Layer . . . 2488
60.4.3 ECU-to-sensor communication . . . 2509

60.4.4 Tx/Rx scenario . . . 2520
60.4.5 Interrupt handling . . . 2521
60.4.6 DMA support . . . 2523
60.4.7 Message read through interrupt generation . . . 2524
60.5 Clock and Reset . . . 2525
60.5.1 Latency and Jitter related requirements . . . 2527
60.5.2 Reset . . . 2527
60.6 Initialization information . . . 2527
60.6.1 Initialization . . . 2527
60.7 Design Assumptions and Limitations . . . 2528
60.7.1 Assumptions . . . 2528
60.7.2 Limitations . . . 2528
60.8 Miscellaneous descriptions . . . 2529
60.8.1 Internal SYNC Pulse generation and coordination across PSI5 channels . . . 2529
60.8.2 Valid states for integrated sync pulse generator . . . 2533
60.8.3 Implementation of EI Diagnostic Register . . . 2535

61 Peripheral Sensor Interface-Support module (PSI5-S) . . . 2536
61.1 Introduction . . . 2536
61.1.1 Overview . . . 2536
61.1.2 Features . . . 2538
61.1.3 PSI5-S global modes of operation . . . 2539
61.1.4 PSI5-S system modes of operation . . . 2542
61.2 External signal description . . . 2542
61.3 Memory map and register description . . . 2543
61.3.1 Memory map . . . 2543
61.3.2 Transfer error generation . . . 2553
61.3.3 UART register configuration . . . 2554
61.3.4 PSI5-S register description . . . 2571
61.4 Functional description . . . 2612
61.4.1 PSI5-S UART message structure . . . 2613
61.4.2 PSI5 received message handling . . . 2615
61.4.3 ECU-to-sensor communication . . . 2623
61.4.4 Timestamp . . . 2630
61.4.5 PSI5-S watchdog operation . . . 2632

61.4.6 Interrupt handling 2634
61.5 PSI5-S clock and reset 2637

62 Deserial Serial Peripheral Interface (DSPI) 2639
62.1 Introduction 2639
62.1.1 Block diagram 2639
62.1.2 Features 2640
62.1.3 DSPI configurations 2641
62.1.4 Modes of operation 2643
62.2 DSPI signal description 2644
62.2.1 PCS0/SS — Peripheral Chip Select/Slave Select 2644
62.2.2 PCS1 – PCS3 — Peripheral Chip Selects 1 – 3 2644
62.2.3 PCS4 — Peripheral Chip Select 4 2644
62.2.4 PCS5/PCSS — Peripheral Chip Select 5/Peripheral Chip Select Strobe 2644
62.2.5 PCS[6] – PCS[7] — Peripheral Chip Selects 6 – 7 2645
62.2.6 SIN — Serial Input 2645
62.2.7 SOUT — Serial Output 2645
62.2.8 SCK — Serial Clock 2645
62.2.9 HT — Hardware Trigger 2645
62.3 Memory map/register definition 2645
62.3.1 DSPI Module Configuration Register (DSPI_MCR) 2648
62.3.2 Hardware Configuration Register (DSPI_HCR) 2651
62.3.3 DSPI Transfer Count Register (DSPI_TCR) 2653
62.3.4 DSPI Clock and Transfer Attributes Register (In Master Mode) (DSPI_CTARn) 2654
62.3.5 DSPI Clock and Transfer Attributes Register (In Slave Mode) (DSPI_CTARn_SLAVE) 2659
62.3.6 DSPI Status Register (DSPI_SR) 2661
62.3.7 DSPI DMA/Interrupt Request Select and Enable Register (DSPI_RSER) 2664
62.3.8 DSPI PUSH FIFO Register In Master Mode (DSPI_PUSHR) 2666
62.3.9 DSPI PUSH FIFO Register In Slave Mode (DSPI_PUSHR_SLAVE) 2668
62.3.10 DSPI POP FIFO Register (DSPI_POPR) 2669
62.3.11 DSPI Transmit FIFO Registers (DSPI_TXFRn) 2670
62.3.12 DSPI Receive FIFO Registers (DSPI_RXFRn) 2670
62.3.13 DSPI DSI Configuration Register 0 (DSPI_DSICR0) 2671
62.3.14 DSPI DSI Serialization Data Register 0 (DSPI_SDR0) 2673

62.3.15 DSPI DSI Alternate Serialization Data Register 0 (DSPI_ASDR0) . . 2674
62.3.16 DSPI DSI Transmit Comparison Register 0 (DSPI_COMPR0) 2675
62.3.17 DSPI DSI Deserialization Data Register 0 (DSPI_DDR0) 2675
62.3.18 DSPI DSI Configuration Register 1 (DSPI_DSICR1) 2676
62.3.19 DSPI DSI Serialization Source Select Register 0 (DSPI_SSR0) 2677
62.3.20 DSPI DSI Parallel Input Select Register 0 (DSPI_PISR0) 2678
62.3.21 DSPI DSI Parallel Input Select Register 1 (DSPI_PISR1) 2680
62.3.22 DSPI DSI Parallel Input Select Register 2 (DSPI_PISR2) 2681
62.3.23 DSPI DSI Parallel Input Select Register 3 (DSPI_PISR3) 2682
62.3.24 DSPI DSI Deserialized Data Interrupt Mask Register 0 (DSPI_DIMR0) 2682
62.3.25 DSPI DSI Deserialized Data Polarity Interrupt Register 0 (DSPI_DPIR0) 2683
62.3.26 DSPI DSI Serialization Data Register 1 (DSPI_SDR1) 2684
62.3.27 DSPI DSI Alternate Serialization Data Register 1 (DSPI_ASDR1) . . 2684
62.3.28 DSPI DSI Transmit Comparison Register 1 (DSPI_COMPR1) 2685
62.3.29 DSPI DSI Deserialization Data Register 1 (DSPI_DDR1) 2686
62.3.30 DSPI DSI Serialization Source Select Register 1 (DSPI_SSR1) 2686
62.3.31 DSPI DSI Parallel Input Select Register 4 (DSPI_PISR4) 2687
62.3.32 DSPI DSI Parallel Input Select Register 5 (DSPI_PISR5) 2689
62.3.33 DSPI DSI Parallel Input Select Register 6 (DSPI_PISR6) 2690
62.3.34 DSPI DSI Parallel Input Select Register 7 (DSPI_PISR7) 2691
62.3.35 DSPI DSI Deserialized Data Interrupt Mask Register 1 (DSPI_DIMR1) 2691
62.3.36 DSPI DSI Deserialized Data Polarity Interrupt Register 1 (DSPI_DPIR1) 2692
62.3.37 DSPI Clock and Transfer Attributes Register Extended (DSPI_CTAREn) 2693
62.3.38 DSPI Status Register Extended (DSPI_SREX) 2694
62.4 Register classification for safety 2694
62.5 Functional description 2695
62.5.1 Start and Stop of DSPI transfers 2696
62.5.2 Serial Peripheral Interface (SPI) configuration 2697
62.5.3 Deserial Serial Interface (DSI) configuration 2701
62.5.4 Combined Serial Interface (CSI) configuration 2705
62.5.5 DSPI baud rate and clock delay generation 2706
62.5.6 Transfer formats 2709
62.5.7 Continuous Serial Communications Clock 2718

62.5.8 Slave mode operation constraints 2720
62.5.9 Timed Serial Bus (TSB) 2720
62.5.10 Interleaved TSB (ITSB) Mode 2723
62.5.11 Parity generation and check 2726
62.5.12 Interrupts/DMA requests 2727
62.5.13 Power saving features 2730
62.6 Initialization/application information 2731
62.6.1 Managing DSPI queues 2731
62.6.2 Switching master and slave mode 2731
62.6.3 Initializing DSPI in Master/Slave Modes 2732
62.6.4 Baud rate settings 2732
62.6.5 Delay settings 2733
62.6.6 Calculation of FIFO pointer addresses 2734

63 LINFlexD 2737
63.1 Introduction 2737
63.2 Main features 2739
63.2.1 LIN mode features 2739
63.2.2 UART mode features 2740
63.2.3 Standard features 2740
63.3 Functional description 2740
63.3.1 LIN protocol 2740
63.3.2 LINFlexD features 2742
63.3.3 Timer 2759
63.3.4 UART mode 2759
63.3.5 DMA interface 2764
63.4 Memory map and register description 2783
63.4.1 Memory map 2783
63.4.2 Register description 2785
63.5 Programming considerations 2818
63.5.1 Master node 2818
63.5.2 Slave node 2820
63.5.3 Extended frames 2824
63.5.4 Timeout 2824
63.5.5 UART mode 2825
63.5.6 Interrupts 2825

64 Reset Generation Module (MC_RGM) 2827
64.1 Introduction 2827
64.1.1 Overview 2827
64.1.2 Features 2828
64.1.3 Reset sources 2829
64.2 External signal description 2830
64.3 Memory map and register definition 2830
64.3.1 Register descriptions 2832
64.4 Functional description 2880
64.4.1 Reset state machine 2880
64.4.2 ‘Destructive’ resets 2884
64.4.3 External reset 2885
64.4.4 ‘Functional’ resets 2886
64.4.5 Alternate event generation 2886
64.4.6 ‘Functional’ reset escalation 2887
64.4.7 ‘Destructive’ reset escalation 2887
64.4.8 Individual peripheral resets 2888

65 Boot Assist Flash (BAF) 2889
65.1 Introduction 2889
65.2 Memory map 2889
65.2.1 BAF image header 2889
65.3 Functional description 2890
65.3.1 Boot modes 2891
65.3.2 Device initialization 2891
65.3.3 Flow of control 2891
65.3.4 Optionally wait for HSM 2893
65.3.5 Initialization of BAF DCF clients 2893
65.3.6 Production disable activation protocol and implementation 2899
65.3.7 TDM Diary Operation 2902
65.3.8 Optionally perform a serial boot 2903
65.3.9 Serial boot configuration 2907
65.4 Resources accessed by BAF code execution 2910

66 System Status and Configuration Module (SSCM) 2911
66.1 Introduction 2911

66.1.1 Overview . . . 2911
66.1.2 Features . . . 2911
66.1.3 Modes of operation . . . 2911
66.2 Memory map and register definition . . . 2911
66.2.1 Register descriptions . . . 2912
66.3 Functional description . . . 2917
66.3.1 DCF mechanism . . . 2917
66.3.2 ECC error monitoring . . . 2919
66.3.3 Boot mode functionality . . . 2920
66.3.4 BAF configuration . . . 2920
66.3.5 User code boot sector search . . . 2920
66.3.6 HSM boot header search . . . 2922
66.3.7 Life Cycle . . . 2923
66.4 Initialization and application information . . . 2924
66.4.1 Reset . . . 2924
66.5 Additional safety measures . . . 2924
66.5.1 Spurious reset protection . . . 2924

67 Power management controller digital interface (PMC_Dig) . . . 2925
67.1 Introduction . . . 2925
67.2 Main features . . . 2927
67.2.1 Standard features . . . 2927
67.3 IPS bus interface . . . 2927
67.4 Memory map/register definition . . . 2927
67.5 Registers description . . . 2931
67.5.1 Event Pending Register (EPR_LV0) . . . 2931
67.5.2 Reset Event Enable Register (REE_LV0) . . . 2932
67.5.3 Reset Event Select Register (RES_LV0) . . . 2933
67.5.4 Interrupt Enable (IE_LV0) . . . 2934
67.5.5 FCCU Event Enable Register (FEE_LV0) . . . 2935
67.5.6 Event Pending Register (EPR_LV1) . . . 2936
67.5.7 Reset Event Enable Register (REE_LV1) . . . 2937
67.5.8 Reset Event Select Register (RES_LV1) . . . 2938
67.5.9 Interrupt Enable (IE_LV1) . . . 2938
67.5.10 FCCU Event Enable Register (FEE_LV1) . . . 2939
67.5.11 Event Pending Register (EPR_HV0) . . . 2940

67.5.12 Reset Event Enable Register (REE_HV0) 2942
67.5.13 Reset Event Select Register (RES_HV0) 2943
67.5.14 Interrupt Enable (IE_HV0) 2945
67.5.15 FCCU Event Enable Register (FEE_HV0) 2946
67.5.16 Event Pending Register (EPR_HV1) 2948
67.5.17 Reset Event Enable Register (REE_HV1) 2949
67.5.18 Reset Event Select Register (RES_HV1) 2951
67.5.19 Interrupt Enable Register (IE_HV1) 2953
67.5.20 FCCU Event Enable Register (FEE_HV1) 2955
67.5.21 Supply Gauge Status Register (GR_S) 2956
67.5.22 Pending Gauge Status Register (GR_P) 2960
67.5.23 Global Interrupt Enable Register (IE_G) 2962
67.5.24 Voltage selection of IO (VSIO) 2964
67.5.25 PMOS/NMOS Regulator Status Register (PNREG_S) 2964
67.5.26 HPREG SMPS SELECT STATUS (HPREG_SMPS_SEL_S) 2965
67.5.27 Miscellaneous CTRL REG (MISC_CTRL_REG) 2966
67.5.28 SSWU CTRL Register (SSWU_CTRL_REG) 2967
67.5.29 Event Pending Register (EPR_TD) 2967
67.5.30 Reset Event Enable Register (REE_TD) 2969
67.5.31 Reset Event Select Register (RES_TD) 2970
67.5.32 Temperature Sensor Configuration Register (CTL_TD) 2971
67.5.33 Temperature Sensor FCCU Event Enable Register (FEE_TD) 2973
67.5.34 BIST Mask Status register (BIST_MASK_STATUS) 2974
67.5.35 User BIST Flags Phase1 register (BIST_FLAGS_PHASE1) 2975
67.5.36 User BIST Flags Phase2 register (BIST_FLAGS_PHASE2) 2977
67.5.37 User BIST Control register (BIST_CTRL) 2978
67.5.38 User BIST Time1 and Time0 register (BIST_TIME10) 2979
67.5.39 User BIST Time3 and Time2 register (BIST_TIME32) 2980
67.5.40 User BIST Time6 and Time5 register (BIST_TIME65) 2980
67.5.41 User BIST VD Under Test Monitor register (BIST_DEBUG) 2981
67.6 Standby entry/exit 2982
67.7 Temperature Sensor Interface Logic 2982
67.8 Power On Reset (MC_RGM Phase Gates) 2983
67.9 Flash interface (DCF client usage) 2983
67.10 Voltage monitor BIST programming 2984

68 Power Control Unit (MC_PCU) 2987

68.1 Introduction . . . 2987
68.1.1 Overview . . . 2987
68.1.2 Features . . . 2988
68.2 External Signal Description . . . 2988
68.3 Memory Map and Register Definition . . . 2989
68.3.1 Memory Map . . . 2989
68.3.2 Register descriptions . . . 2989
68.4 Functional description . . . 2993
68.4.1 General . . . 2993
68.4.2 Reset / Power-On Reset . . . 2993
68.4.3 MC_PCU configuration . . . 2993
68.4.4 Mode transitions . . . 2993
68.5 Application information . . . 2995
68.5.1 STANDBY mode considerations . . . 2995

69 Mode Entry Module (MC_ME) . . . 2996
69.1 Introduction . . . 2996
69.1.1 Overview . . . 2996
69.1.2 Features . . . 2998
69.1.3 Modes of operation . . . 2998
69.2 External signal description . . . 2999
69.3 Memory map and register definition . . . 2999
69.3.1 Memory map . . . 3000
69.3.2 Register description . . . 3005
69.4 Functional description . . . 3039
69.4.1 Mode transition request . . . 3039
69.4.2 Modes details . . . 3040
69.4.3 Mode transition process . . . 3044
69.4.4 Protection of Mode Configuration Registers . . . 3053
69.4.5 Mode Transition Interrupts . . . 3053
69.4.6 Peripheral Clock Gating . . . 3055
69.4.7 Application Example . . . 3056

70 e200z425n3 Core Debug Support . . . 3058
70.1 Overview . . . 3058
70.1.1 Software debug facilities . . . 3058

70.1.2 Additional debug facilities . . . 3059
70.1.3 Hardware debug facilities . . . 3059
70.1.4 Sharing debug resources by software/hardware . . . 3060
70.2 Software debug events and exceptions . . . 3061
70.2.1 Instruction Address Compare event . . . 3062
70.2.2 Data Address Compare event . . . 3063
70.2.3 Linked Instruction Address Compare and Data Address Compare events . . . 3070
70.2.4 Trap debug event . . . 3071
70.2.5 Branch Taken debug event . . . 3071
70.2.6 Instruction Complete debug event . . . 3071
70.2.7 Interrupt Taken debug event . . . 3072
70.2.8 Critical Interrupt Taken debug event . . . 3072
70.2.9 Return debug event . . . 3072
70.2.10 Critical Return Debug Event . . . 3072
70.2.11 External debug event . . . 3072
70.2.12 Unconditional debug event . . . 3073
70.2.13 Performance Monitor Interrupt debug event . . . 3073
70.3 Debug registers . . . 3073
70.3.1 Debug Address and Value registers . . . 3073
70.3.2 Debug Control and Status registers . . . 3074
70.3.3 External Debug Resource Allocation Control (EDBRAC0) register . . 3095
70.3.4 Debug Event Select register (DEVENT) register . . . 3103
70.3.5 Debug Data Acquisition Message (DDAM) register . . . 3103
70.4 Using debug resources for stack limit checking . . . 3104
70.5 External debug support . . . 3106
70.5.1 External debug registers . . . 3107
70.5.2 OnCE introduction . . . 3112
70.5.3 JTAG/OnCE pins . . . 3115
70.5.4 OnCE internal interface signals . . . 3115
70.5.5 OnCE interface signals . . . 3116
70.5.6 OnCE controller and serial interface . . . 3118
70.5.7 Access to debug resources . . . 3125
70.5.8 Methods of entering Debug mode . . . 3127
70.5.9 CPU Status and Control Scan Chain Register (CPUSCR) . . . 3128
70.5.10 Reserved registers (Reserved) . . . 3134
70.6 Watchpoint support . . . 3134

70.7 MPU operation during debug 3137
70.8 Cache array access during debug 3138
70.9 Basic steps for enabling, using, and exiting External Debug mode . . . 3138

71 e200z420n3 Core Debug Support 3140

71.1 Overview 3140
71.1.1 Software debug facilities 3140
71.1.2 Additional debug facilities 3141
71.1.3 Hardware debug facilities 3141
71.1.4 Sharing debug resources by software/hardware 3142
71.2 Software debug events and exceptions 3143
71.2.1 Instruction Address Compare event 3144
71.2.2 Data Address Compare event 3145
71.2.3 Linked Instruction Address Compare and Data Address Compare events 3153
71.2.4 Trap debug event 3154
71.2.5 Branch Taken debug event 3154
71.2.6 Instruction Complete debug event 3154
71.2.7 Interrupt Taken debug event 3155
71.2.8 Critical Interrupt Taken debug event 3155
71.2.9 Return debug event 3155
71.2.10 Critical Return Debug Event 3155
71.2.11 External debug event 3155
71.2.12 Unconditional debug event 3156
71.2.13 Performance Monitor Interrupt debug event 3156
71.3 Debug registers 3156
71.3.1 Debug Address and Value registers 3156
71.3.2 Debug Control and Status registers 3157
71.3.3 External Debug Resource Allocation Control (EDBRAC0) register . . 3178
71.3.4 Debug Event Select register (DEVENT) register 3186
71.3.5 Debug Data Acquisition Message (DDAM) register 3186
71.4 Using debug resources for stack limit checking 3187
71.5 External debug support 3189
71.5.1 External debug registers 3190
71.5.2 OnCE introduction 3195
71.5.3 JTAG/OnCE pins 3198
71.5.4 OnCE internal interface signals 3198

71.5.5 OnCE interface signals . . . 3199
71.5.6 OnCE controller and serial interface . . . 3201
71.5.7 Access to debug resources . . . 3208
71.5.8 Methods of entering Debug mode . . . 3210
71.5.9 CPU Status and Control Scan Chain Register (CPUSCR) . . . 3211
71.5.10 Reserved registers (Reserved) . . . 3217
71.6 Watchpoint support . . . 3217
71.7 MPU operation during debug . . . 3220
71.8 Cache array access during debug . . . 3220
71.9 Basic steps for enabling, using, and exiting External Debug mode . . . 3221

72 Debug and Calibration Interface (DCI) . . . 3223
72.1 Introduction . . . 3223
72.1.1 Features . . . 3223
72.1.2 Overview . . . 3223
72.1.3 Operating modes . . . 3225
72.2 External signal description . . . 3237
72.3 Register description . . . 3237
72.3.1 DCI control register (DCI_CR) . . . 3238
72.3.2 DCI EVTx pin multiplexing control register (DCI_PINCR) . . . 3239
72.4 Functional description . . . 3240
72.4.1 DCI mode transition . . . 3240
72.4.2 LFAST LVDS pad fault . . . 3241

73 JTAG Controller (JTAGC) . . . 3243
73.1 Introduction . . . 3243
73.1.1 Block Diagram . . . 3243
73.1.2 Features . . . 3244
73.1.3 Modes of operation . . . 3244
73.2 External signal description . . . 3245
73.2.1 TCK—Test clock input . . . 3245
73.2.2 TDI—Test data input . . . 3245
73.2.3 TDO—Test data output . . . 3245
73.2.4 TMS—Test mode select . . . 3245
73.2.5 JCOMP—JTAG compliancy . . . 3245
73.3 Register description . . . 3246

73.3.1 Instruction register . . . 3246
73.3.2 Bypass register . . . 3246
73.3.3 Device identification register . . . 3246
73.3.4 JTAG_PASSWORD register . . . 3247
73.3.5 Boundary scan register . . . 3248
73.4 Functional description . . . 3248
73.4.1 JTAGC reset configuration . . . 3248
73.4.2 IEEE 1149.1-2001 (JTAG) Test Access Port . . . 3248
73.4.3 TAP controller state machine . . . 3249
73.4.4 JTAGC block instructions . . . 3251
73.4.5 Boundary scan . . . 3254
73.5 Initialization/Application information . . . 3254

74 IEEE 1149.7 Compact JTAG Test Access Port Controller (CJTAG) . . . 3255
74.1 References . . . 3255
74.2 Introduction . . . 3255
74.2.1 Types of operation . . . 3255
74.2.2 Deployment by class . . . 3256
74.2.3 1149.7 TAP signals . . . 3256
74.2.4 TAP.7 architecture . . . 3257
74.2.5 Protocols . . . 3257
74.3 Operating models . . . 3259
74.4 CJTAG implementation summary . . . 3259
74.4.1 T0 Functions . . . 3259
74.4.2 T1 functions . . . 3259
74.4.3 T2 functions . . . 3259
74.4.4 T3 functions . . . 3260
74.4.5 T4 functions . . . 3260
74.5 Ancillary services . . . 3260
74.5.1 Overview . . . 3260
74.5.2 Resets . . . 3260
74.5.3 Start-up Options . . . 3263
74.5.4 RSU operation . . . 3264
74.5.5 TAPC State Machine . . . 3265
74.6 EPU (Extended Protocol Unit) Operation . . . 3266

74.6.1 EPU Operation . . . 3266
74.6.2 EPU Registers . . . 3268
74.6.3 EPU Commands . . . 3278
74.6.4 EPU operating states . . . 3285
74.6.5 System and EPU Paths . . . 3286
74.7 APU (Advanced Protocol Unit) Operation . . . 3287
74.7.1 Overview . . . 3287
74.7.2 Operation . . . 3289
74.7.3 Escape sequences . . . 3290
74.7.4 Signal behaviors . . . 3290
74.7.5 APU Functions . . . 3291
74.7.6 Configuration Change Packets (CP) . . . 3298
74.7.7 Scan Packet . . . 3298
74.7.8 SP Format . . . 3298
74.8 Functional Description . . . 3304
74.8.1 Switching from Standard Protocol to Advanced Protocol . . . 3304

75 JTAG Data Communication (JDC) . . . 3305
75.1 Introduction . . . 3305
75.2 Overview . . . 3305
75.3 Register definition . . . 3306
75.3.1 Register descriptions . . . 3306
75.3.2 Non-Memory mapped register definition . . . 3308
75.4 Functional description . . . 3309
75.4.1 IPS register access . . . 3310
75.4.2 JTAG register access . . . 3310
75.5 Use case example . . . 3310

76 Sequence Processing Unit (SPU) . . . 3312
76.1 Introduction . . . 3312
76.1.1 SPU block diagram . . . 3312
76.1.2 Overview . . . 3313
76.1.3 Features . . . 3314
76.2 SPU actions . . . 3315
76.3 Operation mode . . . 3316
76.3.1 SPU enable/disable . . . 3316

76.4 SPU module interface 3316
76.5 Register definition 3317
76.5.1 Registers description 3321
76.6 Functional description 3363
76.6.1 Input Mux unit 3363
76.6.2 Processor exception vector encoding 3365
76.6.3 State logic unit (SLU) 3366
76.6.4 Sequence formation 3367
76.6.5 Performance counters/timers unit 3370
76.6.6 Action unit 3372
76.6.7 Optional trace group 3372
76.6.8 Format for CKSRC & CKDATA Values 3372

77 Buddy Device—JTAG Master (JTAGM) 3374
77.1 Introduction 3374
77.1.1 Overview 3374
77.1.2 Feature description 3375
77.2 Functional description 3375
77.2.1 Module architecture 3375
77.2.2 JTAGM JCOMP usage in LFAST 3376
77.2.3 JTAGM data error detection 3377
77.2.4 JTAGM message tagging 3377
77.2.5 JTAGM Ready signal 3377
77.2.6 EVTO and EVTI signals 3378
77.2.7 JTAGM configuration and status monitoring 3378
77.2.8 JTAGM to DCI serial interface 3379
77.2.9 JTAGM data handling and CRC calculation in LFAST mode 3379
77.2.10 Software interface 3380
77.3 Modes of operation 3381
77.4 Memory map and register definition 3381
77.4.1 Register descriptions 3381

78 Debug Zipwire 3390
78.1 Overview 3390
78.2 Debug Zipwire Overview 3390
78.3 Debug Zipwire Block Diagram 3390

78.4 Architecture . . . 3391
78.5 Debug Zipwire implementation . . . 3392
78.6 LFAST clocking . . . 3393
78.7 LFAST switch . . . 3394
78.8 Debug SIPI module . . . 3395
78.8.1 ED power supply constraints . . . 3398
78.9 Debug Zipwire interconnections . . . 3398

79 Debug Serial Interprocessor Interface (SIPI) . . . 3400
79.1 Introduction . . . 3400
79.2 Overview . . . 3400
79.3 Debug SIPI block diagram . . . 3401
79.4 Feature description . . . 3401
79.4.1 Main features . . . 3402
79.4.2 Standard features . . . 3402
79.5 Debug SIPI operation from reset . . . 3402
79.6 Functional description . . . 3402
79.6.1 External signals . . . 3402
79.6.2 Frame format . . . 3402
79.7 Transfer types . . . 3407
79.7.1 Read transfer . . . 3407
79.7.2 Register read answer transfer . . . 3408
79.7.3 Register Write transfer . . . 3410
79.7.4 Write acknowledge transfer . . . 3411
79.7.5 ID request response . . . 3412
79.8 Transfer API and flow charts . . . 3413
79.9 CRC calculation . . . 3414
79.10 Memory map and register description . . . 3414
79.10.1 Register descriptions . . . 3414

80 LVDS Fast Asynchronous Serial Transmission – High Speed Debug . . . 3417
80.1 Introduction . . . 3417
80.2 Block diagram . . . 3417
80.3 External signals . . . 3418

80.3.1 LFAST operating data rates . . . 3418
80.4 LFAST frame structure . . . 3418
80.5 Features . . . 3421
80.6 Memory map and register definition . . . 3421
80.6.1 Memory map . . . 3421
80.6.2 Register descriptions . . . 3423
80.6.3 Register safety classification requirements . . . 3443
80.7 Functional description . . . 3443
80.7.1 Startup procedure . . . 3443
80.7.2 LFAST Interface enable signal (lfast_sysclk_en) . . . 3444
80.7.3 Line Receiver (LR) . . . 3445
80.7.4 Transmit controller . . . 3454
80.7.5 Frames supported . . . 3457
80.7.6 Frame flow . . . 3457
80.7.7 Test and debug support . . . 3460
80.8 Packet memory . . . 3468
80.9 Resets . . . 3468
80.10 Clocks . . . 3469
80.10.1 Clocking strategy . . . 3469
80.10.2 Slow speed clock . . . 3470
80.10.3 Rx controller clocks . . . 3473
80.10.4 Tx controller clocks . . . 3474

81 Development Trigger Semaphore (DTS) . . . 3476
81.1 Introduction . . . 3476
81.2 Overview . . . 3476
81.3 DTS device connections . . . 3478
81.3.1 DTS register access . . . 3479
81.4 Memory map and register definition . . . 3479
81.4.1 Register descriptions . . . 3480
81.5 Example application . . . 3483

82 Nexus Aurora Router (NAR) . . . 3485
82.1 Introduction . . . 3485
82.2 Overview . . . 3486

82.3 Features . . . 3486
82.4 Register definition . . . 3486
82.4.1 Register descriptions . . . 3487
82.4.2 Message filtering registers . . . 3490
82.4.3 Trace memory bus configuration registers . . . 3493
82.4.4 Suppress mode triggers register (NAR_STCR) . . . 3495
82.4.5 Receive queue client disable register (NAR_CDR) . . . 3497
82.4.6 Trace Memory fill level and partition configuration register (NAR_AHFPAR) . . . 3499
82.5 Functional description . . . 3499
82.5.1 Reset/Startup . . . 3499
82.5.2 Operation modes . . . 3500
82.5.3 Data reception and arbitration . . . 3502
82.5.4 Suppress mode . . . 3503
82.5.5 Stall detection . . . 3504
82.5.6 Output arbitration . . . 3505
82.5.7 Message translation and formatting . . . 3506
82.5.8 Aurora interface . . . 3507
82.5.9 Trace memory bus output port usage . . . 3507
82.5.10 Internal message generation . . . 3508
82.5.11 Timestamp function . . . 3512
82.5.12 Overlay and internal trace memory full status generation . . . 3513
82.5.13 Configuration/debug interface JTAG (1149.1) . . . 3514
82.5.14 Trace memory bus controller . . . 3517
82.5.15 Legacy client interface controller . . . 3519
82.6 Initialization information . . . 3520
82.7 Application information . . . 3520

83 e200z420n3 Nexus 3 Module . . . 3521
83.1 Introduction . . . 3521
83.1.1 General Description . . . 3521
83.1.2 Terms and Definitions . . . 3521
83.1.3 Feature List . . . 3522
83.1.4 Functional Block Diagram . . . 3524
83.2 Enabling Nexus 3 Operation . . . 3524
83.2.1 Interaction with Low Power Modes . . . 3525
83.3 TCODEs supported . . . 3525

83.4 Nexus 3 Programmer's Model 3531
83.4.1 Client Select Control (CSC) register 3532
83.4.2 Port Configuration Register (PCR) — reference only 3533
83.4.3 Nexus Development Control Register 1 (DC1) 3534
83.4.4 Nexus Development Control Registers 2 & 3 (DC2, DC3) 3535
83.4.5 Nexus Development Control Register 4 (DC4) 3539
83.4.6 Development Status Register (DS) 3540
83.4.7 Watchpoint Trigger (WT, PTSTC, PTETC, DTSTC, DTETC) registers 3541
83.4.8 Nexus Watchpoint Mask (WMSK) register 3547
83.4.9 Nexus Overrun Control (OVCR) register 3548
83.4.10 Data Trace Control (DTC) register 3549
83.4.11 Data Trace Start Address Registers (DTSA1–4) 3551
83.4.12 Data Trace End Address 1–4 (DTEA1–4) registers 3552
83.4.13 Read/Write Access Control/Status (RWCS) register 3553
83.4.14 Read/Write Access Data (RWD) register 3554
83.4.15 Read/Write Access Address (RWA) 3556
83.5 Nexus 3 Register Access via JTAG/OnCE 3556
83.6 Nexus 3 Register Access via Software 3557
83.7 Nexus Message Fields 3557
83.7.1 TCODE Field 3557
83.7.2 Source ID Field (SRC) 3557
83.7.3 Relative Address Field (U-ADDR) 3558
83.7.4 Full Address Field (F-ADDR) 3558
83.8 Nexus Message Queues 3559
83.8.1 Message Queue Overrun 3559
83.8.2 CPU Stall 3559
83.8.3 Message Suppression 3559
83.8.4 Nexus Message Priority 3560
83.8.5 Data Acquisition Message Priority Loss Response 3561
83.8.6 Ownership Trace Message Priority Loss Response 3561
83.8.7 Program Trace Message Priority Loss Response 3561
83.8.8 Program Trace Sync Message Priority Loss Response 3562
83.8.9 Data Trace Message Priority Loss Response 3562
83.9 Debug Status messages 3562
83.10 Error messages 3562

83.11 Ownership trace 3563
83.11.1 Overview 3563
83.11.2 Ownership Trace Messaging (OTM) 3563
83.12 Program Trace 3564
83.12.1 Branch Trace Messaging types 3564
83.12.2 BTM Message For mats 3566
83.12.3 Program Trace Message Fields 3566
83.12.4 Resource Full Messages 3568
83.12.5 Program Correlation Messages 3569
83.12.6 Program Trace Overflow Error messages 3570
83.12.7 Program Trace Synchronization messages 3570
83.12.8 Program Trace Direct/Indirect Branch with Sync messages 3571
83.12.9 Enabling Program Trace 3573
83.12.10 Program Trace Timing Diagrams (2 MDO / 1 MSEO configuration) . 3573
83.13 Data Trace 3574
83.13.1 Data Trace Messaging (DTM) 3574
83.13.2 DTM Message formats 3575
83.13.3 DTM operation 3577
83.13.4 Data Trace Timing Diagrams (8 MDO / 2 MSEO configuration) 3579
83.14 Data Acquisition messaging 3579
83.14.1 Data Acquisition ID Tag field 3579
83.14.2 Data Acquisition Data field 3579
83.14.3 Data Acquisition Trace event 3580
83.15 Watchpoint Trace messaging 3580
83.15.1 Watchpoint Timing Diagram (2 MDO / 1 MSEO configuration) 3582
83.16 External Hardware Trigger Controls 3582
83.17 Nexus 3 Read/Write access to memory-mapped resources 3583
83.17.1 Single write access 3584
83.17.2 Block write access 3584
83.17.3 Single read access 3585
83.17.4 Block read access 3585
83.17.5 Error handling 3586
83.17.6 Read/Write access error message 3587
83.18 Nexus 3 pin interface 3587
83.18.1 Pins implemented 3587
83.18.2 Pin protocol 3590

83.19 Rules for output messages . . . 3592
83.20 Auxiliary port arbitration . . . 3593
83.21 Examples . . . 3593
83.22 Electrical characteristics . . . 3596
83.23 IEEE 1149.1 (JTAG) RD/WR sequences . . . 3596
83.23.1 JTAG sequence for accessing internal Nexus registers . . . 3596
83.23.2 JTAG sequence for read access of memory-mapped resources . . . 3597
83.23.3 JTAG sequence for write access of memory-mapped resources . . . 3597

84 e200z4256 Nexus 3 module . . . 3598

84.1 Introduction . . . 3598
84.1.1 General description . . . 3598
84.1.2 Terms and definitions . . . 3598
84.1.3 Feature list . . . 3599
84.1.4 Functional block diagram . . . 3601
84.2 Enabling Nexus 3 operation . . . 3601
84.2.1 Interaction with Low Power Modes . . . 3602
84.3 TCODEs supported . . . 3602
84.4 Nexus 3 Programmer's Model . . . 3608
84.4.1 Client Select Control (CSC) register . . . 3610
84.4.2 Port Configuration Register (PCR) — reference only . . . 3610
84.4.3 Nexus Development Control Register 1 (DC1) . . . 3611
84.4.4 Nexus Development Control Registers 2 & 3 (DC2, DC3) . . . 3612
84.4.5 Nexus Development Control Register 4 (DC4) . . . 3617
84.4.6 Development Status Register (DS) . . . 3618
84.4.7 Watchpoint Trigger (WT, PTSTC, PTETC, DTSTC, DTETC) registers . . . 3619
84.4.8 Nexus Watchpoint Mask (WMSK) register . . . 3625
84.4.9 Nexus Overrun Control (OVCR) register . . . 3626
84.4.10 Data Trace Control (DTC) register . . . 3627
84.4.11 Data Trace Start Address Registers (DTSA1–4) . . . 3629
84.4.12 Data Trace End Address 1–4 (DTEA1–4) registers . . . 3630
84.4.13 Read/Write Access Control/Status (RWCS) register . . . 3631
84.4.14 Read/Write Access Data (RWD) register . . . 3632
84.4.15 Read/Write Access Address (RWA) . . . 3634
84.5 Nexus 3 Register Access via JTAG/OnCE . . . 3634

84.6 Nexus 3 Register Access via Software 3635
84.7 Nexus Message Fields 3635
84.7.1 TCODE Field 3635
84.7.2 Source ID Field (SRC) 3635
84.7.3 Relative Address Field (U-ADDR) 3636
84.7.4 Full Address Field (F-ADDR) 3636
84.7.5 Timestamp Field (TSTAMP) 3637
84.8 Nexus Message Queues 3637
84.8.1 Message Queue Overrun 3637
84.8.2 CPU Stall 3638
84.8.3 Message Suppression 3638
84.8.4 Nexus Message Priority 3638
84.8.5 Data Acquisition Message Priority Loss Response 3639
84.8.6 Ownership Trace Message Priority Loss Response 3639
84.8.7 Program Trace Message Priority Loss Response 3640
84.8.8 Program Trace Sync Message Priority Loss Response 3640
84.8.9 Data Trace Message Priority Loss Response 3640
84.9 Debug Status messages 3640
84.10 Error messages 3640
84.11 Ownership trace 3641
84.11.1 Overview 3641
84.11.2 Ownership Trace Messaging (OTM) 3641
84.12 Program Trace 3642
84.12.1 Branch Trace Messaging types 3642
84.12.2 BTM Message For mats 3644
84.12.3 Program Trace Message Fields 3645
84.12.4 Resource Full Messages 3646
84.12.5 Program Correlation Messages 3647
84.12.6 Program Trace Overflow Error messages 3648
84.12.7 Program Trace Synchronization messages 3648
84.12.8 Program Trace Direct/Indirect Branch with Sync messages 3650
84.12.9 Enabling Program Trace 3651
84.12.10 Program Trace Timing Diagrams (2 MDO / 1 MSEO configuration) 3651
84.13 Data Trace 3652
84.13.1 Data Trace Messaging (DTM) 3653

84.13.2 DTM Message formats . . . 3653
84.13.3 DTM operation . . . 3655
84.13.4 Data Trace Timing Diagrams (8 MDO / 2 MSEO configuration) . . . 3657
84.14 Data Acquisition messaging . . . 3657
84.14.1 Data Acquisition ID Tag field . . . 3657
84.14.2 Data Acquisition Data field . . . 3657
84.14.3 Data Acquisition Trace event . . . 3658
84.15 Watchpoint Trace messaging . . . 3658
84.15.1 Watchpoint Timing Diagram (2 MDO / 1 MSEO configuration) . . . 3660
84.16 External Hardware Trigger Controls . . . 3660
84.17 Nexus 3 Read/Write access to memory-mapped resources . . . 3661
84.17.1 Single write access . . . 3662
84.17.2 Block write access . . . 3662
84.17.3 Single read access . . . 3663
84.17.4 Block read access . . . 3663
84.17.5 Error handling . . . 3664
84.17.6 Read/Write access error message . . . 3665
84.18 Nexus 3 pin interface . . . 3665
84.18.1 Pins implemented . . . 3665
84.18.2 Pin protocol . . . 3668
84.19 Rules for output messages . . . 3670
84.20 Auxiliary port arbitration . . . 3671
84.21 Examples . . . 3671
84.22 Electrical characteristics . . . 3674
84.23 IEEE 1149.1 (JTAG) RD/WR sequences . . . 3674
84.23.1 JTAG sequence for accessing internal Nexus registers . . . 3674
84.23.2 JTAG sequence for read access of memory-mapped resources . . . 3675
84.23.3 JTAG sequence for write access of memory-mapped resources . . . 3675

85 Nexus Crossbar Multi-Master Client (NXMC) . . . 3676
85.1 Introduction . . . 3676
85.1.1 Features . . . 3676
85.2 External signal description . . . 3676
85.3 Supported messages and TCODEs . . . 3677
85.4 Register description . . . 3680

85.4.1 Register definitions . . . 3681
85.5 Programming considerations (RESET) . . . 3689
85.5.1 IEEE 1149.1 (JTAG) Test Access Port . . . 3689
85.5.2 Enabling NXMC operation . . . 3689
85.5.3 Enabling NXMC TAP controller . . . 3689
85.5.4 NXMC register access via JTAG . . . 3689
85.6 Functional description . . . 3690
85.6.1 Data trace . . . 3690
85.6.2 Watchpoint support . . . 3695
85.6.3 Peripheral interface . . . 3696
85.6.4 Timestamping feature . . . 3697

86 Cyclic Redundancy Check (CRC) Unit . . . 3699
86.1 Introduction . . . 3699
86.2 Main features . . . 3699
86.2.1 Standard features . . . 3699
86.3 Block diagram . . . 3699
86.4 Signal description . . . 3700
86.4.1 Peripheral bus interface . . . 3700
86.5 Memory map and registers . . . 3700
86.5.1 Register description . . . 3700
86.6 Functional description . . . 3704
86.7 Use cases . . . 3706
86.7.1 Programming example . . . 3706
86.7.2 Register programming . . . 3708

87 Memory Error Management Unit (MEMU) . . . 3709
87.1 Introduction . . . 3709
87.2 Features . . . 3710
87.3 Block diagram . . . 3710
87.4 Design overview . . . 3711
87.5 External signal description . . . 3714
87.6 Memory map and register definition . . . 3714
87.6.1 Memory map . . . 3714
87.6.2 Register descriptions . . . 3716

87.7 Functional description . . . 3731
87.7.1 Initializing MEMU . . . 3731
87.7.2 Reading the reporting table . . . 3732
87.7.3 Handling overflows (Multiple error reporting) . . . 3733

88 Indirect Memory Access (IMA) . . . 3734
88.1 Introduction . . . 3734
88.2 Accessing memory via CPU End2End ECC test . . . 3734
88.2.1 Accessing I-cache and D-cache memory via processor core . . . 3734
88.2.2 Accessing non I-cache or D-cache memory via processor core . . . 3736
88.3 Accessing memory via the IMA module . . . 3738
88.3.1 Features . . . 3739
88.3.2 Block diagram . . . 3739
88.3.3 IMA Module Memory Map and Registers . . . 3740
88.3.4 IMA module functional description . . . 3747
88.3.5 Application information . . . 3749

89 Fault Collection and Control Unit (FCCU) . . . 3750
89.1 Introduction . . . 3750
89.2 Features . . . 3750
89.2.1 Standard features . . . 3751
89.3 Block diagram . . . 3751
89.4 Signal description . . . 3752
89.4.1 Pinout . . . 3752
89.4.2 IPS bus interface . . . 3753
89.5 Functional description . . . 3753
89.5.1 Definitions . . . 3753
89.5.2 FSM description . . . 3754
89.5.3 Reset interface . . . 3755
89.5.4 Fault priority scheme and nesting . . . 3756
89.5.5 Fault recovery . . . 3757
89.5.6 NMI/WKPU interface . . . 3759
89.5.7 FAULT interface . . . 3760
89.5.8 STCU interface . . . 3761
89.5.9 EOUT interface . . . 3762
89.5.10 Error signal flow for RF case . . . 3767

89.6 Register description 3769
89.6.1 FCCU Control Register (FCCU_CTRL) 3771
89.6.2 FCCU CTRL Key Register (FCCU_CTRLK) 3774
89.6.3 FCCU Configuration Register (FCCU_CFG) 3775
89.6.4 FCCU RF Configuration Register n (FCCU_RF_CFGn) 3777
89.6.5 FCCU RFS Configuration Register n (FCCU_RFS_CFGn) 3778
89.6.6 FCCU RF Status register n (FCCU_RF_Sn) 3779
89.6.7 FCCU RF Key register (FCCU_RFK) 3781
89.6.8 FCCU RF Enable Register n (FCCU_RF_En) 3782
89.6.9 FCCU RF Timeout Enable Register n (FCCU_RF_TOEn) 3783
89.6.10 FCCU RF Timeout Register (FCCU_RF_TO) 3784
89.6.11 FCCU CFG Timeout Register (FCCU_CFG_TO) 3785
89.6.12 FCCU IO Control Register (FCCU_EINOUT) 3786
89.6.13 FCCU Status Register (FCCU_STAT) 3787
89.6.14 FCCU NA Freeze Status Register (N2AF_STATUS) 3789
89.6.15 FCCU AF Freeze Status Register (A2FF_STATUS) 3789
89.6.16 FCCU NF Freeze Status Register (N2FF_STATUS) 3790
89.6.17 FCCU FA Freeze Status Register (F2A_STATUS) 3791
89.6.18 FCCU RF Fake Register (FCCU_RFF) 3792
89.6.19 FCCU IRQ Status Register (FCCU_IRQ_STAT) 3793
89.6.20 FCCU IRQ Enable Register (FCCU_IRQ_EN) 3794
89.6.21 FCCU XTMR Register (FCCU_XTMR) 3795
89.6.22 FCCU MCS Register (FCCU_MCS) 3796
89.6.23 FCCU Transient Register (FCCU_TRANS_LOCK) 3799
89.6.24 FCCU Permanent Lock Register (FCCU_PERMNT_LOCK) 3799
89.6.25 FCCU Delta T Register (FCCU_DELTA_T) 3800
89.6.26 FCCU IRQ Alarm Enable register n (FCCU_IRQ_ALARM_ENn) . . . 3800
89.6.27 FCCU NMI Enable registers n (FCCU_NMI_ENn) 3801
89.6.28 FCCU EOUT Signaling Enable registers n (FCCU_EOUT_SIG_ENn) 3802
89.7 FCCU Output Supervision Unit 3803
89.8 Use cases and limitations 3805
89.8.1 Mis-configurations 3805
89.8.2 Recommendations to configure FCCU 3806

90 Self-Test Control Unit (STCU2) 3807
90.1 Introduction 3807

90.2 Main features . . . 3807
90.3 Block diagram . . . 3808
90.3.1 IPS bus interface . . . 3810
90.4 Functional description . . . 3811
90.4.1 FSM description . . . 3811
90.4.2 Reset management . . . 3811
90.4.3 LBIST and MBIST scheduling . . . 3812
90.4.4 ABORT management . . . 3812
90.4.5 PLL interface . . . 3812
90.4.6 Interrupt interface . . . 3813
90.4.7 FCCU interface . . . 3813
90.4.8 Watchdogs . . . 3813
90.4.9 CRC . . . 3814
90.4.10 SSCM interface . . . 3814
90.5 Register description . . . 3815
90.5.1 STCU2 Run Register (STCU_RUN) . . . 3816
90.5.2 STCU2 Run Software Register (STCU_RUNSW) . . . 3817
90.5.3 STCU2 SK Code Register (STCU_SKC) . . . 3819
90.5.4 STCU2 Configuration Register (STCU_CFG) . . . 3820
90.5.5 STCU2 PLL Configuration Register (STCU_PLL_CFG) . . . 3822
90.5.6 STCU2 Watchdog Register Granularity (STCU_WDG) . . . 3823
90.5.7 STCU2 Interrupt Flag Register (STCU_INT_FLG) . . . 3824
90.5.8 STCU2 CRC Expected Status Register (STCU_CRCE) . . . 3825
90.5.9 STCU2 CRC Read Status Register (STCU_CRCR) . . . 3826
90.5.10 STCU2 Error Register (STCU_ERR_STAT) . . . 3827
90.5.11 STCU2 Error FM Register (STCU_ERR_FM) . . . 3830
90.5.12 STCU2 Offline LBIST Status Register (STCU_LBS) . . . 3831
90.5.13 STCU2 Offline LBIST End Flag Register (STCU_LBE) . . . 3831
90.5.14 STCU2 Online LBIST Status Register (STCU_LBSSW) . . . 3832
90.5.15 STCU2 Online LBIST End Flag Register (STCU_LBESW) . . . 3832
90.5.16 STCU2 Online LBIST Reset Management (STCU_LBRMSW) . . . 3833
90.5.17 STCU2 LBIST Unrecoverable FM Register (STCU_LBUFM) . . . 3834
90.5.18 STCU2 Offline MBIST 1 Status Register (STCU_MBS1) . . . 3835
90.5.19 STCU2 Offline MBIST 2 Status Register (STCU_MBS2) . . . 3836
90.5.20 STCU2 Offline MBIST 3 Status Register (STCU_MBS3) . . . 3837
90.5.21 STCU2 Offline MBIST 4 Status Register (STCU_MBS4) . . . 3837
90.5.22 STCU2 Offline MBIST 1 End Flag Register (STCU_MBE1) . . . 3838

90.5.23 STCU2 Offline MBIST 2 End Flag Register (STCU_MBE2) 3839
90.5.24 STCU2 Offline MBIST 3 End Flag Register (STCU_MBE3) 3840
90.5.25 STCU2 Offline MBIST 4 End Flag Register (STCU_MBE4) 3841
90.5.26 STCU2 Online MBIST 1 Status Register (STCU_MBS1SW) 3842
90.5.27 STCU2 Online MBIST 2 Status Register (STCU_MBS2SW) 3843
90.5.28 STCU2 Online MBIST 3 Status Register (STCU_MBS3SW) 3844
90.5.29 STCU2 Online MBIST 4 Status Register (STCU_MBS4SW) 3845
90.5.30 STCU2 Online MBIST 1 End Flag Register (STCU_MBE1SW) 3846
90.5.31 STCU2 Online MBIST 2 End Flag Register (STCU_MBE2SW) 3847
90.5.32 STCU2 Online MBIST 3 End Flag Register (STCU_MBE3SW) 3848
90.5.33 STCU2 Online MBIST 4 End Flag Register (STCU_MBE4SW) 3849
90.5.34 STCU2 MBIST 1 Unrecoverable FM Register (STCU_MBUFM1) . . . 3850
90.5.35 STCU2 MBIST 2 Unrecoverable FM Register (STCU_MBUFM2) . . . 3851
90.5.36 STCU2 MBIST 3 Unrecoverable FM Register (STCU_MBUFM3) . . . 3852
90.5.37 STCU2 MBIST 4 Unrecoverable FM Register (STCU_MBUFM4) . . . 3853
90.5.38 STCU2 LBIST Control n Register (STCU_LB_CTRLn) 3854
90.5.39 STCU2 LBIST PC Stop n Register (STCU_LB_PCSn) 3856
90.5.40 STCU2 LBIST PRPG Low n Register (STCU_LB_PRPGLn) 3857
90.5.41 STCU2 LBIST PRPG High n Register (STCU_LB_PRPGHn) 3858
90.5.42 STCU2 Offline LBIST MISR Expected Low n Register (STCU_LB_MISRELn) 3858
90.5.43 STCU2 Offline LBIST MISR Expected High n Register (STCU_LB_MISREHn) 3859
90.5.44 STCU2 Offline LBIST MISR Read Low n Register (STCU_LB_MISRRLn) 3860
90.5.45 STCU2 Offline LBIST MISR Read High n Register (STCU_LB_MISRRHn) 3860
90.5.46 STCU2 Online LBIST MISR Expected Low n Register (STCU_LB_MISRELSWn) 3861
90.5.47 STCU2 Online LBIST MISR Expected High n Register (STCU_LB_MISREHSWn) 3862
90.5.48 STCU2 Online LBIST MISR Read Low n Register (STCU_LB_MISRRLSWn) 3863
90.5.49 STCU2 Online LBIST MISR Read High n Register (STCU_LB_MISRRHSWn) 3864
90.5.50 STCU2 MBIST Control k Register (STCU_MB_CTRLk) 3865
90.6 Use cases and limitations 3866
90.6.1 Offline self-test sequence 3867
90.6.2 Online self-test Sequence 3868

90.6.3 Bypass USER Mode . . . 3870
90.6.4 Design implementation information . . . 3870

91 Register Protection (REG_PROT) . . . 3871
91.1 Introduction . . . 3871
91.1.1 Overview . . . 3871
91.1.2 Features . . . 3871
91.1.3 Modes of operation . . . 3872
91.2 External signal description . . . 3872
91.3 Memory map and register definition . . . 3872
91.3.1 Memory map . . . 3873
91.3.2 Register descriptions . . . 3874
91.4 Functional description . . . 3876
91.4.1 General . . . 3877
91.4.2 Change lock settings . . . 3877
91.4.3 Access errors . . . 3880
91.5 Initialization/application information . . . 3881
91.5.1 Reset . . . 3881

92 Password and Device Security Module (PASS) . . . 3882
92.1 Introduction . . . 3882
92.1.1 Overview . . . 3882
92.1.2 Features . . . 3882
92.1.3 Modes of operation . . . 3882
92.2 External signal description . . . 3883
92.3 Memory map and register definition . . . 3883
92.3.1 Register descriptions . . . 3885
92.4 DCF clients . . . 3895
92.4.1 Censorship . . . 3895
92.4.2 Production Disable . . . 3896
92.4.3 Fuse Bypass DCF Client . . . 3897
92.4.4 Password Slot Access . . . 3898
92.5 Functional description . . . 3899
92.5.1 Censorship . . . 3899
92.5.2 Debug Interface Access . . . 3899
92.5.3 Flash Memory Read Protection . . . 3900

92.5.4 UTEST protection . . . 3901
92.5.5 Production Disable . . . 3901
92.5.6 Fuse Bypass . . . 3903
92.6 Initialization/application information . . . 3904
92.6.1 Reset . . . 3904
92.6.2 Setting lock bits in a password group . . . 3905
92.6.3 Censoring and uncensoring the device . . . 3906

93 Tamper Detection Module (TDM) . . . 3907
93.1 Overview . . . 3907
93.2 Features . . . 3908
93.3 External signal description . . . 3908
93.4 DCF client . . . 3908
93.4.1 Diary Base Address (DBA) DCF client . . . 3909
93.4.2 Tamper Region Override (TO) DCF client . . . 3909
93.4.3 One Time Programmable Enable (OTPEN0) DCF client . . . 3910
93.4.4 One Time Programmable Enable (OTPEN1) DCF client . . . 3911
93.4.5 One Time Programmable Enable (OTPEN2) DCF client . . . 3912
93.4.6 One Time Programmable Enable (OTPEN3) DCF client . . . 3912
93.4.7 Tamper Region Assignment DCF client (TDRn_LOCK0) . . . 3913
93.4.8 Tamper Region Assignment DCF client (TDRn _LOCK1) . . . 3914
93.4.9 Tamper Region Assignment DCF client (TDRn_LOCK2) . . . 3915
93.4.10 Tamper Region Assignment DCF client (TDRn_LOCK3) . . . 3915
93.5 Memory map and registers . . . 3916
93.5.1 TDR Status Register (TDM_TDRSR) . . . 3916
93.5.2 Last Flash Programed Address Register (TDM_LFPAR) . . . 3918
93.5.3 Diary Base Address (TDM_DBA) . . . 3919
93.5.4 Software Tamper Override Key Region (TDM_STO_KEYn) . . . 3919
93.6 Functional description . . . 3920
93.6.1 Flash erase counter and tamper detection . . . 3920
93.6.2 Assigning blocks to Tamper Detect Regions (TDR) . . . 3921
93.6.3 Diary . . . 3922
93.6.4 Specifying the diary base address . . . 3922
93.6.5 Diary base address verification . . . 3923
93.6.6 One Time Programmable (OTP) . . . 3924

94 ASC Modeling Unit version 2 (AMU2) 3925

94.1 Introduction 3925

94.1.1 Overview 3926

94.1.2 Features 3928

94.1.3 Modes of operation 3928

94.2 Memory map and register definition 3928

94.2.1 AMU memory map 3929

94.2.2 ARAM memory map details 3932

94.2.3 Register descriptions 3933

94.3 Functional description 3986

94.3.1 AMU2 algorithm 3986

94.3.2 The exp(x) function and implementation 3989

94.3.3 AMU data types and formats 3990

94.3.4 Register format and data types 4000

94.3.5 ALU calculation engine 4001

94.3.6 AMU Direct Memory Access (ADMA) 4003

94.4 Use case examples 4008

94.4.1 Single calculation using a CPU as the master device 4008

94.4.2 Single calculation using SDMA for data load, CPU for post-processing 4009

94.4.3 Single calculation using SDMA + ADMA for data load, CPU for post-processing 4010

94.4.4 Single calculation using SDMA for partial data load, streaming ADMA for V[*], CPU for post-processing 4011

94.4.5 Multiple calculations using ADMA for partial data load, SDMA for result retrieval 4011

94.5 Operation details 4012

94.5.1 Details on operation completion status indicators 4012

94.5.2 Details on operation stop/resume 4013

94.5.3 AMU2 Performance 4014

95 Real Time Clock / Autonomous Periodic Interrupt (RTC/API) 4016

95.1 Overview 4016

95.2 Features 4016

95.3 Modes of operation 4018

95.3.1 Functional mode 4018

95.3.2 Debug mode 4018

95.4 Register descriptions 4018
95.4.1 RTC Supervisor Control Register (RTCSUPV) 4019
95.4.2 RTC Control Register (RTCC) 4019
95.4.3 RTC Status Register (RTCS) 4021
95.4.4 RTC Counter Register (RTCCNT) 4023
95.4.5 API Compare Value Register (RTC_APIVAL) 4023
95.4.6 RTC Compare Value Register (RTC_RTCVAL) 4024
95.5 RTC functional description 4024
95.6 API functional description 4025

96 Wakeup unit (WKPU) 4026
96.1 Introduction 4026
96.1.1 Overview 4026
96.1.2 Features 4027
96.2 External signal description 4027
96.3 Memory map and register description 4027
96.3.1 Memory map 4027
96.3.2 Register descriptions 4028
96.4 Functional description 4054
96.4.1 General 4054
96.4.2 Non-maskable interrupts 4054
96.4.3 External wakeups/interrupts 4055
96.4.4 On-Chip wakeups 4057

97 Emulation and Debug Device Introduction 4058
97.1 Functionality 4058
97.2 Feature summary 4059
97.3 Block diagram 4059
97.4 Clocking 4061
97.5 Reset logic 4063
97.6 BD memory map 4064

98 Buddy Device—Data Write Processing Unit (DWPU) 4066
98.1 Introduction 4066
98.2 Features 4067
98.3 Functional description 4067

98.3.1 Nexus Out Interface (NOI) 4067
98.3.2 Message Concatenation and Allocation (MCA) 4068
98.3.3 Client Specific Processing (CSP) 4068
98.3.4 Address Tag Check (ATC) 4071
98.3.5 Out of Range Check (ORC) 4073
98.3.6 Message Formatter (MF) 4074
98.3.7 Nexus Client Interface (NCI) 4075
98.4 Non-memory mapped registers 4075
98.4.1 Register descriptions 4077
98.5 Memory mapped registers 4088
98.5.1 Tag Ram Control (DPWU_TRC_CTRL) 4088
98.5.2 Tag RAM Configuration Address (DWPU_TRC_ADR) 4090
98.5.3 Tag RAM Configuration Data (DWPU_TRC_DATA) 4090

99 Buddy Device—Nexus Read Write Access (RWA) 4092
99.1 Introduction 4092
99.2 Register definition 4092
99.2.1 Register descriptions 4092
99.2.2 Read/write access data (RWD) 4095
99.2.3 Read/write access address (RWA) 4096
99.3 Nexus register access via JTAG 4096
99.4 Memory access examples 4097
99.4.1 Single write access 4097
99.4.2 Block write access 4098
99.4.3 Single read access 4098
99.4.4 Block read access 4098

100 Buddy Device—Crossbar and RAM (BD_XBS_RAM) 4100
100.1 3 x 5 Crossbar switch and attached controllers 4100
100.2 BD_XBS_RAM End-to-End ECC32 Checkbit/ Syndrome Coding Scheme 4101

101 Buddy Device—System Integration Unit Lite (BD_SIUL2) 4107
101.1 Introduction 4107
101.2 Device identification 4107
101.3 Device status information 4108

101.3.1 Initialization status 4108
101.3.2 PD reset status 4108
101.4 Memory map 4108
101.4.1 Register descriptions 4109

102 Buddy Device: Debug and Calibration Interface (DCI) 4115
102.1 Introduction 4115
102.1.1 Features 4115
102.1.2 Overview 4115
102.1.3 Operating modes 4118
102.2 External signal description 4127
102.3 Register description 4127
102.3.1 DCI control register (BD_DCI_CR) 4128
102.3.2 DCI EVTx pin multiplexing control register (DCI_PINCR) 4130
102.4 Functional description 4130
102.4.1 DCI mode transition 4130
102.4.2 LFAST LVDS pad fault 4131

103 Buddy Device—JTAG Controller (JTAGC) 4133
103.1 Introduction 4133
103.1.1 Block Diagram 4133
103.1.2 Features 4134
103.1.3 Modes of operation 4134
103.2 External signal description 4135
103.2.1 TCK—Test clock input 4135
103.2.2 TDI—Test data input 4135
103.2.3 TDO—Test data output 4135
103.2.4 TMS—Test mode select 4135
103.2.5 JCOMP—JTAG compliancy 4135
103.3 Register description 4136
103.3.1 Instruction register 4136
103.3.2 Bypass register 4136
103.3.3 Device identification register 4136
103.3.4 JTAG_PASSWORD register 4137
103.3.5 Boundary scan register 4138
103.4 Functional description 4138

103.4.1 JTAGC reset configuration 4138
103.4.2 IEEE 1149.1-2001 (JTAG) Test Access Port 4139
103.4.3 TAP controller state machine 4139
103.4.4 JTAGC block instructions 4141
103.4.5 Boundary scan 4144
103.5 Initialization/Application information 4145

104 Buddy Device—JTAG Master (JTAGM) 4146
104.1 Introduction 4146
104.1.1 Overview 4146
104.1.2 Feature description 4147
104.2 Functional description 4147
104.2.1 Module architecture 4147
104.2.2 JTAGM JCOMP usage in LFAST 4148
104.2.3 JTAGM data error detection 4149
104.2.4 JTAGM message tagging 4149
104.2.5 JTAGM Ready signal 4149
104.2.6 EVTO and EVTI signals 4150
104.2.7 JTAGM configuration and status monitoring 4150
104.2.8 JTAGM to DCI serial interface 4151
104.2.9 JTAGM data handling and CRC calculation in LFAST mode 4151
104.2.10 Software interface 4152
104.3 Modes of operation 4153
104.4 Memory map and register definition 4153
104.4.1 Register descriptions 4153

105 Buddy Device—Nexus Aurora Router (NAR) 4162
105.1 Introduction 4162
105.2 Overview 4163
105.3 Features 4163
105.4 Register definition 4163
105.4.1 Register descriptions 4164
105.4.2 Message filtering registers 4168
105.4.3 Trace memory bus configuration registers 4171
105.4.4 Suppress mode triggers register (NAR_STCR) 4173
105.4.5 Receive queue client disable register (NAR_CDR) 4175

105.4.6 Nexus Aurora Phy control register (NAR_NAPCR) 4177
105.4.7 Trace Memory fill level and partition configuration register (NAR_AHFPAR) 4177
105.5 Functional description 4179
105.5.1 Reset/Startup 4179
105.5.2 Operation modes 4179
105.5.3 Data reception and arbitration 4181
105.5.4 Suppress mode 4183
105.5.5 Stall detection 4184
105.5.6 Output arbitration 4184
105.5.7 Queue partitioning 4185
105.5.8 Message translation and formatting 4186
105.5.9 Aurora interface 4187
105.5.10 Trace memory bus output port usage 4187
105.5.11 Internal message generation 4188
105.5.12 Timestamp function 4192
105.5.13 Global (Development tool client) stall control 4193
105.5.14 Overlay and internal trace memory full status generation 4193
105.5.15 Configuration/debug interface JTAG (1149.1) 4194
105.5.16 Trace memory bus controller 4197
105.5.17 Aurora client controller 4199
105.6 Initialization information 4199
105.7 Application information 4200

106 Buddy Device—Nexus Aurora Link (NAL) 4201
106.1 Introduction 4201
106.2 Transmit operation 4202
106.3 Nexus Aurora formatting 4202
106.4 Static training 4203
106.5 Programmable registers 4203
106.5.1 Nexus Aurora Link General Status Register (NAL_GSR) 4204
106.5.2 NAL General Control Register (NAL_GCR) 4204
106.5.3 NAL Training Control Register (NAL_TCR) 4206

107 Buddy Device—Nexus Aurora PHY (NAP) 4209
107.1 Introduction 4209

107.1.1 Features 4210
107.1.2 Modes of operation 4210
107.2 External signal description 4210
107.3 Memory map and register definition 4211
107.4 Functional description 4211
107.4.1 Data symbol character accumulator 4211
107.4.2 Serializer 4211
107.4.3 Clock generation and control 4212
107.5 Initialization information 4212

Revision history 4217

1 Preface

1.1 Document organization

This document includes chapters that are divided into parts:

- Part I includes chapters that describe the device as a whole or provide device-specific information.
- Part II includes chapters that describe the functionality of the individual modules on the device.
- Part III includes chapters that describe the functionality of the Emulation and Debug Device (ED).

1.2 Register conventions

Figure 1 and *Table 1* provide the register conventions used throughout this manual.

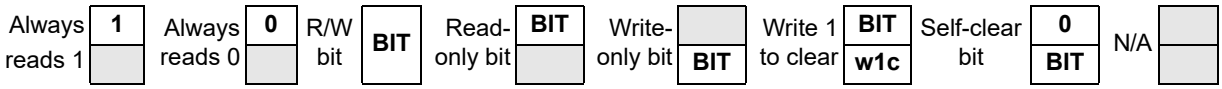


Figure 1. Key to register fields

Table 1. Register conventions

Convention	Description
	Depending on its placement in the read or write row, indicates that the bit is not readable or not writable.
FIELDNAME	Identifies the field. Its presence in the read or write row indicates that it can be read or written.
Register field types	
R	Read only. Writing this bit has no effect.
W	Write only
R/W	Standard read/write bit. Only software can change the bit's value (other than a hardware reset).
rwm	A read/write bit that may be modified by a hardware in some fashion other than by a reset.
w1c	Write one to clear. A status bit that can be read, and is cleared by writing a one.
slfclr	Self-clearing bit. Writing a one has some effect on the module, but it always reads as zero.
Reset values	
0	Resets to zero
1	Resets to one
—	Undefined at reset

1.3 Acronyms and abbreviations

Appendix A: Acronyms and abbreviations lists acronyms and abbreviations used in this document.

1.4 Reference documents

In addition to this reference manual, the following documents provide additional information on the operation of the SPC58xEx/SPC58xGx:

- IEEE-ISTO 5001™-2012 Standard for a Global Embedded Processor Interface (Nexus)
- IEEE 1149.1-2001 standard - IEEE Standard Test Access Port and Boundary-Scan Architecture

Information about Power ISA™ Book is available on the Power Architecture web site

2 Introduction

2.1 SPC58xEx microcontroller

The SPC58xEx microcontroller belongs to a family of devices continuing the SPC57x PWT family. SPC58xEx builds on the legacy of the SPC57x PWT family, while introducing new features coupled with higher throughput to provide substantial reduction of cost per feature and significant power and performance improvement (MIPS per mW). On the SPC58xEx microcontroller there are up to 3 e200z4d processor cores, 2 checkers and one e200z0 core embedded in the Hardware Security Module.

Table 2. Processor cores

Core	Type
Computing Shell 0	
Main Core_0	e200z420
Main Core_1	e200z420
Checker Core_0s	e200z419
Computing Shell 1	
Peripheral Core_2	e200z425
Checker Core_2s	e200z425
Hardware Security Module	
e200z0 Core	

2.1.1 Target applications

This device is targeted at automotive powertrain, chassis applications:

- Powertrain controllers for six to eight cylinder gasoline, diesel engines and advanced combustion systems as well as high end hybrid and transmission control
- Integrated chassis control, high-end steering and braking

These applications are considered to be functionally safe and are designed to achieve ISO26262 ASIL-D compliance.

For this reason, safety measures have been added which are further described in *Chapter 6: Functional safety*.

2.1.2 Overall architecture

The architecture is split into two computing shells as shown in *Table 2*. The computing shell 0 consists of one pair of CPU in lock-step for safety, one CPU without lock-step, a DMA move engine, fast local memories inside the CPUs and fast system memory. All components of the computational shell are connected through a crossbar switch (XBAR_0).

The computing shell 1 consists of two CPUs in lock-step, a DMA move engine, fast local memories inside the CPU, fast system memories, a Hardware Security Module (HSM) and all the other peripherals connected through a crossbar switch (XBAR_1).

2.1.3 Core features

The computing shell 0 consists of dual e200z4 CPUs operating up to 180 MHz, one core paired with a safety checker core in delayed lock-step mode.

The computing shell 1 includes a CPU targeted at managing the peripherals. The shell core is also an e200z4 CPU running at up to 180 MHz, paired with a safety checker core in delayed lock-step, and including enhancements to implement Digital Signal Processing (DSP) algorithms.

The sixth programmable CPU is a e200z0 running up to 90 MHz and is embedded in the HSM.

All CPUs are compatible with the Power Architecture VLE instruction set, that supports code size reduction. The Power Architecture has enhancements that improve the architecture's fit in embedded applications.

Note: The e200z0 core is embedded in the Hardware Security Module.

2.1.4 Core_2

The Core_2 is one of six processor cores on the SPC58xEx device as shown in *Table 2*. In general, the intended use of the Core_2 is to initially configure the machine and then control various peripheral elements of the device. However, the Core_2 is not limited to any particular duty.

The data and the instruction bus structures of the Core_2, which consist of a 64-bit data bus and a 32-bit address bus, are connected to a Crossbar Switch XBAR_1 which operates at 180 MHz.

The Core_2 is automatically started during the boot-up sequence after the release of RESET. This process is fully described in *Chapter 8: Reset and Boot*. During the boot sequence the System Status and Configuration Module (SSCM) reads and interprets a set of device configuration records located in the TEST Flash Area. One of these records is the start address of the code where the Core_2 will begin program execution. This vector will always point to a location in the Boot Assist Flash. Once the System Status and Configuration Module finds and interprets this particular device configuration record, the SSCM will write the start address for the Core_2 to the appropriate location in the Mode Entry module (ME). When the internal reset signal is released, the Mode Entry module transfers the Core_2 start address to the Core_2 and code execution begins at that address.

The code for the Boot Assist Flash, the TEST Device Configuration Records and some of the UTEST Device Configuration Records are written by the device manufacturer and are programmed into the device during production testing. All SPC58xEx devices are shipped with start-up code programmed into the Boot Assist Flash. Also, the SPC58xEx is configured such that code execution by the Core_2 must begin with a start address located in the Boot Assist Flash.

If there is no user application code to execute, the Core_2, executing pre-programmed code in the Boot Assist Flash, will attempt to download user applications code through the LINFlexD port using a specific data transfer protocol. This downloaded code is installed into program memory and code execution will begin at the start address embedded in the downloaded code.

The Core_2 is placed in the SPC58xEx architecture in such a way that enables it to access many of the peripheral elements without affecting the processing speed of Core_0 and

Core_1. On the SPC58xEx device there are several A/D channels taking data which must be analyzed for various spectral components.

This specificity associated with its integrated DSP processor which has special hardware to execute multiply and accumulate functions makes of this core the most avantageous one to manage a system of analog-to-digital converters and fetch the digital data that is generated. The integrated DSP unit uses a signal processing algorithm to digest the incoming data. The results of this signal processing algorithm are used in other engine control algorithms. Using the Core_2 to manage the A/D converters and do the initial signal processing of the data allows the two other cores to focus on other compute intensive applications instead of managing relatively slow peripheral elements.

Once the Core_2 has performed its initialization duties, as determined by pre-loaded instruction code in the Boot Assist Flash and downloaded any needed applications code, the Core_2 can essentially be programmed to perform any duty or access any memory element. When executing instructions from its internal instruction cache and fetching data from peripherals connected to the Cross Bar Switch XBAR_1, the Core_2 operates autonomously from Core_0 and Core_1.

2.1.5 Memory hierarchy

The SPC58xEx computational shell has three levels of memory hierarchy. At the top level is a combination of Instruction Cache, Data Cache, Local Instruction RAM and Local Data RAM, depending on the specific CPU. The local RAM arrays are embedded into the CPU's pipeline to provide the fastest possible access time. The second level of memory is system RAM and flash. These are connected via a 180 MHz, 64-bit wide crossbar. The third level of memory system is the I/O subsystem. All the I/O peripherals are connected together via a 180 MHz 64-bit wide crossbar switch, while peripherals are connected at either 180 MHz or 45 MHz depending on their need. The reduced speed for the peripherals helps to reduce power consumption.

2.1.6 Features

On-chip modules within SPC58xEx include the following features:

- Three main CPUs, dual issue, 32-bit CPU core complexes (e200z4), two of them having a checker core in lock-step
  - Power Architecture embedded specification compliance
  - Instruction set enhancement allowing variable length encoding (VLE), encoding a mix of 16-bit and 32-bit instructions, for code size footprint reduction
  - Single-precision floating point operations
  - Lightweight signal processing auxiliary processing unit (LSP APU) instruction support for digital signal processing (DSP) on Core_2
  - 16 KB Local instruction RAM and 64 KB local data RAM for Core_0 and Core_1, 16 KB Local instruction RAM and 32 KB local data RAM for Core_2
  - 8 KB I-Cache and 4 KB D-Cache for Core_0 and Core_1, 8kB I-Cache for Core_2
- 6582 KB on-chip Flash
  - Supports read during program and erase operations, and multiple blocks allowing EEPROM emulation
  - Supports read while read between the two code Flash partitions.
- 608 KB on-chip general-purpose SRAM (+ 160 KB data RAM included in the CPUs)
- Multi channel direct memory access controllers (eDMA paired in lock-step)
  - One eDMA with 64 channels
  - One eDMA with 32 channels
- One interrupt controller (INTC) in lock-step
- Dual phase-locked loops with stable clock domain for peripherals and FM modulation domain for computational shell
- Dual crossbar switch architecture for concurrent access to peripherals, Flash, or RAM from multiple bus masters with end-to-end ECC
- Hardware security module (HSM) to provide robust integrity checking of Flash memory
- System integration unit lite (SIUL)
- Boot assist Flash (BAF) supports factory programming using a serial bootload through the asynchronous CAN or LIN/UART.
- GTM343 - generic timer module
  - Intelligent complex timer module
  - 144 channels (40 input and 104 output)
  - 5 programmable fine grain multi-threaded cores
  - 61 KB of dedicated RAM
  - 24-bit wide channels
  - Hardware support for engine control, motor control and safety related applications
- Enhanced analog-to-digital converter system with
  - One supervisor 12-bit SAR analog converter
  - Four separate fast 12-bit SAR analog converters
  - Three separate 10-bit SAR analog converters
  - Six separate 16-bit Sigma-Delta analog converters
- Ten deserial serial peripheral interface (DSPI) modules

- Eighteen LIN and UART communication interface (LINFlexD) modules
  - LINFlexD_0 is a Master/Slave
  - All others are Masters
- Seven modular controller area network (MCAN) modules, and one time-triggered controller area network (M-TTCAN), all supporting flexible data rate (CAN-FD)
- Dual-channel FlexRay controller
- Two ethernet controllers 10/100 Mbps, compliant IEEE 802.3-2008
  - IEEE 1588-2008 Time stamping (internal 64-bit time stamp)
  - IEEE 802.1AS and IEEE 802.1Qav (AVB-Feature)
  - IEEE 802.1Q VLAN tag detection
  - IPv4 and IPv6 checksum modules
- Flexible Power Supply options:
  - External Regulators (1.2V core, 3.3V–5V IO)
  - Single internal SMPS regulator (eLQFP176 and Bare Die)
  - Single internal Linear Regulator with external ballast (FPBGA292 and Bare Die)
- Nexus development interface (NDI) per IEEE-ISTO 5001-2003 standard, with some support for 2010 standard.
- Device and board test support per Joint Test Action Group (JTAG) (IEEE 1149.1)
- Standby power domain with smart wake-up sequence (FPBGA292 and Bare Die)

2.1.7 Feature list

Table 3 lists a summary of major features for the SPC58xEx device. The feature column represents a combination of module names and capabilities of certain modules. A detailed description of the functionality provided by each on-chip module is given later in this document.

Table 3. Features list

Feature	Description
SPC58 family	40 nm
Computing Shell 0	
Number of Cores	up to 2
Number of checker cores	up to 1
Local RAM	16 KB Instruction
	64 KB Data
Single Precision Floating Point	Yes
SIMD (LSP)	No
VLE	Yes
Cache	8 KB Instruction
	4 KB Data
Computing Shell 1	

Table 3. Features list (continued)

Feature	Description
Number of Cores	up to 1
Number of checker cores	1
Local RAM	16 KB Instruction
	32 KB Data
Single Precision Floating Point	Yes
SIMD (LSP)	Yes
VLE	Yes
Cache	8 KB Instruction
Other	
Security (HSM Module)	up to 1
MPU	Core MPU: 24 per CPU
	System MPU: 24 per XBAR
Semaphores	Yes
CRC Channels	2 x 4
Software Watchdog Timer (SWT)	4
Core Nexus Class	3+
Event Processor	4 x SCU
	4 x PMC
Run control Module	Yes
System SRAM	up to 608 KB (including 256KB of standby RAM)(1)
User Flash	up to 6144 KB code / 256 KB data
Security Flash	up to 144kB code / 32kB data
Flash fetch accelerator	2 x 2 x 4 x 256-bit
Flash Overlay RAM	2 x 16 KB
Calibration Interface	64-bit IPS Slave
DMA channels	96
DMA Nexus Class	3
LINFlexD	10
M_CAN supporting CAN-FD according to ISO 11898-1 2015 (instances supporting also TTCAN)	8
DSPI	18
Microsecond channel downlink	2
SENT bus	15
I2C	1

Table 3. Features list (continued)

Feature	Description
PSI5 / PSI5-S bus	2 / 1
FlexRay	1 x Dual channel
Ethernet	2 MAC with Time stamping, AVB and VLAN support
SIPI / LFAST Interprocessor bus	High Speed
System Timers	8 PIT channels
	4 AUTOSAR® (STM)
	RTC/API
GTM Timer	40 Input Channels, 104 Output Channels
GTM RAM	61 KB
Interrupt controller	> 710 sources
ADC (SAR)	8
ADC (SD)	6
Temp. sensor	Yes
Self Test Controller	Yes
PLL	Dual PLL with FM
External Power Supplies	3.3 V - 5 V, 1.2 V
Integrated linear voltage regulator	Yes(1)
integrated switch mode voltage regulator (SMPS)	Yes(2)
Low Power Modes	Stop Mode
	Halt Mode
	Smart Standby with output controller, analog and digital inputs(1)
	Standby Mode(1)

1. Except eLQFP176.
2. Except LFBGA292.

2.1.8 Packages

The SPC58xEx is available in these production packages: eLQFP176, FPBGA292 and Bare Die.

All packages support production and emulation devices with compatibility for an emulation package in a production device footprint.

2.1.9 Software debug and calibration

The Production Device (PD) includes many features to facilitate software development and calibration of constant data. These features include:

- Nexus code execution trace on all CPUs
- Nexus data trace on all bus masters (DMA, Interprocessor bus, CPUs, etc.)
- Performance monitors for each CPU
- Sequence Processing Unit for complex trace triggering
- 32 KB of overlay RAM

For more complex software development and calibration of constant data, a Buddy Device (BD) is packaged together with the SPC58xEx production device as a multi-die emulation and is referred to as Emulation Device (ED).

The ED provides exactly the same features as the Production Device (PD). The ED is available in the same package as the PD but has an additional overlay RAM (up to 2M bytes) and a high speed Nexus trace output port consisting of four lanes of Aurora (LVDS) operating up to 1.25 Gbit/s.

2.1.10 Block diagram

The figures below show the top-level block diagrams.

Figure 2. Block diagram

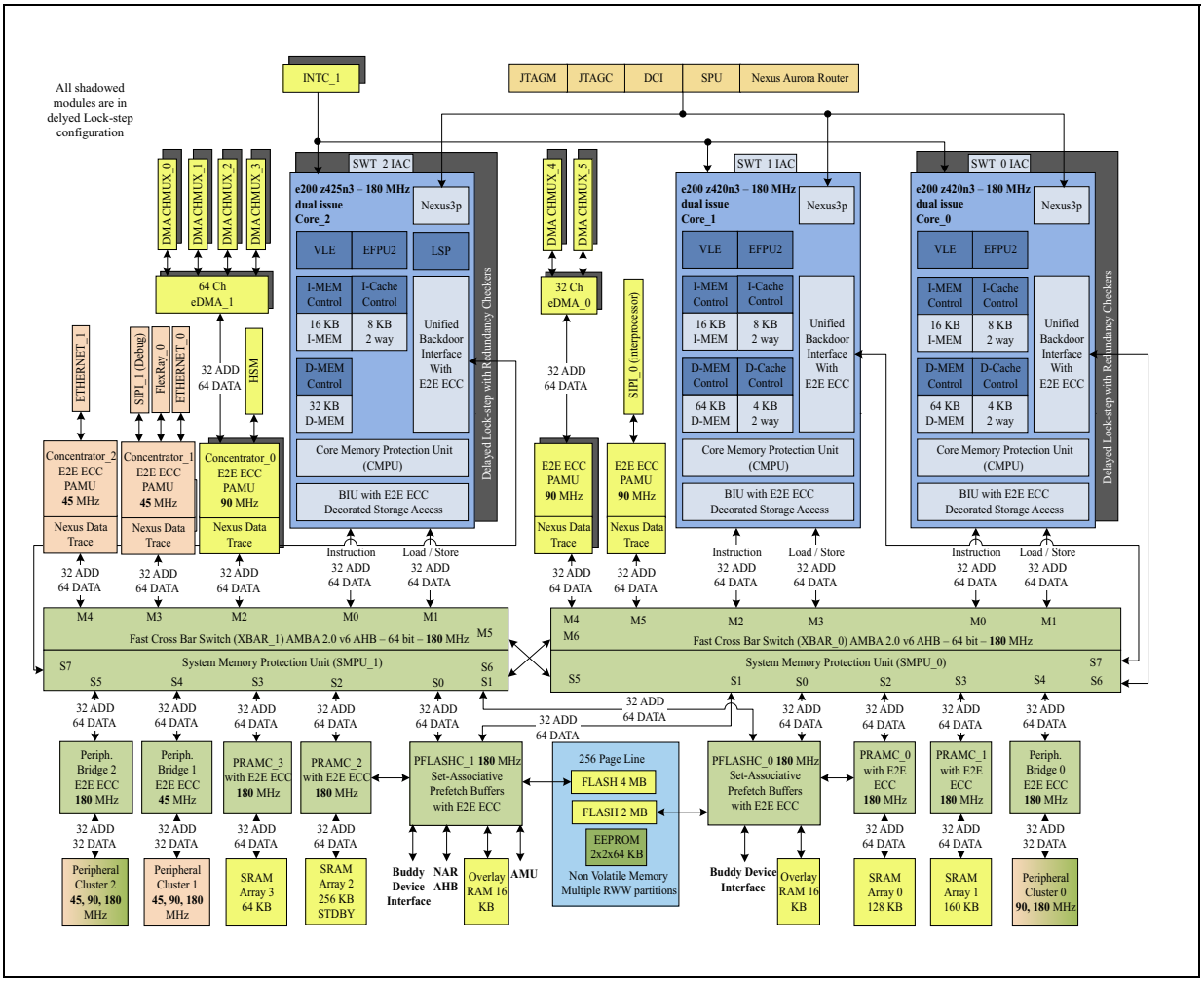
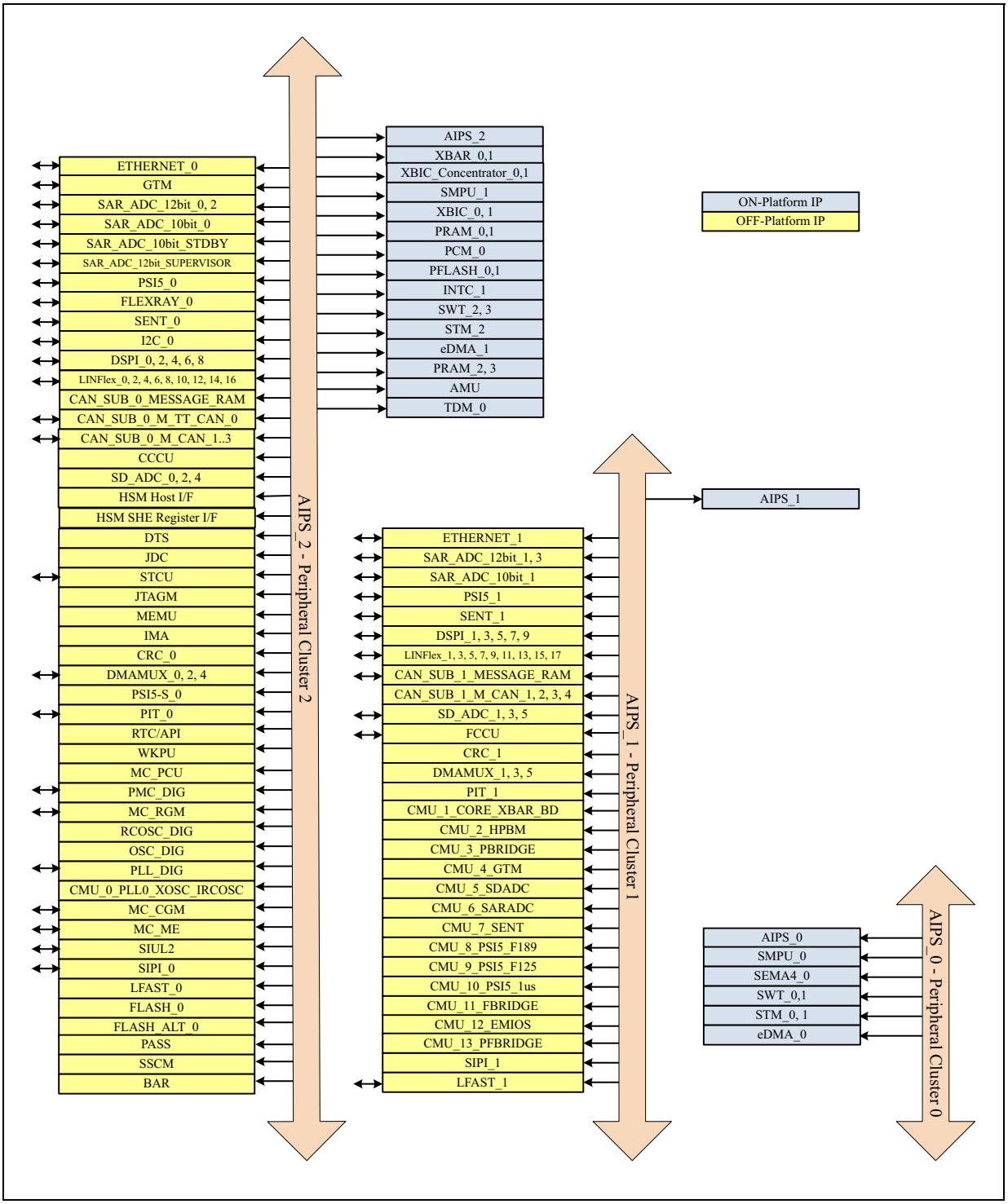


Figure 3. Periphery allocation



2.2 SPC58xGx microcontroller

The SPC58xGx microcontroller belongs to a family of devices continuing the SPC56x Body family. SPC58xGx builds on the legacy of the SPC56x Body family, while introducing new features coupled with higher throughput to provide substantial reduction of cost per feature and significant power and performance improvement (MIPS per mW). On the SPC58xGx microcontroller there are up to 3 e200z4d processor cores, 1 checker and one e200z0 core embedded in the Hardware Security Module.

Table 4. Processor cores

Core	Type
Computing Shell 0	
Main Core_0	e200z420
Main Core_1	e200z420
Checker Core_0s	e200z419
Computing Shell 1	
Peripheral Core_2	e200z425
Hardware Security Module	
e200z0 Core	

2.2.1 Target applications

This device is targeted at automotive high-end central body controller, smart junction boxes, stand-alone and connected gateway.

These applications are considered to be functionally safe and are designed to achieve ISO26262 ASIL-B up to ASIL-D compliance.

For this reason, safety measures have been added which are further described in *Chapter 6: Functional safety*.

2.2.2 Overall architecture

The architecture is split into two computing shells as shown in *Table 4*. The computing shell 0 consists of one pair of CPU in lock-step for safety, one CPU without lock-step, a DMA move engine, fast local memories inside the CPUs and fast system memory. All components of the computational shell are connected through a crossbar switch (XBAR_0).

The computing shell 1 consists of one CPU, a DMA move engine, fast local memories inside the CPU, fast system memories, a Hardware Security Module (HSM) and all the other peripherals connected through a crossbar switch (XBAR_1).

2.2.3 Core features

The computing shell 0 consists of dual e200z4 CPUs operating up to 180 MHz, one core paired with a safety checker core in delayed lock-step mode.

The computing shell 1 includes a CPU targeted at managing the peripherals. The shell core is also an e200z4 CPU running at up to 180 MHz, and including enhancements to implement Digital Signal Processing (DSP) algorithms.

The fifth programmable CPU is a e200z0 running up to 90 MHz and is embedded in the HSM.

All CPUs are compatible with the Power Architecture® VLE instruction set, that supports code size reduction. The Power Architecture has enhancements that improve the architecture's fit in embedded applications.

Note: The e200z0 core is embedded in the Hardware Security Module.

2.2.4 Core_2

The Core_2 is one of five processor cores on the SPC58xGx device as shown in *Table 4*. In general, the intended use of the Core_2 is to initially configure the machine and then control various peripheral elements of the device. However, the Core_2 is not limited to any particular duty.

The data and the instruction bus structures of the Core_2, which consist of a 64-bit data bus and a 32-bit address bus, are connected to a Crossbar Switch XBAR_1 which operates at 180 MHz.

The Core_2 is automatically started during the boot-up sequence after the release of RESET. This process is fully described in *Chapter 8: Reset and Boot*. During the boot sequence the System Status and Configuration Module (SSCM) reads and interprets a set of device configuration records located in the TEST Flash Area. One of these records is the start address of the code where the Core_2 will begin program execution. This vector will always point to a location in the Boot Assist Flash. Once the System Status and Configuration Module finds and interprets this particular device configuration record, the SSCM will write the start address for the Core_2 to the appropriate location in the Mode Entry module (ME). When the internal reset signal is released, the Mode Entry module transfers the Core_2 start address to the Core_2 and code execution begins at that address.

The code for the Boot Assist Flash, the TEST Device Configuration Records and some of the UTEST Device Configuration Records are written by the device manufacturer and are programmed into the device during production testing. All SPC58xGx devices are shipped with start-up code programmed into the Boot Assist Flash. Also, the SPC58xGx is configured such that code execution by the Core_2 must begin with a start address located in the Boot Assist Flash.

If there is no user application code to execute, the Core_2, executing pre-programmed code in the Boot Assist Flash, will attempt to download user applications code through the LINFlexD port using a specific data transfer protocol. This downloaded code is installed into program memory and code execution will begin at the start address embedded in the downloaded code.

The Core_2 is placed in the SPC58xGx architecture in such a way that enables it to access many of the peripheral elements without affecting the processing speed of Core_0 and Core_1. On the SPC58xGx device there are several A/D channels taking data which must be analyzed for various spectral components.

This specificity associated with its integrated DSP processor which has special hardware to execute multiply and accumulate functions makes of this core the most avantageous one to manage a system of analog-to-digital converters and fetch the digital data that is generated.

The integrated DSP unit uses a signal processing algorithm to digest the incoming data. The results of this signal processing algorithm are used in other engine control algorithms. Using the Core_2 to manage the A/D converters and do the initial signal processing of the data allows the two other cores to focus on other compute intensive applications instead of managing relatively slow peripheral elements.

Once the Core_2 has performed its initialization duties, as determined by pre-loaded instruction code in the Boot Assist Flash and downloaded any needed applications code, the Core_2 can essentially be programmed to perform any duty or access any memory element. When executing instructions from its internal instruction cache and fetching data from peripherals connected to the Cross Bar Switch XBAR_1, the Core_2 operates autonomously from Core_0 and Core_1.

2.2.5 Memory hierarchy

The SPC58xGx computational shell has three levels of memory hierarchy. At the top level is a combination of Instruction Cache, Data Cache, Local Instruction RAM and Local Data RAM, depending on the specific CPU. The local RAM arrays are embedded into the CPU's pipeline to provide the fastest possible access time. The second level of memory is system RAM and flash. These are connected via a 180 MHz, 64-bit wide crossbar. The third level of memory system is the I/O subsystem. All the I/O peripherals are connected together via a 180 MHz 64-bit wide crossbar switch, while peripherals are connected at either 180 MHz or 45 MHz depending on their need. The reduced speed for the peripherals helps to reduce power consumption.

2.2.6 Features

The features described in this introduction chapter refer to the superset device, this means all features implemented on the device. To find the possible derivatives and related feature set, please refer to the "Ordering information" chapter in the device datasheet.

On-chip modules within SPC58xGx include the following features:

- Three main CPUs, dual issue, 32-bit CPU core complexes (e200z4), one of them having a checker core in lock-step
  - Power Architecture embedded specification compliance
  - Instruction set enhancement allowing variable length encoding (VLE), encoding a mix of 16-bit and 32-bit instructions, for code size footprint reduction
  - Single-precision floating point operations
  - Lightweight signal processing auxiliary processing unit (LSP APU) instruction support for digital signal processing (DSP) on Core_2
  - 16 KB Local instruction RAM and 64 KB local data RAM for Core_0 and Core_1, 16 KB Local instruction RAM and 32 KB local data RAM for Core_2
  - 8 KB I-Cache and 4 KB D-Cache for Core_0 and Core_1, 8kB I-Cache for Core_2
- 6582 KB on-chip Flash
  - Supports read during program and erase operations, and multiple blocks allowing EEPROM emulation
  - Supports read while read between the two code Flash partitions.
- 608 KB on-chip general-purpose SRAM (+ 160 KB data RAM included in the CPUs)
- Multi channel direct memory access controllers (eDMA paired in lock-step)
  - One eDMA with 64 channels
  - One eDMA with 32 channels
- One interrupt controller (INTC) in lock-step
- Dual phase-locked loops with stable clock domain for peripherals and FM modulation domain for computational shell
- Dual crossbar switch architecture for concurrent access to peripherals, Flash, or RAM from multiple bus masters with end-to-end ECC
- Hardware security module (HSM) to provide robust integrity checking of Flash memory
- System integration unit lite (SIUL)
- Boot assist Flash (BAF) supports factory programming using a serial bootload through the asynchronous CAN or LIN/UART.
- Hardware support for engine control, motor control and safety related applications
- Enhanced modular IO subsystem (eMIOS): up to 64 (2 x 32) timed I/O channels with 16-bit counter resolution
  - Buffered updates
  - Support for shifted PWM outputs to minimize occurrence of concurrent edges
  - Supports configurable trigger outputs for ADC conversion for synchronization to channel output waveforms
  - Shared or independent time bases
  - DMA transfer support available

- Body cross triggering unit (BCTU)
  - Triggers ADC conversions from any eMIOS channel
  - Triggers ADC conversions from up to 2 dedicated PIT_RTIs
  - One event configuration register dedicated to each timer event allows to define the corresponding ADC channel
  - Synchronization with ADC to avoid collision
- Enhanced analog-to-digital converter system with
  - Five separate 12-bit SAR analog converters
  - Three separate 10-bit SAR analog converters
- Ten deserial serial peripheral interface (DSPI) modules
- Eighteen LIN and UART communication interface (LINFlexD) modules
  - LINFlexD_0 is a Master/Slave
  - All others are Masters
- Eight modular controller area network (MCAN) modules, all supporting flexible data rate (CAN-FD)
- Dual-channel FlexRay controller
- One Ethernet controller 10/100 Mbps, compliant IEEE 802.3-2008
  - IEEE 1588-2008 Time stamping (internal 64-bit time stamp)
  - IEEE 802.1AS and IEEE 802.1Qav (AVB-Feature)
  - IEEE 802.1Q VLAN tag detection
  - IPv4 and IPv6 checksum modules
- Nexus development interface (NDI) per IEEE-ISTO 5001-2003 standard, with some support for 2010 standard.
- Device and board test support per Joint Test Action Group (JTAG) (IEEE 1149.1)
- Standby power domain with smart wake-up sequence

2.2.7 Feature list

Table 5 lists a summary of major features for the SPC58xGx device. The feature column represents a combination of module names and capabilities of certain modules. A detailed description of the functionality provided by each on-chip module is given later in this document.

Table 5. Features list

Feature	Description
SPC58 family	40 nm
Computing Shell 0	
Number of Cores	up to 2
Number of checker cores	up to 1
Local RAM	16 KB Instructionω
	64 KB Data
Single Precision Floating Point	Yes
SIMD (LSP)	No

Table 5. Features list (continued)

Feature	Description
VLE	Yes
Cache	8 KB Instruction
	4 KB Data
Computing Shell 1	
Number of Cores	1
Local RAM	16 KB Instruction
	32 KB Data
Single Precision Floating Point	Yes
SIMD (LSP)	No
VLE	Yes
Cache	8 KB Instruction
Other	
Security (HSM Module)	up to 1
MPU	Core MPU: 24 per CPU
	System MPU: 24 per XBAR
Semaphores	Yes
CRC Channels	2 x 4
Software Watchdog Timer (SWT)	up to 4
Core Nexus Class	3+
Event Processor	4 x SCU
	4 x PMC
Run control Module	Yes
System SRAM	up to 608 KB (including 256KB of standby RAM)
User Flash	up to 6144 KB code / 256 KB data
Security Flash	up to 144kB code / 32kB data
Flash fetch accelerator	2 x 2 x 4 x 256-bit
DMA channels	96
DMA Nexus Class	3
LINFlexD	18
M_CAN supporting CAN-FD according to ISO 11898-1 2015	8
DSPI	18
I2C	1
FlexRay	1 x Dual channel
Ethernet	up to 2 MAC with Time stamping, AVB and VLAN support

Table 5. Features list (continued)

Feature	Description
SIPI / LFAST Interprocessor bus	High Speed
System Timers	8 PIT channels
	4 AUTOSAR® (STM)
	RTC/API
eMIOS	2 x 32 channels
BCTU	64 channels
Interrupt controller	> 710 sources
ADC (SAR)	6
Temp. sensor	Yes
Self Test Controller	Yes
PLL	Dual PLL with FM
External Power Supplies	3.3V - 5V
Integrated linear voltage regulator	Yes
Low Power Modes	Stop Mode
	Halt Mode
	Smart Standby with output controller, analog and digital inputs
	Standby Mode

2.2.8 Packages

The SPC58xGx is available in these production packages: eTQFP144, eLQFP176 and FBGA292.

The FPBGA292 packages support production and emulation devices with compatibility for an emulation package in a production device footprint.

2.2.9 Software debug and calibration

The Production Device (PD) includes many features to facilitate software development and calibration of constant data. These features include:

- Nexus code execution trace on all CPUs
- Nexus data trace on all bus masters (DMA, Interprocessor bus, CPUs, etc.)
- Performance monitors for each CPU
- Sequence Processing Unit for complex trace triggering
- 32 KB of overlay RAM

2.2.10 Block diagram

The figures below show the top-level block diagrams.

Figure 4. Block diagram

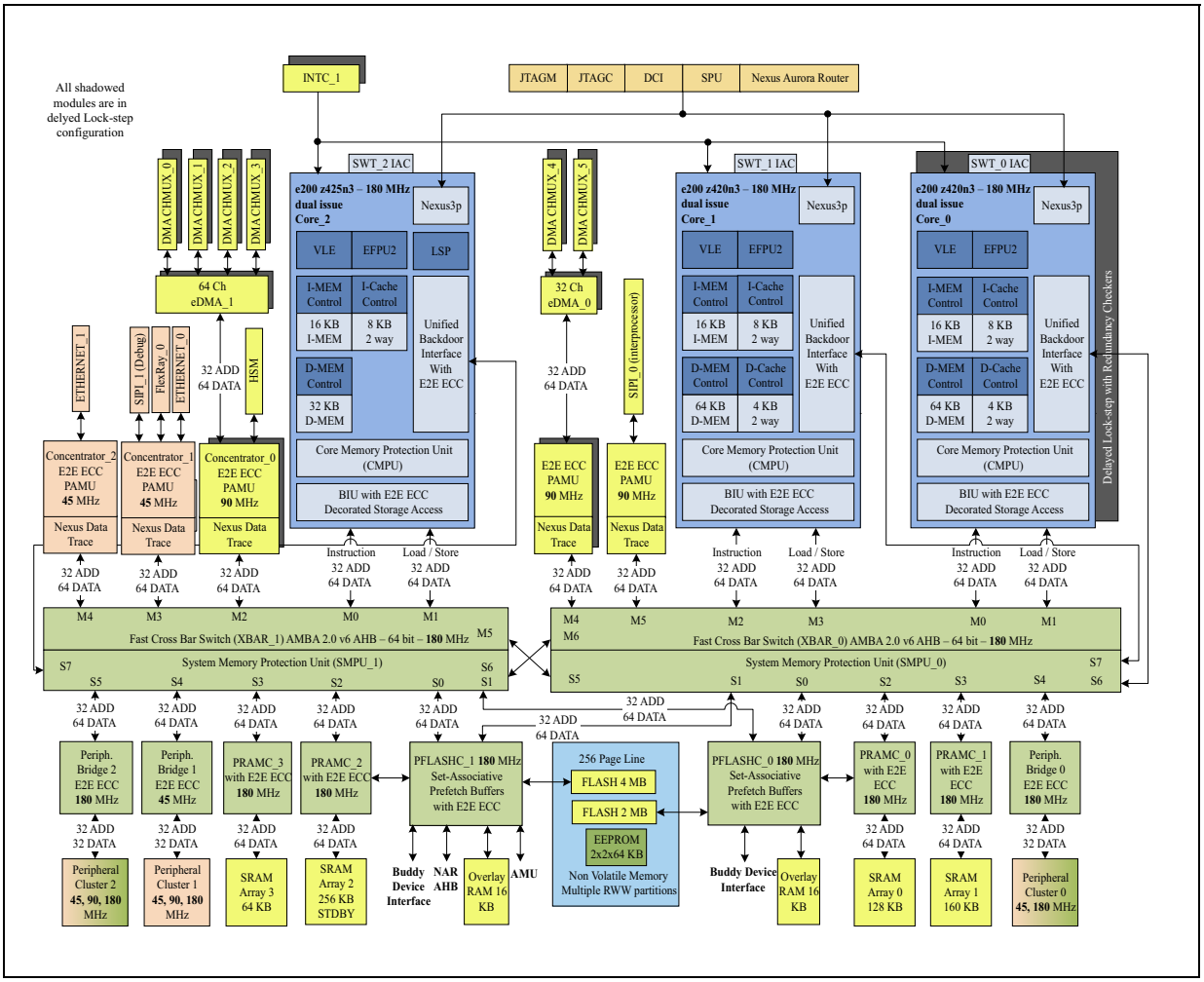
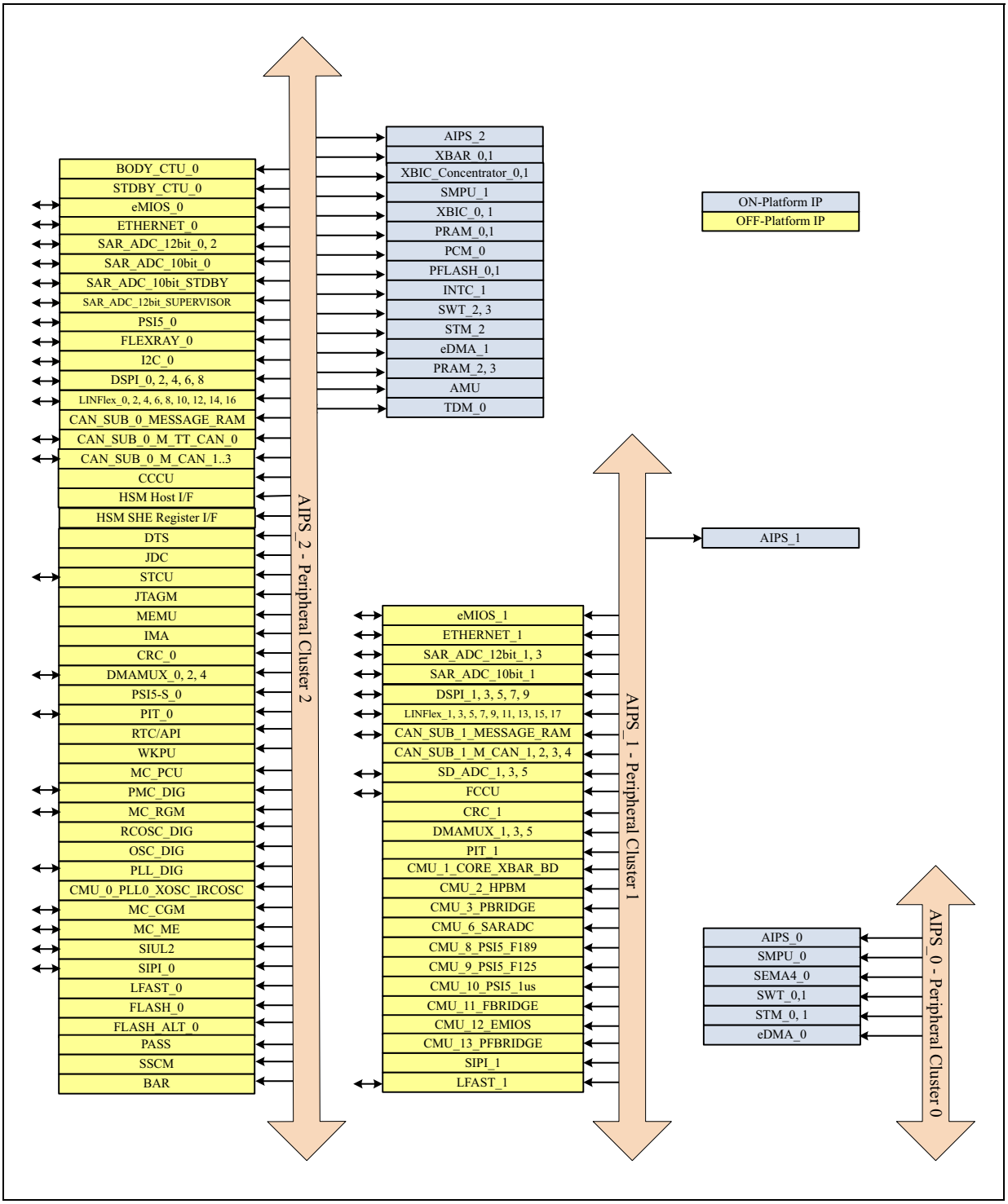


Figure 5. Periphery allocation



3 Embedded memories

3.1 Overview

Features include:

- Onboard SRAM, including system SRAM, local instruction and data memory for each processor core, and overlay SRAM
- Onboard system error correction code (ECC) Flash memory
- End-to-end ECC (e2eECC) error detection and correction
- Built-in Flash memory security features including censorship and a programmable Hardware Security Module (HSM)
- Embedded memories in the peripherals[1]:
  - AMU
  - CAN
  - eDMA
  - Ethernet
  - FlexRay
  - GTM

3.2 SRAM

3.2.1 System SRAM

SPC58xEx/SPC58xGx device includes 608 KB of general-purpose on-chip ECC SRAM, including 256 KB standby RAM. The SRAM can be configured for either zero- or one-wait state read operation latency using the PRCR1 register in the SRAM controller.

- Refer to *Chapter 35: Platform RAM controller AHB (PRAMC_AHB)* for details on configuring the SRAM controller.
- Refer to the Memory Map chapter for the SPC58xEx/SPC58xGx SRAM map.

3.2.2 Standby SRAM regulator

A total of 256 KB of the system SRAM on the device can remain powered when the rest of the device is powered down. Refer to *Chapter 10: Power management* for details.

3.2.3 Overlay SRAM

Overlay SRAM is included in the SPC58xEx/SPC58xGx family of devices as part of a comprehensive set of calibration and debug features. Overlay SRAM can be mapped over specific regions of on-chip Flash memory so that any access to an overlaid Flash address is routed to the overlay SRAM instead. This enables the calibration of constant data without requiring additional external RAMs and calibration memory interfaces.The SPC58xEx/SPC58xGx has two 16 KB banks of ECC-protected RAM, for a total of 32 KB.

1. Refer to the respective module chapter for details.

Note: Overlay SRAM is normally used for system development purposes only. Code can be executed from internal overlay RAM and extended overlay RAM where available. Execution from overlay RAM remapped over Flash can be enabled or disabled by use of PFCRCR[IRMEN]. Code can be executed directly from Extended Overlay RAM (address range beginning 0x0C000000) on the ED device or internal Overlay RAM (address range beginning 0x0D000000) regardless of the PFCRCR[IRMEN] setting.

Overlay memory can also be used to hold the device debug trace stream, without requiring complex debug tool hardware and access to a high-speed trace port. Timing for calibration accesses to a particular overlay SRAM is the same as access to the underlying Flash memory if the overlay SRAM is not being used as a destination for trace streaming at the same time.

The calibration remap function supports different types of overlay SRAM:

- An internal overlay RAM is included on all standard production devices.
- An extended overlay RAM is an additional memory that is available only on special emulation and debug devices.
- A portion of the system RAM can be used as the overlay RAM.

The overlay function is implemented using registers in the Flash memory controller.

1. Using PFlash Calibration Region Descriptors (PFCRD*n*) define one or more (up to 32) overlay regions.
2. Enable individual regions using the appropriate bits in the PFlash Remap Descriptor Enable Register (PFDCRDE).
3. Enable overlay using the PFCRCR[GRMEN] field.

Refer to *Section 36.5.13: PFlash calibration remap support*, for details on memory overlay (remapping).

3.2.4 Processor core local RAM

SPC58xEx/SPC58xGx devices include local instruction and data RAM (I-MEM and D-MEM) for each processor core to provide enhanced performance.

Refer to the Introduction chapter for processor cores local RAM sizes.

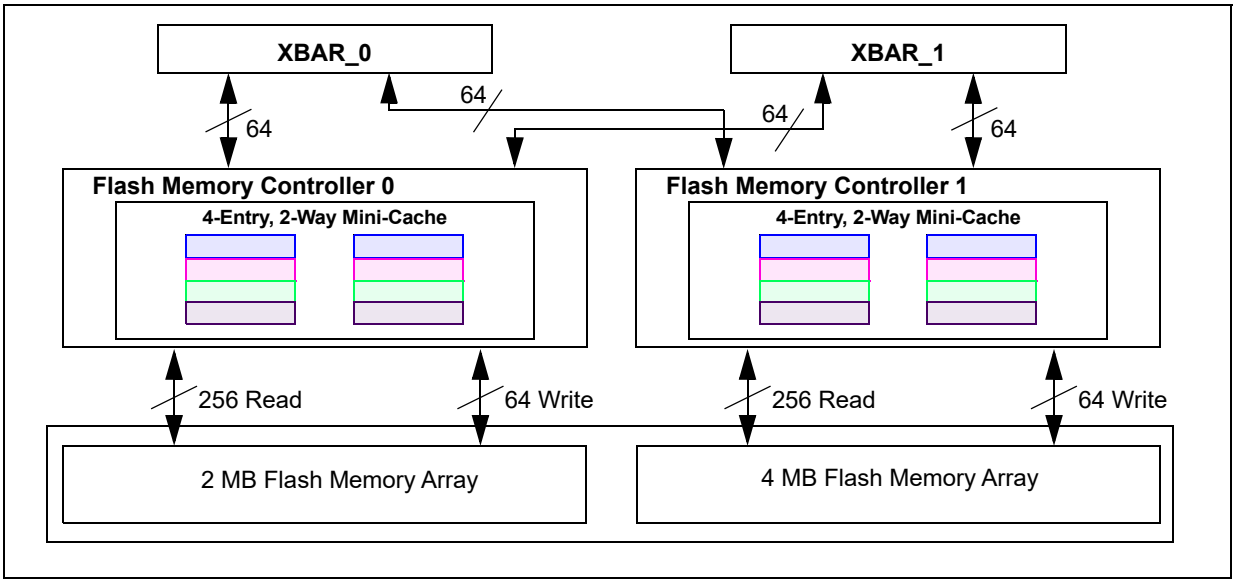
The core RAM supports zero wait-state, single-cycle latency on processor local accesses.

Each area of core RAM is accessible by other processor cores and bus mastering peripheral devices. Refer to the Memory Map chapter for the SPC58xEx/SPC58xGx SRAM map.

3.3 Embedded Flash memory

Flash memory on SPC58xEx/SPC58xGx device consists of two Flash memory controllers and a Flash memory array module. The Flash controller provides Flash configuration and control functions and manages the interface between the Flash memory array and the device crossbar switch. *Figure 6* shows the memory architecture.

Figure 6. SPC58xEx/SPC58xGx Flash memory block diagram



The remaining sections give the functional details of the Flash controller and Flash array, followed by the memory maps.

3.3.1 Flash memory controller

The SPC58xEx/SPC58xGx Flash memory controllers act as an interface between the system bus and the Flash memory array and serves as the interface to the on-chip overlay RAM and off-chip Buddy Device(2).

The Flash memory controller contains a four-entry, two-way set-associative mini-cache that delivers Flash read data with a zero-wait state response on lines that reside in the cache. Each entry contains one Flash read page, a 256-bit (32-byte) memory value. Read requests that miss the cache generate the needed Flash memory array access.

The Flash memory controller contains configuration registers that manage Flash functionality such as read buffering in the mini-cache, access control, calibration RAM overlay remapping, and read wait state management of the Flash memory.

Refer to the *Chapter 36: Flash memory controller* for details.

3.3.2 Flash memory array

SPC58xEx/SPC58xGx device includes two Flash memory arrays, one with 4 MB and one with 2 MB, referred to as “main space,” plus an independent 16 KB block of one-time-programmable (OTP) Flash memory included to support systems that require non-volatile memory for security features or system initialization information. The independent 16 KB block is referred to as “UTEST space.”

For code Flash sectors, reads of the embedded Flash memory return a 256-bit page of data that may be buffered in the mini-cache in the Flash memory controller. Programming of the

2. The Buddy Device is an optional companion chip with additional memory and debug features. A Buddy Device is combined with a Production Device to form an Emulation and Debug Device (ED) for use during system development. It is not used in production systems.

Flash may be done by double word (64-bits), page (256-bits), or quad-page (1024-bits). Flash memory is erased on a block by block basis.

3.3.2.1 Features

- Flash segmentation
  - 16 KB, 32 KB, 64 KB, 128 KB, and 256 KB blocks provided for code or data
  - Four 64 KB blocks provided for EEPROM emulation
  - One 16 KB block and two 64 KB blocks provided for secure code
  - Two 16 KB blocks provided for secure data
  - Support for reading-while-writing when the accesses are to different partitions
- Flash memory protection: write protection and OTP function available for each block
- Test and initialization data stored in a non-volatile 16 KB OTP UTEST block
- ECC with triple bit detection, double and single bit correction
- Erase suspend, program suspend, and erase-suspended program all supported

3.3.2.2 Flash organization

The embedded Flash memory consists of six address spaces in four groupings:

- 16 KB, 32 KB and 64 KB low address spaces
- 16 KB mid address space
- 64 KB high address space
- 256 KB address space

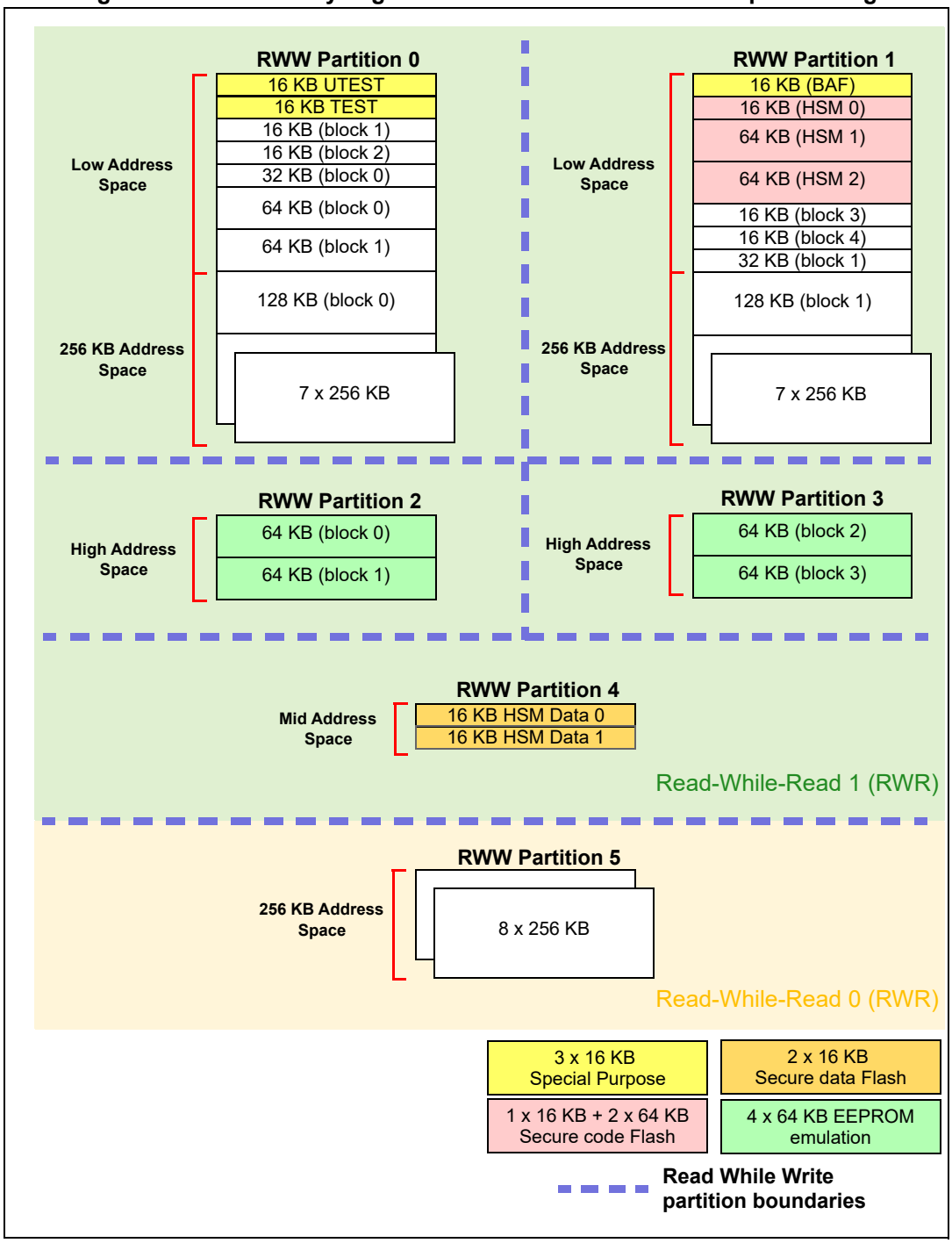
Address space group (low-, mid- or high-address space) determines which registers/fields are used to select blocks for modify operations or lock them from modify operations. Each address space is independent.

The Flash module is further divided into four partitions that determine locations for valid Read While Write (RWW) operations. While the embedded Flash memory is performing a “write” (program or erase) to a given partition, it can simultaneously read from any other partition.

- For program operations, only the address specified by an interlock write determines the partition being written (block locking and block select registers do not determine the RWW partitions being written).
- For erase operations, only blocks that are selected and unlocked determine the RWW partitions being written.
- For read operation two independent read accesses can be done through both PFlash controller to two Read While Read (RWR) partitions.

Figure 7 shows the Flash block segmentation and Read While Write partitioning for SPC58xEx/SPC58xGx devices.

Figure 7. Flash memory segmentation and Read While Write partitioning



Note: Refer to the Chapter 5: Memory Map for the SPC58xEx/SPC58xGx Flash memory map.

3.3.2.3 UTEST memory space

Devices in the SPC58xEx/SPC58xGx family contain a 16 KB area of one-time-programmable (OTP) Flash memory for storing test information and device configuration

data. *Refer to the Memory Map chapter* for the SPC58xEx/SPC58xGx UTEST Flash memory map.

3.3.2.4 Chip-specific C40FMC Flash register information

Refer to *Chapter 37: Embedded Flash Memory* for chip-specific layouts of the on-chip Flash memory control registers that contain bits used to lock individual Flash blocks from accidental erase or select individual Flash blocks for operations.

3.4 End-to-end Error Correction Code (e2eECC)

To support market requirements related to improved functional and transient fault detection capabilities, this family of automotive microcontrollers includes end-to-end ECC (e2eECC) support. This e2eECC is structurally different to traditional “ECC at memory” functionality because it provides robust error detection capabilities from one endpoint of an information transfer to another endpoint, with temporary information storage in one or more intermediate components.

Memory protected by ECC/EDC traditionally generates and checks additional error parity information local to the memory unit to detect or correct errors that have occurred on data stored in the memory, or both. On the other hand, e2eECC generates error protection codes at the source of data generation. It sends the encoded data and error protection codes to intermediate storage when a memory write is initiated by a bus master and performs a data integrity check using the previously stored error protection codes at a data memory when a read of stored information is requested by a bus master. The intermediate storage may transform the generated error protection codes into another format for storage and then regenerate the codes for provision when a request is made to read the stored information or it may simply store the original protection codes unaltered, depending on the particular unit.

Additionally, the error protection codes are generated on the basis of more than just the data associated with a storage location to protect additional information associated with an access. In particular, address information corresponding to the access location of the stored information is combined with the stored data at the data source to generate error protection codes that cover certain types of addressing errors in the system interconnect or in the memory unit. The error protection codes may be checked at the memory unit on a store to ensure that no address information or data has been corrupted while the request has transitioned through the device from the bus master source. The error protection codes may also be checked to ensure that address decoding within the storage memory was performed properly (although not all address decoding errors can be detected this way) or it may simply store the received data and protection codes at the address it receives.

On a read request from a bus master, the memory unit retrieves the data information and error protection codes corresponding to the received address from the storage location or locations and supplies the data along with the error protection codes to the requesting device. The requesting device uses a locally stored address value corresponding to the read request to check the returning data and error protection codes to ensure that no errors have occurred in either addressing the memory or in the retrieved data. This ensures that the address sent for the request was not corrupted, the addressed location was actually accessed (to the extent it is possible to ensure) and the stored data was error free, to the extent the ECC coding scheme is able to detect. As a result, the fault coverage provided by the end-to-end check is considerably more robust than the previous implementation of locally generated and checked/corrected error protection at each memory unit.

A comparison of the traditional and e2eECC approaches is shown in *Table 6*.

Table 6. ECC comparison of data write-then-read sequence

Traditional ECC	End-to-End ECC (e2eECC)
Bus master initiates data write	Bus master initiates data write and generates ECC check-bits based on 29-bit address and 64-bit data fields
Data write transfer routed from bus master to appropriate bus slave	Data write transfer (including check-bits) routed from bus master to appropriate bus slave
For a bus memory slave, generate the ECC check-bits based on the data value and store data + check-bits into the memory	For a bus memory slave, store data and check-bits into the memory
Bus master initiates data read of previously written memory location	Bus master initiates data read of previously written memory location
Data read transfer routed from bus master to appropriate bus slave	Data read transfer routed from bus master to appropriate bus slave
For a bus memory slave, the memory array is accessed, the controller performs the ECC check bit decode and syndrome generation, performs any needed single-bit correction and drives the read data onto the system bus interconnect	For a bus memory slave, the memory array is accessed, and the controller passes the read data and associated check-bits onto the system bus interconnection
The bus master captures the read data and continues	The bus master captures the read data and associated check-bits, performs the ECC check bit decode and syndrome generation, performs any needed single-bit correction and continues

The scope of differences in the operations “covered” by the ECC checks is readily apparent with the e2eECC concept providing improved fault detection capabilities in two important aspects:

- The entire data transaction; from the initiating bus master, through the entire platform crossbar steering mechanism, to the destination slave target is covered during write accesses. Likewise, a read is checked from the initiating bus master, through the crossbar steering mechanism, through the actual memory read and transmission of the data plus check-bits back to the bus master, where the integrity and correctness of the entire transaction is checked.
- The selected ECC provides protection of both the address field as well as the data field, again for improved fault coverage.

Additionally, this particular structure also provides a significant implementation improvement. The traditional ECC at the memory approach places the check-bit decode and error syndrome generation with the error/no_error state, affecting whether the system bus transfer must be stalled (to correct a single or double bit error or report a non-correctable event) or is allowed to complete (for error-free transfers) in a critical timing arc which often sets the upper limit of the operating frequency of the microcontroller. With the e2eECC scheme, the check bit decode and error syndrome logic is located in the requesting bus master where there are generally more degrees of implementation freedom and the error/no_error state determination is removed from the system bus cycle termination logic; producing an improved timing arc and generally higher operating speeds.

Errors that occur within the system interconnect are typically manifested as an incorrect address, incorrect write data, or incorrect read data to be presented to the slave or back to the master. Errors occurring within the storage typically manifest themselves eventually in

corrupted read data or check-bits being returned to a requester. While not all possible errors can be detected this way, this approach provides a substantial improvement versus traditional methods. Additional on-line diagnostics, or additional hardware, or both should be able to catch a majority of the errors not covered directly by the e2eECC scheme.

3.5 Security features

The SPC58xEx/SPC58xGx architecture implements a security framework that makes it possible to protect the micro-controller resources from potential attacks.

Note: Detailed information on the SPC58xEx/SPC58xGx security architecture is published in a separate document and is available on a limited basis to customers who have a non-disclosure agreement (NDA) with STMicroelectronics.

4 Signal Description

4.1 Production packages

The following production packages are for the SPC58xEx/SPC58xGx device:

- eLQFP176 Exposed Pad
- QFP144
- BGA292

The BGA292 is a superset of the eLQFP176 pins. The LFBGA292 adds only:

- Additional GPIO
- More analog inputs
- Added muxing options for the eLQFP176 I/O

There are no I/O interfaces defined for the LFBGA292 that are not included in the eLQFP176 (external bus, etc.).

Case number and outline drawings for each package are provided in the *SPC58xEx/SPC58xGx Microcontroller Data Sheet*.

4.2 Emulation package

The eLQFP176 and LFBGA292 packages also support emulation devices for application debug and calibration support in a production device footprint.

4.3 Package pinouts and ballouts

The SPC58xEx/SPC58xGx pin package pinouts and ballouts are contained in the Microsoft Excel® workbook files attached to the IO_Definition document.

5 Memory Map

Table 7 shows the device memory map for the SPC58xEx/SPC58xGx.

All addresses on the SPC58xEx/SPC58xGx, including those that are reserved, are identified in the table. The addresses represent the physical addresses assigned to each region or module name.

Table 7 summarizes the complete 32-bit address map. A memory access, either read or write, to any region marked as 'Reserved' will result in a Transfer Error. This error can generate an interrupt request in the CPU.

Table 8 lists the memory map of the peripheral region. This region is sub-divided into 16 KB byte regions or 'slots'. A peripheral occupies 1 or more slots. A memory access, either read or write, to any slot marked as 'Reserved' will result in a Transfer Error. This error can generate an interrupt request in the CPU.

Within a 16 KB byte peripheral slot, it is normal for the peripheral registers to only occupy part of that 16 KB byte slot, normally from the lowest address upwards. The individual chapters for each block, define all the registers within that contiguous group of registers.

Any memory access outside the range of the defined register map will result in a transfer error. This error can generate an interrupt request in the CPU.

Table 9, *Table 10*, and *Table 11* define the embedded memory maps and the buddy memory map.

Caution: On the subset devices of the SPC58NE84 and SPC58xG84 device family several memory regions and peripheral instances listed in the global memory map have to be kept reserved. Please note the different system behavior when accessing these reserved areas:

- A memory access, either read or write, to any region marked as 'Reserved' in *Table 11.*, *Table 12.*, and *Table 13.* will result in a Transfer Error. This error can generate an interrupt request in the CPU.
- A memory access, either read or write, to any region marked as 'Used' in *Table 11.*, *Table 12.*, and *Table 13.* but not available in the subset device specification will not result in a Transfer Error. Please take care for an appropriate configuration of the System Memory Protection Unit (SMPU) to protect all these implemented but reserved address areas from being accessed unintentionally.

Table 7. Overview memory map

Start Address	End Address	Allocated Size	Used Size	Description
Flash (XBAR_0 - port 0/1) (XBAR_1 - port 0/1)				
0x00000000	0x003FFFFF	—	Reserved	
0x00400000	0x00403FFF	16 KB	16 KB	UTest NVM Block — no overlay
0x00404000	0x00407FFF	16 KB	16 KB	BAF Flash low block — no overlay
0x00408000	0x005FFFFF	2016 KB	Reserved	
0x00600000	0x0067FFFF	512 KB	144 KB	HSM Code — no overlay
0x00680000	0x007FFFFF	1.5 MB	32 KB	HSM Data — no overlay

Table 7. Overview memory map (continued)

Start Address	End Address	Allocated Size	Used Size	Description
0x00800000	0x009FFFFF	22048 KB	256 KB	Data Flash Blocks — no overlay
0x00A00000	0x00FFFFFF	6 MB	256 KB	Low and Mid Flash Blocks — no overlay
0x01000000	0x01FFFFFF	16 MB	5888 KB	Large Flash Blocks — no overlay
0x02000000	0x089FFFFF	106 MB	Reserved	
0x08A00000	0x08FFFFFF	6 MB	256 KB	Low and mid Flash blocks — overlay enabled mapping
0x09000000	0x09FFFFFF	16 MB	5888 KB	Large Flash blocks — overlay enabled mapping
0x0A000000	0x0BFFFFFF	32 MB	Reserved	
0x0C000000	0x0C1FFFFF	2 MB	2 MB	Extended Overlay RAM(1)
0x0C200000	0x0C7FFFFF	6 MB	Reserved	
0x0C800000	0x0CFFFFFF	8 MB	1 MB	Buddy Device Registers
0x0D000000	0x0D003FFF	16 KB	16 KB	Internal Overlay RAM 0(2)
0x0D004000	0x0D007FFF	16 KB	16 KB	Internal Overlay RAM 1
0x0D008000	0x0FFFFFFF	—	Reserved	
0x10000000	0x1FFFFFFF	256 MB	Reserved	
0x20000000	0x2FFFFFFF	256 MB	Reserved	
0x30000000	0x3FFFFFFF	256 MB	Reserved	
System RAM(XBAR_0 - port 2/3) (XBAR_1 - port 5/6)(3)				
0x40000000	0x401FFFFF	2048 KB	608 KB	System RAM
0x40200000	0x4FFFFFFF	—	Reserved System RAM	
Local Memory (XBAR_0 - port 6) (XBAR_1 - port 7)(4)				
0x50000000	0x5FFFFFFF	256 MB	160 KB (DMEM) 48 KB (IMEM)	Processor Core local SRAM
0x60000000	0x6FFFFFFF	256 MB	Reserved	
0x70000000	0x7FFFFFFF	256 MB	Reserved	
0x80000000	0x8FFFFFFF	256 MB	Reserved	
0x90000000	0x9FFFFFFF	256 MB	Reserved	
HSM (Internal to HSM) (XBAR Slave Port 1–A)				
0xA0000000	0xAFFFFFFF	256 MB		Hardware Security Module (HSM)(5)
Spare Port (XBAR Slave Port 1–B)				
0xB0000000	0xBFFFFFFF	262144 KB	Reserved	
Reserved Port C (XBAR Slave Port 1–C)				

Table 7. Overview memory map (continued)

Start Address	End Address	Allocated Size	Used Size	Description
0xC0000000	0xCFFFFFFF	262144 KB	Reserved	
0xD0000000	0xDFFFFFFF	262144 KB	Reserved	
0xE0000000	0xEFFFFFFF	262144 KB	Reserved	
Peripherals PBRIDGE_0, PBRIDGE_1, PBRIDGE_2(XBAR_0 Port 4, XBAR_1 Port 4/5)				
0xF0000000	0xFFFFFFFF	256 MB		Peripheral address space(6)

1. Address range is for extended overlay RAM for buddy device.
2. This memory range is applicable on PFLASH_0 and is not accessible by NAR.
3. See *Table 11* for details.
4. See *Table 12* for local SRAM addresses.
5. Refer to the Security Reference Manual for details about how the HSM memory space is partitioned.
6. See *Table 8* for details.

5.1 Peripheral memory

Table 8. Peripheral (PBRIDGE_0, PBRIDGE_1, PBRIDGE_2) memory map

Start Address	End Address	Peripheral	"ON-Platform" Peripheral Slot(1)	"OFF-Platform" Peripheral Slot(2)	PBRIDGE	Size [Byte](3)
0xF4000000	0xF4003FFF	PBRIDGE_2	0	—	2	16384
0xF4004000	0xF4007FFF	XBAR_0	1	—	2	16384
0xF4008000	0xF400BFFF	XBAR_1	2	—	2	16384
0xF400C000	0xF400FFFF	XBIC_Concentrator_0	3	—	2	16384
0xF4010000	0xF4013FFF	XBIC_Concentrator_1	4	—	2	16384
0xF4014000	0xF4017FFF	SMPU_1	5	—	2	16384
0xF4018000	0xF401BFFF	XBIC_0 (for XBAR_0)	6	—	2	16384
0xF401C000	0xF401FFFF	XBIC_1 (for XBAR_1)	7	—	2	16384
0xF4020000	0xF4023FFF	PRAM_0	8	—	2	16384
0xF4024000	0xF4027FFF	PRAM_1	9	—	2	16384
0xF4028000	0xF402BFFF	PCM_0	10	—	2	16384
0xF402C000	0xF402FFFF	Reserved				
0xF4030000	0xF4033FFF	PFLASH_0	12	—	2	16384
0xF4034000	0xF4037FFF	PFLASH_1	13	—	2	16384
0xF4038000	0xF4043FFF	Reserved				
0xF4044000	0xF4047FFF	INTC_1	17	—	2	16384

Table 8. Peripheral (PBRIDGE_0, PBRIDGE_1, PBRIDGE_2) memory map (continued)

Start Address	End Address	Peripheral	"ON-Platform" Peripheral Slot(1)	"OFF-Platform" Peripheral Slot(2)	PBRIDGE	Size [Byte](3)
0xF4048000	0xF4057FFF	Reserved				
0xF4058000	0xF405BFFF	SWT_2	22	—	2	16384
0xF405C000	0xF405FFFF	SWT_3	23	—	2	16384
0xF4060000	0xF406FFFF	Reserved				
0xF4070000	0xF4073FFF	STM_2	28	—	2	16384
0xF4074000	0xF40A3FFF	Reserved				
0xF40A4000	0xF40A7FFF	eDMA_1	41	—	2	16384
0xF40A8000	0xF40ABFFF	PRAM_2	42	—	2	16384
0xF40AC000	0xF40AFFFF	PRAM_3	43	—	2	16384
0xF40B0000	0xF40B3FFF	Reserved (XBIC_2)				
0xF40B4000	0xF40BFFFF	Reserved				
0xF40C0000	0xF40C3FFF	AMU_LMEM (16 KB ARAM)	48	—	2	16384
0xF40C4000	0xF40C7FFF	AMU_LMEM (15 KB ARAM + 1KB temp storage)	49	—	2	16384
0xF40C8000	0xF40CFFFF	Reserved				
0xF40D0000	0xF40D3FFF	AMU_REGS	52	—	2	16384
0xF40A8000	0xF40E3FFF	Reserved				
0xF40E4000	0xF40E7FFF	TDM_0	57	—	2	16384
0xF40E8000	0xF40FFFFF	Reserved				
0xF7C00000	0xF7C2BFFF	Reserved				
0xF7C2C000	0xF7C2FFFF	Reserved				
0xF7C30000	0xF7C33FFF	BODY_CTU_0	—	243	2	16384
0xF7C34000	0xF7C37FFF	STDBY_CTU_0	—	242	2	16384
0xF7C38000	0xF7C3BFFF	eMIOS_0	—	241	2	16384
0xF7C3C000	0xF7C53FFF	Reserved				
0xF7C54000	0xF7C57FFF	ETHERNET_0	—	234	2	16384
0xF7C58000	0xF7CFFFFF	Reserved				
0xF7D00000	0xF7D5BFFF	GTM(4)		191 – 169	2	368K
0xF7D5C000	0xF7DFFFFF	Reserved				
0xF7E00000	0xF7E03FFF	SAR_ADC_12bit_0	—	127	2	16384
0xF7E04000	0xF7E07FFF	SAR_ADC_12bit_2	—	126	2	16384

Table 8. Peripheral (PBRIDGE_0, PBRIDGE_1, PBRIDGE_2) memory map (continued)

Start Address	End Address	Peripheral	"ON-Platform" Peripheral Slot(1)	"OFF-Platform" Peripheral Slot(2)	PBRIDGE	Size [Byte](3)
0xF7E08000	0xF7E1FFFF	Reserved				
0xF7E20000	0xF7E23FFF	SAR_ADC_10bit_0	—	119	2	16384
0xF7E24000	0xF7E33FFF	Reserved				
0xF7E34000	0xF7E37FFF	SAR_ADC_10bit_ STDBY	—	114	2	16384
0xF7E38000	0xF7E3BFFF	Reserved				
0xF7E3C000	0xF7E3FFFF	SAR_ADC_12bit _SUPERVISOR (SAR_ADC_B)	—	112	2	16384
0xF7E40000	0xF7E43FFF	PSI5_0	—	111	2	16384
0xF7E44000	0xF7E4FFFF	Reserved				
0xF7E50000	0xF7E53FFF	FLEXRAY_0	—	107	2	16384
0xF7E54000	0xF7E5BFFF	Reserved				
0xF7E5C000	0xF7E5FFFF	SENT_0	—	104	2	16384
0xF7E60000	0xF7E67FFF	Reserved				
0xF7E68000	0xF7E6BFFF	IIC_0	—	101	2	16384
0xF7E6C000	0xF7E6FFFF	Reserved				
0xF7E70000	0xF7E73FFF	DSPI_0	—	99	2	16384
0xF7E74000	0xF7E77FFF	DSPI_2	—	98	2	16384
0xF7E78000	0xF7E7BFFF	DSPI_4	—	97	2	16384
0xF7E7C000	0xF7E7FFFF	DSPI_6	—	96	2	16384
0xF7E80000	0xF7E83FFF	DSPI_8	—	95	2	16384
0xF7E84000	0xF7E87FFF	Reserved				
0xF7E88000	0xF7E8BFFF	Reserved				
0xF7E8C000	0xF7E8FFFF	LINFlexD_0	—	92	2	16384
0xF7E90000	0xF7E93FFF	LINFlexD_2	—	91	2	16384
0xF7E94000	0xF7E97FFF	LINFlexD_4	—	90	2	16384
0xF7E98000	0xF7E9BFFF	LINFlexD_6	—	89	2	16384
0xF7E9C000	0xF7E9FFFF	LINFlexD_8	—	88	2	16384
0xF7EA0000	0xF7EA3FFF	LINFlexD_10	—	87	2	16384
0xF7EA4000	0xF7EA7FFF	LINFlexD_12	—	86	2	16384
0xF7EA8000	0xF7EABFFF	LINFlexD_14	—	85	2	16384
0xF7EAC000	0xF7EAFFFF	LINFlexD_16	—	84	2	16384
0xF7EB0000	0xF7EB3FFF	Reserved				

Table 8. Peripheral (PBRIDGE_0, PBRIDGE_1, PBRIDGE_2) memory map (continued)

Start Address	End Address	Peripheral	"ON-Platform" Peripheral Slot(1)	"OFF-Platform" Peripheral Slot(2)	PBRIDGE	Size [Byte](3)
0xF7EB4000	0xF7EB7FFF	Reserved				
0xF7EB8000	0xF7ED3FFF	Reserved				
0xF7ED4000	0xF7ED7FFF	CAN_SUB_0_ MESSAGE_RAM (15.5 KB implemented)	—	74	2	16384
0xF7ED8000	0xF7EDBFFF	Reserved				
0xF7EDC000	0xF7EDFFFF	CAN_SUB_0_ M_TTCAN_0	—	72	2	16384
0xF7EE0000	0xF7EE7FFF	Reserved				
0xF7EE8000	0xF7EEBFFF	CAN_SUB_0_ M_CAN_1	—	69	2	16384
0xF7EEC000	0xF7EEFFFF	CAN_SUB_0_ M_CAN_2	—	68	2	16384
0xF7EF0000	0xF7EF3FFF	CAN_SUB_0_ M_CAN_3	—	67	2	16384
0xF7EF4000	0xF7F03FFF	Reserved				
0xF7F04000	0xF7F07FFF	CCCU_0	—	62	2	16384
0xF7F08000	0xF7F0BFFF	Reserved				
0xF7F0C000	0xF7F0FFFF	SD_ADC_0	—	60	2	16384
0xF7F10000	0xF7F13FFF	SD_ADC_2	—	59	2	16384
0xF7F14000	0xF7F17FFF	SD_ADC_4	—	58	2	16384
0xF7F18000	0xF7F2FFFF	Reserved				
0xF7F30000	0xF7F33FFF	HSM Host I/F	—	51	2	16384
0xF7F34000	0xF7F37FFF	Reserved				
0xF7F38000	0xF7F3BFFF	DTS	—	49	2	16384
0xF7F3C000	0xF7F3FFFF	JDC	—	48	2	16384
0xF7F40000	0xF7F43FFF	Reserved				
0xF7F44000	0xF7F47FFF	STCU2	—	46	2	16384
0xF7F48000	0xF7F4BFFF	JTAGM	—	45	2	16384
0xF7F4C000	0xF7F4FFFF	Reserved				
0xF7F50000	0xF7F53FFF	MEMU	—	43	2	16384
0xF7F54000	0xF7F57FFF	IMA	—	42	2	16384
0xF7F58000	0xF7F63FFF	Reserved				
0xF7F64000	0xF7F67FFF	CRC_0	—	38	2	16384
0xF7F68000	0xF7F6BFFF	Reserved				

Table 8. Peripheral (PBRIDGE_0, PBRIDGE_1, PBRIDGE_2) memory map (continued)

Start Address	End Address	Peripheral	"ON-Platform" Peripheral Slot(1)	"OFF-Platform" Peripheral Slot(2)	PBRIDGE	Size [Byte](3)
0xF7F6C000	0xF7F6C1FF	DMAMUX_0	—	36	2	512
0xF7F6C200	0xF7F6C3FF	DMAMUX_2	—	36	2	512
0xF7F6C400	0xF7F6C5FF	DMAMUX_4	—	36	2	512
0xF7F6C600	0xF7F73FFF	Reserved				
0xF7F74000	0xF7F77FFF	(ATX_0 - used for internal testing purpose only)	—	34	2	16384
0xF7F78000	0xF7F7BFFF	PSI5-S_0	—	33	2	16384
0xF7F7C000	0xF7F83FFF	Reserved				
0xF7F84000	0xF7F87FFF	PIT_0	—	30	2	16384
0xF7F88000	0xF7F93FFF	Reserved				
0xF7F94000	0xF7F97FFF	RTC/API	—	26	2	16384
0xF7F98000	0xF7F9BFFF	WKPU	—	25	2	16384
0xF7F9C000	0xF7F9FFFF	Reserved				
0xF7FA0000	0xF7FA03FF	MC_PCU	—	23	2	1024
0xF7FA0400	0xF7FA3FFF	PMC_DIG	—	23	2	15360
0xF7FA4000	0xF7FA7FFF	Reserved				
0xF7FA8000	0xF7FABFFF	MC_RGM	—	21	2	16384
0xF7FAC000	0xF7FAFFFF	Reserved				
0xF7FB0000	0xF7FB003F	RC16M_DIG	—	19	2	64
0xF7FB0040	0xF7FB007F	RC1024K_DIG	—	19	2	64
0xF7FB0080	0xF7FB00BF	OSC40M_DIG	—	19	2	64
0xF7FB00C0	0xF7FB00FF	OSC32K_DIG	—	19	2	64
0xF7FB0100	0xF7FB013F	PLL_DIG	—	19	2	64
0xF7FB0140	0xF7FB01FF	Reserved				
0xF7FB0200	0xF7FB023F	CMU_0_PLL0_ XOSC_IRCOSC	—	19	2	64
0xF7FB0240	0xF7FB05FF	Reserved				
0xF7FB0600	0xF7FB3EFF	MC_CGM	—	19	2	14592
0xF7FB3F00	0xF7FB7FFF	Reserved				
0xF7FB8000	0xF7FBBFFF	MC_ME	—	17	2	16384
0xF7FBC000	0xF7FBFFFF	Reserved				
0xF7FC0000	0xF7FC3FFF	SIUL2	—	15	2	16384
0xF7FC4000	0xF7FCFFFF	Reserved				

Table 8. Peripheral (PBRIDGE_0, PBRIDGE_1, PBRIDGE_2) memory map (continued)

Start Address	End Address	Peripheral	"ON-Platform" Peripheral Slot(1)	"OFF-Platform" Peripheral Slot(2)	PBRIDGE	Size [Byte](3)
0xF7FD0000	0xF7FD3FFF	SIPI_0 (Interprocessor)	—	11	2	16384
0xF7FD4000	0xF7FD7FFF	Reserved				
0xF7FD8000	0xF7FDBFFF	LFAST_0 (Interprocessor)	—	9	2	16384
0xF7FDC000	0xF7FDFFFF	Reserved				
0xF7FE0000	0xF7FE3FFF	FLASH_0	—	7	2	16384
0xF7FE4000	0xF7FE7FFF	FLASH_ALT_0	—	6	2	16384
0xF7FE8000	0xF7FF3FFF	Reserved				
0xF7FF4000	0xF7FF7FFF	PASS	—	2	2	16384
0xF7FF8000	0xF7FFBFFF	SSCM	—	1	2	16384
0xF7FFC000	0xF7FFFFFF	BAR	—	0	2	16384
0xF8000000	0xF8003FFF	PBRIDGE_1	0		1	16384
0xF8004000	0xF80FFFFF	Reserved				
0xFBC00000	0xFBC2BFFF	Reserved				
0xFBC2C000	0xFBC2CFFF	Reserved				
0xFBC30000	0xFBC37FFF	Reserved				
0xFBC38000	0xFBC3BFFF	eMIOS_1	—	241	1	16384
0xFBC3C000	0xFBC53FFF	Reserved				
0xFBC54000	0xFBC57FFF	ETHERNET_1	—	234	1	16384
0xFBC58000	0xFBDFFFFF	Reserved				
0xFBE00000	0xFBE03FFF	SAR_ADC_12bit_1	—	127	1	16384
0xFBE04000	0xFBE07FFF	SAR_ADC_12bit_3	—	126	1	16384
0xFBE08000	0xFBE1FFFF	Reserved				
0xFBE20000	0xFBE23FFF	SAR_ADC_10bit_1	—	119	1	16384
0xFBE24000	0xFBE3FFFF	Reserved				
0xFBE40000	0xFBE43FFF	PSI5_1	—	111	1	16384
0xFBE44000	0xFBE4FFFF	Reserved				
0xFBE50000	0xFBE53FFF	Reserved				
0xFBE54000	0xFBE5BFFF	Reserved				
0xFBE5C000	0xFBE5FFFF	SENT_1	—	104	1	16384
0xFBE60000	0xFBE6FFFF	Reserved				
0xFBE70000	0xFBE73FFF	DSPI_1	—	99	1	16384
0xFBE74000	0xFBE77FFF	DSPI_3	—	98	1	16384

Table 8. Peripheral (PBRIDGE_0, PBRIDGE_1, PBRIDGE_2) memory map (continued)

Start Address	End Address	Peripheral	"ON-Platform" Peripheral Slot(1)	"OFF-Platform" Peripheral Slot(2)	PBRIDGE	Size [Byte](3)
0xFBE78000	0xFBE7BFFF	DSPI_5	—	97	1	16384
0xFBE7C000	0xFBE7FFFF	DSPI_7	—	96	1	16384
0xFBE80000	0xFBE83FFF	DSPI_9	—	95	1	16384
0xFBE84000	0xFBE87FFF	Reserved				
0xFBE88000	0xFBE8BFFF	Reserved				
0xFBE8C000	0xFBE8FFFF	LINFlexD_1	—	92	1	16384
0xFBE90000	0xFBE93FFF	LINFlexD_3	—	91	1	16384
0xFBE94000	0xFBE97FFF	LINFlexD_5	—	90	1	16384
0xFBE98000	0xFBE9BFFF	LINFlexD_7	—	89	1	16384
0xFBE9C000	0xFBE9FFFF	LINFlexD_9	—	88	1	16384
0xFBEA0000	0xFBEA3FFF	LINFlexD_11	—	87	1	16384
0xFBEA4000	0xFBEA7FFF	LINFlexD_13	—	86	1	16384
0xFBEA8000	0xFBEABFFF	LINFlexD_15	—	85	1	16384
0xFBEAC000	0xFBEAFFFF	LINFlexD_17	—	84	1	16384
0xFBEB0000	0xFBEB3FFF	Reserved				
0xFBEB4000	0xFBEB7FFF	Reserved				
0xFBEB8000	0xFBED3FFF	Reserved				
0xFBED4000	0xFBED7FFF	CAN_SUB_1_ MESSAGE_RAM (15.5 KB implemented)	—	74	1	16384
0xFBED8000	0xFBEE3FFF	Reserved				
0xFBEE4000	0xFBEE7FFF	CAN_SUB_1_ M_CAN_1	—	70	1	16384
0xFBEE8000	0xFBEEBFFF	CAN_SUB_1_ M_CAN_2	—	69	1	16384
0xFBEEC000	0xFBEEFFFF	CAN_SUB_1_ M_CAN_3	—	68	1	16384
0xFBEF0000	0xFBEF3FFF	CAN_SUB_1_ M_CAN_4	—	67	1	16384
0xFBEF4000	0xFBF0BFFF	Reserved				
0xFBF0C000	0xFBF0FFFF	SD_ADC_1	—	60	1	16384
0xFBF10000	0xFBF13FFF	SD_ADC_3	—	59	1	16384
0xFBF14000	0xFBF17FFF	SD_ADC_5	—	58	1	16384
0xFBF18000	0xFBF57FFF	Reserved				
0xFBF58000	0xFBF5BFFF	FCCU	—	41	1	16384

Table 8. Peripheral (PBRIDGE_0, PBRIDGE_1, PBRIDGE_2) memory map (continued)

Start Address	End Address	Peripheral	"ON-Platform" Peripheral Slot(1)	"OFF-Platform" Peripheral Slot(2)	PBRIDGE	Size [Byte](3)
0xFBF5C000	0xFBF63FFF	Reserved				
0xFBF64000	0xFBF67FFF	CRC_1	—	38	1	16384
0xFBF68000	0xFBF6BFFF	Reserved				
0xFBF6C000	0xFBF6C1FF	DMAMUX_1	—	36	1	512
0xFBF6C200	0xFBF6C3FF	DMAMUX_3	—	36	1	512
0xFBF6C400	0xFBF6C5FF	DMAMUX_5	—	36	1	512
0xFBF6C600	0xFBF83FFF	Reserved				
0xFBF84000	0xFBF87FFF	PIT_1	—	30	1	16384
0xFBF88000	0xFBFB01FF	Reserved				
0xFBFB0200	0xFBFB023F	CMU_1_CORE_ XBAR_BD	—	19	1	64
0xFBFB0240	0xFBFB027F	CMU_2_HPBM	—	19	1	64
0xFBFB0280	0xFBFB02BF	CMU_3_PBRIDGE	—	19	1	64
0xFBFB02C0	0xFBFB02FF	CMU_4_GTM	—	19	1	64
0xFBFB0300	0xFBFB033F	CMU_5_SDADC	—	19	1	64
0xFBFB0340	0xFBFB037F	CMU_6_SARADC	—	19	1	64
0xFBFB0380	0xFBFB03BF	CMU_7_SENT	—	19	1	64
0xFBFB03C0	0xFBFB03FF	CMU_8_PSI5_F189	—	19	1	64
0xFBFB0400	0xFBFB043F	CMU_9_PSI5_F125	—	19	1	64
0xFBFB0440	0xFBFB047F	CMU_10_PSI5_1us	—	19	1	64
0xFBFB0480	0xFBFB04BF	CMU_11_FBRIDGE	—	19	1	64
0xFBFB04C0	0xFBFB04FF	CMU_12_EMIOS	—	19	1	64
0xFBFB0500	0xFBFB053F	CMU_13_PFBRIDGE	—	19	1	64
0xFBFB0540	0xFBFCFFFF	Reserved				
0xFBFD0000	0xFBFD3FFF	SIPI_1 (debug)	—	11	1	16384
0xFBFD4000	0xFBFD7FFF	Reserved				
0xFBFD8000	0xFBFDBFFF	LFAST_1 (debug)	—	9	1	16384
0xFBFDC000	0xFBFFFFFF	Reserved				
0xFC000000	0xFC003FFF	PBRIDGE_0	0	—	0	16384
0xFC004000	0xFC00FFFF	Reserved				
0xFC010000	0xFC013FFF	SMPU_0	4	—	0	16384
0xFC014000	0xFC03BFFF	Reserved				
0xFC03C000	0xFC03FFFF	SEMA4_0	15	—	0	16384

Table 8. Peripheral (PBRIDGE_0, PBRIDGE_1, PBRIDGE_2) memory map (continued)

Start Address	End Address	Peripheral	"ON-Platform" Peripheral Slot(1)	"OFF-Platform" Peripheral Slot(2)	PBRIDGE	Size [Byte](3)
0xFC040000	0xFC043FFF	Reserved				
0xFC044000	0xFC04FFFF	Reserved				
0xFC050000	0xFC053FFF	SWT_0	20	—	0	16384
0xFC054000	0xFC057FFF	SWT_1	21	—	0	16384
0xFC058000	0xFC05FFFF	Reserved				
0xFC060000	0xFC063FFF	Reserved				
0xFC064000	0xFC067FFF	Reserved				
0xFC068000	0xFC06BFFF	STM_0	26	—	0	16384
0xFC06C000	0xFC06FFFF	STM_1	27	—	0	16384
0xFC070000	0xFC073FFF	Reserved				
0xFC074000	0xFC077FFF	Reserved				
0xFC078000	0xFC09FFFF	Reserved				
0xFC0A0000	0xFC0A3FFF	eDMA_0	40	—	0	16384
0xFC0A4000	0xFC0FFFFF	Reserved				
0xFFC00000	0xFFFFFFFF	Reserved				

1. Refer to PACR registers in the Peripheral Bridge chapter
2. Refer to OPACR registers in the Peripheral Bridge chapter
3. This number indicates the assigned address range for the peripheral module on the peripheral bridge. The implemented memory size or the number of implemented peripheral registers might be smaller. Please refer to the dedicated peripheral chapter or device configuration chapter for more details.
4. For details, refer to *Chapter 47: GTM343 Integration (GTMINT) Module*.

5.2 Flash memory

Table 9 is the detailed Flash memory map for the SPC58xEx/SPC58xGx.

Table 9. Flash memory map

Start Address	End Address	Description	RWW partition ID	PFLASH Controller assignment (RWR partition ID)(1)
0x00000000	0x003FFFFF	Reserved	—	—

Table 9. Flash memory map (continued)

Start Address	End Address	Description	RWW partition ID	PFLASH Controller assignment (RWR partition ID)(1)
TEST & BAF Flash Blocks				
0x00400000	0x00403FFF	16 KB UTEST NVM Block Space	0	1
0x00404000	0x00407FFF	16 KB BAF block0	1	
0x00408000	0x0040BFFF	Flash 16 KB Test block	0	
0x0040C000	0x0060BFFF	Reserved	—	
HSM Code Flash Blocks				
0x0060C000	0x0060FFFF	16 KB HSM Code block5	1	1
0x00610000	0x0061FFFF	64 KB HSM Code block2	1	
0x00620000	0x0062FFFF	64 KB HSM Code block3	1	
0x00630000	0x0067FFFF	Reserved	—	
HSM Data Flash Blocks				
0x00680000	0x00683FFF	16 KB HSM EEPROM block0	4	1
0x00684000	0x00687FFF	16 KB HSM EEPROM block1	4	
0x00688000	0x007FFFFF	Reserved	—	
Data Flash Blocks				
0x00800000	0x0080FFFF	64 KB FMC EEPROM Block0	2	1
0x00810000	0x0081FFFF	64 KB FMC EEPROM Block1	2	
0x00820000	0x0082FFFF	64 KB FMC EEPROM Block2	3	
0x00830000	0x0083FFFF	64 KB FMC EEPROM Block3	3	
0x00840000	0x00FBFFFF	Reserved	—	
Low & Mid Flash Blocks				
0x00FC0000	0x00FC3FFF	16 KB Code Flash block3	1	1
0x00FC4000	0x00FC7FFF	16 KB Code Flash block1	0	
0x00FC8000	0x00FCBFFF	16 KB Code Flash block4	1	
0x00FCC000	0x00FCFFFF	16 KB Code Flash block2	0	
0x00FD0000	0x00FD7FFF	32 KB Code Flash block0	0	
0x00FD8000	0x00FDFFFF	32 KB Code Flash block1	1	
0x00FE0000	0x00FEFFFF	64 KB Code Flash block0	0	
0x00FF0000	0x00FFFFFF	64 KB Code Flash block1	0	

Table 9. Flash memory map (continued)

Start Address	End Address	Description	RWW partition ID	PFLASH Controller assignment (RWR partition ID)(1)
Large Flash Blocks				
0x01000000	0x0101FFFF	128 KB Code Flash block0	0	1
0x01020000	0x0103FFFF	128 KB Code Flash block1	1	
0x01040000	0x0107FFFF	256 KB Code Flash block0	0	
0x01080000	0x010BFFFF	256 KB Code Flash block1	0	
0x010C0000	0x010FFFFF	256 KB Code Flash block2	0	
0x01100000	0x0113FFFF	256 KB Code Flash block3	0	
0x01140000	0x0117FFFF	256 KB Code Flash block4	0	
0x01180000	0x011BFFFF	256 KB Code Flash block5	0	
0x011C0000	0x011FFFFF	256 KB Code Flash block6	0	
0x01200000	0x0123FFFF	256 KB Code Flash block7	1	
0x01240000	0x127FFFF	256 KB Code Flash block8	1	
0x01280000	0x012BFFFF	256 KB Code Flash block9	1	
0x012C0000	0x012FFFFF	256 KB Code Flash block10	1	
0x01300000	0x0133FFFF	256 KB Code Flash block11	1	
0x01340000	0x0137FFFF	256 KB Code Flash block12	1	
0x01380000	0x013BFFFF	256 KB Code Flash block13	1	
0x013C0000	0x013FFFFF	256 KB Code Flash block14	5	0
0x01400000	0x0143FFFF	256 KB Code Flash block15	5	
0x01440000	0x0147FFFF	256 KB Code Flash block16	5	
0x01480000	0x014BFFFF	256 KB Code Flash block17	5	
0x014C0000	0x014FFFFF	256 KB Code Flash block18	5	
0x01500000	0x0153FFFF	256 KB Code Flash block19	5	
0x01540000	0x0157FFFF	256 KB Code Flash block20	5	
0x01580000	0x015BFFFF	256 KB Code Flash block21	5	
0x015C0000	0x08FBFFFF	Reserved(2)	—	

Table 9. Flash memory map (continued)

Start Address	End Address	Description	RWW partition ID	PFLASH Controller assignment (RWR partition ID)(1)
Mirror Low and Mid Flash Blocks				
0x08FC0000	0x08FC3FFF	Mirror Flash 16 KB Code Block 3	1	1
0x08FC4000	0x08FC7FFF	Mirror Flash 16 KB Code Block 1	0	
0x08FC8000	0x08FCBFFF	Mirror Flash 16 KB Code Block 4	1	
0x08FCC000	0x08FCFFFF	Mirror Flash 16 KB Code Block 2	0	
0x08FD0000	0x08FD7FFF	Mirror Flash 32 KB Code Block 0	0	
0x08FD8000	0x08FDFFFF	Mirror Flash 32 KB Code Block 1	1	
0x08FE0000	0x08FEFFFF	Mirror Flash 64 KB Code Block 0	0	
0x08FF0000	0x08FFFFFF	Mirror Flash 64 KB Code Block 1	0	
Mirror Large Flash Blocks				
0x09000000	0x901FFFF	Mirror Flash 128 KB Code Block 0	0	1
0x09020000	0x903FFFF	Mirror Flash 128 KB Code Block 1	1	
0x09040000	0x907FFFF	Mirror Flash 256 KB Code Block 1	0	
0x09080000	0x90BFFFF	Mirror Flash 256 KB Code Block 2	0	
0x090C0000	0x90FFFFF	Mirror Flash 256 KB Code Block 3	0	
0x09100000	0x93FFFFF	Mirror Flash 256 KB Code Block 4	0	
0x09140000	0x97FFFFF	Mirror Flash 256 KB Code Block 5	0	
0x09180000	0x91BFFFF	Mirror Flash 256 KB Code Block 6	0	
0x091C0000	0x91FFFFF	Mirror Flash 256 KB Code Block 7	0	
0x09200000	0x923FFFF	Mirror Flash 256 KB Code Block 8	1	
0x09240000	0x927FFFF	Mirror Flash 256 KB Code Block 9	1	
0x09280000	0x92BFFFF	Mirror Flash 256 KB Code Block 10	1	
0x092C0000	0x92FFFFF	Mirror Flash 256 KB Code Block 11	1	
0x09300000	0x933FFFF	Mirror Flash 256 KB Code Block 12	1	
0x09340000	0x937FFFF	Mirror Flash 256 KB Code Block 13	1	
0x09380000	0x93BFFFF	Mirror Flash 256 KB Code Block 14	1	

Table 9. Flash memory map (continued)

Start Address	End Address	Description	RWW partition ID	PFLASH Controller assignment (RWR partition ID)(1)
0x093C0000	0x093FFFFF	Mirror Flash 256 KB Code Block 15	5	0
0x09400000	0x0943FFFF	Mirror Flash 256 KB Code Block 16	5	
0x09440000	0x0947FFFF	Mirror Flash 256 KB Code Block 17	5	
0x09480000	0x094BFFFF	Mirror Flash 256 KB Code Block 18	5	
0x094C0000	0x094FFFFF	Mirror Flash 256 KB Code Block 19	5	
0x09500000	0x0953FFFF	Mirror Flash 256 KB Code Block 20	5	
0x09540000	0x0957FFFF	Mirror Flash 256 KB Code Block 21	5	
0x09580000	0x095BFFFF	Mirror Flash 256 KB Code Block 22	5	
0x95C00000	0x0BFFFFFF	Reserved	—	

1. PFLASH 1 is mapped on Flash read port 0 (RWR0) and PFLASH 0 is mapped on Flash read port 1 (RWR1).
2. Some locations in this area are readable. See RWR1_UTEST virtual block memory map table in the Embedded Flash Memory chapter.

5.3 Emulation and debug memory

Table 10 is the memory map for the SPC58xEx/SPC58xGx buddy device.

Table 10. Emulation and debug map

Start address	End address	Description	PFLASH	Size
Buddy device overlay RAM				
0x0C000000	0x0C1FFFFF	BD extended overlay RAM	PFLASH_1	2 MB
0x0C200000	0x0C7FFFFF	Reserved		
Buddy device registers				
0x0C800000	0x0C803FFF	System Integration Unit Lite 2 (SIUL2)	PFLASH_1	16 KB
0x0C804000	0x0C807FFF	LFAST		16 KB
0x0C808000	0x0C80BFFF	JTAGM		16 KB
0x0C80C000	0x0C80FFFF	Reserved		
0x0C810000	0x0C813FFF	BD XBAR		16 KB
0x0C814000	0x0CFFFFFF	Reserved		
Internal overlay RAM				
0x0D000000	0x0D003FFF	Internal overlay RAM 0	PFLASH_0	16 KB
0x0D004000	0x0D007FFF	Internal overlay RAM 1	PFLASH_1	16 KB
0x0D008000	0x0FFFFFFF	Reserved		

5.4 System memory

The following table details the system RAM memory map.

Table 11. RAM memory map

Start Address	End Address	Allocated Size [KB]	Used size	Assigned SRAM controller	Description
0x40000000	0x4005FFFF	512 KB	—	PRAMC_0	System RAM 0
0x40060000	0x4007FFFF		128 KB		
0x40080000	0x400A7FFF	160 KB	160 KB	PRAMC_1	System RAM 1
0x400A8000	0x400A9FFF	8K	8K	PRAMC_2	System RAM 2/ Stand By RAM
0x400AA000	0x400C7FFF	120K	120K		
0x400C8000	0x400E7FFF	128K	128K		
0x400E8000	0x400F7FFF	96 KB	64 KB	PRAMC_3	System RAM 3
0x400F8000	0x400FFFFF		—		
0x40100000	0x407FFFFF	Reserved			

5.5 Local memory

Table 12. Processor local RAM memory map

Start offset	End offset	Description	Allocated size	Used size
0x50000000	0x5000FFFF	I-MEM CPU_0	64 KB	16 KB
0x50010000	0x507FFFFF	Reserved		
0x50800000	0x5080FFFF	D-MEM CPU_0	64 KB	64 KB
0x50810000	0x50FFFFFF	Reserved		
0x51000000	0x5100FFFF	I-MEM CPU_1	64 KB	16 KB
0x51010000	0x517FFFFF	Reserved		
0x51800000	0x5180FFFF	D-MEM CPU_1	64 KB	64 KB
0x51810000	0x51FFFFFF	Reserved		
0x52000000	0x5200FFFF	I-MEM CPU_2	64 KB	16 KB
0x52010000	0x527FFFFF	Reserved		
0x52800000	0x5280FFFF	D-MEM CPU_2	64 KB	32 KB
0x52810000	0x5FFFFFFF	Reserved		

5.6 UTest Flash memory map

Table 13. UTest Flash memory map

Start Address	End Address	Description
UTest Flash (system base address 0x0040_0000 - Flash base address 0x0080_0000)		
0x00000000	0x00000001	Temperature Sensor Calibration A PASS 'Park' password. Read by PASS module to flush real password data from read buffer
0x00000002	0x00000003	Temperature Sensor Calibration B
0x00000004	0x00000005	Temperature Sensor Calibration C
0x00000006	0x00000007	Temperature Sensor Calibration D (Reserved)
0x00000008	0x0000001F	Reserved
0x00000020	0x0000002F	Factory Erase diary Location
0x00000030	0x0000003F	Test Mode Disable Override Passcode
0x00000040	0x0000004F	Test Mode Disable Seal
0x00000050	0x0000005F	Test Mode Disable Block Select Group A
0x00000060	0x0000006F	Test Mode Disable Block Select Group B
0x00000070	0x0000009F	Reserved
0x000000A0	0x000000BF	Unique Chip Identifier (UID)
0x000000C0	0x000000CF	Soft DCF Record Start Address(1)
0x000000D0	0x000000FF	Reserved
0x00000100	0x0000011F	Password to enable the Flash Test Mode when in FA(2)
0x00000120	0x0000013F	JTAG Password(3)
0x00000140	0x0000015F	PASS Password Group 0(3)
0x00000160	0x0000017F	PASS Password Group 1(3)
0x00000180	0x0000019F	PASS Password Group 2(3)
0x000001A0	0x000001BF	PASS Password Group 3(3)
0x000001C0	0x000001DF	Reserved
0x000001E0	0x000001FF	Lifecycle Slot 0 - ST Production
0x00000200	0x0000021F	Lifecycle Slot 1 - Customer Delivery
0x00000220	0x0000023F	Lifecycle Slot 2 - OEM Production
0x00000240	0x0000025F	Lifecycle Slot 3 - IN_FIELD
0x00000260	0x0000027F	Lifecycle Slot 4 - FA
0x00000280	0x0000029F	Customer Single Bit Correction Area(4)
0x000002A0	0x000002BF	Customer Double Bit Correction Area(5)
0x000002C0	0x000002DF	Customer Triple Bit Correction Area(6)
0x000002E0	0x000002FF	Customer Programmable Detection Area
0x00000300	0x00000307	DCF Start Record(7)

Table 13. UTest Flash memory map (continued)

Start Address	End Address	Description
0x00000308	0x0000036F	Reserved
0x00000370	0x00000FFF	UTEST DCF Records(8)
0x00001000	0x00001FFF	Customer OTP data
BAF Block		
0x00004000	0x00407FFFF	BAF Code

1. The effective data is 4 bytes, but because of the new DEC ECC scheme, 16 bytes are still reserved
2. Used to enable the Flash Test Mode when the eFuse protection is not active
3. Access to this location is conditioned by the Life Cycle
4. This area is written by ST during production phase with a specific content aimed to generate single bit errors, on behalf of diagnostic at application level of the ECC error correction path
5. This area is written by ST during production phase with a specific content aimed to generate double bit errors, on behalf of diagnostic at application level of the ECC error detection path
6. This area is written by ST during production phase with a specific content aimed to generate triple bit errors, on behalf of diagnostic at application level of the ECC error correction path
7. The SSCM “SHD_DCF_OFFSET” parameter shall be aligned to the start of this area. Note that the UTest Flash block is seen by the SSCM at the following address: 0x0080_0000. The SSCM START RECORD DCF (see DCF start record in *Chapter 9: Device Configuration Format (DCF) Records*) must always be programmed at the beginning of this area. Because of the new DEC ECC scheme, the 64b following the START DCF record shall still be programmed in order to let the SSCM complete the DCF parsing (o
8. Factory could program some UTest DCF Records during Production test. Consider that the first free location could change. Refer to Sales to retrieve the information on the first free location or directly read the UTest to find it.

5.7 Security module memory

Table 14 is the memory map for the Hardware Security Module (HSM).

Table 14. HSM memory map

Start offset	End offset	Allocated size	Used size
0xA0000000	0xAFFFFFFF	256 MB	Refer to the Security Reference Manual for details about how the HSM memory space is partitioned.

6 Functional safety

6.1 Introduction

The SPC58xEx/SPC58xGx offers a set of features to support applications that need to fulfill functional safety requirements as defined by automotive safety integrity level ASIL D of ISO 26262. This chapter gives an overview of the safety implementation for the SPC58xEx/SPC58xGx, and discusses the operation of the safety mechanisms.

6.2 Safety overview

The SPC58xEx/SPC58xGx has been designed for automotive applications executing safety-relevant tasks that require a fail-silent or fail-indicate MCU. The SPC58xEx/SPC58xGx supports a Fault Tolerant Time Interval (FTTI) of at least 10 ms. Shorter intervals are possible depending on the actual combination of failure detection and signaling mechanisms but the SPC58xEx/SPC58xGx generally supports an FTTI of 10 ms.

6.3 Module categorization

The definition of the SPC58xEx/SPC58xGx safety concept and its validation (particularly the computation of Probabilistic Metric for Random Hardware Failures (PMHF)) has been done by splitting system modules into four categories as shown in *Table 15*.

Table 15. PMHF module categories

Module category	Category abbreviation	Description
Vital system modules	ViMo	All modules that can directly influence the correct operation of the software execution (by influence of the correct function of CPU, RAM, Flash, and the bus interconnect). Examples of ViMo are the core, the cache, the crossbar, and the interrupt controller (INTC), as well as clock or power generation.
Peripheral modules	PeMo	All microcontroller modules involved in I/O operation, specifically including the I/O bridge.
Non-safety modules	NoSaMo	Modules not used to implement any safety related functionality (for example, debug modules), modules which are functionally similar to modules in the categories above but do not provide any embedded safety provisions by default or are not used for safety in a use case scenario. In case they are used in a safety-relevant way, application measures are necessary to ensure their safe operation. Notice that interference-freedom is guaranteed in both cases.
Other support modules	SuMo	These are modules that, although safety-relevant and in principle potential members of the ViMo category, cannot lead to a violation of the safety goal on their own. Typically, these will only have multiple point faults, but not single point faults. Example SuMo's are the RCCU and Clock Monitoring Unit (CMU).

Table 16 shows module classifications for the SPC58xEx/SPC58xGx.

Table 16. Module classification

Classification	Modules
ViMos	Master Safety Core, Master Safety Core D-Cache controller, Master Safety Core I-Cache controller, Local Memory Controllers Master Core, Safety Core D-Cache array, Safety Core I-Cache array, Safety Core Local D-Mem array, Safety Core Local I-Mem array, Peripheral Core (Core_2), I-Cache Peripheral Core (Core_2), Other local memories of Peripheral Core (Core_2), eDMA, DMACHMUX, Concentrator DMA, Concentrator HSM, INTC, XBAR_0 (Computational), SRAM controller, System RAM array, Flash memory controller, Flash memory array, System timer, MC_RGM, Dual PLL Digital, XOSC, MC_CGM, MC_ME, Voltage regulator, MC_PMC, SSCM, PBRIDGE*n*
PeMos	SAR ADC, PSI5, SENT, LINFlexD, DSPI, M_CAN/M_TTCAN, FlexRay, Ethernet, SIUL2, XBAR_1 (Peripheral), Concentrator Ethernet, FlexRay, LFAST, SD ADC

Table 16. Module classification (continued)

Classification	Modules
NoSaMos	Performance Core (Core_1), D-Cache Performance Core (Core_1), I-Cache Performance Core (Core_1), Other local memories of Performance Core (Core_1) External memory interface (LFAST, SIPI), GTM, HSM, Nexus, JTAG, I2C, SEMA42, WKPU, SPU, Overlay RAM, PIT, STANDBY SAR ADC
SuMos	RCCU, Checker Safety Core, System RAM/local memory controllers of Checker Safety Core (overall 4), Checker Safety Core D-Cache controller, Checker Safety Core I-Cache controller, CRC units, Temperature Sensors, IRCOSC, CMUs, HVDs, Bandgap reference, IMA, REG_PROT, SWT, FCCU, MEMUSTCU2

The SPC58xEx/SPC58xGx implements no specific protection of NoSaMos against Common Cause Failures between them.

6.4 System implementation

6.4.1 General concept

In general, safety integrity is achieved in the following ways:

- For the Cores, Interrupt Controllers, DMA and their closely related periphery, safety is ensured by a delayed lockstep approach (refer to *Section 6.4.2: Dual-core lockstep*).
- The safety of storage and of the data path to storage and periphery is ensured by End-to-End ECC (E2E ECC) with address encoding and selected additional measures for

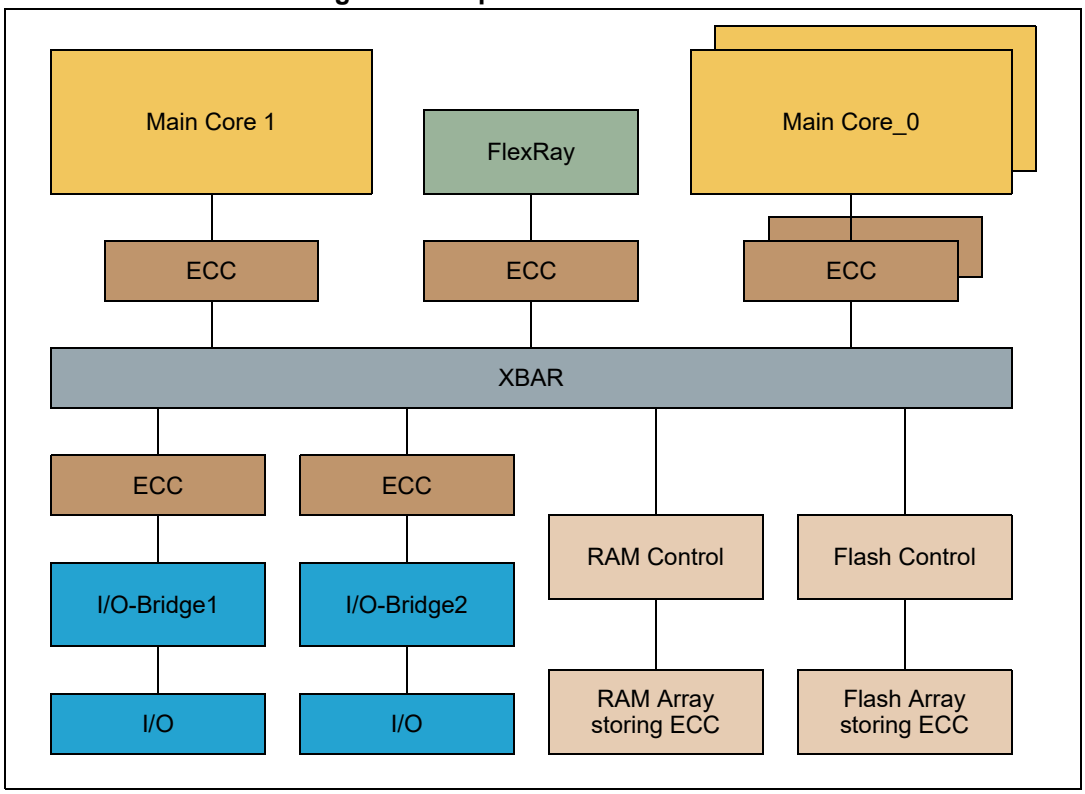
individual modules. For the periphery, end-to-end ends at the I/O bridges (refer to *Section 6.4.1.1: End-to-End protection on data path*).

- Clock and power, generation and distribution, are supervised by dedicated monitors (refer to *Section 6.4.1.2: Clock and power monitoring*).
- The safety of the periphery is ensured by application-level measures (such as connecting one sensor to different I/O modules, sensor validation by sensor fusion, and so on). Hardware supports this application-level redundancy by providing redundant I/O modules connected to different peripheral bridges (PBRIDGE) to maximize the independence between the monitored and monitoring resources (refer to *Section 6.4.1.3: I/O peripherals*, and *Section 6.4.1.4: Communication controllers*).
- MBIST is provided to avoid the accumulation of latent faults in the functional logic as well as the safety integrity mechanisms (refer to *Section 6.4.1.5: Built-In Self-Tests (BIST)*). Dedicated mechanisms are provided to check the availability of safety mechanisms and the functionality of each error reaction path (such as by fake fault injection).
- The Fault Collection and Control Unit (refer to “Fault Collection and Control Unit (FCCU)” chapter for FCCU details) is responsible for collecting and reacting to failure notifications (refer to *Section 6.4.1.7: FCCU and failure monitoring*). MEMU (refer to *Section 6.4.4.4: Memory Error Management Unit (MEMU)*) is responsible for the collection and reporting (to the FCCU) of error events in system memories, and Flash memory, as well as E2E ECC errors caused by the XBAR, or RAM or Flash memory controllers. Error events include ECC corrections and detections as well as faults detected by memory BISTs.
- Common Cause Failures (CCF) are dealt with by a set of measures for both control and avoidance of CCFs spanning system-level approaches (such as temperature and nonfunctional signal monitoring) and back-end techniques (such as isolated silicon areas and routing constraints) (refer to *Section 6.4.3: Common cause failure measures*).

6.4.1.1 End-to-End protection on data path

Connections between XBAR masters and slaves (clients) are denoted as data paths. Data corruption on all data paths between the Safety Core and any client is detected with at least 99% coverage via two main safety mechanisms: data from the replicated cores is encoded using Error Correcting Code (ECC), which is implemented with a single-error correction, double-error detection (SECDED) code with a Hamming distance of 4 and includes coverage of addressing information; control signals and address decoding are monitored to verify the data reaches all of the intended clients, from all possible connections to these clients and the intended operation is performed on the target address. *Figure 8* shows a general view of ECC schema.

Figure 8. Simplified view of E2E ECC



Note: Specific implementation for the SPC58xEx/SPC58xGx varies depending on the special requirements of RAM and Flash memory concerning ECC handling.

ECC bits are generated on writes by XBAR masters (including, but not limited to the Safety Core) and checked on reads. The ECC correction bits are stored alongside the data in Flash memory and RAM so, in principle, no ECC logic is necessary at the memories themselves. For this reason the ECC schema is referred to as End-to-End ECC (E2E ECC) in the following sections. For XBAR slaves, apart from memories, new ECC logic is added as these clients cannot store or produce the ECC correction bits. This resolves the problem where ECC needs to be calculated in real time before entering or exiting the ECC-protected data path. This setup leaves the data path downstream of the ECC units (for example, starting at, and downstream of, the peripheral bridges) unprotected by ECC. To detect corruptions introduced on those unprotected data paths, the PBRIDGEs and peripherals can be used redundantly (refer to *Section 6.4.1.3: I/O peripherals*), or other mechanisms may be used to validate peripheral operation.

The E2E ECC schema provides high detection capabilities against failures affecting the data content of the transaction. The inclusion of the target address in the computation of the redundancy bits (8 ECC bits) does allow the partial detection of addressing faults as well. To reach the desired integrity level, additional dedicated safety mechanisms are implemented in the data path particularly to:

- Improve the detection capability over addressing failures (no/multiple/wrong address selected), considering faults affecting address transmission (from master to client) as well as the decoding of the address.
- Provide coverage for control failures affecting, for example, the type (read vs. write) or size of a transaction.

Though safety mechanisms protecting the XBAR, the RAM controller, or the Flash memory controller are different, they are all based on the feedback of address and control information from the target to the source of the transaction, which is responsible for checking for consistency with respect to the intended transaction. Depending on the portion of the data path covered by the specific safety mechanism, the source can be an XBAR master port rather than the XBAR interface of the RAM or Flash controllers; the target is respectively an XBAR slave port, the RAM array, or the Flash module (refer to “Crossbar Switch (XBAR)”, “Flash memory controller”, and “RAM controller (PRAM)” chapters).

6.4.1.2 Clock and power monitoring

6.4.1.2.1 Clock

Clocks in the SPC58xEx/SPC58xGx are supervised by Clock Monitor Units (CMUs). The CMUs are driven by the IRCOSC (16 MHz internal oscillator) for independent operation from the monitored clocks. CMUs flag errors associated with conditions due to loss of lock, clock out of a programmable bound (over and under frequency), and loss of reference. If a supervised clock leaves the specified range for the device, a signal is sent to the FCCU.

6.4.1.2.2 Power

There are four types of voltage supervisors on the SPC58xEx/SPC58xGx, Low Voltage Detect (LVD), High Voltage Detect (HVD), Upper Voltage Detector (UVD) and Minimum Voltage Detector (MVD) monitors. Safety-relevant voltages are supervised for voltages that are out of these ranges. Since safety relevant voltages have the potential to disable the failure indication mechanisms of the MCU (such as FCCU, pads, and so on) the indication of these errors can be used to cause a direct transition of the MCU into the safe state (reset assertion) (refer to “Power management” and “Power Control Unit (MC_PCU)” chapters for details on power monitoring).

6.4.1.3 I/O peripherals

To allow a safety application to make redundant use of all I/O peripherals, each peripheral has at least two instances (except for the GTM), and each instance is connected to a different PBRIDGE. This means, for example, that if DSPI is provided by the MCU, two DSPI modules (DSPI*n*, DSPI*m*) are included and connected externally through different pins. Internally, DSPI*n* would then be connected to PBRIDGE_0 and DSPI*m* to PBRIDGE_1 and be accessible via different addresses.

The arrangement of I/O peripherals onto two PBRIDGEs, as well as further CCF prevention measures, allow redundant use of peripherals while limiting possible causes of CCFs. Redundant usage includes usage of equivalent peripherals in a replicated way as well as usage of functionally different peripherals in, for example, feedback measurement loops. Comparison of redundant operation is under the responsibility of the application software, and not the safety hardware mechanism.

6.4.1.4 Communication controllers

Communication controllers provide the ability to exchange information with external components and therefore fall under the same safety reasoning as I/O peripherals. Yet we assume that for high bandwidth communication controllers additional software measures are employed that do not require redundant communication peripherals.

The following communication controllers do not contain special safety mechanisms (above what is included in them by their protocol specifications) nor are they duplicated or spread over the PBRIDGE (in contrast to other I/Os, refer to *Section 6.4.1.3: I/O peripherals*):

- FlexRay
- Ethernet
- M_CAN
- M_TTCAN

Typically, software measures for the communication controllers (also called fault-tolerant communication layer) could contain E2E CRC data protection, sender identification, sequence numbering, and an acknowledgment mechanism.

PSI5 represents an exception to the replication of low bandwidth communications controllers. The module itself is replicated, but the two PSI5 modules share a single synchronization bit needed for staggered sync mode. This single bit is a single source of failures for both PSI5 modules if it triggers spurious synchronizations. The failure mode has to be covered by application level countermeasures, which are in any case needed to cover the single transceiver used in such a configuration.

6.4.1.4.1 Disabling of communication controllers

In the event of a dangerous failure, the MCU offers the capability of disabling transmission of individual channels of communication controllers (such as CAN modules and FlexRay). This prevents the transmission of erroneous messages while preserving the capability of communicating over the diagnostic bus. Disabling outputs is controlled by resetting the SIUL2_MSCR*n*[SMC] for the pins that are associated with communication controllers where this feature is needed (refer to the “System Integration Unit Lite2 (SIUL2)” chapter for details on disabling communication signals).

FCCU can drive a fault state on FI[0] with either one of the following two mechanisms:

- Error pin reaction due to a fault
- Software writing FCCU_CFG[FCCU_SET_CLEAR] = 01b

When the FCCU drives FI[0] to a fault state, the SIUL2 disables the output drivers of all pins whose SIUL2_MSCR*n*[SMC] = 0. This disables the possible transmission of erroneous messages. As long as the FCCU drives FI[0] to faulty state, the communication controllers’ Tx pins are disabled and internal pull-up are enabled. As soon as FI[0] is driven to non-faulty state, the communication controllers’ Tx pins are automatically enabled and state of internal pull-up/down restored.

The application should configure SIUL2_MSCR*n*[SMC] for pins that have active mapping of CAN modules (M_CAN, M_TTCAN) or FlexRay functionality and ensure that pin does not stay in undriven state.

Note: The FCCU uses an internal signal to disable the communication controller transmission, so transmission is disabled even when the FCCU cannot drive the FI[0] pin because the pin is configured as GPIO. Also, if FI[0] is used as error input, externally pulling it down will disable the communication controller transmission if this event drives FCCU to a fault state.

Disabling of communication controllers occurs on the Tx pins of the communication controller (for example, M_TTCAN, M_CAN, FlexRay) of the MCU even while the communication controller continues operating. Failed transmissions, when detected by the communication controller, may lead to additional error events.

6.4.1.5 Built-In Self-Tests (BIST)

The term BIST indicates the set of built-in hardware mechanisms that can be used (typically at startup) to avoid the accumulation of latent faults. BIST is a mechanism that permits a device to test itself. On the SPC58xEx/SPC58xGx, BIST is the principal means of meeting the requirement on latent faults as defined by the ISO 26262 standard.

Different types of BIST are implemented in the SPC58xEx/SPC58xGx:

- BIST for memories and logic: managed by the Self-Test Control Unit
- BIST for analog peripherals: requires software intervention

Refer to “Self-Test Control Unit (STCU2)” chapter for further details.

6.4.1.5.1 BIST during boot (off-line)

A device BIST is performed every time the device boots. BIST is performed transparently for the application while the device is still under reset. BIST failure will hold the device in reset, but the BIST will continue running where possible, and inform the system of the failure. Application software can start executing when the BIST finishes successfully without detecting a fault.

The boot time BIST, configured by DCF, comprises:

- Memory BIST (MBIST) for all RAMs and ROM.
- Logic BIST (LBIST) for digital logic.

6.4.1.6 Redundancy Control and Checker Units (RCCUs)

The RCCU compares a set of equivalent input signals provided by different sources and issues an alarm in case of a mismatch. For example, an RCCU compares the output of the checker core against the output of the main core and issues an alarm if they do not match. Signals are arranged in comparison groups, such as, all signals in one group are compared in one RCCU.In case of a compare mismatch the fault stays stable until clearing the FCCU channel status.

6.4.1.7 FCCU and failure monitoring

The FCCU offers a hardware mechanism to aggregate error notifications and a configurable means to bring the device to a safe state. No CPU intervention is required for collection and control operation. Error indications are passed from the individual hardware components to the FCCU where the appropriate action is decided (according to the FCCU configuration).

6.4.1.7.1 External error indication

Failure of the MCU is signaled to one or two pins, FI[0] (ERROR0) and FI[1] (ERROR1). FI[0] can also serve as an error input mechanism (refer to “Fault Collection and Control Unit (FCCU)” chapter, and “FCCU configuration” section of the “Device configuration” chapter, for details on the fault output signals). Both pins can be multiplexed with other functions with different defaults.

The error indication on pins FI[0] and FI[1] is controlled by SIUL2 and FCCU. The mapping of the FCCU output signals on the relevant pins and the associated electrical characteristics are enabled by the SIUL2_MSCR[*n*] register (refer to table “SIUL2_MSCR0 - SIUL2_MSCR511 field description” in the “System Integration Unit Lite2 (SIUL2)” chapter for signal mapping details). Once the FCCU is mapped onto the pin, the logic level is driven by the FCCU.

FI[0] is controlled by the FCCU at startup by default, but the state of FI[1] is like other I/O signals (refer to table “Pin startup and reset states” in chapter “Signal Description” for default states of FI[0] and FI[1]).

Since it is possible to reset FCCU and SIUL2 independently, the behavior of FI[0] and FI[1] is the result of the combined state of SIUL2_MSCR[*n*] and the FCCU. (refer to section “Reset interface” in chapter “Fault Collection and Control Unit (FCCU)” for FCCU reset details, and chapter “Reset and Boot” for the state of the FCCU relative to the system during different reset phases).

During functional reset, the application shall consider only the PAD PB[11] as FCCU error out. The FCCU EOUT1, that is PC[2], is not enabled during functional reset.

The FCCU_STAT[ESTAT] error status flag can be read to show whether the FCCU is in an error state. This flag can be written by software to either a 1 (fault) or 0 (operational) when the FCCU is in operational state.

FCCU_EINOUT(EINO): The value of this register depends on the state of the pad. If the IBE functionality of the SIUL is enabled, the register bit reflects the physical status of the pin as seen by the input buffer. In case the protocol selected for EOUT interface is a toggling one (Dual rail or Time switching), this may not be true due to synchronization in the safety clock domain.

The register reflects the physical status of the pin from application point of view (external status).

FCCU_STAT(PhysicalErrorPin): This register field indicates the actual status of the Error pin as driven by FCCU to the pad based on different FCCU configurations and state of the failure input channels. The value reported in the FCCU_STAT (PhysicalErrorPin) could be different from what is seen at the pad.

6.4.1.7.2 Failure handling

The FCCU is an autonomous module that is responsible for reacting to failure indicators. A different reaction can be configured for each failure source. Overall failure reaction time requires time for detecting, processing, and indicating the error. During this time, the SPC58xEx/SPC58xGx could provide wrong results to the system.

Failure sources include:

- All failure indication signals from modules within the MCU.
- Control logic and signals monitored by the FCCU itself (refer to chapter “Fault Collection and Control Unit (FCCU)”).
- Software-initiated failure indications. For example, software signals the FCCU that it has evidence of a failure. Keep in mind that software can also directly influence the state of the F*n* pins.
- External failure input.

Available failure reactions are:

- Assertion of an interrupt (maskable or non-maskable)
- Resetting the MCU
- Changing the state of the failure indication pins, ERROR*n*
- Disabling the transmission capabilities of communication controllers (FlexRay, CAN, Ethernet. Note that, it is only possible in conjunction with changing the state of the failure indication pins
- No reaction

Software can read the failure source that caused a fault, either before or after a functional reset (the condition indicators are not volatile). Software can also reset the failure, but the external failure indication will stay in failure mode for a configurable minimum time. If necessary, software can also reset the MCU.

6.4.1.7.3 FCCU failure inputs from other modules

Table 17 shows the different failure input signals of the FCCU, how they can be tested, and their default configuration after reset(3).

Table 17. FCCU failure inputs

FCCU channel(1)	Failure	Failure description	Error reaction path check	Default reaction configuration after POR or destructive reset	Reference Manual related information	Recommended Recovery mechanism(2)
0	TEMP_ERROR	Event indication from temperature sensor. The fault is cleared by clearing the status in PMC_dig_EPR_TD, followed by clearing the FCCU channel status, FCCU_RF_Sn[RFSm].	Not testable	No reaction	*Section 67.5.29: Event Pending Register (EPR_TD)*	Correction of violating condition and power reset cycle
1	LVD_ERROR	Voltage out of range indication from LVDs. The fault is cleared by clearing the status in PMC_dig_EPR_VDn, followed by clearing the FCCU channel status, FCCU_RF_Sn[RFSm].	Not testable	No reaction	PMC_Dig registers EPR_LV0 EPR_LV1 *Section 67.4: Memory map/register definition*	Correction of violating condition and power reset cycle

3. All of these faults require a long functional reset.

Table 17. FCCU failure inputs (continued)

FCCU channel[1]	Failure	Failure description	Error reaction path check	Default reaction configuration after POR or destructive reset	Reference Manual related information	Recommended Recovery mechanism[2]
2	HVD_ERROR	Voltage out of range indication from HVDs. The fault is cleared by clearing the status in PMC_dig_EPR_VDn, followed by clearing the FCCU channel status, FCCU_RF_Sn[RFSm].	Not testable	No reaction	PMC_Dig registers EPR_HV0 EPR_HV1 *Section 67.4: Memory map/register definition*	Correction of violating condition and power reset cycle
3	DPMC_DCF_SAFETY_ERR	Digital PMC initialization error during DCF data load. The fault is cleared by clearing the FCCU channel status, FCCU_RF_Sn[RFSm]. The status gets cleared if the fault is not persistent"	Not testable	No reaction	*Chapter 67: Power management controller digital interface (PMC_Dig)*	POR, Destructive Reset
4	DPMC_VD_BIST	Digital PMC Voltage detector BIST	Feedback loop via FCCU Fake bit.	No reaction	*Chapter 67: Power management controller digital interface (PMC_Dig)*	POR, Destructive Reset
5	SSCM_XFER_FLASH_ERR_ MEMORY_ERR	SSCM transfer error (during the SSCM and STCU DCF loading), or Flash memory initialization error, or Memory error.	Not testable (but wrong DCF configuration or spurious data from Flash can trigger other failures)	No reaction	*Table 1736: SSCM memory map*	POR, Destructive Reset
6	STCU_UF	STCU2 unrecoverable fault indication during self-test. It is simultaneously reported to the corresponding fault line to the FCCU. Fault is cleared by clearing STCU2_ERR_STAT[UFSF] followed by clearing the FCCU channel status, FCCU_RF_Sn[RFSm].	Feedback loop via FCCU Fake bit	Destructive reset by MC_RGM directly during offline self-test	*Section 90.5.10: STCU2 Error Register (STCU_ERR_STAT)* *Section 64.3.1.2: 'Destructive' Event Reset Disable Register (RGM_DERD)*	POR, Destructive reset. During offline test, assertion of this fault will automatically cause a destructive reset request by MC_RGM.

Table 17. FCCU failure inputs (continued)

FCCU channel[1]	Failure	Failure description	Error reaction path check	Default reaction configuration after POR or destructive reset	Reference Manual related information	Recommended Recovery mechanism[2]
7	STCU_RF	STCU2 recoverable fault indication during self-test. In the STCU2, self-test failures can be configured as recoverable or unrecoverable faults which would then lead to assertion of the corresponding fault line to the FCCU. Fault is cleared by clearing STCU2_ERR_STAT [RFSF] followed by clearing the FCCU channel status, FCCU_RF_Sn[RFSm].	Feedback loop via FCCU Fake bit	No reaction	*Section 90.5.10: STCU2 Error Register (STCU_ERR_STAT)*	POR, Destructive Reset
8	STCU_LMBIST_USR_ERR	Activation of L/MBIST control during application mode. Fault is cleared by clearing the FCCU channel status, FCCU_RF_Sn[RFSm]. The status will clear if the fault is not persistent.	Feedback loop via FCCU Fake bit	No reaction	—	POR, Destructive Reset
9	SPURIOUS_DEBUG_SSCM_ACTIVATION	JTAG, NPC or Debug functionality moved out of reset or SSCM activation during application mode. Fault is cleared by clearing the FCCU channel status, FCCU_RF_Sn[RFSm].	Feedback loop via FCCU Fake bit	No reaction	—	Long Functional Reset
10	RCCU_Core 0_ALARM	Indication of lockstep out of sync due to a fault for Core_0 or associated DSMC. Fault is cleared by clearing the FCCU channel status, FCCU_RF_Sn[RFSm].	Feedback loop via FCCU Fake bit	No reaction	*Section 6.4.1.6: Redundancy Control and Checker Units (RCCUs)*	Functional Reset
11	—	—	—	—	—	—
12	RCCU_Core 2_ALARM	Indication of lockstep out of sync due to a fault for Core_2 or associated DSMC. Fault is cleared by clearing the FCCU channel status, FCCU_RF_Sn[RFSm].	Feedback loop via FCCU Fake bit	No reaction	*Section 6.4.1.6: Redundancy Control and Checker Units (RCCUs)*	Functional Reset

Table 17. FCCU failure inputs (continued)

FCCU channel[1]	Failure	Failure description	Error reaction path check	Default reaction configuration after POR or destructive reset	Reference Manual related information	Recommended Recovery mechanism[2]
13	—	—	—	—	—	—
14	RCCU_INTC 1 _ALARM	INTC_1 out of sync during lockstep. Fault is cleared by clearing the FCCU channel status, FCCU_RF_Sn[RFSm].	Feedback loop via FCCU Fake bit	No reaction	*Section 6.4.1.6: Redundancy Control and Checker Units (RCCUs)*	Functional Reset
15	RCCU_DMA 0	DMA_0 out of sync during lockstep. Fault is cleared by clearing the FCCU channel status, FCCU_RF_Sn[RFSm].	Feedback loop via FCCU Fake bit.	No reaction	*Section 6.4.1.6: Redundancy Control and Checker Units (RCCUs)*	Functional Reset
16	RCCU_DMA 1	DMA_1 out of sync during lockstep. Fault is cleared by clearing the FCCU channel status, FCCU_RF_Sn[RFSm].	Feedback loop via FCCU Fake bit.	No reaction	*Section 6.4.1.6: Redundancy Control and Checker Units (RCCUs)*	Functional Reset
17	SWT3_ ResetReq	Reset request from SWT_3. The fault is cleared by clearing FCCU_RF_Sn[RFSm] after the status has been cleared inside SWT3 (after a functional reset).	Test in software by enabling the SWT and failing to service it.	No reaction	*Chapter 46: Software Watchdog Timer (SWT)*	Functional Reset
18	SWT2 _ResetReq	Reset request from SWT_2. The fault is cleared by clearing FCCU_RF_Sn[RFSm] after the status has been cleared inside SWT2 (after a functional reset).	Test in software by enabling the SWT and failing to service it.	Long Function Reset	*Chapter 46: Software Watchdog Timer (SWT)*	Functional Reset
19	SWT1 _ResetReq	Reset request from SWT_1. The fault is cleared by clearing FCCU_RF_Sn[RFSm] after the status has been cleared inside SWT1 (after a functional reset).	Test in software by enabling the SWT and failing to service it.	No reaction	*Chapter 46: Software Watchdog Timer (SWT)*	Functional Reset
20	SWT0 _ResetReq	Reset request from SWT_0. The fault is cleared by clearing FCCU_RF_Sn[RFSm] after the status has been cleared inside SWT0 (after a functional reset).	Test in software by enabling the SWT and failing to service it.	No reaction	*Chapter 46: Software Watchdog Timer (SWT)*	Functional Reset

Table 17. FCCU failure inputs (continued)

FCCU channel[1]	Failure	Failure description	Error reaction path check	Default reaction configuration after POR or destructive reset	Reference Manual related information	Recommended Recovery mechanism[2]
21	MEMU_RAM_CE	Indication by MEMU of occurrence of correctable errors in system RAMs. The fault is cleared by clearing MEMU_ERR_FLAG[SR_CE], followed by clearing the FCCU channel status, FCCU_RF_Sn[RFSm].	Injection of RAM ECC error via ECC bit access and later reading	No reaction	*Section 87.6.2.2: Error Flag register (ERR_FLAG)*	
22	MEMU_RAM_UCE	Indication by MEMU of occurrence of uncorrectable errors in system RAMs. The fault is cleared by the clearing MEMU_ERR_FLAG[SR_UCE], followed by clearing the FCCU channel status, FCCU_RF_Sn[RFSm].	Injection of RAM ECC error via ECC bit access and later reading	No reaction	*Section 87.6.2.2: Error Flag register (ERR_FLAG)*	—
23	MEMU_RAM_OV	Indication of overflow of system RAM ECC buffer in MEMU. The fault is cleared by the clearing MEMU_ERR_FLAG[SR_EBO], followed by clearing the FCCU channel status, FCCU_RF_Sn[RFSm].	Functional test	No reaction	*Section 87.6.2.2: Error Flag register (ERR_FLAG)*	—
24	MEMU_PER_CE	Indication by MEMU of the occurrence of correctable errors in peripheral RAMs. The fault is cleared by clearing MEMU_ERR_FLAG[PR_CE], followed by clearing the FCCU channel status, FCCU_RF_Sn[RFSm].	Functional test	No reaction	*Section 87.6.2.2: Error Flag register (ERR_FLAG)*	—
25	MEMU_PER_UCE	Indication by MEMU of the occurrence of uncorrectable errors in peripheral RAMs. The fault is cleared by clearing MEMU_ERR_FLAG[PR_UCE], followed by clearing the FCCU channel status, FCCU_RF_Sn[RFSm]	Functional test	No reaction	*Section 87.6.2.2: Error Flag register (ERR_FLAG)*	—

Table 17. FCCU failure inputs (continued)

FCCU channel[1]	Failure	Failure description	Error reaction path check	Default reaction configuration after POR or destructive reset	Reference Manual related information	Recommended Recovery mechanism[2]
26	MEMU_PER_OV	Indication of overflow of peripheral RAM ECC buffer in MEMU. The fault is cleared by clearing MEMU_ERR_FLAG[PR_EBO], followed by clearing the FCCU channel status, FCCU_RF_Sn[RFSm]	Functional test	No reaction	*Section 87.6.2.2: Error Flag register (ERR_FLAG)*	—
27	MEMU_FLS_CE	Indication by MEMU of occurrence of correctable errors in Flash memory. The fault is cleared by the clearing MEMU_ERR_FLAG[F_CE], followed by clearing the FCCU channel status, FCCU_RF_Sn[RFSm].	Reading preprogrammed correctable ECC error from UTEST Flash memory	No reaction	*Section 87.6.2.2: Error Flag register (ERR_FLAG)*	—
28	MEMU_FLS_UCE	Indication by MEMU of occurrence of uncorrectable errors in Flash memory. The fault is cleared by the clearing MEMU_ERR_FLAG[F_UCE], followed by clearing the FCCU channel status, FCCU_RF_Sn[RFSm].	Reading preprogrammed correctable ECC error from UTEST Flash memory	No reaction	*Section 87.6.2.2: Error Flag register (ERR_FLAG)*	—
29	MEMU_FLS_OV	Indication of overflow of Flash memory ECC buffer in MEMU. The fault is cleared by the clearing MEMU_ERR_FLAG[F_EBO], followed by clearing the FCCU channel status, FCCU_RF_Sn[RFSm].	Functional test	No reaction	*Section 87.6.2.2: Error Flag register (ERR_FLAG)*	—
30	IMA_SoC	IMA SoC Active The fault is cleared by disabling IMA (IMA.SLCT.R = 0x0), followed by clearing the FCCU channel status, FCCU_RF_Sn[RFSm].	IMA read/write	No reaction	*Section 88.3: Accessing memory via the IMA module*	Functional Reset

Table 17. FCCU failure inputs (continued)

FCCU channel[1]	Failure	Failure description	Error reaction path check	Default reaction configuration after POR or destructive reset	Reference Manual related information	Recommended Recovery mechanism[2]
31	SMPU_MONITORXBAR 0	Indication of fault in SMPU_0 logic fault resulting in an incorrectly refused access, but failure to indicate the error to the requesting bus master. The fault is cleared by clearing FCCU_RF_Sn[RFSm].	Not testable	No reaction	*Chapter 21: System Memory Protection Unit (SMPU)*	POR, Destructive reset
32	SMPU_MONITOR XBAR 1	Indication of fault in SMPU_1 logic fault resulting in an incorrectly refused access, but failure to indicate the error to the requesting bus master. The fault is cleared by clearing FCCU_RF_Sn[RFSm].	Not testable	No reaction	*Chapter 21: System Memory Protection Unit (SMPU)*	POR, Destructive reset
33	SMPU ERROR XBAR 0	Indication of the SMPU_0 that it prevented an unauthorized access. The fault is cleared by clearing the FCCU channel status, FCCU_RF_Sn[RFSm].	Willfully causing an access to an unauthorized location	No reaction	*Chapter 21: System Memory Protection Unit (SMPU)*	—
34	SMPU ERROR XBAR 1	Indication of the SMPU_1 that it prevented an unauthorized access. The fault is cleared by clearing the FCCU channel status, FCCU_RF_Sn[RFSm].	Willfully causing an access to an unauthorized location	No reaction	*Chapter 21: System Memory Protection Unit (SMPU)*	—

Table 17. FCCU failure inputs (continued)

FCCU channel[1]	Failure	Failure description	Error reaction path check	Default reaction configuration after POR or destructive reset	Reference Manual related information	Recommended Recovery mechanism[2]
35	CORE 0 I-MEM address ERR	Indication of an addressing error (address, CS and WE feedback) inside dedicated memories of Core_0 I-MEM. The fault is cleared by clearing FCCU_RF_Sn[RFSm].	Feedback loop via FCCU Fake bit	No reaction	*Section 35.5.2: Transaction monitor*	—
36	CORE 0 D-MEM address ERR	Indication of an addressing error (address, CS and WE feedback) inside dedicated memories of Core_0 D-MEM. The fault is cleared by clearing FCCU_RF_Sn[RFSm].	Feedback loop via FCCU Fake bit	No reaction	*Section 35.5.2: Transaction monitor*	—
37	CORE 0 I-Cache address ERR	Indication of an addressing error (adddress, CS and WE feedback) inside dedicated memories of Core_0 I-Cache. The fault is cleared by clearing FCCU_RF_Sn[RFSm].	Feedback loop via FCCU Fake bit	No reaction	*Section 35.5.2: Transaction monitor*	—
38	CORE 0 D-Cache address ERR	Indication of an addressing error (adddress, CS and WE feedback) inside dedicated memories of Core_0 D-Cache. The fault is cleared by clearing FCCU_RF_Sn[RFSm].	Feedback loop via FCCU Fake bit	No reaction	*Section 35.5.2: Transaction monitor*	—
39	CORE 1 I-MEM address ERR	Indication of an addressing error (adddress, CS and WE feedback) inside dedicated memories of Core_1 I-MEM. The fault is cleared by clearing FCCU_RF_Sn[RFSm].	Feedback loop via FCCU Fake bit	No reaction	*Section 35.5.2: Transaction monitor*	—

Table 17. FCCU failure inputs (continued)

FCCU channel[1]	Failure	Failure description	Error reaction path check	Default reaction configuration after POR or destructive reset	Reference Manual related information	Recommended Recovery mechanism[2]
40	CORE 1 D-MEM address ERR	Indication of an addressing error (adddress, CS and WE feedback) inside dedicated memories of Core_1 D-MEM. The fault is cleared by clearing FCCU_RF_Sn[RFSm].	Feedback loop via FCCU Fake bit	No reaction	*Section 35.5.2: Transaction monitor*	—
41	CORE 1 I-Cache address ERR	Indication of an addressing error (adddress, CS and WE feedback) inside dedicated memories of Core_1 I-Cache. The fault is cleared by clearing FCCU_RF_Sn[RFSm].	Feedback loop via FCCU Fake bit	No reaction	*Section 35.5.2: Transaction monitor*	—
42	CORE 1 D-Cache address ERR	Indication of an addressing error (address, CS and WE feedback) inside dedicated memories of Core_1 D-Cache. The fault is cleared by clearing FCCU_RF_Sn[RFSm].	Feedback loop via FCCU Fake bit	No reaction	*Section 35.5.2: Transaction monitor*	—
43	CORE 2 I-MEM address ERR	Indication of an addressing error (address, CS and WE feedback) inside dedicated memories of Core_2 I-MEM. The fault is cleared by clearing FCCU_RF_Sn[RFSm].	Feedback loop via FCCU Fake bit	No reaction	*Section 35.5.2: Transaction monitor*	—
44	CORE 2 D-MEM address ERR	Indication of an addressing error (address, CS and WE feedback) inside dedicated memories of Core_2 D-MEM. The fault is cleared by clearing FCCU_RF_Sn[RFSm].	Feedback loop via FCCU Fake bit	No reaction	*Section 35.5.2: Transaction monitor*	—

Table 17. FCCU failure inputs (continued)

FCCU channel[1]	Failure	Failure description	Error reaction path check	Default reaction configuration after POR or destructive reset	Reference Manual related information	Recommended Recovery mechanism[2]
45	CORE 2 I-Cache address ERR	Indication of an addressing error (address, CS and WE feedback) inside dedicated memories of Core_2 I-Cache. The fault is cleared by clearing FCCU_RF_Sn[RFSm].	Feedback loop via FCCU Fake bit	No reaction	*Section 35.5.2: Transaction monitor*	—
46	—	—	—	—	—	—
47	DMA 0 TCD address ERR	Error while reading a DMA 0 TCD. The fault is cleared by clearing FCCU_RF_Sn[RFSm].	Feedback loop via FCCU Fake bit	No reaction	—	—
48	DMA 1 TCD address ERR	Error while reading a DMA 1 TCD. The fault is cleared by clearing FCCU_RF_Sn[RFSm].	Feedback loop via FCCU Fake bit	No reaction	—	—
49	FM_PLL_0	PLL0 loss of lock. In order for this fault source to be asserted in the event of loss of lock, the Loss of Lock Interrupt needs to be enabled by writing (PLLDIG_PLL0CR[LOLIE]) = 1. The fault is cleared by clearing PLLDIG_PLL0SR[LOLF], followed by clearing FCCU_RF_Sn[RFSm].	Forced loss by PLL reconfiguration	No reaction	*Section 28.5.2.2: PLL0 status register (PLL0SR)*	Application integrity checks
50	FM_PLL_1	PLL1 loss of lock. In order for this fault source to be asserted in the event of loss of lock, the Loss of Lock Interrupt needs to be enabled by writing (PLLDIG_PLL0CR[LOLIE]) = 1. The fault is cleared by clearing PLLDIG_PLL0SR[LOLF], followed by clearing FCCU_RF_Sn[RFSm].	Forced loss by PLL reconfiguration	No reaction	*Section 28.5.2.5: PLL1 status register (PLL1SR)*	Application integrity checks

Table 17. FCCU failure inputs (continued)

FCCU channel[1]	Failure	Failure description	Error reaction path check	Default reaction configuration after POR or destructive reset	Reference Manual related information	Recommended Recovery mechanism[2]
51	CMU_0_OSC	Loss of XOSC clock. The fault is cleared by clearing CMU_ISR[OLRI] of CMU_0, followed by clearing the FCCU channel status, FCCU_RF_Sn[RFSm].	Not testable	No reaction	*Section 29.4.1.5: CMU Interrupt Status Register (CMU_ISR)*	Functional Reset
52	CMU_0_PLL	SYSCLK frequency out of range. The fault is cleared by clearing either CMU_ISR[FHHI] or CMU_ISR[FLLI] (or both) of CMU_0, followed by clearing the FCCU channel status, FCCU_RF_Sn[RFSm]	Test by mis-configuration	No reaction	*Section 29.4.1.5: CMU Interrupt Status Register (CMU_ISR)*	Functional Reset
53	CMU_Platform	Exception from monitoring internal clocks by the CMU_Platform. The fault is cleared by clearing either CMU_ISR[FHHI] or CMU_ISR[FLLI] (or both) of CMU_n, followed by clearing the FCCU channel status, FCCU_RF_Sn[RFSm]	Test by mis-configuration	No reaction	*Section 29.4.1.5: CMU Interrupt Status Register (CMU_ISR)*	Functional Reset
54	CMU_other	Exception from monitoring internal clocks by other CMU. The fault is cleared by clearing either CMU_ISR[FHHI] or CMU_ISR[FLLI] (or both) of CMU_n, followed by clearing the FCCU channel status, FCCU_RF_Sn[RFSm]	Test by mis-configuration	No reaction	*Section 29.4.1.5: CMU Interrupt Status Register (CMU_ISR)*	Functional Reset
55	GASKET XBAR 2 BACKDOOR TO XBAR0	Indication of a hardware fault resulting in corrupted transaction through the peripheral XBAR or gaskets	Not testable	No reaction	*Chapter 19: Crossbar Integrity Checker (XBIC)*	POR, Destructive reset

Table 17. FCCU failure inputs (continued)

FCCU channel[1]	Failure	Failure description	Error reaction path check	Default reaction configuration after POR or destructive reset	Reference Manual related information	Recommended Recovery mechanism[2]
56	XBAR 1 XBIC	Indication of a hardware fault resulting in corrupted transaction through the peripheral XBAR or gaskets	Not testable	No reaction	*Chapter 19: Crossbar Integrity Checker (XBIC)*	POR, Destructive reset
57	XBAR 0 XBIC	Indication of a hardware fault resulting in corrupted transaction through the computational XBAR or gaskets	Not testable	No reaction	*Chapter 19: Crossbar Integrity Checker (XBIC)*	POR, Destructive reset
58	EDC_ECC _FLASH 0	EDC after ECC for FLASH Array. Indication of a hardware fault in the ECC of the Flash memory.	Feedback loop via FCCU Fake bit or test by accessing to Flash address of the Customer Programmable Detection area in the UTEST block	No reaction	*Section 36.5.14: Safety considerations*	POR, Destructive reset
59	EDC_ECC _FLASH_C 0	EDC after ECC for FLASH Controller. Indication of hardware fault in the Flash memory controller resulting in corrupted ECC detection / correction.	Not testable	No reaction	*Section 36.5.14: Safety considerations*	POR, Destructive reset
60	ENC_ERR _FLASH 0	Flash Encoding Error. Indication of hardware fault resulting in corrupted Flash memory access.	Test by accessing to Flash address of the Customer Programmable Detection area in the UTEST block	No reaction	*Section 36.5.14: Safety considerations*	Application dependent

Table 17. FCCU failure inputs (continued)

FCCU channel[1]	Failure	Failure description	Error reaction path check	Default reaction configuration after POR or destructive reset	Reference Manual related information	Recommended Recovery mechanism[2]
61	ADDR_FDBK_ERR_FLASH_C 0	Indication of an addressing error in the Flash memory controller resulting in corrupted Flash memory access.	Test by accessing to Flash address of the Customer Programmable Detection area in the UTEST block	No reaction	*Section 36.5.14: Safety considerations*	Application dependent
62	SAFE_CAL_ERR 0	alarm indicating the Flash controller detected a transaction monitor mismatch in the pseudo-replicated calibration evaluation hardware (PFLASHC_0).	Not testable	No reaction	—	—
63	FLASH_REF_ERR	FLASH read reference error	Test by accessing to Flash address of the Customer Programmable Detection area in the UTEST block.	No reaction	*Section 36.5.14: Safety considerations*	POR, Destructive reset
64	EDC_ECC_FLASH 1	EDC after ECC for FLASH Array. Indication of a hardware fault in the ECC of the Flash memory resulting in corrupted ECC detection/correction.	Feedback loop via FCCU Fake bit or test by accessing to Flash address of the Customer Programmable Detection area in the UTEST block.	No reaction	*Section 36.5.14: Safety considerations*	POR, Destructive reset

Table 17. FCCU failure inputs (continued)

FCCU channel[1]	Failure	Failure description	Error reaction path check	Default reaction configuration after POR or destructive reset	Reference Manual related information	Recommended Recovery mechanism[2]
65	EDC_ECC_FLASH_C 1	EDC after ECC for FLASH Controller. Indication of hardware fault in the Flash memory controller resulting in corrupted ECC detection / correction.	Not testable	No reaction	*Section 36.5.14: Safety considerations*	POR, Destructive reset
66	ENC_ERR_FLASH 1	Flash Encoding Error. Indication of hardware fault resulting in corrupted Flash memory access.	Test by access to Flash address in Customer Programmable Detection area in the UTEST block	No reaction	*Section 36.5.14: Safety considerations*	—
67	ADDR_FDBK_ERR_FLASH_C 1	Indication of an addressing error in the Flash memory controller resulting in corrupted Flash memory access.	Test by access to Flash address in Customer Programmable Detection area in the UTEST block	No reaction	*Section 36.5.14: Safety considerations*	—
68	SAFE_CAL_ERR 1	alarm indicating the Flash controller detected a transaction monitor mismatch in the pseudo-replicated calibration evaluation hardware (PFLASHC_1).	Not testable	No reaction	—	—
69	ON_CAL_RAM_ERR	alarm indicating the Flash controller detected a transaction monitor mismatch when compared to the on-chip calibration RAM safety feedback outputs.	Not testable	No reaction	—	—

Table 17. FCCU failure inputs (continued)

FCCU channel[1]	Failure	Failure description	Error reaction path check	Default reaction configuration after POR or destructive reset	Reference Manual related information	Recommended Recovery mechanism[2]
70	ADDR_FDBK_ERR_PRAM 0 RAM_LWB_ERR	Indication of an addressing error in system RAM or a write error in the RAM controller resulting in corrupted RAM access or a RAM Late-Write Buffer mismatch.	Not testable	No reaction	*Chapter 35: Platform RAM controller AHB (PRAMC_AHB)*	—
71	EDC_ERR _PRAM 0	Indication of a fault in the controller PRAM of the system RAM resulting in a potentially corrupted RAM word during RMW access or stale word during read.	Not testable	No reaction	*Chapter 35: Platform RAM controller AHB (PRAMC_AHB)*	—
72	ADDR_FDBK_ERR_PRAM 1 RAM_LWB_ERR	Indication of an addressing error in system RAM or a write error in the RAM controller resulting in corrupted RAM access.	Not testable	No reaction	*Chapter 35: Platform RAM controller AHB (PRAMC_AHB)*	—
73	EDC_ERR _PRAM 1	Indication of a fault in the controller PRAM of the system RAM resulting in a potentially corrupted RAM word during RMW access or stale word during read.	Not testable	No reaction	*Chapter 35: Platform RAM controller AHB (PRAMC_AHB)*	—
74	ADDR_FDBK_ERR_PRAM 2 RAM_LWB_ERR	Indication of an addressing error in system RAM or a write error in the RAM controller resulting in corrupted RAM access.	Not testable	No reaction	*Chapter 35: Platform RAM controller AHB (PRAMC_AHB)*	—

Table 17. FCCU failure inputs (continued)

FCCU channel[1]	Failure	Failure description	Error reaction path check	Default reaction configuration after POR or destructive reset	Reference Manual related information	Recommended Recovery mechanism[2]
75	EDC_ERR_PRAM 2	Indication of a fault in the controller PRAM of the system RAM resulting in a potentially corrupted RAM word during RMW access or stale word during read.	Not testable	No reaction	*Chapter 35: Platform RAM controller AHB (PRAMC_AHB)*	—
76	ADDR_FDBK_ERR_PRAM 3 RAM_LWB_ERR	Indication of an addressing error in system RAM or a write error in the RAM controller resulting in corrupted RAM access.	Not testable	No reaction	*Chapter 35: Platform RAM controller AHB (PRAMC_AHB)*	—
77	EDC_ERR_PRAM 3	Indication of a fault in the controller PRAM of the system RAM resulting in a potentially corrupted RAM word during RMW access or stale word during read.	Not testable	No reaction	*Chapter 35: Platform RAM controller AHB (PRAMC_AHB)*	—
78	DFT1	Test circuitry Group 1 activation.	Not testable	No reaction	—	POR, Destructive reset
79	DFT2[3]	Test circuitry Group 2 activation.	Not testable	No reaction	—	POR, Destructive reset
80	DFT3	Test circuitry Group 3 activation.	Not testable	No reaction	—	POR, Destructive reset
81	DFT4	Test circuitry Group 4 activation.	Not testable	No reaction	—	POR, Destructive reset
82	Core 0 Exception	Core exception indication. The fault is cleared by clearing the status in MCSR register of the Safety core followed by clearing the FCCU channel status, FCCU_RF_Sn[RFSm].	Willfully cause an appropriate exception(for example, access a non-existing address)	No reaction	*Section 16.7: Exceptions*	—

Table 17. FCCU failure inputs (continued)

FCCU channel[1]	Failure	Failure description	Error reaction path check	Default reaction configuration after POR or destructive reset	Reference Manual related information	Recommended Recovery mechanism[2]
83	Core 1 Exception	Core exception indication. The fault is cleared by clearing the status in MCSR register of the Safety core followed by clearing the FCCU channel status, FCCU_RF_Sn[RFSm].	Willfully cause an appropriate exception(for example, access a non-existing address)	No reaction	*Section 16.7: Exceptions*	—
84	Core 2 Exception	Core exception indication. The fault is cleared by clearing the status in MCSR register of the Safety core followed by clearing the FCCU channel status, FCCU_RF_Sn[RFSm].	Willfully cause an appropriate exception (for example, access a non-existing address)	No reaction	*Section 16.7: Exceptions*	—
85	Checker Core 0 disable alarm	Indication of disablement of lockstep mode	Feedback loop via FCCU Fake bit	No reaction	*Section 6.4.2: Dual-core lockstep*	POR, Destructive reset
86	AIC_0_ALARM	Error detected by the PBRIDGE monitor during a read/write transaction through the PBRIDGE	Feedback loop via FCCU Fake bit	No reaction	*Chapter 20: Peripheral Bridge (PBRIDGE)*	—
87	PBRIDGE_0_e2eEDC_ERR	Error detected by the e2eEDC during a read/write transaction through the PBRIDGE	Feedback loop via FCCU Fake bit	No reaction	*Chapter 20: Peripheral Bridge (PBRIDGE)*	—
88	AIC_1_ALARM	Error detected by the PBRIDGE monitor during a read/write transaction through the PBRIDGE	Feedback loop via FCCU Fake bit	No reaction	*Chapter 20: Peripheral Bridge (PBRIDGE)*	—

Table 17. FCCU failure inputs (continued)

FCCU channel[1]	Failure	Failure description	Error reaction path check	Default reaction configuration after POR or destructive reset	Reference Manual related information	Recommended Recovery mechanism[2]
89	PBRIDGE_1_e2eEDC_ERR	Error detected by the e2eEDC during a read/write transaction through the PBRIDGE	Feedback loop via FCCU Fake bit	No reaction	*Chapter 20: Peripheral Bridge (PBRIDGE)*	—
90	AIC_2_ALARM	Error detected by the PBRIDGE monitor during a read/write transaction through the PBRIDGE	Feedback loop via FCCU Fake bit	No reaction	*Chapter 20: Peripheral Bridge (PBRIDGE)*	—
91	PBRIDGE_2_e2eEDC_ERR	Error detected by the e2eEDC during a read/write transaction through the PBRIDGE	Feedback loop via FCCU Fake bit	No reaction	*Chapter 20: Peripheral Bridge (PBRIDGE)*	—
92	Xbic_Concentrator_0_alarm_Alarm	XBIC Concentrator 0	Not testable	No reaction	—	—
93	Monitor_Concentrator0_edc_Alarm_Alarm	EDC after ECC error alarm for any master in concentrator 0	Feedback loop via FCCU Fake bit.	No reaction	—	—

Table 17. FCCU failure inputs (continued)

FCCU channel[1]	Failure	Failure description	Error reaction path check	Default reaction configuration after POR or destructive reset	Reference Manual related information	Recommended Recovery mechanism[2]
94	Safe Mode[4]	Indication of safe mode request assertion from MC_RGM. The event is also captured in MC_ME and available as status for crosscheck. Status should be cleared first in the MC_RGM_FES, then followed by clearing the FCCU channel status, FCCU_RF_Sn[RFSm].	Not testable	No reaction	*Chapter 64: Reset Generation Module (MC_RGM)*	—
95	Compensation Disable	Indication of undesired deactivation of pad compensation. Fault may affect a distorted transmissions on I/Os associated. The indication is cleared by clearing the FCCU channel status, FCCU_RF_Sn[RFSm][5]	Not testable	No reaction	—	—
96	EIN_ERR	Error input pin (FI[0], external of the MCU) signalling an error condition. The fault is cleared by clearing the fault driving FI[0] followed by clearing the FCCU channel status, FCCU_RF_Sn[RFSm].	External activation of EIN. External entity has to pull the error input low.	No reaction	*Chapter 89: Fault Collection and Control Unit (FCCU)*	—
97	Reserved - Fake fault error injection should be avoided.					
98	Reserved - Fake fault error injection should be avoided.					
99	Monitor_Concentrator_0_alarm	Error in any frequency gasket in concentrator 0	Feedback loop via FCCU Fake bit	No reaction	—	—

Table 17. FCCU failure inputs (continued)

FCCU channel[1]	Failure	Failure description	Error reaction path check	Default reaction configuration after POR or destructive reset	Reference Manual related information	Recommended Recovery mechanism[2]
100	SWT3_first_timeout	First Timeout event which is also mapped to INTC	SW test by not resetting the SWT during startup	No reaction	*Chapter 46: Software Watchdog Timer (SWT)*	—
101	SWT2_first_timeout	First Timeout event which is also mapped to INTC	SW test by not resetting the SWT during startup	No reaction	*Chapter 46: Software Watchdog Timer (SWT)*	—
102	SWT1_first_timeout	First Timeout event which is also mapped to INTC	SW test by not resetting the SWT during startup	No reaction	*Chapter 46: Software Watchdog Timer (SWT)*	—
103	SWT0_first_timeout	First Timeout event which is also mapped to INTC	SW test by not resetting the SWT during startup	No reaction	*Chapter 46: Software Watchdog Timer (SWT)*	—
104	Monitor_Concentrator1_ma_Alarm	EDC after ECC error alarm for SIPI 1	Feedback loop via FCCU Fake bit	No reaction	—	—
105	Monitor_Concentrator1_mb_Alarm	EDC after ECC error alarm for Flexray 0	Feedback loop via FCCU Fake bit	No reaction	—	—

Table 17. FCCU failure inputs (continued)

FCCU channel[1]	Failure	Failure description	Error reaction path check	Default reaction configuration after POR or destructive reset	Reference Manual related information	Recommended Recovery mechanism[2]
106	Monitor_Concentrator1_mc_Alarm	EDC after ECC error alarm for Ether 0	Feedback loop via FCCU Fake bit	No reaction	—	—
107	Monitor_Concentrator2_ma_Alarm	EDC after ECC error alarm for Ether 1	Feedback loop via FCCU Fake bit	No reaction	—	—
108	Monitor_Concentrator1_Alarm	Error in the frequency gasket in the concentrator 1	Feedback loop via FCCU Fake bit	No reaction	—	—
109	Monitor_Concentrator2_Alarm	Error in the frequency gasket in the concentrator 2	Feedback loop via FCCU Fake bit.	No reaction	—	—

Table 17. FCCU failure inputs (continued)

FCCU channel[1]	Failure	Failure description	Error reaction path check	Default reaction configuration after POR or destructive reset	Reference Manual related information	Recommended Recovery mechanism[2]
110	Xbic_Concentrator_1_alarm	XBIC Concentrator 1	Not testable	No reaction	—	—
111	checker core 2 disable alarm	Indication of disablement of lockstep mode	Feedback loop via FCCU Fake bit	No reaction	*Section 6.4.2: Dual-core lockstep*	POR, Destructive Reset
112	checker intc disable alarm	Indication of disablement of lockstep mode	Feedback loop via FCCU Fake bit	No reaction	*Section 6.4.2: Dual-core lockstep*	POR, Destructive Reset
113	checker dma_0 disable alarm	Indication of disablement of lockstep mode	Feedback loop via FCCU Fake bit	No reaction	*Section 6.4.2: Dual-core lockstep*	POR, Destructive Reset

Table 17. FCCU failure inputs (continued)

FCCU channel[1]	Failure	Failure description	Error reaction path check	Default reaction configuration after POR or destructive reset	Reference Manual related information	Recommended Recovery mechanism[2]
114	checker dma_1 disable alarm	Indication of disablement of lockstep mode	Feedback loop via FCCU Fake bit	No reaction	*Section 6.4.2: Dual-core lockstep*	POR, Destructive Reset
115	Flash Reset Error	The pin/flag signals the following unrecoverable errors: – ECC errors on Flash internal reads during configuration loading (startup) – ECC errors on Flash internal reads during fw copy (startup) – Double ECC errors on FLASH controller internal RAM during internal self-check routine (always running)	Not testable	No reaction	—	POR, Destructive Reset

1. Refer to “Fault Collection and Control Unit (FCCU)” chapter for channel mapping details.
2. This column recommends the reaction of the application in case an FCCU channel triggers an error during the execution of the safety function. An empty cell means that the reaction is application dependent..
3. In the event of activation of RF[94] (that is safe mode request), RF[79] is also activated. If the RGM trigger of safe mode is functionally used, during servicing of the response to activation of the RF[79], software should check that the device is not in safe mode to discriminate between an unwanted activation of test circuitry and a safe mode request.
4. In the event of spurious activation of this fault source, RF[79] is activated together with RF[94]. In scenarios where safe mode trigger from RGM is a valid functional use case, the RF[94] should be disabled.
5. Compensation cell reduces the spread of some circuit parameters (slew rate of the output signal and the output impedance) in the IO buffers over temperature, process and voltage. Compensation cell might remain disabled until supplies have reached LVD290 threshold.

6.4.1.8 Operational interference protection

Being a multi-master system, SPC58xEx/SPC58xGx provides safety mechanisms to prevent non-safety masters from interfering with the operation of the Safety Core, as well as mechanisms to handle the concurrent execution of software with different (lower) ASIL. Interference freedom is guaranteed via a hierarchical memory protection schema including:

- MPUs
- PBRIDGEs
- Register protection

There are two Memory Protection Unit levels included in the SPC58xEx/SPC58xGx. The Core Memory Protection Unit (CMPU) is a mechanism included in each core to protect

address ranges against access by software developed according to lower ASIL. The CMPU is typically used by the operating system to ensure inter-task interference protection.

The second MPU level is provided by the System MPUs (SMPU) located in each XBAR. The MPUs prevent access of different bus masters to address ranges and are typically used by the safety application to prevent NoSaMos (such as the Performance Core) to access the application's safety-relevant resources.

Furthermore, the PBRIDGE can restrict read and write access to individual I/O modules based on the origin of the access and its state (user mode/supervisor mode).

Finally, the register protection included allows individual register to be locked against any manipulation without unlocking.

These safety mechanisms are further described in the chapters System Memory Protection Unit (SMPU), Peripheral Bridge (PBRIDGE) and Registers Protection (REG_PROT).

6.4.1.9 FCCU supervision (FOSU)

As the FCCU is a central component in reacting to errors, it is itself supervised even though an error in it can only cause a latent failure. This supervision is provided by the FCCU Output Supervision Unit (FOSU). The FOSU receives failure indications at the same time as the FCCU. Unless the respective failure is switched off, the FOSU will observe the outputs of the FCCU (IRQ, RESET, F*n*). If the FCCU does not react within a predefined interval as specified in the "FCCU configuration" section of the "Device configuration" chapter, on one of those outputs to the incoming failure indication, the FOSU assumes the FCCU has failed, and causes a destructive reset.

The FOSU does not require any configuration by software.

6.4.2 Dual-core lockstep

The SPC58xEx/SPC58xGx duplicates its safety-relevant processing elements and compares their operation in Lockstep mode (LSM).

The safety core and its strictly connected modules are replicated.

Note: As software is concerned, they behave as one core.

Each replicated processing channels contains:

- Core
- Cache control
- Local memory control
- Core MPU
- Core System Bus Interface, including e2eECC logic

Other modules which are replicated are:

- INTC
- eDMA controller
- DMA muxes

Together each set of replicated elements forms a channel (for example, the Main channel and the Checker channel).

Elements that are not replicated are:

- Cache and local memory arrays
- Interconnect (XBAR)
- RAM controller
- Flash memory controller

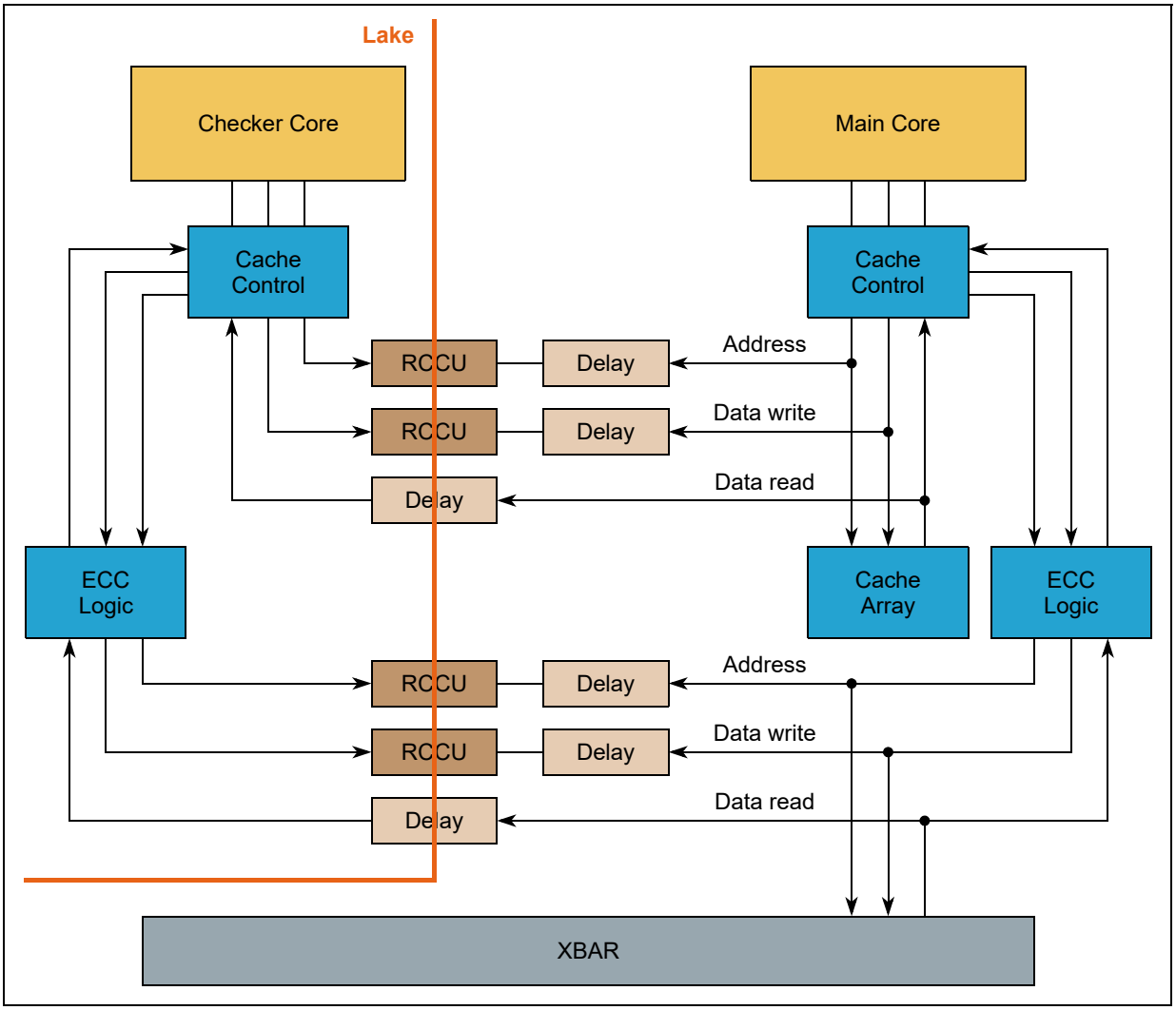
Note: This is not a complete list of all non-replicated elements.

Equivalent operation of replicated resources is supervised by comparators on all functional signals leaving the channels for the rest of the MCU. Any operational deviations between the supervised signals will cause the FCCU to be notified of the discrepancy.

The Checker Core does not have a direct connection to the XBAR. All of the outputs of Checker Core that target the XBAR (as well as any other non-duplicated resource, like local memories) will end in an RCCU for verification, and all the inputs to Checker Core from the XBAR will be split off from the Main Core XBAR inputs.

An abstract view of the implementation including delays and RCCUs is shown in *Figure 9*, but does not show local memory.

Figure 9. Dual core concept



A delay of two clock cycles exists between the Main and Checker channel to reduce their susceptibility to CCFs caused by power and clock disturbances. Outputs of the Main Core (such as XBAR addresses) are registered and delayed before being forwarded to the RCCUs, while inputs signals (such as XBAR read data) are registered and delayed before being fed to the Checker channel.

LSM can be disabled during boot by reprogramming the Flash memory (UTEST Miscellaneous DCF Client). LSM is also automatically disabled when Main Core enters debug mode and automatically re-enabled when it leaves it. If LSM is disabled, the RCCUs are disabled, and a signal is sent to the FCCU. Thus no user-initiated dynamic enabling of LSM is possible. If the cores are out of lock, a destructive or functional reset is needed to reestablish LSM. Also, no Decoupled Parallel Mode (DPM) is available (Main Core and Checker Core cannot operate as two independent cores running different software). Main Core is able to operate as an independent core if Checker Core is disabled. The status of LSM for the Safety core can be verified by checking the appropriate FCCU register.

6.4.3 Common cause failure measures

Various measures are included to prevent CCFs from endangering the effectiveness of the replication of the Safety Core or of the peripherals. These measures include physical separation of the components on the die, routing restrictions and supervision of clock, power, temperature, test and debug signals. In general these measures are independent from the software.

There are also configuration registers that, on change, can affect the execution of the MCU's safety function and, simultaneously, disable the respective safety mechanism. These are either protected against bit flips or those flips are detected by independent measures. These same registers are also protected against accidental software writes by employing the register protection described above.

6.4.4 ECC

Error correcting codes are used for end-to-end protection from cores to system storage as well as for individual protection of peripheral RAMs.

6.4.4.1 ECC for storage

All storage used in normal operation is protected by ECC with:

- SEC/DED (single bit error correction and double bit error detection) for core, local memory, peripheral RAM
- SEC/DEC/TED (single bit error correction, double bit error correction and triple error bit detection) for Flash memory

The following exceptions exist:

- The caches in the cores are protected by EDC (Error Detection Code), only detecting, but not correcting errors.
- SRAM is protected by SEC/DEC ECC in PRAM controller only during partial writes (RMW operations) otherwise it is protected by E2E-ECC mechanism
- FlexRay's schedule information RAM (lookup table RAM) is protected by EDC only.
- The HSM's AES program extension RAM are not protected by ECC.

A list showing the implementation of RAMs with ECC (including address protection) and for the SPC58xEx/SPC58xGx is shown in *Table 18*.

Table 18. ECC RAM implementations

Location	Memory	MEMU Classification	ECC	Address is ECC	Muxing Factor	Reacts to uncorrectable error
Core_0	I-MEM	System RAM	SEC/DED(1)	Y	8	Y
	D-MEM	System RAM	SEC/DED(1)	Y	16	Y
	I-Cache	System RAM	EDC	Y	4	Y
	D-Cache	System RAM	EDC	Y	4	Y
	ITAG	System RAM	SEC/DED	Y	4	Y
	DTAG	System RAM	SEC/DED	Y	4	Y

Table 18. ECC RAM implementations (continued)

Location	Memory	MEMU Classification	ECC	Address is ECC	Muxing Factor	Reacts to uncorrectable error
Core_1	I-MEM	System RAM	SEC/DED(1)	Y	8	Y
	D-MEM	System RAM	SEC/DED(1)	Y	16	Y
	I-Cache	System RAM	EDC	Y	4	Y
	D-Cache	System RAM	EDC	Y	4	Y
	ITAG	System RAM	SEC/DED	Y	4	Y
	DTAG	System RAM	SEC/DED	Y	4	Y
Core_2	I-MEM	System RAM	SEC/DED(1)	Y	8	Y
	D-MEM	System RAM	SEC/DED(1)	Y	16	Y
	I-Cache	System RAM	EDC	Y	4	Y
	D-Cache	System RAM	EDC	Y	4	Y
	ITAG	System RAM	SEC/DED	Y	4	Y
	DTAG	System RAM	SEC/DED	Y	4	Y
HSM	PRAM	Peripheral	SEC/DED	Y	8	N
	I-Cache	Peripheral	SEC/DED	Y	4	N
	ITAG	Peripheral	SEC/DED	Y	4	N
	C3	None	NO	N	4	N
FlexRay	DRAM	Peripheral	SEC/DED	N	4	Y
	LRAM	Peripheral	EDC	N	4	N
Ethernet	FIFO RX	Peripheral	SEC/DED	Y	4	Y
	FIFO TX	Peripheral	SEC/DED	Y	4	Y
GTM	FIFO	Peripheral	SEC/DED	Y	4	N
	(MCSi)-RAM0, i=0~3	Peripheral	SEC/DED	Y	4	N
	(MCSi)-RAM1, i=0~3	Peripheral	SEC/DED	Y	4	N
	DPLL-1A	Peripheral	SEC/DED	Y	4	N
	DPLL-1B	Peripheral	SEC/DED	Y	4	N
	DPLL-2	Peripheral	SEC/DED	Y	4	N
CAN	CAN	Peripheral	SEC/DED	N	8	Y
DMA	DMA	Peripheral	SEC/DED	Y	4	Y
Platform	SRAM	System RAM	E2E-ECC(2)	Y	8	Y
	Calibration	Peripheral	SEC/DED	Y	8	Y
	Nexus	Peripheral	NO	—	4	N
	AMU	Peripheral	SEC/DED	Y	8	N

1. Only when accessed by its own core. During access from other cores it is protected by E2E-ECC.
2. SEC/DEC for partial write operation (RMW), refer to *Section 36.5.14: Safety considerations*.

As shown in *Table 18*, some memories, particularly system storage, use an ECC computed over data and address to detect data and addressing faults (no/wrong/multiple selection). In addition, these same memories include dedicated measures against addressing and control faults (such as address/control feedback). This is different for storage related to PeMos. PeMos, in general, use an ECC without address error protections and do not include additional measures to detect them. All ECC calculations for RAMs in PeMos are accomplished by logic outside of the RAM, not in the RAM block.

An exception to this is the GTM's RAMs. They include addressing failure detection measures in the ECC. In addition, the individual RAM ports of the GTM do not share common parts and connect to individual (non-interfering) RAM arrays.

6.4.4.2 All-x words and ECC

There is a special case for valid ECC values in the SPC58xEx/SPC58xGx. Memory entries that are all zeros (All-0) or all ones (All-1), including the ECC parity bits, are not valid for memory that is checked by ECC. Memories that include addresses in the ECC calculation do not specifically protect against All-0 or All-1. This means that for some addresses All-0 or All-1 may be valid. The Flash memory allows All-1, corresponding to the status of an erase block, as a valid codeword.

Memories that include addresses in the ECC calculation do not specifically protect against All-0 or All-1. This means that for some addresses All-0 or All-1 may be valid.

In case of not valid codeword, All-0 and All-1 memory content is indicated in different ways. For memories that do not include address into the ECC calculation, All-0 and All-1 will be always uncorrectable errors. For all memories that include address into the ECC code-bit calculation, since the ECC checkbits depend on the address, it is not possible to generate an uncorrectable error indication for all the possible addresses. Therefore, a not valid All-x codeword may result in a correctable error.

Note: For Flash memory, additional dedicated safety mechanisms exist to detect failures that have the potential of leading to an All-1 word (refer to Chapter 37: Embedded Flash Memory, for more details on Flash memory safety mechanisms).

6.4.4.3 ECC failure handling

Single-bit and double-bit errors (correctable and uncorrectable errors) are signaled to the FCCU unless filtered by the MEMU. The MEMU (refer to *Section 6.4.4.4: Memory Error Management Unit (MEMU)*) may filter ECC error notification for known ECC error addresses with same (optional) syndrome (known permanent correctable errors). Actual implementation will signal errors, not to the FCCU, but to the MEMU, which filters and then forwards unfiltered notifications in an aggregated manner to the FCCU.

6.4.4.4 Memory Error Management Unit (MEMU)

The MEMU is responsible for the collection and reporting of error events generated by the ECC logic used to protect system RAMs (in this context they are those RAMs with "System RAM" memory classification as shown in *Table 18*), peripheral RAM, and Flash memory. In addition to storage errors, e2eECC errors caused by XBAR, RAM or flash memory controllers and memory related faults detected during MBIST are also routed to MEMU.

When ECC error events occur, the MEMU receives an error signal that causes an event to be recorded, and possibly sets corresponding error flags that are reported to the FCCU.

Refer to the Device Configuration chapter for MEMU connections.

The MEMU stores the addresses of ECC errors that have occurred in a table as follows:

- Uniqueness of the addresses with same (optional) syndrome in the table is ensured. For example, if an address with same (optional) syndrome is already stored in the MEMU's table, correctable errors at that address will no longer be reported to the FCCU
- Software can read and modify the MEMU table and remove individual entries (by marking them as invalid). For example, this allows software to invalidate a new entry in the MEMU table and wait for a repeat occurrence of the ECC error indicating a permanent error in memory
- If the MEMU table overflows, an additional signal (instead of the ECC error signal) is sent to the FCCU

As the MEMU aggregates address information from several sources, it might not be able to process all simultaneously arriving reports. This is called a simultaneous overflow and is signaled to the FCCU as a buffer overflow (refer to *Chapter 87: Memory Error Management Unit (MEMU)* for more information).

6.4.4.4.1 Interface to ECC units

The MEMU receives data according to the following interface per ECC unit connected:

- Whether the error is a Single-Bit-Error (SBE) or Uncorrectable Error (UCE) type
- Address of the memory where the error occurred
- The (optional) ECC-syndrome or MBIST bad-bit information in case of SBE
- A configuration specifying whether the ECC unit is attached to a peripheral RAM, to a System RAM or to Flash memory

Not all ECC units provide a syndrome. If an ECC unit does not provide the syndrome, 0xFF is stored in the MEMU error table instead of the syndrome. If an error is signaled, it is compared against all errors known for that storage:

- If no entry of that address with same (optional) syndrome is already in the buffer, it is to be added into the appropriate table depending on the SBE versus UCE indicator
- If there is no free entry left in that buffer, the overflow flag is set

Note: If a previously known error has been marked invalid by software, no comparison against the address will occur.

If a valid entry of that address is in the buffer, and if the entry indicates an SBE in that address, and the error indicated is uncorrectable, a new entry is added in the MEMU buffer. In case of the same error is reported by ECC which reports the syndrome and additionally by ECC which does not, the reporting order becomes important.

- If the entry indicates an SBE in that address, and the error indicated is uncorrectable, a new entry is added in the MEMU buffer. If ECC which provides the syndrome reports first, the second reporting without syndrome is discarded.
- If ECC which does not provide the syndrome reports first, a second entry with the syndrome is stored in the table.

7 Device configuration

7.1 Introduction

This chapter provides details on the individual modules of the microcontroller. It includes:

- Module block diagrams showing immediate connections within the device,
- Specific module-to-module interactions not necessarily discussed in the individual Module chapters, and
- Links for more information.

7.2 Core modules

The SPC58xEx/SPC58xGx has cores in three distinct modules: HSM (described in the HSM chapter of the SPC58xEx/SPC58xGx Microcontroller Security Reference Manual); GTM (described in the GTM documentation); and the main platform (described below).

The main platform has five separate cores that perform various computational and control functions:

- The main computational shell consists of 2 processing channels (Main Core_0 and Main Core_1), both of which use an e200z420n3 core. These two cores are used for general computational functions.
- Only Main Core_0 uses one e200z419 core (Checker Core_0) as lock step replica. The e200z419 is a true subset of the e200z420n3 core. When enabled, the checker core operates in lock step mode with the associated main core executing exactly the same instructions. Thus, the checker core insures that the main core is executing correctly.
- The third processing channel is implemented with a lock-step pair of 2 e200z425s and is generally used to control various I/O modules.
- Core0 and Core2 are capable of locking their respective crossbar master ports using reservation instructions for data accesses. These cores are not capable of locking their crossbar master ports for instruction accesses using reservation instructions.
- Core1 is not allowed to lock both instruction and data crossbar ports using reservation instructions.

PowerISA 2.06 and implementation-specific SPRs for the e200z420n3, e200z419, and e200z425n3 cores are listed in *Table 19*. All registers are 32 bits in size. Register bits are numbered from bit 0 to bit 31 (most-significant to least-significant). Shaded entries represent optional registers. An SPR register may be read or written with the **mfspr** and **mtspr** instructions. In the instruction syntax, compilers should recognize the mnemonic name given in *Table 19*.

Table 19. Special Purpose Registers (SPRs)

Mnemonic	Name	SPR Number	Access	Privileged	e200z Specific
XER	Integer Exception Register	1	R/W	No	No
LR	Link Register	8	R/W	No	No
CTR	Count Register	9	R/W	No	No
SRR0	Save/Restore Register 0	26	R/W	Yes	No

Table 19. Special Purpose Registers (SPRs) (continued)

Mnemonic	Name	SPR Number	Access	Privileged	e200z Specific
SRR1	Save/Restore Register 1	27	R/W	Yes	No
PID0	Process ID Register	48	R/W	Yes	No
CSRR0	Critical Save/Restore Register 0	58	R/W	Yes	No
CSRR1	Critical Save/Restore Register 1	59	R/W	Yes	No
DEAR	Data Exception Address Register	61	R/W	Yes	No
ESR	Exception Syndrome Register	62	R/W	Yes	No
IVPR	Interrupt Vector Prefix Register	63	R/W	Yes	No
SPRG0	SPR General 0	272	R/W	Yes	No
SPRG1	SPR General 1	273	R/W	Yes	No
SPRG2	SPR General 2	274	R/W	Yes	No
SPRG3	SPR General 3	275	R/W	Yes	No
PIR	Processor ID Register	286	R/W	Yes	No
PVR	Processor Version Register	287	Read-only	Yes	No
DBSR	Debug Status Register	304	Read/Clear (1)	Yes	No
DBCR0	Debug Control Register 0	308	R/W	Yes	No
DBCR1	Debug Control Register 1	309	R/W	Yes	No
DBCR2	Debug Control Register 2	310	R/W	Yes	No
IAC1	Instruction Address Compare 1	312	R/W	Yes	No
IAC2	Instruction Address Compare 2	313	R/W	Yes	No
IAC3	Instruction Address Compare 3	314	R/W	Yes	No
IAC4	Instruction Address Compare 4	315	R/W	Yes	No
DAC1	Data Address Compare 1	316	R/W	Yes	No
DAC2	Data Address Compare 2	317	R/W	Yes	No
DVC1	Data Value Compare 1(2)	318	R/W	Yes	No
DVC2	Data Value Compare 2(2)	319	R/W	Yes	No
SPEFSCR	LSP/EFP APU status and control register	512	R/W	No	No
L1CFG0	L1 cache config register 0	515	Read-only	No	Yes
L1CFG1	L1 cache config register 1	516	Read-only	No	Yes
NPIDR	Nexus 3 Process ID register	517	R/W	No	Yes
DBCR3	Debug control register 3	561	R/W	Yes	Yes
DBCNT	Debug Counter register	562	R/W	Yes	Yes
DBCR4	Debug control register 4	563	R/W	Yes	Yes
DBCR5	Debug control register 5	564	R/W	Yes	Yes

Table 19. Special Purpose Registers (SPRs) (continued)

Mnemonic	Name	SPR Number	Access	Privileged	e200z Specific
IAC5	Instruction Address Compare 5	565	R/W	Yes	Yes
IAC6	Instruction Address Compare 6	566	R/W	Yes	Yes
IAC7	Instruction Address Compare 7	567	R/W	Yes	Yes
IAC8	Instruction Address Compare 8	568	R/W	Yes	Yes
MCSRR0	Machine Check Save/Restore Register 0	570	R/W	Yes	Yes
MCSRR1	Machine Check Save/Restore Register 1	571	R/W	Yes	Yes
MCSR	Machine Check Syndrome Register	572	Read/Clear (3)	Yes	Yes
MCAR	Machine Check Address Register	573	R/W	Yes	Yes
DSRR0	Debug save/restore register 0	574	R/W	Yes	Yes
DSRR1	Debug save/restore register 1	575	R/W	Yes	Yes
DDAM	Debug Data Acquisition Messaging register	576	R/W	No	Yes
DAC3	Data Address Compare 3	592	R/W	Yes	Yes
DAC4	Data Address Compare 4	593	R/W	Yes	Yes
DBCR7	Debug control register 7	596	R/W	Yes	Yes
DBCR8	Debug control register 8	597	R/W	Yes	Yes
DDEAR	Debug Data Effective Address register	600	R/W	Yes	Yes
DVC1U(4)	Data Value Compare 1 Upper	601	R/W	Yes	No
DVC2U(4)	Data Value Compare 2 Upper	602	R/W	Yes	No
DBCR6	Debug control register 6	603	R/W	Yes	Yes
MAS0	MPU assist register 0	624	R/W	Yes	Yes
MAS1	MPU assist register 1	625	R/W	Yes	Yes
MAS2	MPU assist register 2	626	R/W	Yes	Yes
MAS3	MPU assist register 3	627	R/W	Yes	Yes
EDBRAC0	External debug resource allocation control register 0	638	Read only	Yes	Yes
MPU0CFG	MPU0 configuration register	692	Read-only	Yes	Yes
L1FINV1	L1 cache flush and invalidate control register 0	959	R/W	Yes	Yes
DEVENT	Debug Event register	975	R/W	No	Yes
HID0	Hardware implementation dependent reg 0	1008	R/W	Yes	Yes
HID1	Hardware implementation dependent reg 1	1009	R/W	Yes	Yes
L1CSR0	L1 cache control and status register 0	1010	R/W	Yes	Yes
L1CSR1	L1 cache control and status register 1	1011	R/W	Yes	Yes
BUCSR	Branch Unit Control and Status Register	1013	R/W	Yes	Yes
MPU0CSR0	MPU0 configuration register	1014	R/W	Yes	Yes

Table 19. Special Purpose Registers (SPRs) (continued)

Mnemonic	Name	SPR Number	Access	Privileged	e200z Specific
MMUCFG	MMU/MPU configuration register	1015	Read-only	Yes	Yes
L1FINV0	L1 cache flush and invalidate control register 0	1016	R/W	Yes	Yes
SVR	System Version Register	1023	Read-only	Yes	Yes

1. The Debug Status Register can be read using *mfspr RT,DBSR*. The Debug Status Register cannot be directly written to. Instead, bits in the Debug Status Register corresponding to '1' bits in GPR(RS) can be cleared using *mtspr DBSR,RS*.
2. Undefined on **POR** assertion, unchanged on **external PORESET** assertion.
3. The Machine Check Syndrome Register can be read using *mfspr RT,MCSR*. The Machine Check Syndrome Register cannot be directly written to. Instead, bits in the Machine Check Syndrome Register corresponding to '1' bits in GPR(RS) can be cleared using *mtspr MCSR,RS*.
4. The 64-bit DVC registers are accessed by using the DVC1,2 spr #s for the lower 32 bits, and the DVC1,2U spr #s for the upper 32 bits.

Table 20 shows the state of the *PowerISA 2.06* architected registers and other optional resources immediately following a system reset.

Table 20. Reset settings for e200z4 resources

Resource	System reset setting
Program Counter	p_rstbase[0:29] || 2'b00(1)
GPRs	Unaffected(2)
CR	Unaffected(2)
BUCSR	0x0000_0000
CSRR0	Unaffected(2)
CSRR1	Unaffected(2)
CTR	Unaffected(2)
DAC1	0x0000_0000(3)
DAC2	0x0000_0000(3)
DAC3	0x0000_0000(3)
DAC4	0x0000_0000(3)
DBCR0	0x0000_0000(3)
DBCR1	0x0000_0000(3)
DBCR2	0x0000_0000(3)
DBCR4	0x0000_0000(3)
DBCR5	0x0000_0000(3)
DBCR6	0x0000_0000(3)
DBCR7	0x0000_0000(3)
DBCR8	0x0000_0000(3)
DBSR	0x1000_0000(3)
DDAM	0x0000_0000(3)

Table 20. Reset settings for e200z4 resources (continued)

Resource	System reset setting
DDEAR	Unaffected(2)
DEAR	Unaffected(2)
DEVENT	0x0000_0000(3)
DSRR0	Unaffected(2)
DSRR1	Unaffected(2)
DVC1	Unaffected(2)
DVC2	Unaffected(2)
ESR	0x0000_0000
HID0	0x0000_0000
HID1	0x0000_0000
IAC1	0x0000_0000(3)
IAC2	0x0000_0000(3)
IAC3	0x0000_0000(3)
IAC4	0x0000_0000(3)
IAC5	0x0000_0000(3)
IAC6	0x0000_0000(3)
IAC7	0x0000_0000(3)
IAC8	0x0000_0000(3)
IVPR	Unaffected(2)
LR	Unaffected(2)
L1CFG0, L1CFG1(4)	—
L1CSR0, 1	0x0000_0000
L1FINV0, 1	0x0000_0000
MAS1	Unaffected(2)
MAS2	Unaffected(2)
MAS3	Unaffected(2)
MAS4	Unaffected(2)
MCAR	Unaffected(2)
MCSR	0x0000_0000
MCSRR0	Unaffected(2)
MCSRR1	Unaffected(2)
MMUCFG(4)	—
MPU0CSR0	0x0000_0000
MPU0CFG(4)	—
MSR	0x0000_0000

Table 20. Reset settings for e200z4 resources (continued)

Resource	System reset setting
NPIDR	0x0000_0000
PID0	0x0000_0000
PIR	0x0000_00 || p_cpuid[0:7]1
PVR(4)	—
SPEFSCR	0x0000_0000
SPRG0	Unaffected(2)
SPRG1	Unaffected(2)
SPRG2	Unaffected(2)
SPRG3	Unaffected(2)
SRR0	Unaffected(2)
SRR1	Unaffected(2)
SVR(4)	—
XER	0x0000_0000
IMEMCTL0	22'b0 || p_imem_rstcfg[0] || 4'b0 || p_imem_rstcfg[1:5](5)

1. The levels that determine these bits are set by the factory and may change between revisions of the device.
   Main_Core_0: p_cpuid[0:7] = 0x00
   Main_core_1: p_cpuid[0:7] = 0x01
   Peripheral_Core_2: p_cpuid[0:7] = 0x02
2. Undefined on **POR** assertion, unchanged on **external PORESET** assertion.
3. Reset by processor reset **external PORESET** if DBCR0[EDM]=0, as well as unconditionally by **POR**.
4. Read-only registers.
5. For all three cores (0, 1, 2) p_imem_rstcfg[0:5] = 0x0

Table 21. Core configuration inputs

CPU ID	Inputs	Value	Affected registers
Core_0	p_dmem_rstcfg[0:8]	0x0DA	DMEMCTL0
	p_imem_rstcfg[0:5]	0x1B	IMEMCTL0
	p_e2e_rstcfg[0:3]	0x5	E2ECTL0
Core_1	p_dmem_rstcfg[0:8]	0x0DA	DMEMCTL0
	p_imem_rstcfg[0:5]	0x1B	IMEMCTL0
	p_e2e_rstcfg[0:3]	0x5	E2ECTL0
Core_2	p_dmem_rstcfg[0:8]	0x0DA	DMEMCTL0
	p_imem_rstcfg[0:5]	0x00	IMEMCTL0
	p_e2e_rstcfg[0:3]	0x5	E2ECTL0

7.3 System modules

7.3.1 SIUL2 configuration

The System Integration Unit Lite 2 (SIUL2) controls the MCU pad configuration, ports, general-purpose input and output (GPIO) signals and external interrupts with trigger event configuration.

For details, refer *Chapter 17: System Integration Unit Lite2 (SIUL2)*.

Note: All output pins, whose SIUL2_MSCRn[SMC] = 0, disable their output drivers when the device enters SAFE mode while the internal weak pull-up of GPIO is enabled automatically. Coherently, ME_SAFE_MC.PDO must be set to 1 (default value) when disabling the output buffer.
Each other configuration has no effect on output driver of pin, refer to Table 22.

Table 22. Output buffer configuration

ME_SAFE_MC [PDO]	SIUL2.MSCR_IOn[SMC]	DRUN Mode	SAFE Mode
0 (no safe gating of I/O)	0 (output buffer disabled)	Output buffer enabled	Output buffer enabled
0 (no safe gating of I/O)	1 (output buffer enabled)	Output buffer enabled	Output buffer enabled
1 (output buffer driver disabled in SAFE mode)	0 (output buffer disabled)	Output buffer enabled	Output buffer disabled, internal weak pull-up enabled
1 (output buffer driver disabled in SAFE mode)	1 (output buffer enabled)	Output buffer enabled	Output buffer enabled

7.3.2 Semaphores2 (SEMA42) configuration

The peripheral platform shell includes a memory-mapped module (SEMA42) that provides

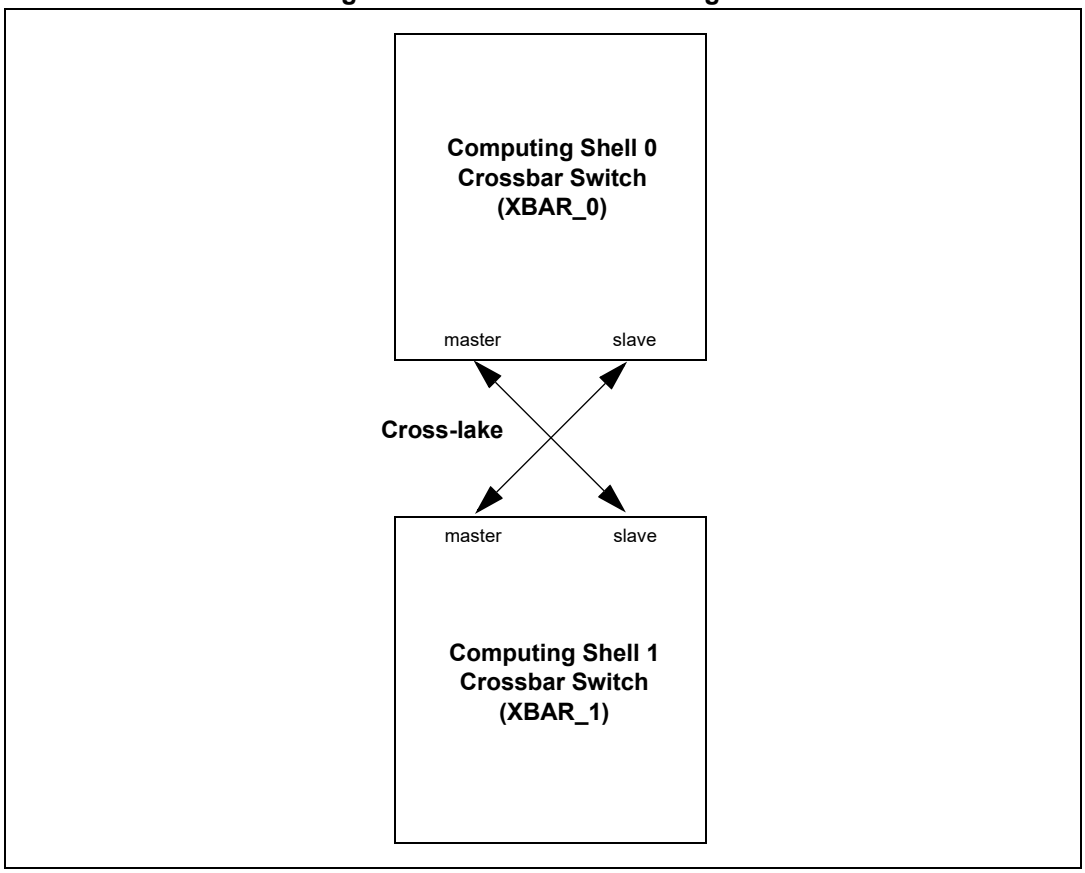
- Robust hardware support needed in multi-core systems for implementing semaphores
- A simple mechanism to achieve "lock/unlock" operations via a single write access

The hardware semaphores module is an architecture-neutral solution that provides sixteen hardware-enforced gates for SPC58xEx/SPC58xGx as well as other useful system functions related to the gating mechanisms. The capabilities provided by the hardware semaphore module are currently utilized in dual-core AUTOSAR® implementations and its use is expected to continue in future multi-core ports. The result is a second-generation module implementation known as Semaphores2 or SEMA42.

7.3.3 Crossbar switch

This device contains two crossbar switch modules. *Figure 10* illustrates that they are connected via a cross-lake, which is comprised of intelligent bridging bus gaskets. These bus gaskets handle the traffic which crosses between the shells and do not require any interaction with the user. This dual-crossbar architecture allows all masters to access all slaves across computing shell 0 and computing shell 1.

Figure 10. Crossbar switch diagram



The following sections explain how each crossbar switch is configured on this device.

7.3.3.1 Computing shell 0 crossbar switch (XBAR_0) configuration

This section summarizes how the module has been configured in the chip. For a comprehensive description of the module itself, please refer to the Crossbar switch (XBAR) chapter.

Figure 11. Crossbar_0 switch integration

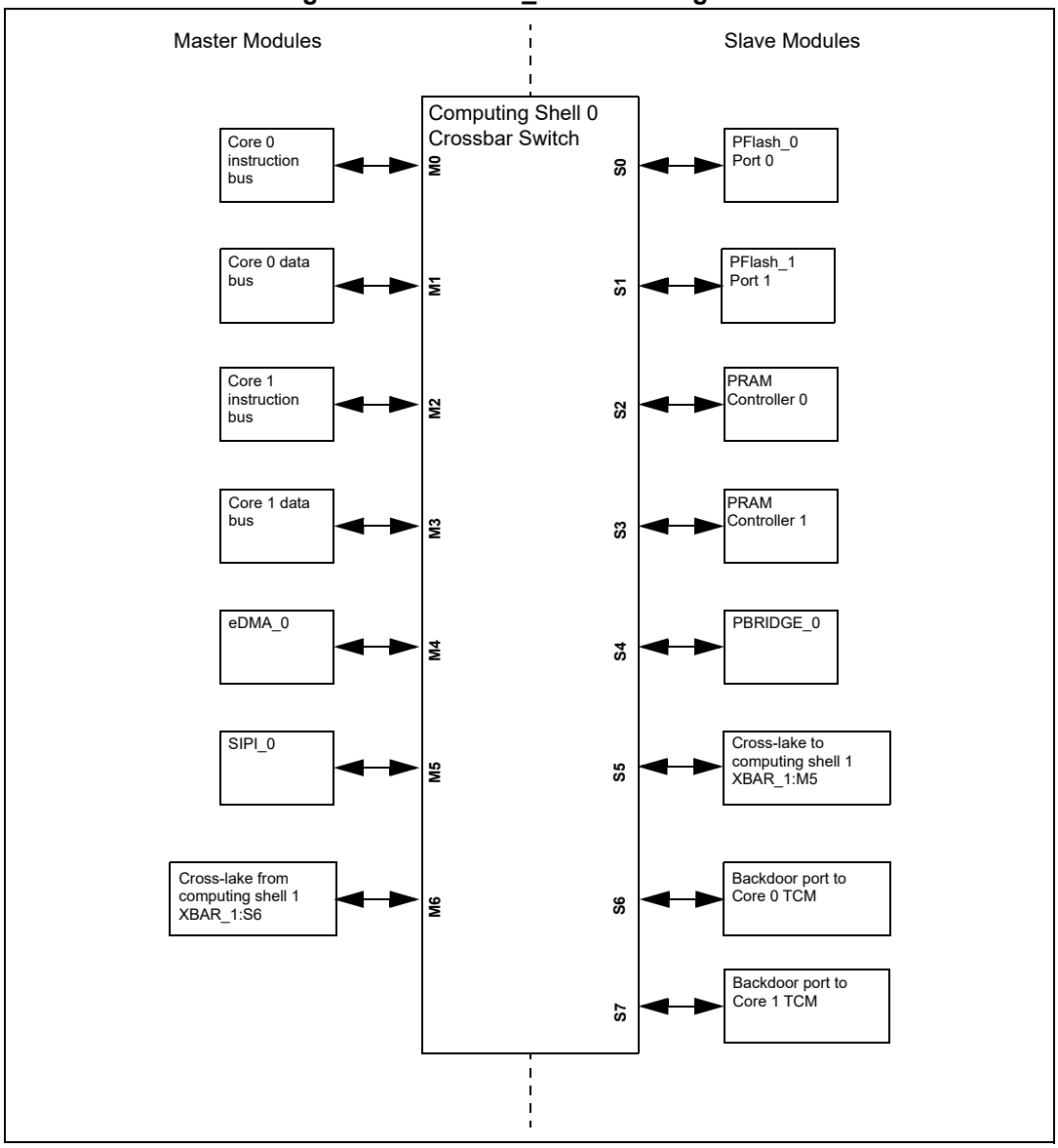


Table 23. Reference links to related information

Topic	Related module	Reference
Full description	Crossbar switch	*Chapter 18: Crossbar switch (XBAR)*
System memory map	—	*Chapter 5: Memory Map*
Clocking	—	*Chapter 27: Clocking*
Crossbar switch master	Cores	*Section 7.2: Core modules*
Crossbar switch slave	Flash memory controller	*Section 3.3.1: Flash memory controller*
Crossbar switch slave	Processor core local memory	*Section 3.2.4: Processor core local RAM*
Crossbar switch slave	System RAM	*Section 3.2.1: System SRAM*

7.3.3.1.1 Crossbar_0 switch master assignments

This device contains seven masters connected to the computing shell 0 crossbar switch.

The master assignment is shown in *Table 24*.

Table 24. Crossbar_0 switch master assignments

Master module	Physical master port number
z420n3_0 Instruction AHB	M0
z420n3_0 Data Bus + Nexus AHB	M1
z420n3_1 Instruction AHB	M2
z420n3_1 Data Bus + Nexus AHB	M3
e2eECC Master Gasket for eDMA_0	M4
e2eECC Master Gasket for SIPI_0	M5
Cross-lake from computing shell 1 crossbar XBAR_1:S6	M6

7.3.3.1.2 Crossbar_0 switch slave assignments

This device contains eight slaves connected to the computing shell 0 crossbar switch.

The slave assignment is shown in *Table 25*.

Table 25. Crossbar_0 switch slave assignments

Slave module	Physical slave port number
PFLASHC_0:p0	S0
PFLASHC_1:p1	S1
PRAMC_0	S2
PRAMC_1	S3
PBRIDGE_0	S4
Cross-lake to computing shell 1 XBAR_1:M5	S5
Backdoor port to Core 0 local memory	S6
Backdoor port to Core 1 local memory	S7

7.3.3.1.3 PRS register reset values

The computing shell 0 XBAR_0 PRS*n* registers reset to 0x66301425. Note that these registers are read-only.

7.3.3.1.4 MGPCR register reset values

The streaming shell 0 XBAR_0 MGPCRn registers reset to 0x00000000.

7.3.3.2 Computing shell 1 crossbar switch (XBAR_1) configuration

This section summarizes how the module has been configured in the chip. For a comprehensive description of the module itself, please refer to the Crossbar switch (XBAR) chapter.

Figure 12. Crossbar_1 switch integration

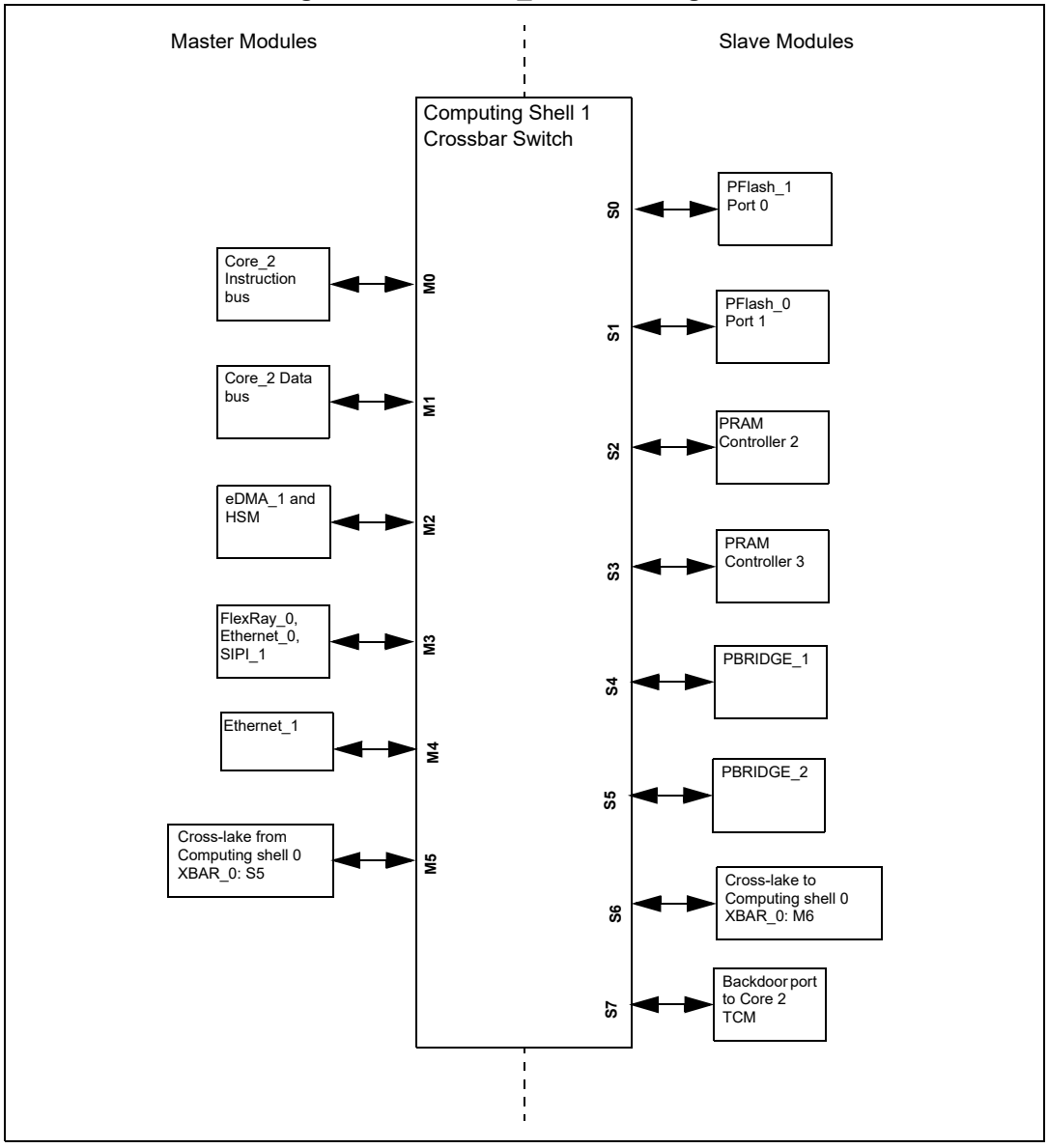


Table 26. Reference links to related information

Topic	Related module	Reference
Full description	Crossbar switch	*Chapter 18: Crossbar switch (XBAR)*
System memory map	—	*Chapter 5: Memory Map*
Clocking	—	*Chapter 27: Clocking*

Table 26. Reference links to related information (continued)

Topic	Related module	Reference
Crossbar switch master	Cores	*Section 7.2: Core modules*
Crossbar switch master	DMA and HSM	*Section 7.3.7: DMA controller configuration*
Crossbar switch master	Ethernet, LFAST, and FlexRay	*Section 7.8.1: Ethernet configuration*, *Section 7.8.2: CAN subsystem configuration*, *Section 7.8.5: FlexRay configuration*
Crossbar switch slave	Processor core local memory	*Section 3.2.4: Processor core local RAM*
Crossbar switch slave	Peripheral bridge	*Chapter 20: Peripheral Bridge (PBRIDGE)*

7.3.3.2.1 Crossbar_1 switch master assignments

This device contains six masters connected to the computing shell 1 crossbar switch.

The master assignment is shown in *Table 27*.

Table 27. Crossbar_1 switch master assignments

Master module	Physical master port number
z425n3 Instruction AHB	M0
z425n3 Data Bus + Nexus AHB	M1
AHB_Concentrator_0	M2
AHB_Concentrator_1	M3
AHB_Concentrator_2	M4
Cross-lake from Computing shell 0 XBAR_0:S5	M5

7.3.3.2.2 Crossbar_1 switch slave assignments

This device contains eight slaves connected to the computing shell 1 crossbar switch.

The slave assignment is shown in *Table 28*.

Table 28. Crossbar_1 switch slave assignments

Slave module	Physical slave port number
PFLASHC_1:p0	S0
PFLASHC_0:p1	S1
PRAMC_2	S2
PRAMC_3	S3
PBRIDGE_1	S4
PBRIDGE_2	S5
Cross-lake to Computing shell 0 XBAR_0:M6	S6
Backdoor port to Core 2 local memory	S7

7.3.3.2.3 PRS register reset values

The computing shell 1 XBAR_1 PRS*n* registers reset to 0x50531024. Note that these registers are read-only.

7.3.3.2.4 MGPCR register reset values

The streaming shell 1 XBAR_1 MGPCRn registers reset to 0x00000000.

7.3.3.3 AHB Master ID Assignments

Table 29. AHB Master ID Assignments (SMPU0)

SMPU logical bus master number	Bus master
0	Core0 (z420n3_0)
1	Core1 (z420n3_1)
2	Core2 (z425n3)
3	eDMA_0
4	Ethernet_0
5	FlexRay_0
6	Reserved
7	SIPI_1
8	Core0_Nexus Interface
9	Core1_Nexus Interface
10	Core2_Nexus Interface
11	eDMA_1
12	SIPI_0
13	Ethernet_1
14	HSM
15	NAR (Nexus Aurora Router)

Table 30. AHB Master ID Assignments (SMPU1)

SMPU logical bus master number	Bus master
0	Core0 (z420n3_0)
1	Core1 (z420n3_1)
2	Core2 (z425n3)
3	eDMA_0
4	Ethernet_0
5	FlexRay_0
6	Reserved
7	SIPI_1

Table 30. AHB Master ID Assignments (SMPU1) (continued)

SMPU logical bus master number	Bus master
8	Core0_Nexus Interface
9	Core1_Nexus Interface
10	Core2_Nexus Interface
11	eDMA_1
12	SIPI_0
13	Ethernet_1
14	HSM
15	NAR (Nexus Aurora Router)

7.3.4 System Memory Protection Unit (SMPU) configuration

This chip contains three instances of the System Memory Protection Unit (SMPU):

- SMPU_HSM for the Hardware Security Module
- SMPU_0 for the Computational Shell
- SMPU_1 for the Peripheral Shell

The following sections explain how SMPU_0 and SMPU_1 are configured on this chip. See the *SPC58xEx/SPC58xGx Microcontroller Security Reference Manual* for configuration details of SMPU_HSM.

Note: SMPU_0 can only protect slaves of XBAR_0 except slave S5 (Slave port that connects to XBAR_1). Bus masters with Master IDs as mentioned in Table 27 are considered valid by SMPU_0.

SMPU_1 can only protect slaves of XBAR_1 except slave S6 (Slave port that connects to XBAR_0). Bus masters with Master IDs as mentioned in Table 28 are considered valid by SMPU_1.

7.3.4.1 SMPU0 configuration

This section summarizes how the module instance has been configured in the chip. For a comprehensive description of the module itself, see the chapter for that module.

Figure 13. SMPU0 configuration

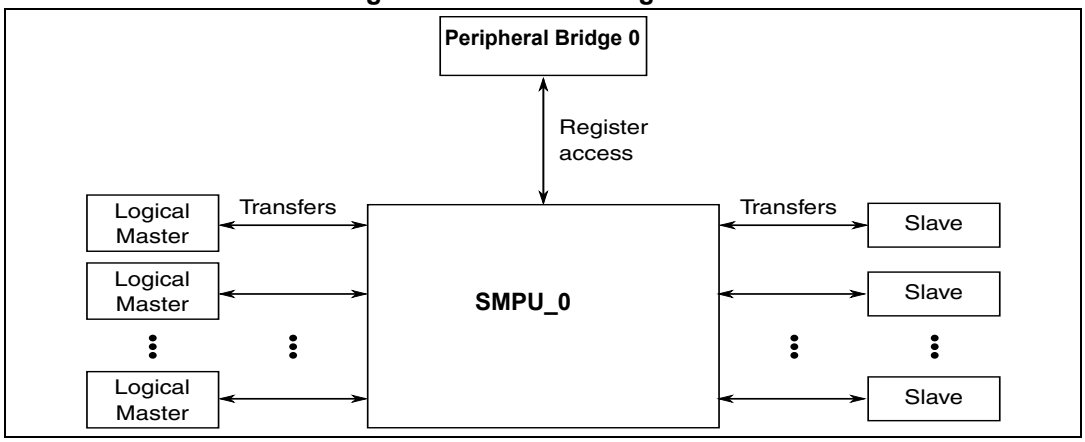


Table 31. Reference links to related information

Topic	Related module	Reference
Full description	System Memory Protection Unit (SMPU)	*Chapter 21: System Memory Protection Unit (SMPU)*
System memory map	—	*Chapter 5: Memory Map*
Clocking	—	*Chapter 27: Clocking*
Power management	—	*Chapter 10: Power management*
Logical masters	—	*Section 7.3.3.3: AHB Master ID Assignments*
Region descriptors	—	*Section 7.3.4.1.2: Number of region descriptors supported by SMPU0*
RGD_WORD2 formats	—	*Section 7.3.4.1.3: RGD_WORD2 formats supported by SMPU0*
Master Access Permissions	—	*Section 7.3.4.1.4: Master Access permissions for SMPU0*

7.3.4.1.1 SMPU0 logical bus master assignments

The logical bus master assignments for the SMPU0 are present in *Chapter 7: Device configuration*, *Section 7.3.3.3: AHB Master ID Assignments*.

7.3.4.1.2 Number of region descriptors supported by SMPU0

This instance of the SMPU module supports a maximum of twenty-four region descriptors.

7.3.4.1.3 RGD_WORD2 formats supported by SMPU0

For RGD_WORD2, this instance of the SMPU module supports only the FMT0 format. The RGD_WORD3[FMT] field is read-only with a fixed value of 0 and only RGD_WORD2_FMT0 applies.

7.3.4.1.4 Master Access permissions for SMPU0

The only masters allowed to access the SMPU0 registers are HSM, Core 2, Core 1, and Core 0. It is also required that the core be in supervisor mode.

7.3.4.2 SMPU1 configuration

This section summarizes how the module instance has been configured in the chip. For a comprehensive description of the module itself, see the chapter for that module.

Figure 14. SMPU1 configuration

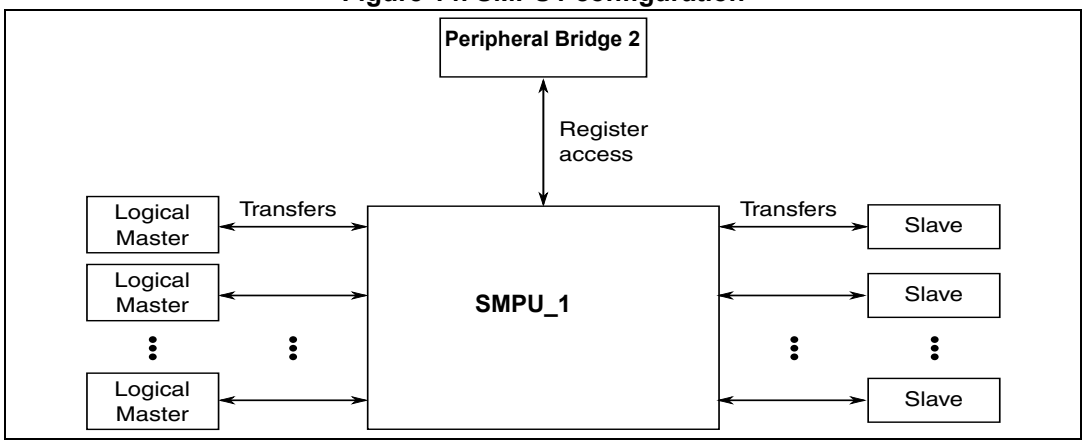


Table 32. Reference links to related information

Topic	Related module	Reference
Full description	System Memory Protection Unit (SMPU)	*Chapter 21: System Memory Protection Unit (SMPU)*
System memory map	—	*Chapter 5: Memory Map*
Clocking	—	*Chapter 27: Clocking*
Power management	—	*Chapter 10: Power management*
Logical masters	—	*Section 7.3.4.2.1: SMPU1 logical bus master assignments*

Table 32. Reference links to related information (continued)

Topic	Related module	Reference
Region descriptors	—	*Section 7.3.4.2.2: Number of region descriptors supported by SMPU1*
RGD_WORD2 formats	—	*Section 7.3.4.2.3: RGD_WORD2 formats supported by SMPU1*
Master Access Permissions	—	*Section 7.3.4.2.4: Master Access permissions for SMPU1*

7.3.4.2.1 SMPU1 logical bus master assignments

The logical bus master assignments for the SMPU1 are present in *Chapter 7: Device configuration*, *Section 7.3.3.3: AHB Master ID Assignments*.

7.3.4.2.2 Number of region descriptors supported by SMPU1

This instance of the SMPU module supports a maximum of twenty-four region descriptors.

7.3.4.2.3 RGD_WORD2 formats supported by SMPU1

For RGD_WORD2, this instance of the SMPU module supports only the FMT0 format. The RGD_WORD3[FMT] field is read-only with a fixed value of 0 and only RGD_WORD2_FMT0 applies.

7.3.4.2.4 Master Access permissions for SMPU1

The only masters allowed to access the SMPU1 registers are HSM, Core 2, Core 1, and Core 0. It is also required that the core be in supervisor mode.

7.3.5 Peripheral bridge configuration

This section summarizes how the module has been configured in the chip. For a comprehensive description of the module itself, see the module's dedicated chapter.

Figure 15. Peripheral bridge configuration

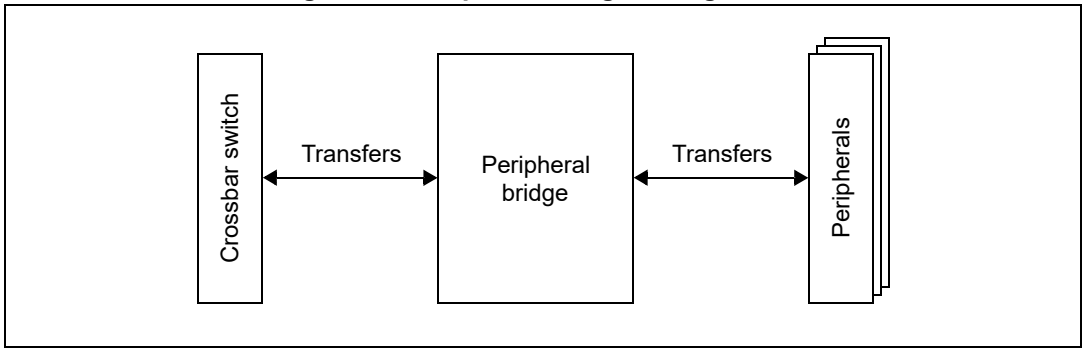


Table 33. Reference links to related information

Topic	Related module	Reference
Full description	Peripheral bridge	*Chapter 20: Peripheral Bridge (PBRIDGE)*
System memory map	—	*Chapter 5: Memory Map*
Clocking	—	*Chapter 27: Clocking*
Crossbar switch	Crossbar switch	*Section 7.3.3: Crossbar switch*

7.3.5.1 Peripheral bridge instantiation

This device contains three identical peripheral bridges.

7.3.5.2 Memory maps

The peripheral bridges are used to access the registers of most of the modules on this device. See *Chapter 5: Memory Map* for the memory slot assignment for each module.

7.3.5.3 MPR registers

Each of the three peripheral bridges supports up to sixteen crossbar switch masters, each assigned to a MPROT*n* field in the MPRA-B registers. However, fewer are supported on this device. See *Section 7.3.3: Crossbar switch* for details of the master port assignments for this device.

7.3.5.4 PACR/OPACR registers

The peripherals attached to the peripheral bridges each assigned to a PACR*n* or OPACR*n* field within the PACRA-H/OPACRA-AF registers. However, fewer peripherals are supported on this device. See *Chapter 5: Memory Map* for details of the peripheral slot assignments for this device. Unused PACR*n* fields are reserved.

7.3.6 Interrupt controller (INTC) configuration

7.3.6.1 INTC configurable parameters

The INTC on this chip supports three processors—the safety core, the computational core and the I/O processor (IOP).

7.3.6.2 Interrupt channel map

Table 34 defines the interrupt sources for the INTC on this chip. All IRQ numbers not specifically listed are not used.

Table 34. Interrupt sources

Number	Description	Register bit
0	Software interrupt flag 0	INTC_SSCIR0_3[CLR0]
1	Software interrupt flag 1	INTC_SSCIR0_3[CLR1]
2	Software interrupt flag 2	INTC_SSCIR0_3[CLR2]
3	Software interrupt flag 3	INTC_SSCIR0_3[CLR3]

Table 34. Interrupt sources (continued)

Number	Description	Register bit
4	Software interrupt flag 4	INTC_SSCIR4_7[CLR4]
5	Software interrupt flag 5	INTC_SSCIR4_7[CLR5]
6	Software interrupt flag 6	INTC_SSCIR4_7[CLR6]
7	Software interrupt flag 7	INTC_SSCIR4_7[CLR7]
8	Software interrupt flag 8	INTC_SSCIR8_11[CLR8]
9	Software interrupt flag 9	INTC_SSCIR8_11[CLR9]
10	Software interrupt flag 10	INTC_SSCIR8_11[CLR10]
11	Software interrupt flag 11	INTC_SSCIR8_11[CLR11]
12	Software interrupt flag 12	INTC_SSCIR12_15[CLR12]
13	Software interrupt flag 13	INTC_SSCIR12_15[CLR13]
14	Software interrupt flag 14	INTC_SSCIR12_15[CLR14]
15	Software interrupt flag 15	INTC_SSCIR12_15[CLR15]
16	Software interrupt flag 16	INTC_SSCIR16_19[CLR16]
17	Software interrupt flag 17	INTC_SSCIR16_19[CLR17]
18	Software interrupt flag 18	INTC_SSCIR16_19[CLR18]
19	Software interrupt flag 19	INTC_SSCIR16_19[CLR19]
20	Software interrupt flag 20	INTC_SSCIR20_23[CLR20]
21	Software interrupt flag 21	INTC_SSCIR20_23[CLR21]
22	Software interrupt flag 22	INTC_SSCIR20_23[CLR22]
23	Software interrupt flag 23	INTC_SSCIR20_23[CLR23]
24	Software interrupt flag 24	INTC_SSCIR24_27[CLR24]
25	Software interrupt flag 25	INTC_SSCIR24_27[CLR25]
26	Software interrupt flag 26	INTC_SSCIR24_27[CLR26]
27	Software interrupt flag 27	INTC_SSCIR24_27[CLR27]
28	Software interrupt flag 28	INTC_SSCIR28_31[CLR28]
29	Software interrupt flag 29	INTC_SSCIR28_31[CLR29]
30	Software interrupt flag 30	INTC_SSCIR28_31[CLR30]
31	Software interrupt flag 31	INTC_SSCIR28_31[CLR31]
32	Watchdog timer 0 (SWT_0)	SWT_0_IR[TIF]
33	Watchdog timer 1 (SWT_1)	SWT_1_IR[TIF]
34	Watchdog timer 2 (SWT_2)	SWT_2_IR[TIF]
35	Watchdog timer 3 (SWT_3)	SWT_3_IR[TIF]

Table 34. Interrupt sources (continued)

Number	Description	Register bit
36	Platform periodic timer 0 (STM_0)	STM_0_CIR0[CIF]
37		STM_0_CIR1[CIF]
38		STM_0_CIR2[CIF]
39		STM_0_CIR3[CIF]
40	Platform periodic timer 1 (STM_1)	STM_1_CIR0[CIF]
41		STM_1_CIR1[CIF]
42		STM_1_CIR2[CIF]
43		STM_1_CIR3[CIF]
44	Platform periodic timer 2 (STM_2)	STM_2_CIR0[CIF]
45		STM_2_CIR1[CIF]
46		STM_2_CIR2[CIF]
47		STM_2_CIR3[CIF]
48 - 51	Reserved	—
52	eDMA Combined Channel Error Flags 0 - 95	DMA_1_ERRL[INT0 - INT31] :OR: DMA_1_ERRH[INT32 - INT63] :OR: DMA_0_ERRL[INT0 - INT31]

Table 34. Interrupt sources (continued)

Number	Description	Register bit
53	Enhanced Direct Memory Access 1 (eDMA_1)	DMA_1_INTL[INT0]
54		DMA_1_INTL[INT1]
55		DMA_1_INTL[INT2]
56		DMA_1_INTL[INT3]
57		DMA_1_INTL[INT4]
58		DMA_1_INTL[INT5]
59		DMA_1_INTL[INT6]
60		DMA_1_INTL[INT7]
61		DMA_1_INTL[INT8]
62		DMA_1_INTL[INT9]
63		DMA_1_INTL[INT10]
64		DMA_1_INTL[INT11]
65		DMA_1_INTL[INT12]
66		DMA_1_INTL[INT13]
67		DMA_1_INTL[INT14]
68		DMA_1_INTL[INT15]
69		DMA_1_INTL[INT16]
70		DMA_1_INTL[INT17]
71		DMA_1_INTL[INT18]
72		DMA_1_INTL[INT19]
73		DMA_1_INTL[INT20]
74		DMA_1_INTL[INT21]
75		DMA_1_INTL[INT22]
76		DMA_1_INTL[INT23]
77		DMA_1_INTL[INT24]
78		DMA_1_INTL[INT25]
79		DMA_1_INTL[INT26]
80		DMA_1_INTL[INT27]
81		DMA_1_INTL[INT28]
82		DMA_1_INTL[INT29]
83		DMA_1_INTL[INT30]
84		DMA_1_INTL[INT31]

Table 34. Interrupt sources (continued)

Number	Description	Register bit
85	Enhanced Direct Memory Access 1 (eDMA_1)	DMA_1_INTH[INT32]
86		DMA_1_INTH[INT33]
87		DMA_1_INTH[INT34]
88		DMA_1_INTH[INT35]
89		DMA_1_INTH[INT36]
90		DMA_1_INTH[INT37]
91		DMA_1_INTH[INT38]
92		DMA_1_INTH[INT39]
93		DMA_1_INTH[INT40]
94		DMA_1_INTH[INT41]
95		DMA_1_INTH[INT42]
96		DMA_1_INTH[INT43]
97		DMA_1_INTH[INT44]
98		DMA_1_INTH[INT45]
99		DMA_1_INTH[INT46]
100		DMA_1_INTH[INT47]
101		DMA_1_INTH[INT48]
102		DMA_1_INTH[INT49]
103		DMA_1_INTH[INT50]
104		DMA_1_INTH[INT51]
105		DMA_1_INTH[INT52]
106		DMA_1_INTH[INT53]
107		DMA_1_INTH[INT54]
108		DMA_1_INTH[INT55]
109		DMA_1_INTH[INT56]
110		DMA_1_INTH[INT57]
111		DMA_1_INTH[INT58]
112		DMA_1_INTH[INT59]
113		DMA_1_INTH[INT60]
114		DMA_1_INTH[INT61]
115		DMA_1_INTH[INT62]
116		DMA_1_INTH[INT63]

Table 34. Interrupt sources (continued)

Number	Description	Register bit
117	Enhanced Direct Memory Access 0 (eDMA_0)	DMA_0_INTL[INT64]
118		DMA_0_INTL[INT65]
119		DMA_0_INTL[INT66]
120		DMA_0_INTL[INT67]
121		DMA_0_INTL[INT68]
122		DMA_0_INTL[INT69]
123		DMA_0_INTL[INT70]
124		DMA_0_INTL[INT71]
125		DMA_0_INTL[INT72]
126		DMA_0_INTL[INT73]
127		DMA_0_INTL[INT74]
128		DMA_0_INTL[INT75]
129		DMA_0_INTL[INT76]
130		DMA_0_INTL[INT77]
131		DMA_0_INTL[INT78]
132		DMA_0_INTL[INT79]
133		DMA_0_INTL[INT80]
134		DMA_0_INTL[INT81]
135		DMA_0_INTL[INT82]
136		DMA_0_INTL[INT83]
137		DMA_0_INTL[INT84]
138		DMA_0_INTL[INT85]
139		DMA_0_INTL[INT86]
140		DMA_0_INTL[INT87]
141		DMA_0_INTL[INT88]
142		DMA_0_INTL[INT89]
143		DMA_0_INTL[INT90]
144		DMA_0_INTL[INT91]
145		DMA_0_INTL[INT92]
146		DMA_0_INTL[INT93]
147		DMA_0_INTL[INT94]
148		DMA_0_INTL[INT95]
149 - 184	Reserved	—
185	Flash controller Prog/Erase/Suspend IRQ	MCR[DONE]

Table 34. Interrupt sources (continued)

Number	Description	Register bit
186 - 188	Reserved	—
189	AMU DMA interrupt for AMU channel 0	STCFG0[WDMA]
190	AMU DMA interrupt for AMU channel 1	STCFG1[WDMA]
191	AMU Error interrupt for AMU channel 0	STCFG0[EINT]
192	AMU Error interrupt for AMU channel 1	STCFG1[EINT]
193	AMU Calculation interrupt for AMU channel 0	STCFG0[VINT] :OR: STCFG0[OINT] :OR: STCFG0[NINT]
194	AMU Calculation interrupt for AMU channel 1	STCFG1[VINT] :OR: STCFG1[OINT] :OR: STCFG1[NINT]
195 - 211	Reserved	—
212	Ethernet 0 (SYNOPSYS IP)	ETHERNET_0[sbd_intr_o] combined interrupts from all EQOS sources
213		ETHERNET_0[pmt_intr_o]
214		ETHERNET_0[lpi_intr_o]
215 - 217	Reserved	—
218	Ethernet 1 (SYNOPSYS IP)	ETHERNET_1[sbd_intr_o] combined interrupts from all EQOS sources
219		ETHERNET_1[pmt_intr_o]
220		ETHERNET_1[lpi_intr_o]
221 - 223	Reserved	—
224	Real Time Counter (RTC/API)	RTC
225		API wakeup
226	Periodic Interrupt Timer 0 (PIT_0)	PIT_0_TFLG0[TIF]
227		PIT_0_TFLG1[TIF]
228		PIT_0_TFLG2[TIF]
229		PIT_0_TFLG3[TIF]
230		PIT_0_TFLG4[TIF]
231		PIT_0_TFLG5[TIF]
232		PIT_0_TFLG6[TIF]
233		PIT_0_TFLG7[TIF]
234 - 238	Reserved	—
239	Periodic Interrupt Timer 0 (PIT_0)	PIT_0_RTI_TFLG[TIF]
240	Periodic Interrupt Timer 1 (PIT_1)	PIT_1_TFLG0[TIF]
241		PIT_1_TFLG1[TIF]

Table 34. Interrupt sources (continued)

Number	Description	Register bit
242	XOSC counter	XOSC_CTL[I_OSC]
243	SIUL External Interrupt 0	SIUL2_DISR0[EIF0] (EXT_INT 0 signal)
244	SIUL External Interrupt 1	SIUL2_DISR0[EIF1] (EXT_INT 1 signal)
245 - 246	Reserved	—
247	WakeUp unit (WKPU)	WakeUp_IRQ_0
248		WakeUp_IRQ_1
249		WakeUp_IRQ_2
250		WakeUp_IRQ_3
251	Magic Carpet Module - ME 0	ME_IS[I_SAFE]
252	Magic Carpet Module - ME 1	ME_IS[I_MTC]
253	Magic Carpet Module - ME 2	ME_IS[I_IMODE]
254	Magic Carpet Module - ME 3	ME_IS[I_ICONF]
255	Magic Carpet Module - RGM 0	MC_RGM Functional and destructive reset alternate event interrupt
256 - 258	Reserved	—
259	Serial Peripheral Interface 0 (DSPI_0)	DSPI_0_SR[TFUF] :OR: DSPI_0_SR[RFOF] :OR: DSPI_0_SR[TFIWF]
260		DSPI_0_SR[EOQF]
261		DSPI_0_SR[TFFF]
262		DSPI_0_SR[TCF]
263		DSPI_0_SR[RFDF]
264		DSPI_0_SR[CMD_TCF] :OR: DSPI_0_SR[SPI_TCF]
265		DSPI_0_SR[CMDFFF]
266		DSPI_0_SR[SPEF]
267	Reserved	—

Table 34. Interrupt sources (continued)

Number	Description	Register bit
268	Serial Peripheral Interface 1 (DSPI_1)	DSPI_1_SR[TFUF] :OR: DSPI_1_SR[RFOF] :OR: DSPI_1_SR[TFIWF]
269		DSPI_1_SR[EOQF]
270		DSPI_1_SR[TFFF]
271		DSPI_1_SR[TCF]
272		DSPI_1_SR[RFDF]
273		DSPI_1_SR[CMD_TCF] :OR: DSPI_1_SR[SPI_TCF]
274		DSPI_1_SR[CMDFFF]
275		DSPI_1_SR[SPEF]
276	Reserved	—
277	Serial Peripheral Interface 2 (DSPI_2)	DSPI_2_SR[TFUF] :OR: DSPI_2_SR[RFOF] :OR: DSPI_2_SR[TFIWF]
278		DSPI_2_SR[EOQF]
279		DSPI_2_SR[TFFF]
280		DSPI_2_SR[TCF]
281		DSPI_2_SR[RFDF]
282		DSPI_2_SR[CMD_TCF] :OR: DSPI_2_SR[SPI_TCF]
283		DSPI_2_SR[CMDFFF]
284		DSPI_2_SR[SPEF]
285	Reserved	—
286	Serial Peripheral Interface 3 (DSPI_3)	DSPI_3_SR[TFUF] :OR: DSPI_3_SR[RFOF] :OR: DSPI_3_SR[TFIWF]
287		DSPI_3_SR[EOQF]
288		DSPI_3_SR[TFFF]
289		DSPI_3_SR[TCF]
290		DSPI_3_SR[RFDF]
291		DSPI_3_SR[CMD_TCF] :OR: DSPI_3_SR[SPI_TCF]
292		DSPI_3_SR[CMDFFF]
293		DSPI_3_SR[SPEF]
294	Reserved	—

Table 34. Interrupt sources (continued)

Number	Description	Register bit
295	Serial Peripheral Interface 4 (DSPI_4)	DSPI_4_SR[TFUF] :OR: DSPI_4_SR[RFOF] :OR: DSPI_4_SR[TFIWF]
296		DSPI_4_SR[EOQF]
297		DSPI_4_SR[TFFF]
298		DSPI_4_SR[TCF]
299		DSPI_4_SR[RFDF]
300		DSPI_4_SR[CMD_TCF] :OR: DSPI_4_SR[SPI_TCF]
301		DSPI_4_SR[CMDFFF] :OR: DSPI_4_SR[DSI_TCF]
302		DSPI_4_SR[SPEF] :OR: DSPI_4_SR[DPEF]
303		DSPI_4_SR[DDIF]
304	Serial Peripheral Interface 5 (DSPI_5)	DSPI_5_SR[TFUF] :OR: DSPI_5_SR[RFOF] :OR: DSPI_5_SR[TFIWF]
305		DSPI_5_SR[EOQF]
306		DSPI_5_SR[TFFF]
307		DSPI_5_SR[TCF]
308		DSPI_5_SR[RFDF]
309		DSPI_5_SR[CMD_TCF] :OR: DSPI_5_SR[SPI_TCF]
310		DSPI_5_SR[CMDFFF] :OR: DSPI_5_SR[DSI_TCF]
311		DSPI_5_SR[SPEF] :OR: DSPI_5_SR[DPEF]
312		DSPI_5_SR[DDIF]
313	Serial Peripheral Interface 6 (DSPI_6)	DSPI_6_SR[TFUF] :OR: DSPI_6_SR[RFOF] :OR: DSPI_6_SR[TFIWF]
314		DSPI_6_SR[EOQF]
315		DSPI_6_SR[TFFF]
316		DSPI_6_SR[TCF]
317		DSPI_6_SR[RFDF]
318		DSPI_6_SR[CMD_TCF] :OR: DSPI_6_SR[DSI_TCF]
319		DSPI_6_SR[CMDFFF]
320		DSPI_6_SR[SPEF]
321	Reserved	—

Table 34. Interrupt sources (continued)

Number	Description	Register bit
322	Serial Peripheral Interface 7 (DSPI_7)	DSPI_7_SR[TFUF] :OR: DSPI_7_SR[RFOF] :OR: DSPI_7_SR[TFIWF]
323		DSPI_7_SR[EOQF]
324		DSPI_7_SR[TFFF]
325		DSPI_7_SR[TCF]
326		DSPI_7_SR[RFDF]
327		DSPI_7_SR[CMD_TCF] :OR: DSPI_7_SR[DSI_TCF]
328		DSPI_7_SR[CMDFFF]
329		DSPI_7_SR[SPEF]
330 - 330	Reserved	—
331	Serial Peripheral Interface 8 (DSPI_8)	DSPI_8_SR[TFUF] :OR: DSPI_8_SR[RFOF] :OR: DSPI_8_SR[TFIWF]
332		DSPI_8_SR[EOQF]
333		DSPI_8_SR[TFFF]
334		DSPI_8_SR[TCF]
335		DSPI_8_SR[RFDF]
336		DSPI_8_SR[CMD_TCF] :OR: DSPI_8_SR[DSI_TCF]
337		DSPI_8_SR[CMDFFF]
338		DSPI_8_SR[SPEF]
339	Reserved	—
340	Serial Peripheral Interface 9 (DSPI_9)	DSPI_9_SR[TFUF] :OR: DSPI_9_SR[RFOF] :OR: DSPI_9_SR[TFIWF]
341		DSPI_9_SR[EOQF]
342		DSPI_9_SR[TFFF]
343		DSPI_9_SR[TCF]
344		DSPI_9_SR[RFDF]
345		DSPI_9_SR[CMD_TCF] :OR: DSPI_9_SR[SPI_TCF]
346		DSPI_9_SR[CMDFFF]
347		DSPI_9_SR[SPEF]
348 - 375	Reserved	—

Table 34. Interrupt sources (continued)

Number	Description	Register bit
376	UART/LIN 0 (LINFlexD_0)	LINFlexD_0_RX
377		LINFlexD_0_TX
378		LINFlexD_0_ERR
379	Reserved	—
380	UART/LIN 1 (LINFlexD_1)	LINFlexD_1_RX
381		LINFlexD_1_TX
382		LINFlexD_1_ERR
383	Reserved	—
384	UART/LIN 2 (LINFlexD_2)	LINFlexD_2_RX
385		LINFlexD_2_TXI
386		LINFlexD_2_ERR
387	Reserved	—
388	UART/LIN 3 (LINFlexD_3)	LINFlexD_3_RX
389		LINFlexD_3_TX
390		LINFlexD_3_ERR
391	Reserved	—
392	UART/LIN 4 (LINFlexD_4)	LINFlexD_4_RX
393		LINFlexD_4_TX
394		LINFlexD_4_ERR
395	Reserved	—
396	UART/LIN 5 (LINFlexD_5)	LINFlexD_5_RX
397		LINFlexD_5_TX
398		LINFlexD_5_ERR
399	Reserved	—
400	UART/LIN 6 (LINFlexD_6)	LINFlexD_6_RX
401		LINFlexD_6_TX
402		LINFlexD_6_ERR
403	Reserved	—
404	UART/LIN 7 (LINFlexD_7)	LINFlexD_7_RX
405		LINFlexD_7_TX
406		LINFlexD_7_ERR
407	Reserved	—
408	UART/LIN 8 (LINFlexD_8)	LINFlexD_8_RX
409		LINFlexD_8_TX
410		LINFlexD_8_ERR

Table 34. Interrupt sources (continued)

Number	Description	Register bit
411	Reserved	—
412	UART/LIN 9 (LINFlexD_9)	LINFlexD_9_RX
413		LINFlexD_9_TX
414		LINFlexD_9_ERR
415	Reserved	—
416	UART/LIN 10 (LINFlexD_10)	LINFlexD_10_RX
417		LINFlexD_10_TX
418		LINFlexD_10_ERR
419	Reserved	—
420	UART/LIN 11 (LINFlexD_11)	LINFlexD_11_RX
421		LINFlexD_11_TX
422		LINFlexD_11_ERR
423	Reserved	—
424	UART/LIN 12 (LINFlexD_12)	LINFlexD_12_RX
425		LINFlexD_12_TXI
426		LINFlexD_12_ERR
427	Reserved	—
428	UART/LIN 13 (LINFlexD_13)	LINFlexD_13_RX
429		LINFlexD_13_TX
430		LINFlexD_13_ERR
431	Reserved	—
432	UART/LIN 14 (LINFlexD_14)	LINFlexD_14_RX
433		LINFlexD_14_TX
434		LINFlexD_14_ERR
435	Reserved	—
436	UART/LIN 15 (LINFlexD_15)	LINFlexD_15_RX
437		LINFlexD_15_TX
438		LINFlexD_15_ERR
439	Reserved	—
440	I2C_0	I2C0_SR[IBAL] :OR: I2C0_SR[TCF] :OR: I2C0_SR[IAAS]
441	Reserved	—
442	UART/LIN 16 (LINFlexD_16)	LINFlexD_16_RX
443		LINFlexD_16_TX
444		LINFlexD_16_ERR

Table 34. Interrupt sources (continued)

Number	Description	Register bit
445 - 445	Reserved	—
446	UART/LIN 17 (LINFlexD_17)	LINFlexD_17_RX
447		LINFlexD_17_TX
448		LINFlexD_17_ERR
449 - 452	Reserved	—
453	FlexRay_0	FR_0_LRNEIF:OR: FR_0_DNREIF
454		FR_0_LRCEIF :OR: FR_0_DRCEIF
455		FR_0_FNEAIF
456		FR_0_FNEBIF
457		FR_0_WUPIF
458		FR_0_PRIF
459		FR_0_CHIF
460		FR_0_TBIF
461		FR_0_RBIF
462		FR_0_MIF
463 - 476	Reserved	—
477	Power Monitor Unit	OR-ed GR_S voltage detector flags
478	Power management Unit (temperature sensor)	OR-ed EPR_TD temperature detector flags
479	XOSC32k counter	counter expired
480 - 487	Reserved	—
488	FCCU	FCCU_IRQ_STAT[ALRM_STAT]
489		FCCU_IRQ_STAT[CFG_TO_STAT]
490		FCCU_SCFS[RCCS0]
491		FCCU_SCFS[RCCS1]
492 - 493	Reserved	—
494	STCU	STCU_RUNSW[LBIE]
495		STCU_RUNSW[MBIE]

Table 34. Interrupt sources (continued)

Number	Description	Register bit
496	HSM	HSM2HTIE[0]
497		HSM2HTIE[1]
498		HSM2HTIE[2]
499		HSM2HTIE[3]
500		HSM2HTIE[4]
501		HSM2HTIE[5]
502		HSM2HTIE[6]
503		HSM2HTIE[7]
504		HSM2HTIE[8]
505		HSM2HTIE[9]
506		HSM2HTIE[10]
507		HSM2HTIE[11]
508		HSM2HTIE[12]
509		HSM2HTIE[13]
510		HSM2HTIE[14]
511		HSM2HTIE[15]
512		HSM2HTIE[16]
513		HSM2HTIE[17]
514		HSM2HTIE[18]
515		HSM2HTIE[19]
516		HSM2HTIE[20]
517		HSM2HTIE[21]
518		HSM2HTIE[22]
519		HSM2HTIE[23]
520		HSM2HTIE[24]
521		HSM2HTIE[25]
522		HSM2HTIE[26]
523		HSM2HTIE[27]
524		HSM2HTIE[28]
525		HSM2HTIE[29]
526		HSM2HTIE[30]
527		HSM2HTIE[31]

Table 34. Interrupt sources (continued)

Number	Description	Register bit
528	SAR_ADC_12bit_0	combined interrupt signal: SARADC12_0 [NEOC] :OR: SARADC12_0 [NECH] :OR: SARADC12_0 [JEOC] :OR: SARADC12_0 [JECH]:OR: SARADC12_0 [EOCTU] :OR: SARADC12_0 [WDGxL] :OR: SARADC12_0 [WDGxH]
529	SAR_ADC_12bit_1	combined interrupt signal: SARADC12_1 [NEOC] :OR: SARADC12_1 [NECH] :OR: SARADC12_1 [JEOC] :OR: SARADC12_1 [JECH]:OR: SARADC12_1 [EOCTU] :OR: SARADC12_1 [WDGxL] :OR: SARADC12_1 [WDGxH]
530	SAR_ADC_12bit_2	combined interrupt signal: SARADC12_2 [NEOC] :OR: SARADC12_2 [NECH] :OR: SARADC12_2 [JEOC] :OR: SARADC12_2 [JECH]:OR: SARADC12_2 [EOCTU] :OR: SARADC12_2 [WDGxL] :OR: SARADC12_2 [WDGxH]
531	SAR_ADC_12bit_3	combined interrupt signal: SARADC12_3 [NEOC] :OR: SARADC12_3 [NECH] :OR: SARADC12_3 [JEOC] :OR: SARADC12_3 [JECH]:OR: SARADC12_3 [EOCTU] :OR: SARADC12_3 [WDGxL] :OR: SARADC12_3 [WDGxH]
532 - 542	Reserved	—
543	SAR_ADC_12bit_SUPERVISOR	combined interrupt signal: SARADC12_SV [NEOC] :OR: SARADC12_SV [NECH] :OR: SARADC12_SV [JEOC] :OR: SARADC12_SV [JECH]:OR: SARADC12_SV [EOCTU] :OR: SARADC12_SV [WDGxL] :OR: SARADC12_SV [WDGxH]
544	SD_ADC_0	SD_ADC_0_SFR[DFEF] :OR: SD_ADC_0_SFR[DFORF] :OR: SD_ADC_0_SFR[DFFF]
545	SD_ADC_1	SD_ADC_1_SFR[DFEF] :OR: SD_ADC_1_SFR[DFORF] :OR: SD_ADC_1_SFR[DFFF]

Table 34. Interrupt sources (continued)

Number	Description	Register bit
546	SD_ADC_2	SD_ADC_2_SFR[DFEF] :OR: SD_ADC_2_SFR[DFORF] :OR: SD_ADC_2_SFR[DFFF]
547	SD_ADC_3	SD_ADC_3_SFR[DFEF] :OR: SD_ADC_3_SFR[DFORF] :OR: SD_ADC_3_SFR[DFFF]
548	SD_ADC_4	SD_ADC_4_SFR[DFEF] :OR: SD_ADC_4_SFR[DFORF] :OR: SD_ADC_4_SFR[DFFF]
549	SD_ADC_5	SD_ADC_5_SFR[DFEF] :OR: SD_ADC_5_SFR[DFORF] :OR: SD_ADC_5_SFR[DFFF]
550 - 553	Reserved	—
554	SAR_ADC_10bit_0	combined interrupt signal: SARADC10_0 [NEOC] :OR: SARADC10_0 [NECH] :OR: SARADC10_0 [JEOC] :OR: SARADC10_0 [JECH]:OR: SARADC10_0 [EOCTU] :OR: SARADC10_0 [WDGxL] :OR: SARADC10_0 [WDGxH]
555	SAR_ADC_10bit_1	combined interrupt signal: SARADC10_1 [NEOC] :OR: SARADC10_1 [NECH] :OR: SARADC10_1 [JEOC] :OR: SARADC10_1 [JECH]:OR: SARADC10_1 [EOCTU] :OR: SARADC10_1 [WDGxL] :OR: SARADC10_1 [WDGxH]
556 - 556	Reserved	—
557	SAR_ADC_10bit_STDBY	combined interrupt signal: SARADC10_SB [NEOC] :OR: SARADC10_SB [NECH] :OR: SARADC10_SB [JEOC] :OR: SARADC10_SB [JECH]:OR: SARADC10_SB [EOCTU] :OR: SARADC10_SB [WDGxL] :OR: SARADC10_SB [WDGxH]

Table 34. Interrupt sources (continued)

Number	Description	Register bit
558	SENT_0	SENT combined fast (SENT_0_FMSG_RDY[0] :OR: SENT_0_FMSG_RDY[1] :OR: SENT_0_FMSG_RDY[2] :OR: SENT_0_FMSG_RDY[3] :OR: SENT_0_FMSG_RDY[4] :OR: SENT_0_FMSG_RDY[5] :OR: SENT_0_FMSG_RDY[6] :OR: SENT_0_FMSG_RDY[7])
559		SENT combined slow (SENT_0_SMSG_RDY[0] :OR: SENT_0_SMSG_RDY[1] :OR: SENT_0_SMSG_RDY[2] :OR: SENT_0_SMSG_RDY[3] :OR: SENT_0_SMSG_RDY[4] :OR: SENT_0_SMSG_RDY[5] :OR: SENT_0_SMSG_RDY[6] :OR: SENT_0_SMSG_RDY[7])
560		SENT combined error (SENT_0_GBL_STATUS[FMDU] :OR: SENT_0_GBL_STATUS[SMDU] :OR: SENT_0_CHn_STATUS[4:7] :OR: SENT_0_CHn_STATUS[9:15])
561	SENT_1	SENT combined fast (SENT_1_FMSG_RDY[0] :OR: SENT_1_FMSG_RDY[1] :OR: SENT_1_FMSG_RDY[2] :OR: SENT_1_FMSG_RDY[3]] :OR: SENT_1_FMSG_RDY[4] :OR: SENT_1_FMSG_RDY[5]] :OR: SENT_1_FMSG_RDY[6])
562		SENT combined slow (SENT_1_SMSG_RDY[0] :OR: SENT_1_SMSG_RDY[1] :OR: SENT_1_SMSG_RDY[2] :OR: SENT_1_SMSG_RDY[3] :OR: SENT_1_SMSG_RDY[4] :OR: SENT_1_SMSG_RDY[5] :OR: SENT_1_SMSG_RDY[6])
563		SENT combined error (SENT_1_GBL_STATUS[FMDU] :OR: SENT_1_GBL_STATUS[SMDU] :OR: SENT_1_CHn_STATUS[4:7] :OR: SENT_1_CHn_STATUS[9:15])

Table 34. Interrupt sources (continued)

Number	Description	Register bit
564	SENT_0	Channel 0 fast SENT_0_FMSG_RDY[0]
565		Channel 0 slow SENT_0_SMSG_RDY[0]
566		Channel 0 error SENT_0_CH0_STATUS[4:7] :OR: SENT_0_CH0_STATUS[9:15]
567		Channel 1 fast SENT_0_FMSG_RDY[1]
568		Channel 1 slow SENT_0_SMSG_RDY[1]
569		Channel 1 error SENT_0_CH1_STATUS[4:7] :OR: SENT_0_CH1_STATUS[9:15]
570		Channel 2 fast SENT_0_FMSG_RDY[2]
571		Channel 2 slow SENT_0_SMSG_RDY[2]
572		Channel 2 error SENT_0_CH2_STATUS[4:7] :OR: SENT_0_CH2_STATUS[9:15]
573		Channel 3 fast SENT_0_FMSG_RDY[3]
574		Channel 3 slow SENT_0_SMSG_RDY[3]
575		Channel 3 error SENT_0_CH3_STATUS[4:7] :OR: SENT_0_CH3_STATUS[9:15]
576		Channel 4 fast SENT_0_FMSG_RDY[4]
577		Channel 4 slow SENT_0_SMSG_RDY[4]
578		Channel 4 error SENT_0_CH4_STATUS[4:7] :OR: SENT_0_CH4_STATUS[9:15]

Table 34. Interrupt sources (continued)

Number	Description	Register bit
579	SENT_1	Channel 0 fast SENT_1_FMSG_RDY[0]
580		Channel 0 slow SENT_1_SMSG_RDY[0]
581		Channel 0 error SENT_1_CH0_STATUS[4:7] :OR: SENT_1_CH0_STATUS[9:15]
582		Channel 1 fast SENT_1_FMSG_RDY[1]
583		Channel 1 slow SENT_1_SMSG_RDY[1]
584		Channel 1 error SENT_1_CH1_STATUS[4:7] :OR: SENT_1_CH1_STATUS[9:15]
585		Channel 2 fast SENT_1_FMSG_RDY[2]
586		Channel 2 slow SENT_1_SMSG_RDY[2]
587		Channel 2 error SENT_1_CH2_STATUS[4:7] :OR: SENT_1_CH2_STATUS[9:15]
588		Channel 3 fast SENT_1_FMSG_RDY[3]
589		Channel 3 slow SENT_1_SMSG_RDY[3]
590		Channel 3 error SENT_1_CH3_STATUS[4:7] :OR: SENT_1_CH3_STATUS[9:15]
591		Channel 4 fast SENT_1_FMSG_RDY[4]
592		Channel 4 slow SENT_1_SMSG_RDY[4]
593		Channel 4 error SENT_1_CH4_STATUS[4:7] :OR: SENT_1_CH4_STATUS[9:15]

Table 34. Interrupt sources (continued)

Number	Description	Register bit
594	SENT_0	Channel 5 fast SENT_0_FMSG_RDY[5]
595		Channel 5 slow SENT_0_SMSG_RDY[5]
596		Channel 5 error SENT_0_CH4_STATUS[4:7] :OR: SENT_0_CH4_STATUS[9:15]
597		Channel 6 fast SENT_0_FMSG_RDY[6]
598		Channel 6 slow SENT_0_SMSG_RDY[6]
599		Channel 6 error SENT_0_CH4_STATUS[4:7] :OR: SENT_0_CH4_STATUS[9:15]
600		Channel 7 fast SENT_0_FMSG_RDY[7]
601		Channel 7 slow SENT_0_SMSG_RDY[7]
602		Channel 7 error SENT_0_CH4_STATUS[4:7] :OR: SENT_0_CH4_STATUS[9:15]
603	SENT_1	Channel 5 fast SENT_1_FMSG_RDY[5]
604		Channel 5 slow SENT_1_SMSG_RDY[5]
605		Channel 5 error SENT_1_CH4_STATUS[4:7] :OR: SENT_1_CH4_STATUS[9:15]
606		Channel 6 fast SENT_1_FMSG_RDY[6]
607		Channel 6 slow SENT_1_SMSG_RDY[6]
608		Channel 6 error SENT_1_CH4_STATUS[4:7] :OR: SENT_1_CH4_STATUS[9:15]
609 - 611	Reserved	—

Table 34. Interrupt sources (continued)

Number	Description	Register bit
612	PSI5_0 CH_0	Channel 0 DMA combined interrupt: PSI5_0_CH_0_DSR[IS_DMA_TF_PM_DS] :OR: PSI5_0_CH_0_DSR[IS_DMA_TF_SF] :OR: PSI5_0_CH_0_DSR[IS_DMA_PM_DS_FIFO_FULL] :OR: PSI5_0_CH_0_DSR[IS_DMA_SFUF] :OR: PSI5_0_CH_0_DSR[IS_DMA_PM_DS_UF]
613		Channel 0 combined general interrupt: PSI5_0_CH_0_GISR[IS_CESM[6:1]] :OR: PSI5_0_CH_0_GISR[IS_STS] :OR: PSI5_0_CH_0_GISR[IS_DTS] :OR: PSI5_0_CH_0_GISR[IS_DSROW] :OR: PSI5_0_CH_0_GISR[IS_BROW] :OR: PSI5_0_CH_0_GISR[DSR_RDY] :OR: PSI5_0_CH_0_GISR[DBR_RDY] :OR: PSI5_0_CH_0_GISR[DPR_RDY] :OR: PSI5_0_CH_0_GISR[IS_OWSM[6:1]] :OR: PSI5_0_CH_0_GISR[IS_NVSM[6:1]]
614		Channel 0 new message combined interrupt: PSI5_0_CH_0_NDSR[NDS31] :OR: PSI5_0_CH_0_NDSR[NDS30] :OR: … :OR: PSI5_0_CH_0_NDSR[NDS1] :OR: PSI5_0_CH_0_NDSR[NDS0]
615		Channel 0 message overwrite: PSI5_0_CH_0_OWSR[NDS31] :OR: PSI5_0_CH_0_OWSR[NDS30] :OR: … :OR: PSI5_0_CH_0_OWSR[NDS1] :OR: PSI5_0_CH_0_OWSR[NDS0]
616		Channel 0 error combined interrupt: PSI5_0_CH_0_EISR[NDS31] :OR: PSI5_0_CH_0_EISR[NDS30] :OR: … :OR: PSI5_0_CH_0_EISR[NDS1] :OR: PSI5_0_CH_0_EISR[NDS0]
617		Channel 0 global interrupt: PSI5_0_CH_0_DSR[*] :OR: PSI5_0_CH_0_GISR[*] :OR: PSI5_0_CH_0_NDSR[31:0] :OR: PSI5_0_CH_0_OWSR[31:0] :OR: PSI5_0_CH_0_EISR[31:0]
618 - 623	Reserved	—

Table 34. Interrupt sources (continued)

Number	Description	Register bit
624	PSI5_1 CH_0	Channel 0 DMA combined interrupt: PSI5_1_CH_0_DSR[IS_DMA_TF_PM_DS] :OR: PSI5_1_CH_0_DSR[IS_DMA_TF_SF] :OR: PSI5_1_CH_0_DSR[IS_DMA_PM_DS_FIFO_FULL] :OR: PSI5_1_CH_0_DSR[IS_DMA_SFUF] :OR: PSI5_1_CH_0_DSR[IS_DMA_PM_DS_UF]
625		Channel 0 combined general interrupt: PSI5_1_CH_0_GISR[IS_CESM[6:1]] :OR: PSI5_1_CH_0_GISR[IS_STS] :OR: PSI5_1_CH_0_GISR[IS_DTS] :OR: PSI5_1_CH_0_GISR[[IS_DSROW] :OR: PSI5_1_CH_0_GISR[IS_BROW] :OR: PSI5_1_CH_0_GISR[[DSR_RDY] :OR: PSI5_1_CH_0_GISR[DBR_RDY] :OR: PSI5_1_CH_0_GISR[DPR_RDY] :OR: PSI5_1_CH_0_GISR[IS_OWSM[6:1]] :OR: PSI5_1_CH_0_GISR[IS_NVSM[6:1]]
626		Channel 0 new message combined interrupt: PSI5_1_CH_0_NDSR[NDS31] :OR: PSI5_1_CH_0_NDSR[NDS30] :OR: … :OR: PSI5_1_CH_0_NDSR[NDS1] :OR: PSI5_1_CH_0_NDSR[NDS0]
627		Channel 0 message overwrite: PSI5_1_CH_0_OWSR[NDS31] :OR: PSI5_1_CH_0_OWSR[NDS30] :OR: … :OR: PSI5_1_CH_0_OWSR[NDS1] :OR: PSI5_1_CH_0_OWSR[NDS0]
628		Channel 0 error combined interrupt: PSI5_1_CH_0_EISR[NDS31] :OR: PSI5_1_CH_0_EISR[NDS30] :OR: … :OR: PSI5_1_CH_0_EISR[NDS1] :OR: PSI5_1_CH_0_EISR[NDS0]
629		Channel 0 global interrupt: PSI5_1_CH_0_DSR[*] :OR: PSI5_1_CH_0_GISR[*] :OR: PSI5_1_CH_0_NDSR[31:0] :OR: PSI5_1_CH_0_OWSR[31:0] :OR: PSI5_1_CH_0_EISR[31:0]
630 - 653	Reserved	—

Table 34. Interrupt sources (continued)

Number	Description	Register bit
654	SIPI_0 (interprocessor communication)	combined error interrupt: SIPI_ERR[TOEn] :OR: SIPI_ERR[TIDEn] :OR: SIPI_ERR[ACKEn]
655		CRC error interrupt: SIPI_SR[GCRCE)
656		Channel 0 receiption combined interrupt: SIPI_CSR0[RAR]:OR: SIPI_CSR0[ACKR]
657		Channel 1 receiption combined interrupt: SIPI_CSR1[RAR]:OR: SIPI_CSR1[ACKR]
658		Channel 2 receiption combined interrupt: SIPI_CSR2[RAR]:OR: SIPI_CSR2[ACKR]
659		Channel 3 receiption combined interrupt: SIPI_CSR3[RAR]:OR: SIPI_CSR3[ACKR]
660		combined event interrupt: SIPI_SR[TE0]:OR: SIPI_SR[TE1]:OR:SIPI_SR[TE2]:OR:SIPI_SR[TE3]:OR:SIPI_SR[MCR]

Table 34. Interrupt sources (continued)

Number	Description	Register bit
661	LFAST_0 (interprocessor communication)	transmit complete combined interrupt: LFAST_0_TISR[TXPNGF] :OR: LFAST_0_TISR[TXUNSF] :OR: LFAST_0_TISR[TXICLCF] :OR: LFAST_0_TISR[TXDTF]
662		transmit error combined interrupt: LFAST_0_TISR[TXIEF] :OR: LFAST_0_TISR[TXOVF]
663		receive combined interrupt: LFAST_0_RISR[RXCTSF] :OR: LFAST_0_RISR[RXDF] :OR: LFAST_0_RISR[RXUNSF]
664		receive error interrupt: LFAST_0_RISR[RXUOF] :OR: LFAST_0_RISR[RXMNF] :OR: LFAST_0_RISR[RXMXF] :OR: LFAST_0_RISR[RXUFF] :OR: LFAST_0_RISR[RXOFF] :OR: LFAST_0_RISR[RXSZF] :OR: LFAST_0_RISR[RXICF] :OR: LFAST_0_RISR[RXLCEF]
665		ICLC receive combined interrupt: LFAST_0_RIISR[ICPFF] :OR: LFAST_0_RIISR[ICPSF] :OR: LFAST_0_RIISR[ICPRF] :OR: LFAST_0_RIISR[ICTOF] :OR: LFAST_0_RIISR[ICLPF] :OR: LFAST_0_RIISR[ICCTF] :OR: LFAST_0_RIISR[ICTDF] :OR: LFAST_0_RIISR[ICTEF] :OR: LFAST_0_RIISR[ICRFF] :OR: LFAST_0_RIISR[ICRSF] :OR: LFAST_0_RIISR[ICTFF] :OR: LFAST_0_RIISR[ICTSF] :OR: LFAST_0_RIISR[ICPOFF] :OR: LFAST_0_RIISR[ICPONF]
666 - 673	Reserved	—
674	JTAGM	JTAGM_SR[SPU_INT] :OR: JTAGM_SR[IDLE]
675	JDC	JDC_MSR[JIN_INT] :OR: JDC_MSR[JOUT_INT]
676	Reserved	—

Table 34. Interrupt sources (continued)

Number	Description	Register bit
677	CAN_SUB_0_M_TTCAN_0	Line 0 combined interrupt: (M_TTCAN_0_IR[0] & ~M_TTCAN_0_ILS[0]):OR: ... :OR: (M_TTCAN_0_IR[11] & ~M_TTCAN_0_ILS[11]) :OR: (M_TTCAN_0_IR[13] & ~M_TTCAN_0_ILS[13]) :OR: … :OR: (M_TTCAN_0_IR[31] & ~M_TTCAN_0_ILS[31])
678		Line 1 combined interrupt: (M_TTCAN_0_IR[0] & M_TTCAN_0_ILS[0]):OR: ... :OR: (M_TTCAN_0_IR[11] & M_TTCAN_0_ILS[11]) :OR: (M_TTCAN_0_IR[13] & M_TTCAN_0_ILS[13]) :OR: … :OR: (M_TTCAN_0_IR[31] & M_TTCAN_0_ILS[31])
679		Register Time Mark interrupt: M_TTCAN_0_TTIR[RTMI]
680 - 687	Reserved	—
688	CAN_SUB_0_M_CAN_1	Line 0 combined interrupt CAN_SUBSYSTEM_0: (M_CAN_1_IR[0] & ~M_CAN_1_ILS[0]):OR: ... :OR: (M_CAN_1_IR[11] & ~M_CAN_1_ILS[11]) :OR: (M_CAN_1_IR[13] & ~M_CAN_1_ILS[13]) :OR: … :OR: (M_CAN_1_IR[31] & ~M_CAN_1_ILS[31])
689		Line 1 combined interrupt CAN_SUBSYSTEM_0: (M_CAN_1_IR[0] & M_CAN_1_ILS[0]):OR: ... :OR: (M_CAN_1_IR[11] & M_CAN_1_ILS[11]) :OR: (M_CAN_1_IR[13] & M_CAN_1_ILS[13]) :OR: … :OR: (M_CAN_1_IR[31] & M_CAN_1_ILS[31])

Table 34. Interrupt sources (continued)

Number	Description	Register bit
690	CAN_SUB_0_M_CAN_2	Line 0 combined interrupt CAN_SUBSYSTEM_0: (M_CAN_2_IR[0] & ~M_CAN_2_ILS[0]):OR: ... :OR: (M_CAN_2_IR[11] & ~M_CAN_2_ILS[11]) :OR: (M_CAN_2_IR[13] & ~M_CAN_2_ILS[13]) :OR: … :OR: (M_CAN_2_IR[31] & ~M_CAN_2_ILS[31])
691		Line 1 combined interrupt CAN_SUBSYSTEM_0: (M_CAN_2_IR[0] & M_CAN_2_ILS[0]):OR: ... :OR: (M_CAN_2_IR[11] & M_CAN_2_ILS[11]) :OR: (M_CAN_2_IR[13] & M_CAN_2_ILS[13]) :OR: … :OR: (M_CAN_2_IR[31] & M_CAN_2_ILS[31])
692	CAN_SUB_0_M_CAN_3	Line 0 combined interrupt CAN_SUBSYSTEM_0: (M_CAN_3_IR[0] & ~M_CAN_3_ILS[0]):OR: ... :OR: (M_CAN_3_IR[11] & ~M_CAN_3_ILS[11]) :OR: (M_CAN_3_IR[13] & ~M_CAN_3_ILS[13]) :OR: … :OR: (M_CAN_3_IR[31] & ~M_CAN_3_ILS[31])
693		Line 1 combined interrupt CAN_SUBSYSTEM_0: (M_CAN_3_IR[0] & M_CAN_3_ILS[0]):OR: ... :OR: (M_CAN_3_IR[11] & M_CAN_3_ILS[11]) :OR: (M_CAN_3_IR[13] & M_CAN_3_ILS[13]) :OR: … :OR: (M_CAN_3_IR[31] & M_CAN_3_ILS[31])
694	CAN_SUB_1_M_CAN_1	Line 0 combined interrupt CAN_SUBSYSTEM_1: (M_CAN_1_IR[0] & ~M_CAN_1_ILS[0]):OR: ... :OR: (M_CAN_1_IR[11] & ~M_CAN_1_ILS[11]) :OR: (M_CAN_1_IR[13] & ~M_CAN_1_ILS[13]) :OR: … :OR: (M_CAN_1_IR[31] & ~M_CAN_1_ILS[31])
695		Line 1 combined interrupt CAN_SUBSYSTEM_1: (M_CAN_1_IR[0] & M_CAN_1_ILS[0]):OR: ... :OR: (M_CAN_1_IR[11] & M_CAN_1_ILS[11]) :OR: (M_CAN_1_IR[13] & M_CAN_1_ILS[13]) :OR: … :OR: (M_CAN_1_IR[31] & M_CAN_1_ILS[31])

Table 34. Interrupt sources (continued)

Number	Description	Register bit
696	CAN_SUB_1_M_CAN_2	Line 0 combined interrupt CAN_SUBSYSTEM_1: (M_CAN_2_IR[0] & ~M_CAN_2_ILS[0]):OR: ... :OR: (M_CAN_2_IR[11] & ~M_CAN_2_ILS[11]) :OR: (M_CAN_2_IR[13] & ~M_CAN_2_ILS[13]) :OR: … :OR: (M_CAN_2_IR[31] & ~M_CAN_2_ILS[31])
697		Line 1 combined interrupt CAN_SUBSYSTEM_1: (M_CAN_2_IR[0] & M_CAN_2_ILS[0]):OR: ... :OR: (M_CAN_2_IR[11] & M_CAN_2_ILS[11]) :OR: (M_CAN_2_IR[13] & M_CAN_2_ILS[13]) :OR: … :OR: (M_CAN_2_IR[31] & M_CAN_2_ILS[31])
698	CAN_SUB_1_M_CAN_3	Line 0 combined interrupt CAN_SUBSYSTEM_1: (M_CAN_3_IR[0] & ~M_CAN_3_ILS[0]):OR: ... :OR: (M_CAN_3_IR[11] & ~M_CAN_3_ILS[11]) :OR: (M_CAN_3_IR[13] & ~M_CAN_3_ILS[13]) :OR: … :OR: (M_CAN_3_IR[31] & ~M_CAN_3_ILS[31])
699		Line 1 combined interrupt CAN_SUBSYSTEM_1: (M_CAN_3_IR[0] & M_CAN_3_ILS[0]):OR: ... :OR: (M_CAN_3_IR[11] & M_CAN_3_ILS[11]) :OR: (M_CAN_3_IR[13] & M_CAN_3_ILS[13]) :OR: … :OR: (M_CAN_3_IR[31] & M_CAN_3_ILS[31])
700	CAN_SUB_1_M_CAN_4	Line 0 combined interrupt CAN_SUBSYSTEM_1: (M_CAN_4_IR[0] & ~M_CAN_4_ILS[0]):OR: ... :OR: (M_CAN_4_IR[11] & ~M_CAN_4_ILS[11]) :OR: (M_CAN_4_IR[13] & ~M_CAN_4_ILS[13]) :OR: … :OR: (M_CAN_4_IR[31] & ~M_CAN_4_ILS[31])
701		Line 1 combined interrupt CAN_SUBSYSTEM_1: (M_CAN_4_IR[0] & M_CAN_4_ILS[0]):OR: ... :OR: (M_CAN_4_IR[11] & M_CAN_4_ILS[11]) :OR: (M_CAN_4_IR[13] & M_CAN_4_ILS[13]) :OR: … :OR: (M_CAN_4_IR[31] & M_CAN_4_ILS[31])
702 - 705	Reserved	—
706	GTM_AEI	GTM_ICM_IRQG_0[AEI_IRQ]

Table 34. Interrupt sources (continued)

Number	Description	Register bit
707	GTM_ARU	GTM_ICM_IRQG_0[ARU_NEW_DATA0_IRQ]
708		GTM_ICM_IRQG_0[ARU_NEW_DATA1_IRQ]
709		GTM_ICM_IRQG_0[ARU_ACC_ACK_IRQ]
710	GTM_BRC	GTM_ICM_IRQG_0[BRC_IRQ]
711	GTM_CMP	GTM_ICM_IRQG_0[CMP_IRQ]
712	GTM_SPE_0	GTM_ICM_IRQG_0[SPE0_IRQ]
713	GTM_SPE_1	GTM_ICM_IRQG_0[SPE1_IRQ]
714	GTM_PSM_0	GTM_ICM_IRQG_0[PSM0_CH0_IRQ]
715		GTM_ICM_IRQG_0[PSM0_CH1_IRQ]
716		GTM_ICM_IRQG_0[PSM0_CH2_IRQ]
717		GTM_ICM_IRQG_0[PSM0_CH3_IRQ]
718		GTM_ICM_IRQG_0[PSM0_CH4_IRQ]
719		GTM_ICM_IRQG_0[PSM0_CH5_IRQ]
720		GTM_ICM_IRQG_0[PSM0_CH6_IRQ]
721		GTM_ICM_IRQG_0[PSM0_CH7_IRQ]

Table 34. Interrupt sources (continued)

Number	Description	Register bit
722	GTM_DPLL	GTM_ICM_IRQG_1[DPLL_DCGI_IRQ]
723		GTM_ICM_IRQG_1[DPLL_EDI_IRQ]
724		GTM_ICM_IRQG_1[DPLL_TINI_IRQ]
725		GTM_ICM_IRQG_1[DPLL_TAXI_IRQ]
726		GTM_ICM_IRQG_1[DPLL_SISI_IRQ]
727		GTM_ICM_IRQG_1[DPLL_TISI_IRQ]
728		GTM_ICM_IRQG_1[DPLL_MSI_IRQ]
729		GTM_ICM_IRQG_1[DPLL_MTI_IRQ]
730		GTM_ICM_IRQG_1[DPLL_SASI_IRQ]
731		GTM_ICM_IRQG_1[DPLL_TASI_IRQ]
732		GTM_ICM_IRQG_1[DPLL_PWI_IRQ]
733		GTM_ICM_IRQG_1[DPLL_W2I_IRQ]
734		GTM_ICM_IRQG_1[DPLL_W1I_IRQ]
735		GTM_ICM_IRQG_1[DPLL_GL1I_IRQ]
736		GTM_ICM_IRQG_1[DPLL_LL1I_IRQ]
737		GTM_ICM_IRQG_1[DPLL_EI_IRQ]
738		GTM_ICM_IRQG_1[DPLL_GL2I_IRQ]
739		GTM_ICM_IRQG_1[DPLL_LL2I_IRQ]
740		GTM_ICM_IRQG_1[DPLL_TE0I_IRQ]
741		GTM_ICM_IRQG_1[DPLL_TE1I_IRQ]
742		GTM_ICM_IRQG_1[DPLL_TE2I_IRQ]
743		GTM_ICM_IRQG_1[DPLL_TE3I_IRQ]
744		GTM_ICM_IRQG_1[DPLL_TE4I_IRQ]
745		GTM_ICM_IRQG_1[DPLL_CDTI_IRQ]
746		GTM_ICM_IRQG_1[DPLL_CDSI_IRQ]
747		GTM_ICM_IRQG_1[DPLL_TORI_IRQ]
748		GTM_ICM_IRQG_1[DPLL_SORI_IRQ]
749	GTM	gtm_err_irq

Table 34. Interrupt sources (continued)

Number	Description	Register bit
750	GTM_MCS_0	MCS0_CH0_IRQ
751		MCS0_CH1_IRQ
752		MCS0_CH2_IRQ
753		MCS0_CH3_IRQ
754		MCS0_CH4_IRQ
755		MCS0_CH5_IRQ
756		MCS0_CH6_IRQ
757		MCS0_CH7_IRQ
758	GTM_MCS_1	MCS1_CH0_IRQ
759		MCS1_CH1_IRQ
760		MCS1_CH2_IRQ
761		MCS1_CH3_IRQ
762		MCS1_CH4_IRQ
763		MCS1_CH5_IRQ
764		MCS1_CH6_IRQ
765		MCS1_CH7_IRQ
766	GTM_MCS_2	MCS2_CH0_IRQ
767		MCS2_CH1_IRQ
768		MCS2_CH2_IRQ
769		MCS2_CH3_IRQ
770		MCS2_CH4_IRQ
771		MCS2_CH5_IRQ
772		MCS2_CH6_IRQ
773		MCS2_CH7_IRQ
774	GTM_MCS_3	MCS3_CH0_IRQ
775		MCS3_CH1_IRQ
776		MCS3_CH2_IRQ
777		MCS3_CH3_IRQ
778		MCS3_CH4_IRQ
779		MCS3_CH5_IRQ
780		MCS3_CH6_IRQ
781		MCS3_CH7_IRQ

Table 34. Interrupt sources (continued)

Number	Description	Register bit
782	GTM_MCS_4	MCS4_CH0_IRQ
783		MCS4_CH1_IRQ
784		MCS4_CH2_IRQ
785		MCS4_CH3_IRQ
786		MCS4_CH4_IRQ
787		MCS4_CH5_IRQ
788		MCS4_CH6_IRQ
789		MCS4_CH7_IRQ
790	GTM_TIM_0	TIM0_CH0_IRQ
791		TIM0_CH1_IRQ
792		TIM0_CH2_IRQ
793		TIM0_CH3_IRQ
794		TIM0_CH4_IRQ
795		TIM0_CH5_IRQ
796		TIM0_CH6_IRQ
797		TIM0_CH7_IRQ
798	GTM_TIM_1	TIM1_CH0_IRQ
799		TIM1_CH1_IRQ
800		TIM1_CH2_IRQ
801		TIM1_CH3_IRQ
802		TIM1_CH4_IRQ
803		TIM1_CH5_IRQ
804		TIM1_CH6_IRQ
805		TIM1_CH7_IRQ
806	GTM_TIM_2	TIM2_CH0_IRQ
807		TIM2_CH1_IRQ
808		TIM2_CH2_IRQ
809		TIM2_CH3_IRQ
810		TIM2_CH4_IRQ
811		TIM2_CH5_IRQ
812		TIM2_CH6_IRQ
813		TIM2_CH7_IRQ

Table 34. Interrupt sources (continued)

Number	Description	Register bit
814	GTM_TIM_3	TIM3_CH0_IRQ
815		TIM3_CH1_IRQ
816		TIM3_CH2_IRQ
817		TIM3_CH3_IRQ
818		TIM3_CH4_IRQ
819		TIM3_CH5_IRQ
820		TIM3_CH6_IRQ
821		TIM3_CH7_IRQ
822	GTM_TIM_4	TIM4_CH0_IRQ
823		TIM4_CH1_IRQ
824		TIM4_CH2_IRQ
825		TIM4_CH3_IRQ
826		TIM4_CH4_IRQ
827		TIM4_CH5_IRQ
828		TIM4_CH6_IRQ
829		TIM4_CH7_IRQ
830	GTM_TOM_0 (each TOM IRQ covers two channel events)	TOM0_CH0_IRQ :OR: TOM0_CH1_IRQ
831		TOM0_CH2_IRQ :OR: TOM0_CH3_IRQ
832		TOM0_CH4_IRQ :OR: TOM0_CH5_IRQ
833		TOM0_CH6_IRQ :OR: TOM0_CH7_IRQ
834		TOM0_CH8_IRQ :OR: TOM0_CH9_IRQ
835		TOM0_CH10_IRQ :OR: TOM0_CH11_IRQ
836		TOM0_CH12_IRQ :OR: TOM0_CH13_IRQ
837		TOM0_CH14_IRQ :OR: TOM0_CH15_IRQ
838	GTM_TOM_1 (each TOM IRQ covers two channel events)	TOM1_CH0_IRQ :OR: TOM1_CH1_IRQ
839		TOM1_CH2_IRQ :OR: TOM1_CH3_IRQ
840		TOM1_CH4_IRQ :OR: TOM1_CH5_IRQ
841		TOM1_CH6_IRQ :OR: TOM1_CH7_IRQ
842		TOM1_CH8_IRQ :OR: TOM1_CH9_IRQ
843		TOM1_CH10_IRQ :OR: TOM1_CH11_IRQ
844		TOM1_CH12_IRQ :OR: TOM1_CH13_IRQ
845		TOM1_CH14_IRQ :OR: TOM1_CH15_IRQ

Table 34. Interrupt sources (continued)

Number	Description	Register bit
846	GTM_TOM_2 (each TOM IRQ covers two channel events)	TOM2_CH0_IRQ :OR: TOM2_CH1_IRQ
847		TOM2_CH2_IRQ :OR: TOM2_CH3_IRQ
848		TOM2_CH4_IRQ :OR: TOM2_CH5_IRQ
849		TOM2_CH6_IRQ :OR: TOM2_CH7_IRQ
850		TOM2_CH8_IRQ :OR: TOM2_CH9_IRQ
851		TOM2_CH10_IRQ :OR: TOM2_CH11_IRQ
852		TOM2_CH12_IRQ :OR: TOM2_CH13_IRQ
853		TOM2_CH14_IRQ :OR: TOM2_CH15_IRQ
854	GTM_TOM_3 (each TOM IRQ covers two channel events)	TOM3_CH0_IRQ :OR: TOM3_CH1_IRQ
855		TOM3_CH2_IRQ :OR: TOM3_CH3_IRQ
856		TOM3_CH4_IRQ :OR: TOM3_CH5_IRQ
857		TOM3_CH6_IRQ :OR: TOM3_CH7_IRQ
858		TOM3_CH8_IRQ :OR: TOM3_CH9_IRQ
859		TOM3_CH10_IRQ :OR: TOM3_CH11_IRQ
860		TOM3_CH12_IRQ :OR: TOM3_CH13_IRQ
861		TOM3_CH14_IRQ :OR: TOM3_CH15_IRQ
862	GTM_ATOM_0 (each ATOM IRQ covers two channel events)	ATOM0_CH0_IRQ :OR: ATOM0_CH1_IRQ
863		ATOM0_CH2_IRQ :OR: ATOM0_CH3_IRQ
864		ATOM0_CH4_IRQ :OR: ATOM0_CH5_IRQ
865		ATOM0_CH6_IRQ :OR: ATOM0_CH7_IRQ
866	GTM_ATOM_1 (each ATOM IRQ covers two channel events)	ATOM1_CH0_IRQ :OR: ATOM1_CH1_IRQ
867		ATOM1_CH2_IRQ :OR: ATOM1_CH3_IRQ
868		ATOM1_CH4_IRQ :OR: ATOM1_CH5_IRQ
869		ATOM1_CH6_IRQ :OR: ATOM1_CH7_IRQ
870	GTM_ATOM_2 (each ATOM IRQ covers two channel events)	ATOM2_CH0_IRQ :OR: ATOM2_CH1_IRQ
871		ATOM2_CH2_IRQ :OR: ATOM2_CH3_IRQ
872		ATOM2_CH4_IRQ :OR: ATOM2_CH5_IRQ
873		ATOM2_CH6_IRQ :OR: ATOM2_CH7_IRQ
874	GTM_ATOM_3 (each ATOM IRQ covers two channel events)	ATOM3_CH0_IRQ :OR: ATOM3_CH1_IRQ
875		ATOM3_CH2_IRQ :OR: ATOM3_CH3_IRQ
876		ATOM3_CH4_IRQ :OR: ATOM3_CH5_IRQ
877		ATOM3_CH6_IRQ :OR: ATOM3_CH7_IRQ

Table 34. Interrupt sources (continued)

Number	Description	Register bit
878	GTM_ATOM_4 (each ATOM IRQ covers two channel events)	ATOM4_CH0_IRQ :OR: ATOM4_CH1_IRQ
879		ATOM4_CH2_IRQ :OR: ATOM4_CH3_IRQ
880		ATOM4_CH4_IRQ :OR: ATOM4_CH5_IRQ
881		ATOM4_CH6_IRQ :OR: ATOM4_CH7_IRQ
882 - 899	Reserved	—
900	eMIOS_0	EMIOS_GFR[F0,F1]
901		EMIOS_GFR[F2,F3]
902		EMIOS_GFR[F4,F5]
903		EMIOS_GFR[F6,F7]
904		EMIOS_GFR[F8,F9]
905		EMIOS_GFR[F10,F11]
906		EMIOS_GFR[F12,F13]
907		EMIOS_GFR[F14,F15]
908		EMIOS_GFR[F16,F17]
909		EMIOS_GFR[F18,F19]
910		EMIOS_GFR[F20,F21]
911		EMIOS_GFR[F22,F23]
912		EMIOS_GFR[F24,F25]
913		EMIOS_GFR[F26,F27]
914		EMIOS_GFR[F28,F29]
915		EMIOS_GFR[F30,F31]

Table 34. Interrupt sources (continued)

Number	Description	Register bit
916	eMIOS_1	EMIOS_GFR[F0,F1]
917		EMIOS_GFR[F2,F3]
918		EMIOS_GFR[F4,F5]
919		EMIOS_GFR[F6,F7]
920		EMIOS_GFR[F8,F9]
921		EMIOS_GFR[F10,F11]
922		EMIOS_GFR[F12,F13]
923		EMIOS_GFR[F14,F15]
924		EMIOS_GFR[F16,F17]
925		EMIOS_GFR[F18,F19]
926		EMIOS_GFR[F20,F21]
927		EMIOS_GFR[F22,F23]
928		EMIOS_GFR[F24,F25]
929		EMIOS_GFR[F26,F27]
930		EMIOS_GFR[F28,F29]
931		EMIOS_GFR[F30,F31]
932 - 934	Reserved	—
935	CCCU	cu_int
936 - 945	Reserved	—

Table 34. Interrupt sources (continued)

Number	Description	Register bit
946	PSI5-S_0	PS_MBOX_SR_IRQ[0] PS_ERR_SR_IRQ[0]
947		PS_MBOX_SR_IRQ[1] PS_ERR_SR_IRQ[1]
948		PS_MBOX_SR_IRQ[2] PS_ERR_SR_IRQ[2]
949		PS_MBOX_SR_IRQ[3] PS_ERR_SR_IRQ[3]
950		PS_MBOX_SR_IRQ[4] PS_ERR_SR_IRQ[4]
951		PS_MBOX_SR_IRQ[5] PS_ERR_SR_IRQ[5]
952		PS_MBOX_SR_IRQ[6] PS_ERR_SR_IRQ[6]
953		PS_MBOX_SR_IRQ[7] PS_ERR_SR_IRQ[7]
954		PSI5_E2SSR[1]
955		PSI5_E2SSR[2]
956		PSI5_E2SSR[3]
957		PSI5_E2SSR[4]
958		PSI5_E2SSR[5]
959		PSI5_E2SSR[6]
960		PSI5_E2SSR[7]
961		PS_GLSR
962		PSI5_S_0_UART_RX
963		PSI5_S_0_UART_TX
964		PSI5_S_0_UART_ERR
965 - 1023	Reserved	—

7.3.6.3 Interrupt latency

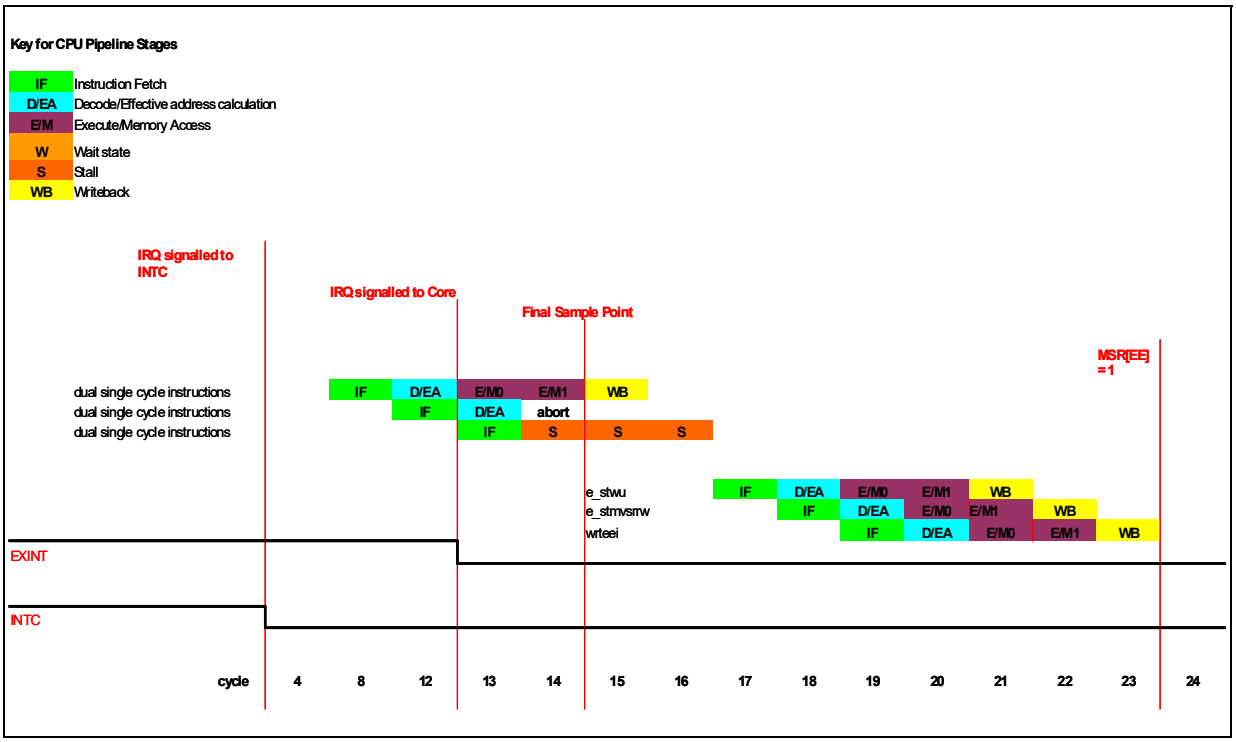
The Interrupt system is designed to provide a very short latency between the assertion of an interrupt request from a peripheral and the first instruction of the Interrupt Service Routine.

The latency time is difficult to define exactly because it depends on what interrupts the INTC is processing and what the CPU is doing.

In general, the latency time is divided into three subsections:

- INTC Latency
  - 3 clock cycles (INTC clock domain[4]) from the assertion of the interrupt request from the peripheral to assertion of the interrupt request to the CPU
- CPU Latency
  - 4 clock cycles (CPU clock domain[5]) from the assertion of the interrupt request from the INTC to the execution of the first instruction
- Software Latency
  - 7 clock cycles (CPU clock domain[5]) to execute the three instructions needed to save essential registers and enable interrupts

Figure 16. Interrupt timing



7.3.6.4 INTC (hsm_z0_platform) configuration

For interrupts specific to the Hardware Security Module (HSM) refer to the SPC58xEx/SPC58xGx Microcontroller Security Reference Manual.

7.3.7 DMA controller configuration

This section summarizes how the module has been configured in the chip. For a comprehensive description of the module itself, see the module's dedicated chapter.

4. 90 MHz
5. 180 MHz

Figure 17. DMA controllers configuration

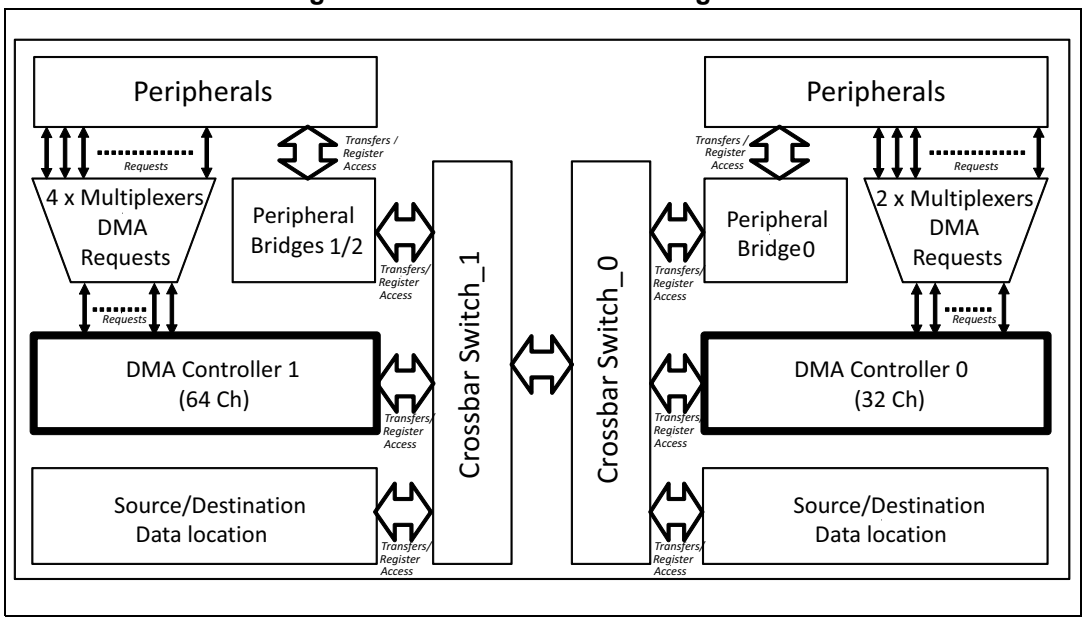


Table 35. Reference links to related information

Topic	Related module	Reference
Full description	DMA Controller 0/1	*Chapter 25: Enhanced Direct Memory Access (eDMA)*
Full description	DMACHMUX 0/1/2/3/4/5	*Chapter 26: DMA channel multiplexer (DMACHMUX)*
System memory map	—	*Chapter 5: Memory Map*
Register access	Peripheral bridge 0/1/2	*Chapter 20: Peripheral Bridge (PBRIDGE)*
Clocking	—	*Chapter 27: Clocking*
Transfers	Crossbar switch 0/1	*Chapter 18: Crossbar switch (XBAR)*

7.3.7.1 DMA implemented registers

Table 36 shows the registers implemented for each DMA instance on the SPC58xEx/SPC58xGx chip.

Table 36. DMA register comparison map

Register description	eDMA_0 Address offset (hex)	eDMA_1 Address offset (hex)
eDMA Control Register (eDMA_CR)	0x0000	0x0000
eDMA Error Status (eDMA_ES)	0x0004	0x0004
eDMA Enable Request High (eDMA_ERQH, Channels 63-32)	—	0x0008
eDMA Enable Request Low (eDMA_ERQL, Channels 31-00)	0x000C	0x000C
eDMA Enable Error Interrupt High (eDMA_EEIH, Channels 63-32)	0x0010	0x0010

Table 36. DMA register comparison map (continued)

Register description				eDMA_0 Address offset (hex)	eDMA_1 Address offset (hex)
eDMA Enable Error Interrupt Low (eDMA_EEIL, Channels 31-00)				0x0014	0x0014
eDMA Set Enable Request (eDMA_SERQ)	eDMA Clear Error (eDMA_CERR)	eDMA Set Enable Error Interrupt (eDMA_SEEI)	eDMA Clear Enable Error Interrupt (eDMA_CEEI)	0x0018	0x0018
eDMA Clear Interrupt Request (eDMA_CINT)	eDMA Clear Error (eDMA_CERR)	eDMA Set Start Bit (eDMA_SSRT)	eDMA Clear Done Status Bit (eDMA_CDNE)	0x001C	0x001C
eDMA Interrupt Request High (eDMA_INTH, Channels 63-32)				—	0x0020
eDMA Interrupt Request Low (eDMA_INTL, Channels 31-00)				0x0024	0x0024
eDMA Error High (eDMA_ERRH, Channels 63-32)				—	0x0028
eDMA Error Low (eDMA_ERRL, Channels 31-00)				0x002C	0x002C
eDMA Hardware Request Status High (eDMA_HRSH, Channels 63-32)				—	0x0030
eDMA Hardware Request Status Low (eDMA_HRSL, Channels 31-00)				0x0034	0x0034
Reserved				0x0038–0x00FF	0x0038–0x00FF
eDMA Channel 0 Priority (eDMA_DCHPRI0)	eDMA Channel 1 Priority (eDMA_DCHPRI1)	eDMA Channel 2 Priority (eDMA_DCHPRI2)	eDMA Channel 3 Priority (eDMA_DCHPRI3)	0x0100	0x0100
eDMA Channel 4 Priority (eDMA_DCHPRI4)	eDMA Channel 5 Priority (eDMA_DCHPRI5)	eDMA Channel 6 Priority (eDMA_DCHPRI6)	eDMA Channel 7 Priority (eDMA_DCHPRI7)	0x0104	0x0104
eDMA Channel 8 Priority (eDMA_DCHPRI8)	eDMA Channel 9 Priority (eDMA_DCHPRI9)	eDMA Channel 10 Priority (eDMA_DCHPRI10)	eDMA Channel 11 Priority (eDMA_DCHPRI11)	0x0108	0x0108
eDMA Channel 12 Priority (eDMA_DCHPRI12)	eDMA Channel 13 Priority (eDMA_DCHPRI13)	eDMA Channel 14 Priority (eDMA_DCHPRI14)	eDMA Channel 15 Priority (eDMA_DCHPRI15)	0x010C	0x010C
eDMA Channel 16 Priority (eDMA_DCHPRI16)	eDMA Channel 17 Priority (eDMA_DCHPRI17)	eDMA Channel 18 Priority (eDMA_DCHPRI18)	eDMA Channel 19 Priority (eDMA_DCHPRI9)	0x0110	0x0110
eDMA Channel 20 Priority (eDMA_DCHPRI20)	eDMA Channel 21 Priority (eDMA_DCHPRI21)	eDMA Channel 22 Priority (eDMA_DCHPRI22)	eDMA Channel 23 Priority (eDMA_DCHPRI23)	0x0114	0x0114
eDMA Channel 24 Priority (eDMA_DCHPRI24)	eDMA Channel 25 Priority (eDMA_DCHPRI25)	eDMA Channel 26 Priority (eDMA_DCHPRI26)	eDMA Channel 27 Priority (eDMA_DCHPRI27)	0x0118	0x0118

Table 36. DMA register comparison map (continued)

Register description				eDMA_0 Address offset (hex)	eDMA_1 Address offset (hex)
eDMA Channel 28 Priority (eDMA_DCHPRI28)	eDMA Channel 29 Priority (eDMA_DCHPRI29)	eDMA Channel 30 Priority (eDMA_DCHPRI30)	eDMA Channel 31 Priority (eDMA_DCHPRI31)	0x011C	0x011C
eDMA Channel 32 Priority (eDMA_DCHPRI32)	eDMA Channel 33 Priority (eDMA_DCHPRI33)	eDMA Channel 34 Priority (eDMA_DCHPRI34)	eDMA Channel 35 Priority (eDMA_DCHPRI35)	—	0x0120
eDMA Channel 36 Priority (eDMA_DCHPRI36)	eDMA Channel 37 Priority (eDMA_DCHPRI37)	eDMA Channel 38 Priority (eDMA_DCHPRI38)	eDMA Channel 39 Priority (eDMA_DCHPRI39)	—	0x0124
eDMA Channel 40 Priority (eDMA_DCHPRI40)	eDMA Channel 41 Priority (eDMA_DCHPRI41)	eDMA Channel 42 Priority (eDMA_DCHPRI42)	eDMA Channel 43 Priority (eDMA_DCHPRI43)	—	0x0128
eDMA Channel 44 Priority (eDMA_DCHPRI44)	eDMA Channel 45 Priority (eDMA_DCHPRI45)	eDMA Channel 46 Priority (eDMA_DCHPRI46)	eDMA Channel 47 Priority (eDMA_DCHPRI47)	—	0x012C
eDMA Channel 48 Priority (eDMA_DCHPRI48)	eDMA Channel 49 Priority (eDMA_DCHPRI49)	eDMA Channel 50 Priority (eDMA_DCHPRI50)	eDMA Channel 51 Priority (eDMA_DCHPRI51)	—	0x0130
eDMA Channel 52 Priority (eDMA_DCHPRI52)	eDMA Channel 53 Priority (eDMA_DCHPRI53)	eDMA Channel 54 Priority (eDMA_DCHPRI54)	eDMA Channel 55 Priority (eDMA_DCHPRI55)	—	0x0134
eDMA Channel 56 Priority (eDMA_DCHPRI56)	eDMA Channel 57 Priority (eDMA_DCHPRI57)	eDMA Channel 58 Priority (eDMA_DCHPRI58)	eDMA Channel 59 Priority (eDMA_DCHPRI59)	—	0x0138
eDMA Channel 60 Priority (eDMA_DCHPRI60)	eDMA Channel 61 Priority (eDMA_DCHPRI61)	eDMA Channel 62 Priority (eDMA_DCHPRI62)	eDMA Channel 63 Priority (eDMA_DCHPRI63)	—	0x013C
Reserved				0x0140-0x0FFC	0x0140-0x0FFC
TCD00-TCD15				0x1000-0x11FC	0x1000-0x11FC
TCD16-TCD31				0x1200-0x13FC	0x1200-0x13FC
TCD32-TCD47				—	0x1400-0x15FC
TCD48-TCD63				—	0x1600-0x17FC

7.3.8 DMACHMUX configuration

The peripheral DMA requests are serviced by a DMA controller. A static multiplexing scheme allows the support of more peripheral requests than available DMA channels.

7.3.8.1 DMACHMUX parameters

SPC58xEx/SPC58xGx has six DMA channel multiplexers.

Each DMACHMUX includes 16 channels.

Table 37 lists the number of DMA channels, always slots and DMA triggers.

Table 37. DMACHMUX controller generics

Signal name Description	Description	DMACHMUX_0	DMACHMUX_1	DMACHMUX_2	DMACHMUX_3	DMACHMUX_4	DMACHMUX_5
NUMBER_OF_DMA_CHANNELS	Number of DMA channels	16	16	16	16	16	16
NUMBER_OF_ALWAYS_SLOTS	Module-internal slots which will always do a request	2	2	2	2	2	2
NUMBER_OF_DMA_TRIGGERS	Number of DMA channels which will be gated with a trigger pulse	3	3	3	3	3	3

7.3.8.2 DMA channel assignment

Table 38 lists which channels apply to each multiplexer.

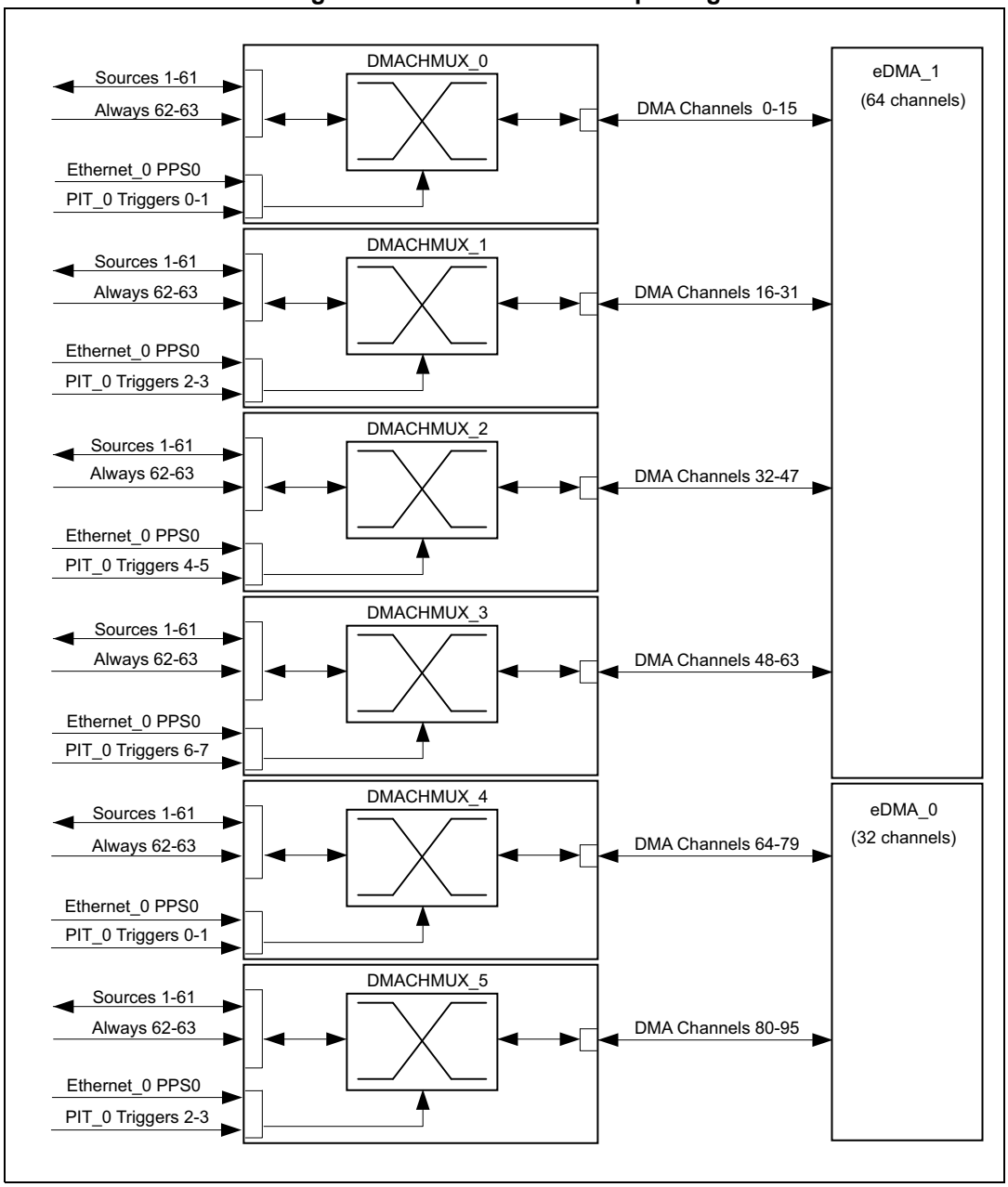
Table 38. SPC58xEx/SPC58xGx multiplexer and channel list

DMACHMUX instance	DMA channel	
	eDMA_0 channel	eDMA_1 channel
0	—	0-15
1	—	16-31
2	—	32-47
3	—	48-63
4	64-79	—
5	80-95	—

7.3.8.3 DMA channel multiplexing block diagram

The connections of the DMA channel multiplexers to the eDMA are shown in *Figure 18*. One DMACHMUX is instantiated for every sixteen DMA channels.

Figure 18. DMA channel multiplexing



7.3.8.4 DMA request mapping

The mapping of the peripheral DMA requests to the DMACHMUX input sources is shown in *Table 39*.

Table 39. DMAMUX Map(1)

	DMA_1 (64 channels)				DMA_0 (32 channels)	
	DMACHMUX_0 (16 DMA channels) source name	**DMACHMUX_1 (16 DMA channels) source name**	**DMACHMUX_2 (16 DMA channels) source name**	**DMACHMUX_3 (16 DMA channels) source name**	**DMACHMUX_4 (16 DMA channels) source name**	**DMACHMUX_5 (16 DMA channels) source name**
0	not available	not available	not available	not available	not available	not available
1	ADC_SAR_12bit_ SUPERVISOR EOC	ADC_SAR_12bit_1 EOC	ADC_SAR_12bit_2 EOC	ADC_SAR_12bit_3 EOC	AMU0	AMU1
2	ADC_SAR_12bit_0 EOC	ADC_SD_0	ADC_SD_1	ADC_SD_2	SIPI CH0	SIPI CH1
3	DSPI_0 RX	ADC_SD_3	ADC_SD_4	ADC_SD_5	SIPI CH2	SIPI CH3
4	DSPI_0 TX	DSPI_2 RX	DSPI_8 RX	DSPI_9 RX	eCTU_0_REQ_0	eCTU_1_REQ_0
5	DSPI_0 CMD	DSPI_2 TX	DSPI_8 TX	DSPI_9 TX	eCTU_0_REQ_1	eCTU_1_REQ_1
6	DSPI_1 RX	DSPI_2 CMD	DSPI_8 CMD	DSPI_9 CMD	eCTU_0_REQ_2	eCTU_1_REQ_2
7	DSPI_1 TX	DSPI_3 RX	LINFlexD_12 RX	LINFlexD_13 RX	eCTU_0_REQ_3	eCTU_1_REQ_3
8	DSPI_1 CMD	DSPI_3 TX	LINFlexD_12 TX	LINFlexD_13 TX	eCTU_0_GLOBAL	eCTU_1_GLOBAL
9	DSPI_4 RX	DSPI_3 CMD	LINFlexD_14 RX	LINFlexD_15 RX	tbd	tbd
10	DSPI_4 TX	DSPI_6 RX	LINFlexD_14 TX	LINFlexD_15 TX	tbd	tbd
11	DSPI_4 CMD	DSPI_6 TX	SIUL2_REQ5	SIUL2_REQ8	tbd	tbd
12	DSPI_5 RX	DSPI_6 CMD	ADC_SAR_10bit_0 EOC	ADC_SAR_10bit_1 EOC	tbd	tbd
13	DSPI_5 TX	DSPI_7 RX	DSPI_4 RX	DSPI_5 RX	SENT_0 RX FAST	SENT_1 RX FAST
14	DSPI_5 CMD	DSPI_7 TX	DSPI_4 TX	DSPI_5 TX	SENT_0 RX SLOW	SENT_1 RX SLOW
15	LINFlexD_0 RX	DSPI_7 CMD	DSPI_4 CMD	DSPI_5 CMD	PSI5_0 CH0 RX PSI5	PSI5-S RX
16	LINFlexD_0 TX	LINFlexD_6 RX	GTM_MCS2_IRQ0	GTM_MCS3_IRQ0	PSI5_0 CH0 RX SMC	PSI5-S TX
17	LINFlexD_1 RX	LINFlexD_6 TX	GTM_MCS2_IRQ1	GTM_MCS3_IRQ1	PSI5_1 CH0 RX PSI5	PSI5-S PS
18	LINFlexD_1 TX	LINFlexD_7 RX	GTM_MCS2_IRQ2	GTM_MCS3_IRQ2	PSI5_1 CH0 RX SMC	SIUL2_REQ10
19	LINFlexD_2 RX	LINFlexD_7 TX	GTM_MCS2_IRQ3	GTM_MCS3_IRQ3	SIUL2_REQ9	SIUL2_REQ11
20	LINFlexD_2 TX	LINFlexD_8 RX	GTM_MCS2_IRQ4	GTM_MCS3_IRQ4	LINFlexD_0 RX	LINFlexD_2 RX
21	LINFlexD_3 RX	LINFlexD_8 TX	GTM_MCS2_IRQ5	GTM_MCS3_IRQ5	LINFlexD_0 TX	LINFlexD_2 TX
22	LINFlexD_3 TX	LINFlexD_9 RX	GTM_MCS2_IRQ6	GTM_MCS3_IRQ6	LINFlexD_1 RX	ADC_SAR_12bit_ SUPERVISOR EOC
23	LINFlexD_4 RX	LINFlexD_9 TX	GTM_MCS2_IRQ7	GTM_MCS3_IRQ7	LINFlexD_1 TX	ADC_SAR_10bit_ STDBY EOC
24	LINFlexD_4 TX	LINFlexD_10 RX	GTM_TIM0_IRQ0	GTM_TIM1_IRQ0	GTM_MCS4_IRQ0	GTM_MCS4_IRQ4
25	LINFlexD_5 RX	LINFlexD_10 TX	GTM_TIM0_IRQ1	GTM_TIM1_IRQ1	GTM_MCS4_IRQ1	GTM_MCS4_IRQ5
26	LINFlexD_5 TX	LINFlexD_11 RX	GTM_TIM0_IRQ2	GTM_TIM1_IRQ2	GTM_MCS4_IRQ2	GTM_MCS4_IRQ6
27	I2C_0 RX	LINFlexD_11 TX	GTM_TIM0_IRQ3	GTM_TIM1_IRQ3	GTM_MCS4_IRQ3	GTM_MCS4_IRQ7
28	I2C_0 TX	CAN_SUB_0_ M_CAN_2	GTM_TIM0_IRQ4	GTM_TIM1_IRQ4	GTM_TIM2_IRQ0	GTM_TIM3_IRQ0
29	CAN_SUB_0_ M_CAN_1	SIUL2_REQ2	GTM_TIM0_IRQ5	GTM_TIM1_IRQ5	GTM_TIM2_IRQ1	GTM_TIM3_IRQ1
30	SIUL2_REQ0	SIUL2_REQ4	GTM_TIM0_IRQ6	GTM_TIM1_IRQ6	GTM_TIM2_IRQ2	GTM_TIM3_IRQ2

Table 39. DMAMUX Map[1] (continued)

	DMA_1 (64 channels)				DMA_0 (32 channels)	
	DMACHMUX_0 (16 DMA channels) source name	DMACHMUX_1 (16 DMA channels) source name	DMACHMUX_2 (16 DMA channels) source name	DMACHMUX_3 (16 DMA channels) source name	DMACHMUX_4 (16 DMA channels) source name	DMACHMUX_5 (16 DMA channels) source name
31	SIUL2_REQ1	eMIOS_1_CH0	GTM_TIM0_IRQ7	GTM_TIM1_IRQ7	GTM_TIM2_IRQ3	GTM_TIM3_IRQ3
32	eMIOS_0_CH0	eMIOS_1_CH9	GTM_TOM0_IRQ0	GTM_TOM1_IRQ0	GTM_TIM2_IRQ4	GTM_TIM3_IRQ4
33	eMIOS_0_CH1	eMIOS_1_CH17	GTM_TOM0_IRQ1	GTM_TOM1_IRQ1	GTM_TIM2_IRQ5	GTM_TIM3_IRQ5
34	eMIOS_0_CH9	eMIOS_1_CH18	GTM_TOM0_IRQ2	GTM_TOM1_IRQ2	GTM_TIM2_IRQ6	GTM_TIM3_IRQ6
35	eMIOS_0_CH18	eMIOS_1_CH25	GTM_TOM0_IRQ3	GTM_TOM1_IRQ3	GTM_TIM2_IRQ7	GTM_TIM3_IRQ7
36	eMIOS_0_CH25	eMIOS_1_CH26	GTM_TOM0_IRQ4	GTM_TOM1_IRQ4	GTM_TOM2_IRQ0	GTM_TOM3_IRQ0
37	eMIOS_0_CH26	GTM_SPE0	GTM_TOM0_IRQ5	GTM_TOM1_IRQ5	GTM_TOM2_IRQ1	GTM_TOM3_IRQ1
38	ADC_SAR_10bit_ STDBY EOC	GTM_SPE1	GTM_TOM0_IRQ6	GTM_TOM1_IRQ6	GTM_TOM2_IRQ2	GTM_TOM3_IRQ2
39	GTM_MCS0_IRQ0	GTM_MCS1_IRQ0	GTM_TOM0_IRQ7	GTM_TOM1_IRQ7	GTM_TOM2_IRQ3	GTM_TOM3_IRQ3
40	GTM_MCS0_IRQ1	GTM_MCS1_IRQ1	GTM_PSM0_IRQ0	GTM_PSM0_IRQ4	GTM_TOM2_IRQ4	GTM_TOM3_IRQ4
41	GTM_MCS0_IRQ2	GTM_MCS1_IRQ2	GTM_PSM0_IRQ1	GTM_PSM0_IRQ5	GTM_TOM2_IRQ5	GTM_TOM3_IRQ5
42	GTM_MCS0_IRQ3	GTM_MCS1_IRQ3	GTM_PSM0_IRQ2	GTM_PSM0_IRQ6	GTM_TOM2_IRQ6	GTM_TOM3_IRQ6
43	GTM_MCS0_IRQ4	GTM_MCS1_IRQ4	GTM_PSM0_IRQ3	GTM_PSM0_IRQ7	GTM_TOM2_IRQ7	GTM_TOM3_IRQ7
44	GTM_MCS0_IRQ5	GTM_MCS1_IRQ5	GTM_ATOM2_IRQ0	GTM_ATOM3_IRQ0	GTM_TIM4_IRQ0	GTM_TIM4_IRQ4
45	GTM_MCS0_IRQ6	GTM_MCS1_IRQ6	GTM_ATOM2_IRQ1	GTM_ATOM3_IRQ1	GTM_TIM4_IRQ1	GTM_TIM4_IRQ5
46	GTM_MCS0_IRQ7	GTM_MCS1_IRQ7	GTM_ATOM2_IRQ2	GTM_ATOM3_IRQ2	GTM_TIM4_IRQ2	GTM_TIM4_IRQ6
47	GTM_ATOM0_IRQ0	GTM_ATOM1_IRQ0	GTM_ATOM2_IRQ3	GTM_ATOM3_IRQ3	GTM_TIM4_IRQ3	GTM_TIM4_IRQ7
48	GTM_ATOM0_IRQ1	GTM_ATOM1_IRQ1	GTM_TIM4_IRQ0	GTM_TIM4_IRQ4	GTM_ATOM4_IRQ0	GTM_ATOM4_IRQ2
49	GTM_ATOM0_IRQ2	GTM_ATOM1_IRQ2	GTM_TIM4_IRQ1	GTM_TIM4_IRQ5	GTM_ATOM4_IRQ1	GTM_ATOM4_IRQ3
50	GTM_ATOM0_IRQ3	GTM_ATOM1_IRQ3	GTM_TIM4_IRQ2	GTM_TIM4_IRQ6	GTM_PSM0_IRQ0	GTM_PSM0_IRQ4
51	GTM_TIM2_IRQ0	GTM_TIM3_IRQ0	GTM_TIM4_IRQ3	GTM_TIM4_IRQ7	GTM_PSM0_IRQ1	GTM_PSM0_IRQ5
52	GTM_TIM2_IRQ1	GTM_TIM3_IRQ1	GTM_TOM3_IRQ0	GTM_ATOM4_IRQ0	GTM_PSM0_IRQ2	GTM_PSM0_IRQ6
53	GTM_TIM2_IRQ2	GTM_TIM3_IRQ2	GTM_TOM3_IRQ1	GTM_ATOM4_IRQ1	GTM_PSM0_IRQ3	GTM_PSM0_IRQ7
54	GTM_TIM2_IRQ3	GTM_TIM3_IRQ3	GTM_TOM3_IRQ2	GTM_ATOM4_IRQ2	ADC_SAR_10bit_0 EOC	ADC_SAR_10bit_1 EOC
55	GTM_TIM2_IRQ4	GTM_TIM3_IRQ4	GTM_TOM3_IRQ3	GTM_ATOM4_IRQ3	LINFlexD_16 RX	LINFlexD_17 TX
56	GTM_TIM2_IRQ5	GTM_TIM3_IRQ5	GTM_TOM3_IRQ4	—	LINFlexD_16 TX	LINFlexD_17 TX
57	GTM_TIM2_IRQ6	GTM_TIM3_IRQ6	GTM_TOM3_IRQ5	—	SIUL2_REQ3	SIUL2_REQ6
58	GTM_TIM2_IRQ7	GTM_TIM3_IRQ7	GTM_TOM3_IRQ6	—	SIUL2_REQ7	SIUL2_REQ12
59	GTM_TOM2_IRQ0	GTM_TOM2_IRQ3	GTM_TOM3_IRQ7	—	SIUL2_REQ13	SIUL2_REQ14
60	GTM_TOM2_IRQ1	GTM_TOM2_IRQ4	GTM_TOM2_IRQ6	—	SIUL2_REQ15	—
61	GTM_TOM2_IRQ2	GTM_TOM2_IRQ5	GTM_TOM2_IRQ7	—	—	—
62	Always on	Always on	Always on	Always on	Always on	Always on
63	Always on	Always on	Always on	Always on	Always on	Always on

Table 39. DMAMUX Map[1] (continued)

	DMA_1 (64 channels)				DMA_0 (32 channels)	
	DMACHMUX_0 (16 DMA channels) source name	DMACHMUX_1 (16 DMA channels) source name	DMACHMUX_2 (16 DMA channels) source name	DMACHMUX_3 (16 DMA channels) source name	DMACHMUX_4 (16 DMA channels) source name	DMACHMUX_5 (16 DMA channels) source name
Trigger	PIT_0_CH0	PIT_0_CH2	PIT_0_CH4	PIT_0_CH6	PIT_0_CH0	PIT_0_CH2
	PIT_0_CH1	PIT_0_CH3	PIT_0_CH5	PIT_0_CH7	PIT_0_CH1	PIT_0_CH3
	Ethernet_0 PPS	Ethernet_0 PPS	Ethernet_0 PPS	Ethernet_0 PPS	Ethernet_0 PPS	Ethernet_0 PPS

1. Some sources are double mapped to increase the flexibility but only one source shall be selected at one time.

Note: Just as multiple CHCFG registers within the same instance must not select the same source value, CHCFG registers among multiple instances must not select the same source value.

The mapping of the PIT periodic triggers to the DMA channel sources is shown in *Table 40*.

Table 40. PIT_0 periodic trigger to DMA channel mapping

PIT_0 periodic trigger	DMA channel
0	0, 64
1	1, 65
2	16, 80
3	17, 81
4	32
5	33
6	48
7	49

The mapping of the Ethernet periodic triggers to the DMA channel sources is shown in *Table 41*.

Table 41. Ethernet_0 periodic trigger to DMA channel mapping

Ethernet_0 periodic trigger	DMA channel
PPS	2, 18, 34, 50, 66, 82

7.3.8.5 DMACHMUX Memory map

The 6 instances of DMACHMUX (0–5) are distributed on two PBRIDGE slots. For the DMACHMUX instances addresses, refer to *Chapter 5: Memory Map*.

The DMACHMUX address offsets are shown in *Table 42*.

Table 42. DMACHMUX register comparison map

Channel Configuration register description	DMACHMUX_0 address offset	DMACHMUX_1 address offset	DMACHMUX_2 address offset	DMACHMUX_3 address offset	DMACHMUX_4 address offset	DMACHMUX_5 address offset
CHCFG0	0x0000	0x0000	0x0000	0x0000	0x0000	0x0000
CHCFG1	0x0001	0x0001	0x0001	0x0001	0x0001	0x0001
CHCFG2	0x0002	0x0002	0x0002	0x0002	0x0002	0x0002
CHCFG3	0x0003	0x0003	0x0003	0x0003	0x0003	0x0003
CHCFG4	0x0004	0x0004	0x0004	0x0004	0x0004	0x0004
CHCFG5	0x0005	0x0005	0x0005	0x0005	0x0005	0x0005
CHCFG6	0x0006	0x0006	0x0006	0x0006	0x0006	0x0006
CHCFG7	0x0007	0x0007	0x0007	0x0007	0x0007	0x0007
CHCFG8	0x0008	0x0008	0x0008	0x0008	0x0008	0x0008
CHCFG9	0x0009	0x0009	0x0009	0x0009	0x0009	0x0009
CHCFG10	0x000A	0x000A	0x000A	0x000A	0x000A	0x000A
CHCFG11	0x000B	0x000B	0x000B	0x000B	0x000B	0x000B
CHCFG12	0x000C	0x000C	0x000C	0x000C	0x000C	0x000C
CHCFG13	0x000D	0x000D	0x000D	0x000D	0x000D	0x000D
CHCFG14	0x000E	0x000E	0x000E	0x000E	0x000E	0x000E
CHCFG15	0x000F	0x000F	0x000F	0x000F	0x000F	0x000F

Table 43. Reference links to related information

Topic	Related module	Reference
Full description	DMA Channel Multiplexer	*Chapter 26: DMA channel multiplexer (DMACHMUX)*
DMA controller description	DMA Controller	*Chapter 25: Enhanced Direct Memory Access (eDMA)*
System memory map	—	*Chapter 5: Memory Map*

7.3.9 Platform Configuration Module (PCM) configuration

The Platform Configuration Module (PCM) contains three registers. These PCM registers are used for controlling different aspects of the Platform functionality including the Frequency Gasket Configurations of the Intelligent Bus Bridging Gaskets.

The PCM module is mapped to PBRIDGE_2 on platform slot 10. Its base address is 0xF4028000.

7.3.9.1 Platform Configuration Module 0 register (PCM0)

Offset: 0x0000 Access: Supervisor read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	PRE_CONC2	BRE_CONC2	BWE_CONC2	0	PRE_CONC1_GSKT	BRE_CONC1_GSKT	BWE_CONC1_GSKT	0	PRE_CONC0_GSKT	BRE_CONC0_GSKT	BWE_CONC0_GSKT	0	PRE_CONC1	BRE_CONC1	BWE_CONC1
W																
Reset	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	PRE_CONC0_HSM	BRE_CONC0_HSM	BWE_CONC0_HSM	0	PRE_CONC0_DMA1	BRE_CONC0_DMA1	BWE_CONC0_DMA1	0	PRE_XBAR1_S6_GSKT	BRE_XBAR1_S6_GSKT	BWE_XBAR1_S6_GSKT	0	PRE_XBAR1_M5_GSKT	BRE_XBAR1_M5_GSKT	BWE_XBAR1_M5_GSKT
W																
Reset	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1

Figure 19. Platform Configuration Module 0 register (PCM0)

Table 44. PCM0 field descriptions

Field	Description
1 PRE_CONC2	Pending read enable (PRE) used for controlling the burst parameter of frequency gasket used for Concentrator 2 This bit controls the bus gasket's handling of pending read transactions. 0 Pending reads are disabled. 1 Pending reads are enabled.
2 BRE_CONC2	Burst read enable (BRE) used for controlling the burst parameter of frequency gasket used for Concentrator 2 This bit controls the bus gasket's handling of burst read transactions. 0 Burst reads are converted into a series of single transactions on the slave side of the gasket. 1 Burst reads are optimized for best system performance.
3 BWE_CONC2	Burst write enable (BWE) used for controlling the burst parameter of frequency gasket used for Concentrator 2 This bit controls the bus gasket's handling of burst write transactions. 0 Burst writes are converted into a series of single transactions on the slave side of the gasket. 1 Burst writes are optimized for best system performance.
5 PRE_CONC1_GSKT	0 Pending reads are disabled. 1 Pending reads are enabled. **Note:** This bit has same value as bitfield PRE_CONC1.

Table 44. PCM0 field descriptions (continued)

Field	Description
6 BRE_CONC1_GSKT	0 Burst reads are converted into a series of single transactions on the slave side of the gasket. 1 Burst reads are optimized for best system performance. **Note:** This bit has same value as bitfield BRE_CONC1.
7 BWE_CONC1_GSKT	0 Burst writes are converted into a series of single transactions on the slave side of the gasket. 1 Burst writes are optimized for best system performance. **Note:** This bit has same value as bitfield BWE_CONC1.
9 PRE_CONC0_GSKT	Pending read enable (PRE) used for controlling the burst parameter of frequency gasket used for Concentrator 0 This bit controls the bus gasket's handling of pending read transactions. 0 Pending reads are disabled. 1 Pending reads are enabled.
10 BRE_CONC0_GSKT	Burst read enable (BRE) used for controlling the burst parameter of frequency gasket used for Concentrator 0 This bit controls the bus gasket's handling of burst read transactions. 0 Burst reads are converted into a series of single transactions on the slave side of the gasket. 1 Burst reads are optimized for best system performance.
11 BWE_CONC0_GSKT	Burst write enable (BWE) used for controlling the burst parameter of frequency gasket used for Concentrator 0 This bit controls the bus gasket's handling of burst write transactions. 0 Burst writes are converted into a series of single transactions on the slave side of the gasket. 1 Burst writes are optimized for best system performance.
13 PRE_CONC1	Pending read enable (PRE) used for controlling the burst parameter of frequency gasket used for Concentrator 1 This bit controls the bus gasket's handling of pending read transactions. 0 Pending reads are disabled. 1 Pending reads are enabled.
14 BRE_CONC1	Burst read enable (BRE) used for controlling the burst parameter of frequency gasket used for Concentrator 1 This bit controls the bus gasket's handling of burst read transactions. 0 Burst reads are converted into a series of single transactions on the slave side of the gasket. 1 Burst reads are optimized for best system performance.
15 BWE_CONC1	Burst write enable (BWE) used for controlling the burst parameter of frequency gasket used for Concentrator 1 This bit controls the bus gasket's handling of burst write transactions. 0 Burst writes are converted into a series of single transactions on the slave side of the gasket. 1 Burst writes are optimized for best system performance.

Table 44. PCM0 field descriptions (continued)

Field	Description
17 PRE_CONC0_HSM	Pending read enable (PRE) used for controlling the burst parameter of frequency gasket used for Concentrator 0 HSM (HSMGasket) This bit controls the bus gasket's handling of pending read transactions. 0 Pending reads are disabled. 1 Pending reads are enabled.
18 BRE_CONC0_HSM	Burst read enable (BRE) used for controlling the burst parameter of frequency gasket used for Concentrator 0 HSM (HSMGasket) This bit controls the bus gasket's handling of burst read transactions. 0 Burst reads are converted into a series of single transactions on the slave side of the gasket. 1 Burst reads are optimized for best system performance.
19 BWE_CONC0_HSM	Burst write enable (BWE) used for controlling the burst parameter of frequency gasket used for Concentrator 0 HSM (HSMGasket) This bit controls the bus gasket's handling of burst write transactions. 0 Burst writes are converted into a series of single transactions on the slave side of the gasket. 1 Burst writes are optimized for best system performance.
21 PRE_CONC0_DMA1	Pending read enable (PRE) used for controlling the burst parameter of frequency gasket used for Concentrator 0 DMA1 (DMA1Gasket) This bit controls the bus gasket's handling of pending read transactions. 0 Pending reads are disabled. 1 Pending reads are enabled.
22 BRE_CONC0_DMA1	Burst read enable (BRE) used for controlling the burst parameter of frequency gasket used for Concentrator 0 DMA1 (DMA1Gasket) This bit controls the bus gasket's handling of burst read transactions. 0 Burst reads are converted into a series of single transactions on the slave side of the gasket. 1 Burst reads are optimized for best system performance.
23 BWE_CONC0_DMA1	Burst write enable (BWE) used for controlling the burst parameter of frequency gasket used for Concentrator 0 DMA1 (DMA1Gasket) This bit controls the bus gasket's handling of burst write transactions. 0 Burst writes are converted into a series of single transactions on the slave side of the gasket. 1 Burst writes are optimized for best system performance.
25 PRE_XBAR1_S6_GSKT	Pending read enable (PRE) used for controlling the burst parameter of frequency gasket used for XBAR1 S6 This bit controls the bus gasket's handling of pending read transactions. 0 Pending reads are disabled. 1 Pending reads are enabled.
26 BRE_XBAR1_S6_GSKT	Burst read enable (BRE) used for controlling the burst parameter of frequency gasket used for XBAR1 S6 This bit controls the bus gasket's handling of burst read transactions. 0 Burst reads are converted into a series of single transactions on the slave side of the gasket. 1 Burst reads are optimized for best system performance.

Table 44. PCM0 field descriptions (continued)

Field	Description
27 BWE_XBAR1_S6_GSKT	Burst write enable (BWE) used for controlling the burst parameter of frequency gasket used for XBAR1 S6 This bit controls the bus gasket's handling of burst write transactions. 0 Burst writes are converted into a series of single transactions on the slave side of the gasket. 1 Burst writes are optimized for best system performance.
29 PRE_XBAR1_M5_GSKT	Pending read enable (PRE) used for controlling the burst parameter of frequency gasket used for XBAR1 M5 This bit controls the bus gasket's handling of pending read transactions. 0 Pending reads are disabled. 1 Pending reads are enabled.
30 BRE_XBAR1_M5_GSKT	Burst read enable (BRE) used for controlling the burst parameter of frequency gasket used for XBAR1 M5 This bit controls the bus gasket's handling of burst read transactions. 0 Burst reads are converted into a series of single transactions on the slave side of the gasket. 1 Burst reads are optimized for best system performance.
31 BWE_XBAR1_M5_GSKT	Burst write enable (BWE) used for controlling the burst parameter of frequency gasket used for XBAR1 M5 This bit controls the bus gasket's handling of burst write transactions. 0 Burst writes are converted into a series of single transactions on the slave side of the gasket. 1 Burst writes are optimized for best system performance.

7.3.9.2 Platform Configuration Module 1 register (PCM1)

Offset: 0x0004 Access: Supervisor read/write

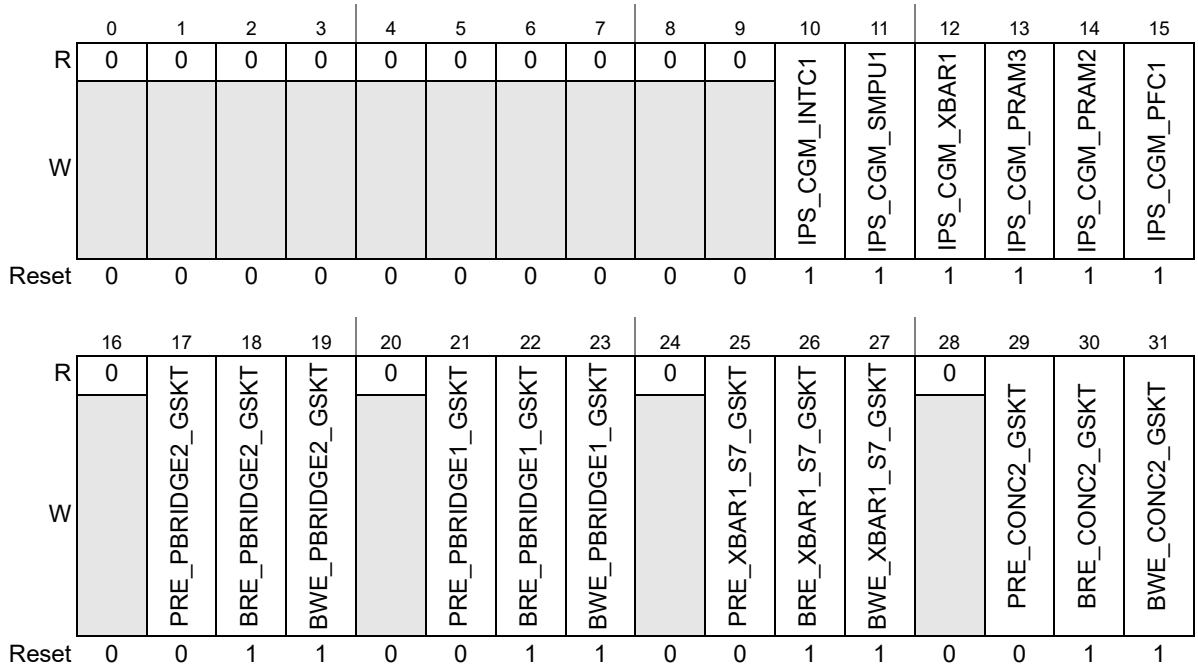


Figure 20. Platform Configuration Module 1 register (PCM1)

Table 45. PCM1 field descriptions

Field	Description
10 IPS_CGM_INTC1	IPS clock gating enable for respective Module Enables/Disables IPG Clock to the respective module 0 Disable clock gating. 1 Enable clock gating, one IPS clock delay is introduced between master and slave.
11 IPS_CGM_SMPU1	IPS clock gating enable for respective Module Enables/Disables IPG Clock to the respective module 0 Disable clock gating. 1 Enable clock gating, one IPS clock delay is introduced between master and slave.
12 IPS_CGM_XBAR1	IPS clock gating enable for respective Module Enables/Disables IPG Clock to the respective module 0 Disable clock gating. 1 Enable clock gating, one IPS clock delay is introduced between master and slave.
13 IPS_CGM_PRAM3	IPS clock gating enable for respective Module Enables/Disables IPG Clock to the respective module 0 Disable clock gating. 1 Enable clock gating, one IPS clock delay is introduced between master and slave.

Table 45. PCM1 field descriptions (continued)

Field	Description
14 IPS_CGM_PRAM2	IPS clock gating enable for respective Module Enables/Disables IPG Clock to the respective module 0 Disable clock gating. 1 Enable clock gating, one IPS clock delay is introduced between master and slave.
15 IPS_CGM_PFC1	IPS clock gating enable for respective Module Enables/Disables IPG Clock to the respective module 0 Disable clock gating. 1 Enable clock gating, one IPS clock delay is introduced between master and slave.
17 PRE_PBRIDGE2_GSKT	Pending read enable (PRE) used for controlling the burst parameter of frequency gasket used for PBRIDGE2 This bit controls the bus gasket's handling of pending read transactions. 0 Pending reads are disabled. 1 Pending reads are enabled.
18 BRE_PBRIDGE2_GSKT	Burst read enable (BRE) used for controlling the burst parameter of frequency gasket used for PBRIDGE2 This bit controls the bus gasket's handling of burst read transactions. 0 Burst reads are converted into a series of single transactions on the slave side of the gasket. 1 Burst reads are optimized for best system performance.
19 BWE_PBRIDGE2_GSKT	Burst write enable (BWE) used for controlling the burst parameter of frequency gasket used for PBRIDGE2 This bit controls the bus gasket's handling of burst write transactions. 0 Burst writes are converted into a series of single transactions on the slave side of the gasket. 1 Burst writes are optimized for best system performance.
21 PRE_PBRIDGE1_GSKT	Pending read enable (PRE) used for controlling the burst parameter of frequency gasket used for PBRIDGE1 This bit controls the bus gasket's handling of pending read transactions. 0 Pending reads are disabled. 1 Pending reads are enabled.
22 BRE_PBRIDGE1_GSKT	Burst read enable (BRE) used for controlling the burst parameter of frequency gasket used for PBRIDGE1 This bit controls the bus gasket's handling of burst read transactions. 0 Burst reads are converted into a series of single transactions on the slave side of the gasket. 1 Burst reads are optimized for best system performance.
23 BWE_PBRIDGE1_GSKT	Burst write enable (BWE) used for controlling the burst parameter of frequency gasket used for PBRIDGE1 This bit controls the bus gasket's handling of burst write transactions. 0 Burst writes are converted into a series of single transactions on the slave side of the gasket. 1 Burst writes are optimized for best system performance.

Table 45. PCM1 field descriptions (continued)

Field	Description
25 PRE_XBAR1_S7_GSKT	Pending read enable (PRE) used for controlling the burst parameter of frequency gasket used for XBAR1 S7 This bit controls the bus gasket's handling of pending read transactions. 0 Pending reads are disabled. 1 Pending reads are enabled.
26 BRE_XBAR1_S7_GSKT	Burst read enable (BRE) used for controlling the burst parameter of frequency gasket used for XBAR1 S7 This bit controls the bus gasket's handling of burst read transactions. 0 Burst reads are converted into a series of single transactions on the slave side of the gasket. 1 Burst reads are optimized for best system performance.
27 BWE_XBAR1_S7_GSKT	Burst write enable (BWE) used for controlling the burst parameter of frequency gasket used for XBAR1 S7 This bit controls the bus gasket's handling of burst write transactions. 0 Burst writes are converted into a series of single transactions on the slave side of the gasket. 1 Burst writes are optimized for best system performance.
29 PRE_CONC2_GSKT	Pending read enable (PRE) used for controlling the burst parameter of frequency gasket used for Concentrator 2 This bit controls the bus gasket's handling of pending read transactions. 0 Pending reads are disabled. 1 Pending reads are enabled.
30 BRE_CONC2_GSKT	Burst read enable (BRE) used for controlling the burst parameter of frequency gasket used for Concentrator 2 This bit controls the bus gasket's handling of burst read transactions. 0 Burst reads are converted into a series of single transactions on the slave side of the gasket. 1 Burst reads are optimized for best system performance.
31 BWE_CONC2_GSKT	Burst write enable (BWE) used for controlling the burst parameter of frequency gasket used for Concentrator 2 This bit controls the bus gasket's handling of burst write transactions. 0 Burst writes are converted into a series of single transactions on the slave side of the gasket. 1 Burst writes are optimized for best system performance.

7.3.9.3 Platform Configuration Module 2 register (PCM2)

Offset: 0x0008 Access: Supervisor read/write

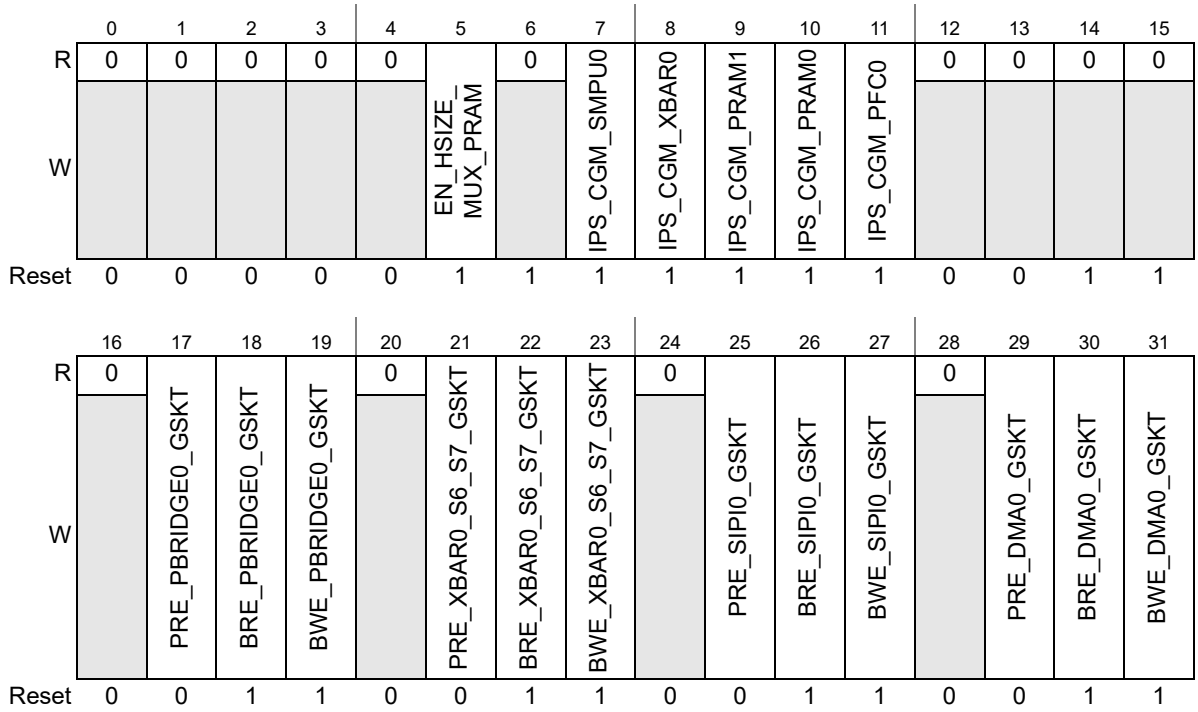


Figure 21. Platform Configuration Module 2 register (PCM2)

Table 46. PCM2 field descriptions

Field	Description
5 EN_HSIZE_MUX_PRAM	DMC operations HSIZE control for PRAM 0, 1, 2 and 3 Force HSIZE=32 bit for Decorated Memory operations on PRAM as decorated operations compute write checkbits (ECC) on 32-bit word. 0 Any decorated write to SRAM peripherals will be done with native HSIZE of the instruction (8b/16b/32b) 1 Any decorated write to SRAM peripherals will be done with HSIZE=32bit
7 IPS_CGM_SMPU0	IPS clock gating enable for respective Module Enables/Disables IPG Clock to the respective module 0 Disable clock gating. 1 Enable clock gating, one IPS clock delay is introduced between master and slave.
8 IPS_CGM_XBAR0	IPS clock gating enable for respective Module Enables/Disables IPG Clock to the respective module 0 Disable clock gating. 1 Enable clock gating, one IPS clock delay is introduced between master and slave.

Table 46. PCM2 field descriptions (continued)

Field	Description
9 IPS_CGM_PRAM1	IPS clock gating enable for respective Module Enables/Disables IPG Clock to the respective module 0 Disable clock gating. 1 Enable clock gating, one IPS clock delay is introduced between master and slave.
10 IPS_CGM_PRAM0	IPS clock gating enable for respective Module Enables/Disables IPG Clock to the respective module 0 Disable clock gating. 1 Enable clock gating, one IPS clock delay is introduced between master and slave.
11 IPS_CGM_PFC0	IPS clock gating enable for respective Module Enables/Disables IPG Clock to the respective module 0 Disable clock gating. 1 Enable clock gating, one IPS clock delay is introduced between master and slave.
17 PRE_PBRIDGE0_GSKT	Pending read enable (PRE) used for controlling the burst parameter of frequency gasket used for PBRIDGE0 This bit controls the bus gasket's handling of pending read transactions. 0 Pending reads are disabled. 1 Pending reads are enabled.
18 BRE_PBRIDGE0_GSKT	Burst read enable (BRE) used for controlling the burst parameter of frequency gasket used for PBRIDGE0 This bit controls the bus gasket's handling of burst read transactions. 0 Burst reads are converted into a series of single transactions on the slave side of the gasket. 1 Burst reads are optimized for best system performance.
19 BWE_PBRIDGE0_GSKT	Burst write enable (BWE) used for controlling the burst parameter of frequency gasket used for PBRIDGE0 This bit controls the bus gasket's handling of burst write transactions. 0 Burst writes are converted into a series of single transactions on the slave side of the gasket. 1 Burst writes are optimized for best system performance.
21 PRE_XBAR0_S6_S7_GSKT	Pending read enable (PRE) used for controlling the burst parameter of frequency gasket used for XBAR0_S6_S7 This bit controls the bus gasket's handling of pending read transactions. 0 Pending reads are disabled. 1 Pending reads are enabled.
22 BRE_XBAR0_S6_S7_GSKT	Burst read enable (BRE) used for controlling the burst parameter of frequency gasket used for XBAR0_S6_S7 This bit controls the bus gasket's handling of burst read transactions. 0 Burst reads are converted into a series of single transactions on the slave side of the gasket. 1 Burst reads are optimized for best system performance.

Table 46. PCM2 field descriptions (continued)

Field	Description
23 BWE_XBAR0_S6_S7_GSKT	Burst write enable (BWE) used for controlling the burst parameter of frequency gasket used for XBAR0_S6_S7 This bit controls the bus gasket's handling of burst write transactions. 0 Burst writes are converted into a series of single transactions on the slave side of the gasket. 1 Burst writes are optimized for best system performance.
25 PRE_SIPI0_GSKT	Pending read enable (PRE) used for controlling the burst parameter of frequency gasket used for SIPI0 This bit controls the bus gasket's handling of pending read transactions. 0 Pending reads are disabled. 1 Pending reads are enabled.
26 BRE_SIPI0_GSKT	Burst read enable (BRE) used for controlling the burst parameter of frequency gasket used for SIPI0 This bit controls the bus gasket's handling of burst read transactions. 0 Burst reads are converted into a series of single transactions on the slave side of the gasket. 1 Burst reads are optimized for best system performance.
27 BWE_SIPI0_GSKT	Burst write enable (BWE) used for controlling the burst parameter of frequency gasket used for SIPI0 This bit controls the bus gasket's handling of burst write transactions. 0 Burst writes are converted into a series of single transactions on the slave side of the gasket. 1 Burst writes are optimized for best system performance.
29 PRE_DMA0_GSKT	Pending read enable (PRE) used for controlling the burst parameter of frequency gasket used for DMA0 This bit controls the bus gasket's handling of pending read transactions. 0 Pending reads are disabled. 1 Pending reads are enabled.
30 BRE_DMA0_GSKT	Burst read enable (BRE) used for controlling the burst parameter of frequency gasket used for DMA0 This bit controls the bus gasket's handling of burst read transactions. 0 Burst reads are converted into a series of single transactions on the slave side of the gasket. 1 Burst reads are optimized for best system performance.
31 BWE_DMA0_GSKT	Burst write enable (BWE) used for controlling the burst parameter of frequency gasket used for DMA0 This bit controls the bus gasket's handling of burst write transactions. 0 Burst writes are converted into a series of single transactions on the slave side of the gasket. 1 Burst writes are optimized for best system performance.

7.3.10 Wakeup unit (WKPU) configuration

The Wakeup unit supports 32 external sources that can generate CPU exceptions or wakeup events and 2 external sources, driven by PA[4] and PD[15], that can cause non-maskable interrupt requests or wakeup events.

The effect of the signals coming from PA[4] and PD[15] depends on the execution mode.

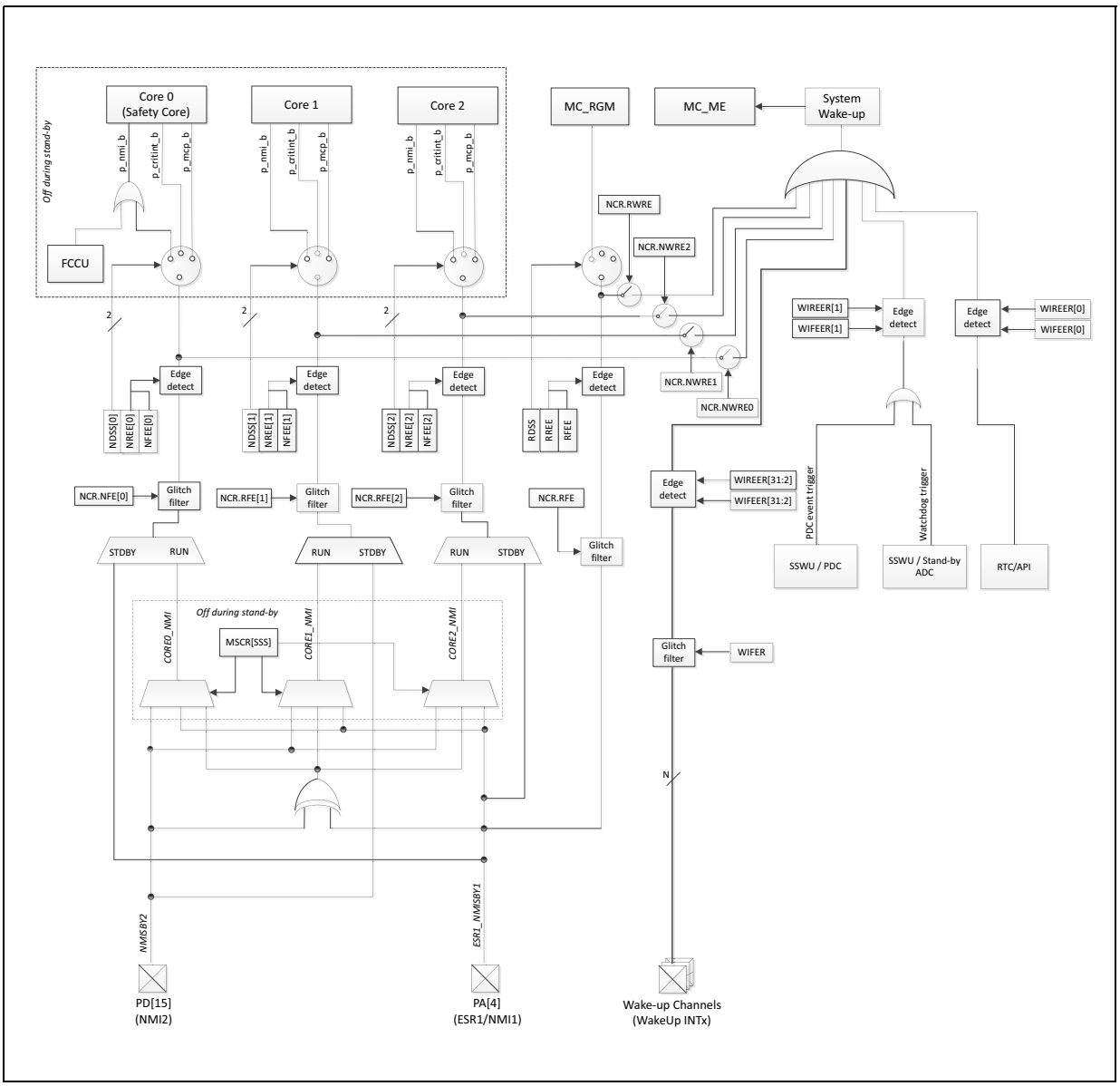
In any run mode, three signals (called CORE0_NMI, CORE1_NMI, and CORE2_NMI) are input to each core. Each of these signals is the output of one multiplexer, which makes it possible to select (via SIUL2 register) any of the following sources:

- PA[4]
- PD[15]
- Logical XOR'ed of PA[4] and PD[15]

In standby mode, the signals driven by PA[4] and PD[15] (called ESR1_NMISBY1 and NMISBY2 respectively) can be used to wake-up the system.

PA[4] can also be used to trigger a reset to the whole system.

Figure 22. NMI block diagram SPC58xEx/SPC58xGx



In *Figure 2509: Wakeup unit block diagram* of *Chapter 96: Wakeup unit (WKPU)*, it has to be noted that only one filter is implemented.

7.3.10.1 Features

The Wakeup unit supports these distinctive features:

- Non-maskable Interrupt support with:
  - 3 NMI source with bypassable glitch filter
  - Independent interrupt destination: non-maskable interrupt, critical interrupt, or machine check request
  - Edge detection
- External wakeup/interrupt support with:
  - 30 wakeup/interrupt sources
  - Analog glitch filter per each wakeup line
  - Independent interrupt mask
  - Edge detection
  - Configurable system wakeup triggering from all interrupt sources
  - Configurable pull
- On-chip wakeup support
  - 2 wakeup sources
  - Wakeup status mapped to same register as external wakeup/interrupt status
  - Independent interrupt mask
  - Configurable system wakeup triggering from all interrupt sources

7.3.10.2 Wakeup vector mapping

Table 47. Wakeup vector mapping

Wakeup number	Port	Port Input Function	WKPU IRQ to INTC	IRQ#	WISR	Register Bit position(1)
WKUP0	SSWU / API	Refer to SPC58xEx/SPC58xGx pinout Excel® file	WakeUp_IRQ_0	Table 34: Interrupt sources	EISR0	31
WKUP1	RTC				EISR1	30
WKUP2	Refer to SPC58xEx/SPC58xGx pinout Excel® file				EISR2	29
WKUP3					EISR3	28
WKUP4					EISR4	27
WKUP5					EISR5	26
WKUP6					EISR6	25
WKUP7					EISR7	24
WKUP8			WakeUp_IRQ_1	Table 34: Interrupt sources	EISR8	23
WKUP9					EISR9	22
WKUP10					EISR10	21
WKUP11					EISR11	20
WKUP12					EISR12	19
WKUP13					EISR13	18
WKUP14					EISR14	17
WKUP15					EISR15	16
WKUP16			WakeUp_IRQ_2	Table 34: Interrupt sources	EISR16	15
WKUP17					EISR17	14
WKUP18					EISR18	13
WKUP19					EISR19	12
WKUP20					EISR20	11
WKUP21					EISR21	10
WKUP22					EISR22	9
WKUP23					EISR23	8
WKUP24			WakeUp_IRQ_3	Table 34: Interrupt sources	EISR24	7
WKUP25					EISR25	6
WKUP26					EISR26	5
WKUP27					EISR27	4
WKUP28					EISR28	3
WKUP29					EISR29	2
WKUP30		—			EISR30	1
WKUP31		—			EISR31	0

1. applicable for registers WISR, IRER, WRER, WIFEER, WIFEEF, WIFER, WIPUER

7.3.11 Crossbar integrity checker (XBIC)

The device implements the following XBIC instances:

- XBIC_Concentrator_0
- XBIC_Concentrator_1
- XBIC_0
- XBIC_1

Master and slave assignments are defined as follows:

- XBIC master definition follows the AHB master (see *Section 7.3.3.1.1: Crossbar_0 switch master assignments* and *Section 7.3.3.2.1: Crossbar_1 switch master assignments*).
- XBIC slave definition follows the XBAR slave numbering (see *Section 7.3.3.1.2: Crossbar_0 switch slave assignments* and *Section 7.3.3.2.2: Crossbar_1 switch slave assignments*).

7.4 Clocking

See *Chapter 27: Clocking* for a description of the architecture for the system level clocks including clock generation, the clock sources and the peripheral clocks. The chapter also provides configuration details for the Clock Monitor Unit (CMU).

7.5 Memories and memory interfaces

See the following chapters for detailed information on memories and memory interfaces.

Table 48. Reference links to related information

Topic	Related module	Reference
Embedded memories	—	*Chapter 3: Embedded memories*
System memory map	—	*Chapter 5: Memory Map*
Platform Flash	—	*Chapter 36: Flash memory controller*
Platform RAM controller	—	*Chapter 35: Platform RAM controller AHB (PRAMC_AHB)*
Embedded flash memory	—	*Chapter 37: Embedded Flash Memory*
Decorated Storage Memory Controller	—	*Chapter 39: Decorated Storage Memory Controller (DSMC)*

7.5.1 Flash memory controller (PFLASH) configuration

Two flash controllers and arrays are implemented. The flash controllers provide flash configuration and control functions and manage the interface between the flash memory array and the crossbar switch. The configuration and control functions are accessed via control registers, which are read/write accessible only in supervisor mode.

Three of the registers, PFCR1, PFCR2, and PFAPR, control the interaction of master modules with the flash array by enabling/disabling prefetch or by controlling access to the

flash array on a per-master basis. Use of the fields in these registers requires knowledge of the number assigned to each master. For example, the PFAPR[M2AP] field controls flash array access by master 2, the PFAPR[M3AP] field controls flash array access by master 3, and so on. See the crossbar switch configuration section for details. See the PFCR3 register details in the flash memory controller chapter for more details on configuring the flash controller.

Table 49. Reference links to related information

Topic	Related module	Reference
Full description	Flash Memory Controller (PFLASH)	*Chapter 36: Flash memory controller*
Flash memory architecture overview	Flash memory, Flash controller	*Section 3.3.1: Flash memory controller*
Flash memory	Flash memory	*Chapter 37: Embedded Flash Memory*
Flash port assignments	Crossbar switch	*Section 7.3.3: Crossbar switch*
System memory map	—	*Chapter 5: Memory Map*

The master assignments are shown in *Section 7.3.3.3: AHB Master ID Assignments*. Note that these are the same master assignments used elsewhere in the chip, for example, the SMPU.

7.5.1.1 Calibration remapping support

The PFLASH flash memory controller supports calibration development by providing a remapping function to route flash accesses to overlay RAM or on-chip system RAM that can be used as overlay RAM. The relationship between flash access latency and remapped overlay RAM latency is contingent upon operating frequency and intrinsic memory access times. *Table 50* describes the intrinsic cycle latency on a random, initial access for each supported overlay RAM target.

(See *Section 36.5.13: PFlash calibration remap support* for more details on the relationship between flash access latency and overlay RAM access latency.

Note: Cycle times measured based on the PFLASH clock. See Chapter 27: Clocking for more details on clocking.

Table 50. Calibration memory access times

Calibration memory	Burst access time (cycles)
On-chip overlay RAM	5
Buddy device overlay RAM	8
System RAM	9

7.5.1.2 Wait state settings for overlay flash latency matching

A remapped calibration load access has a fixed, intrinsic 4-beat latency, where the access time of each individual beat depends on the overlay target - system RAM, on-chip overlay RAM or buddy device Therefore, in order for an initial, random calibration load access to mimic the flash access time, PFCR1[RWSC] read wait state setting must be programmed to a value which sufficiently covers the intrinsic access time for a flash access as well as the

required access time to complete a four-beat burst sequence from the slowest available overlay target at the specified device operating frequency. *Table 51* describes the minimum supported flash wait state setting to guarantee a remapped access matches the response latency of a flash access.

Table 51. Minimum wait state settings for overlay flash latency match

Overlay Target	PFCR1[RWSC] Minimum Value
BD RAM	6
On-Chip Overlay RAM	3
System RAM	6

If PFCR1[RWSC] is set sufficiently high such that the remapped burst access completes before flash read data is expected to be returned, the flash controller stalls the return of overlay data the appropriate number of additional cycles to ensure the access time matches the response latency of a flash access. Once the overlay access is stored in the flash controller mini-cache, subsequent requests that hit in the mini-cache are returned with zero wait state latency.

7.5.2 Platform RAM Controller (PRAMC) configuration

7.5.2.1 Wait state management

PRAMC_CR1[FT_DIS] should be set to 1’b1 to enable 1 cycle of wait state during read operation when the controller is working @180 MHz (refer to the device datasheet to check from which frequency this wait state cycle is needed).

7.5.3 Decorated Storage Memory Controller (DSMC)

The DSMC supports five store and three load operations, including:

- Bit field inserts
- Compare-and-store
- Bitwise AND, OR, and XOR operators
- Simple memory load
- Swap
- Load-and-set-1(bit)

As the DSMC generates the atomic read-modify-write bus transactions to the attached slave memory controller, the two transactions (read, write) are fully pipelined with no idle cycles introduced. During these transactions, the AHB hlock control signal is asserted to the slave during the entire read-modify-write.

The DSMC is instantiated multiple times within the core platform and physically resides between the CPU data AHB bus and the associated crossbar master port.

For SPC58xEx/SPC58xGx, the DSMC is instantiated six times: three times on every CPU's AHB data bus and three lock-step replicas in the sphere of replication, on for each functional DSMC instance.

Due to this architecture choice, each DSMC can access the whole addressing space.

7.6 Analog modules

This section highlights the following analog modules that are implemented on SPC58xEx/SPC58xGx:

Table 52. Reference links to related information

Module or Subsystem	Device-specific information	Block details
Temperature sensor	*Section 7.6.1: Temperature sensor configuration*	*Chapter 43: Temperature Sensor*
ADC subsystem	*Section 40.1: ADC overview*	—
Successive Approximation Register Analog-to-Digital Converters (SARADCs)	*Section 7.6.2: SARADC*	*Section 42: Successive Approximation Register Analog-to-Digital Converter (SARADC) on page 1118*
Sigma-Delta Analog-to-Digital Converter (SDADC)	*Section 7.6.3: SDADC*	*Chapter 41: Sigma-Delta Analog-to-Digital Converter (SDADC) Digital Interface*

7.6.1 Temperature sensor configuration

The temperature sensor is used to measure the temperature of the SPC58xEx/SPC58xGx device. The temperature sensor has the following outputs:

- Two analog voltages
  - A voltage signal that is linearly increasing with internal junction temperature (PTAT that is proportional to absolute temperature)
  - A voltage signal that is linearly decreasing with internal junction temperature (CTAT that is complementary proportional to absolute temperature)
- Three digital outputs that signal under- or over-temperature operating conditions.

During production testing of the device, the output voltages of the temperature sensor are sampled by SARB input channel 120 and 121 of the onboard ADC module at a predefined high temperature and also at a predefined low temperature. These calibration parameters are stored during production test into UTEST Flash memory. See Section UTEST in Chapter Flash Memory.

By converting the two analog outputs of the temperature sensor through ADC, one can find out a constant reference code which does not change with temperature or the ADC reference values. It is also called 'Digital Bandgap Voltage'. See *Section 43.3.2: Equations for converting TSENS voltages into constant reference (Digital Bandgap Voltage)* in *Chapter 43: Temperature Sensor*.

For details, see *Chapter 43: Temperature Sensor*.

7.6.2 SARADC

The Successive Approximation Register Analog-to-Digital Converter (SARADC) digital interface block controls the on-chip SARADC analog block and holds control and status registers accessible for an application. The SARADC provides accurate and fast conversion data for a wide range of applications. Each SARADC analog block has a corresponding digital interface.

SPC58xEx ncludes:

- five (5) independent 12-bit SARADCs.
- three (3) independent 10-bit SARADCs (2 + 1 standby).

SPC58xGx includes:

- five (5) independent 12-bit SARADCs.
- one independent 10-bit SARADCs (standby).

The 12-bit SARADC_B has access to all external analog input pins, access to the external multiplexed inputs, and has special safety features.

For details, refer to the section Analog-to-Digital Converters (ADC) Configuration.

7.6.3 SDADC

The Sigma-Delta Analog-to-Digital Converter (SDADC) digital interface block controls the on-chip SDADC and holds control and status registers accessible for application. It provides the accurate conversion data for a wide range of applications.

SPC58xEx/SPC58xGx includes six independent 16-bit SDADCs.

For details, see *Chapter 40: Analog-to-Digital Converters (ADC) Configuration*.

7.6.4 Body Cross Triggering Unit (BCTU) configuration

This section summarizes the Body Cross Triggering Unit (BCTU) configuration in the SPC58xEx/SPC58xGx device. For a comprehensive description of the BCTU, please refer to the BCTU's dedicated chapter.

Module overview

One single BCTU instance (BCTU0) is implemented. It is interconnected to:

- ADC_SAR_B (12-bit Supervisor)
- eMIOS
- PIT and GTM via IOMUX

The Body Cross Triggering Unit is used to prioritize the trigger events from eMIOS/PIT/GTM.

The BCTU synchronizes the ADC conversion with PWM signal coming from eMIOS/PIT/GTM (refer to *Table 53*). When the event is programmed to be an ADC trigger, the 7-bit channel number is provided to ADC. This channel number is used to communicate which particular channel has to be converted (refer to *Table 54*).

The BCTU is also used to route the trigger events from eMIOS as a DMA request in case the event is not programmed to be a trigger to ADC.

The supported trigger sources are the following:

- 62 eMIOS channels
- 2 PIT channels and 2 GTM channels: the trigger source selection is done via IOMUX. User selects between PIT and GTM source event by means of IOMUX selection.

Figure 23. BCTU overview

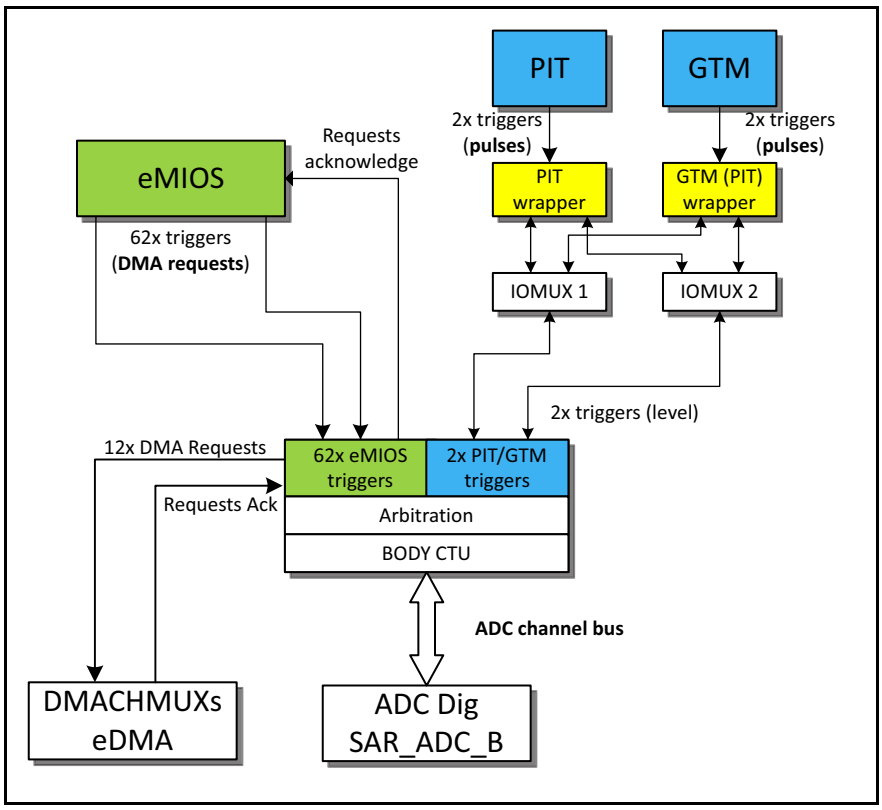


Table 53. BCTU input triggers allocation(1)

Body CTU trigger input	Trigger source	Connected signal
0–22	eMIOS_0	eMIOS_0 CH0–CH22
23	PIT_RTI_0 via IOMUX	PIT_0 CH1
	GTM via IOMUX	GTM_TOM2_IRQ7
24–31	eMIOS_0	eMIOS_0 CH24–CH31
32–54	eMIOS_1	eMIOS_1 CH0–CH22
55	PIT_RTI_0 via IOMUX	PIT_0 CH3
	GTM via IOMUX	GTM_TOM2_IRQ0
56–63	eMIOS_1	eMIOS_1 CH24–CH31

1. Refer to the excel file attached to this Reference Manual (IO Definition) for further information on PIT and GTM input signals selection.

Table 54. Channel value ADC mapping

SAR_ADC_B	BCTU channel number
Internal channels	
0	0x00

Table 54. Channel value ADC mapping (continued)

SAR_ADC_B	BCTU channel number
1	0x01
...	...
63	0x3F
External channels	
128	0x40
129	0x41
...	...
191	0x7F

7.7 Timers

7.7.1 System Timer Module (STM) configuration

The System Timer Module (STM) is a 32-bit timer designed to support commonly required system and application software timing functions. The STM includes a 32-bit up counter and four 32-bit compare channels with a separate interrupt source for each channel. The counter is driven by the FBRIDGE_CLK divided by an 8-bit prescale value (1 to 256).

SPC58xEx/SPC58xGx includes three STMs, one for each core, with four 32-bit compare channels in each STM.

Table 55. System Timer Module (STM) instances

Instance	Description	Number of channels	Number of registers (32-bit)
STM_0	Clocked by FBRIDGE_CLK (Core_0)	4	14
STM_1	Clocked by FBRIDGE_CLK (Core_1)	4	14
STM_2	Clocked by FBRIDGE_CLK (Core_2)	4	14

7.7.2 Software Watchdog Timer (SWT) configuration

SPC58xEx/SPC58xGx includes four SWTs.

Table 56. Software Watchdog Timer (SWT) instances

Instance	Description
SWT0	Clocked by IRCOSC clock; using value of IAC8 register of Core 0 for instruction address compare feature
SWT1	Clocked by IRCOSC clock; using value of IAC8 register of Core 1 for instruction address compare feature

Table 56. Software Watchdog Timer (SWT) instances (continued)

Instance	Description
SWT2	Clocked by IRCOSC clock; using value of IAC8 register of Core 2 for instruction address compare feature
SWT3	Clocked by IRCOSC clock; not supporting the address compare feature

The SWT has the following features:

- 32-bit time-out register to set the time-out period
- Internal 16 MHz RC oscillator clock that is used for timer operation
- Programmable selection of window mode or regular servicing
- Programmable selection of reset or interrupt on an initial time-out
- Programmable selection of the servicing mode
- Master access protection
- Hard and soft configuration lock bits

7.7.2.1 IAC (Instruction Address Compare)

The IAC feature makes it possible to serve a SWT when the CPU executes an instruction at a specific address location. The purpose is to guarantee that the application regularly executes an instruction within some fundamental section of the code. If this address location is not executed regularly, it implies that a critical error has occurred and the device should be eventually reset.

Not all the SWT embeds the IAC feature: *Table 56* shows for which watchdog the IAC feature is enabled.

If the IAC feature is enabled for a specific SWT, the address to be executed is set by the Instruction Address Compare 8 (IAC8) register, whereas the IAC feature can be activated by the register field SWT_CR[SMD] to one of the following settings:

- 2'b10 for Fixed Address Execution; the IAC8 register cannot be updated while the watchdog is enabled. If the SWT is in 'Fixed Address Execution' mode, an SWT output signal tells the core; this 'freezes' changes to the IAC8 register when the SWT is enabled
- 2'b11 for Incremental Address Execution; the IAC8 register can be updated

7.7.2.2 Reset assertion

The SWT can assert a reset when the watchdog timer expires. See *Chapter 8: Reset and Boot* chapter for details.

7.7.2.3 Default configuration

On SPC58xEx/SPC58xGx, the SWTs come out of reset with the default reset values summarized in *Table 57*. Minimum timeout value is 256 cycles for SWT0, SWT1, SWT2, and SWT3. For details, please refer to *Section 65.3.5: Initialization of BAF DCF clients*

Table 57. Register reset values

SWT instance	Register	Reset value
SWT0	SWT Control Register (SWT_CR)	0xFF00_010A
SWT0	SWT Time-out (SWT_TO)	0x0005_FCD0
SWT1	SWT Control Register (SWT_CR)	0xFF00_010A
SWT1	SWT Time-out (SWT_TO)	0x0005_FCD0
SWT2	SWT Control Register (SWT_CR)(1)	0xFF00_011B
SWT2	SWT Time-out (SWT_TO)	0x0003_FDE0
SWT3	SWT Control Register (SWT_CR)	0xFF00_010A
SWT3	SWT Time-out (SWT_TO)	0x0005_FCD0

1. The SWT can be disabled by accepting a DCI-driven signal to clear the SWT_CR[WEN] bit. For SWT2, the default reset value for that bit is '1', but the DCI can override that default value by setting it to '0'. If the signal is de-asserted, there is no change to the SWT_CR[WEN] value. (The DCI can only disable the SWT; it cannot enable it.) For details refer to *Section 72.1.3: Operating modes*.

7.7.3 PIT configuration

This section summarizes how the module has been configured in the chip. For a comprehensive description of the module itself, see the module's dedicated chapter.

Figure 24. PIT configuration

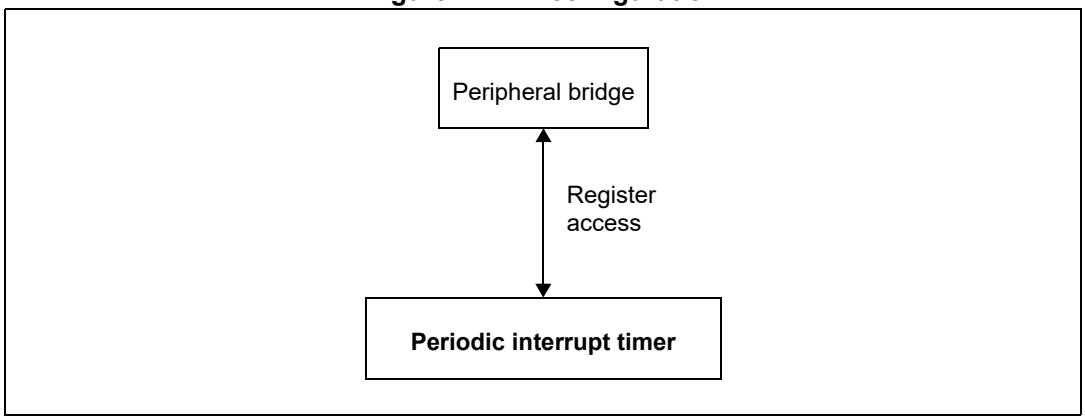


Table 58. Reference links to related information

Topic	Related module	Reference
Full description	PIT	*Chapter 50: Periodic Interrupt Timer (PIT)*
System memory map	—	*Chapter 5: Memory Map*
Clocking	—	*Chapter 27: Clocking*
Power management	—	*Chapter 10: Power management*
DMA channel assignments	Direct memory access channel multiplexer	*Chapter 26: DMA channel multiplexer (DMACHMUX)*

7.7.3.1 PIT instantiation

Table 59. PIT information

PIT module	PIT channels	Description
PIT_0	8 + RTI	The PIT has eight standard channels. All PIT registers are cleared by any reset.
PIT_1	2	PIT1 has chaining mode to implement a 64-bit timer and the timer value is unchanged by a functional reset.

7.7.3.2 PIT/DMA periodic trigger assignments

The PIT generates periodic trigger events to the DMACHMUX. To check the mapping of the PIT channel number with the corresponding DMA channel number, refer to the *Table 40: PIT_0 periodic trigger to DMA channel mapping*.

7.7.3.3 PIT Registers

The following tables show the implementation of PIT registers in this MCU.

Table 60. PIT_0 memory map

Absolute address (hex)	Register name
F7F8_4000	PIT Module Control Register (PIT0_MCR)(1)
F7F8_40E0 - F7F8_40E4	Reserved
F7F8_40F0 - F7F8_40FC	RTI Timer Channel Registers (PIT0_RTI_LDVAL, PIT0_RTI_CVAL, PIT0_RTI_TCTRL, PIT0_RTI_TFLG)
F7F8_4100 - F7F8_417C	Timer Channel [0:7] Registers (PIT0_LDVAL*n*, PIT0_CVAL*n*, PIT0_TCTRL*n*, PIT0_TFLG*n*)
F7F8_4180 - F7F8_7FFF	Reserved

1. The reset value of MCR is 0x06.

Table 61. PIT_1 memory map

Absolute address (hex)	Register name
FBF8_4000	PIT Module Control Register (PIT1_MCR)(1)
FBF8_40E0	PIT Upper Lifetime Timer Register (PIT1_LTMR64H)
FBF8_40E4	PIT Lower Lifetime Timer Register (PIT1_LTMR64L)
FBF8_40F0 - FBF8_40F8	Reserved

Table 61. PIT_1 memory map (continued)

Absolute address (hex)	Register name
FBF8_4100 - FBF8_411C	Timer Channel [0:1] Registers (PIT1_LDVAL*n*, PIT1_CVAL*n*, PIT1_TCTRL*n*, PIT1_TFLG*n*)
F7F8_4120 - F7F8_7FFF	Reserved

1. The reset value of MCR is 0x02.

7.7.3.4 PIT clocking

The PIT_0 standard channels are clocked with the peripheral clock (PER_CLK).

The PIT_0_RTI channel clock source is selectable between XOSC and RCOSC via the auxiliary clock selector 9.

The PIT_1 channels are clocked with the peripheral clock (PER_CLK).

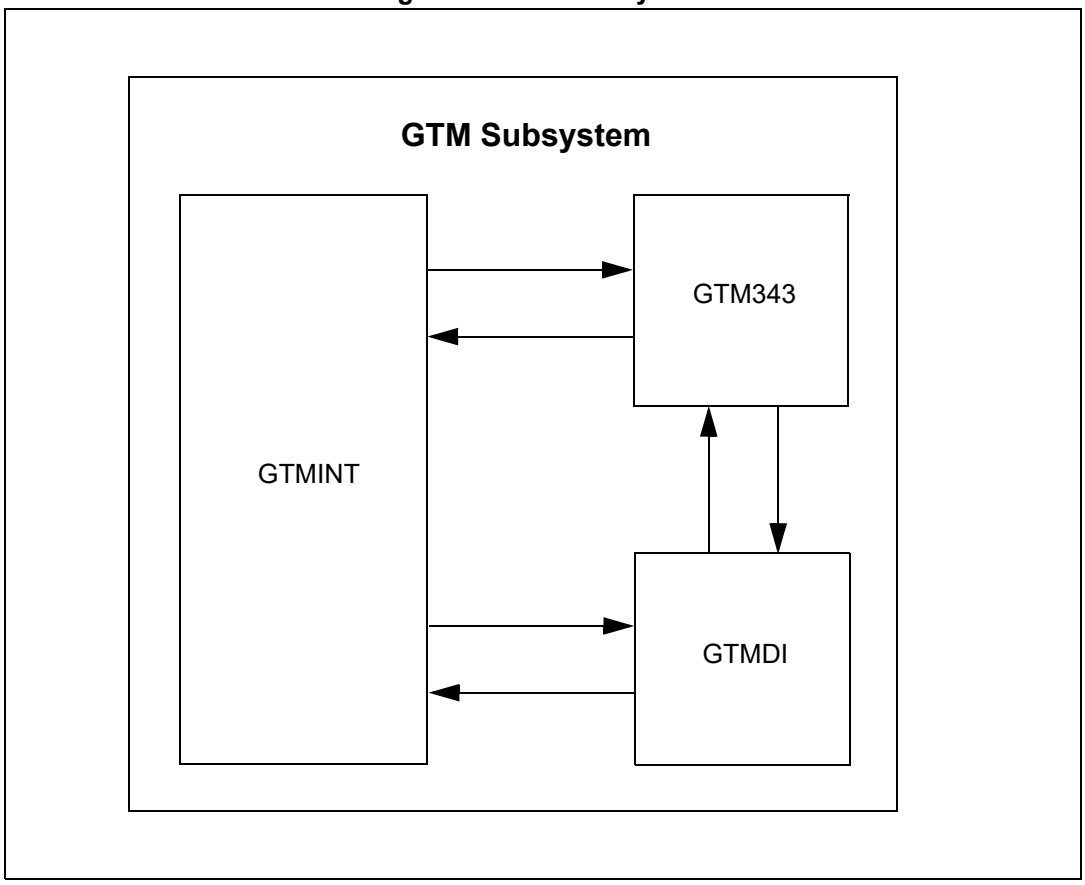
7.7.4 GTM subsystem configuration

The GTM343 integration module (GTMINT) is an interface to the General Timer Module 343 (GTM343) that adds ECC functionality and a Nexus-based system debugger interface (GTMDI). It has two control registers and provides access to the GTM343's memory-mapped registers. The GTM343 is detailed separately in the GTM-IP_343_Specification. See *Chapter 47: GTM343 Integration (GTMINT) Module*, for more information.

7.7.4.1 GTM subsystem

The diagram provided in *Figure 25* illustrates the parts of the GTM subsystem.

Figure 25. GTM subsystem



7.7.4.2 Parts of the GTM subsystem

The *Table 62* describes the parts of the GTM subsystem.

Table 62. Parts of the GTM subsystem

Part	Function
GTMINT	Provides reset control, clock control, error-correcting code (ECC) functionality, interrupt and DMA control, and access to GTM343's memory mapped registers. For debug accesses from GTMDI to GTM343 registers, prevents destructive read accesses into the GTM343 registers.
GTM343	Provides programmable timer functionality.
GTMDI	Provides real-time development capabilities for the GTM subsystem.

7.7.4.3 GTM subsystem clock sources

The system clock and the peripheral clocks to the GTM subsystem are provided by the Clock Generation Module (MC_CGM).

Table 63. GTM subsystem clock sources

GTM subsystem clock	MC_CGM source	Maximum frequency
SYS_CLK	XBAR_CLK	90 MHz
GTMIP_CLK	PER_CLK	100 MHz
GTMDI_CLK		
GTMINT_CLK		

7.8 Communication interfaces

7.8.1 Ethernet configuration

7.8.1.1 Ethernet interface selection

On this chip, the Ethernet interface depends on the SIUL2_SCR0 settings (in the SIUL2), as shown in *Table 64*.

Table 64. Ethernet interface selection

SIUL2_SCR0[Ethernet_MODE]	Interface used by the Ethernet
0	RMII
1	MII

7.8.1.2 Ethernet timestamp snapshot trigger

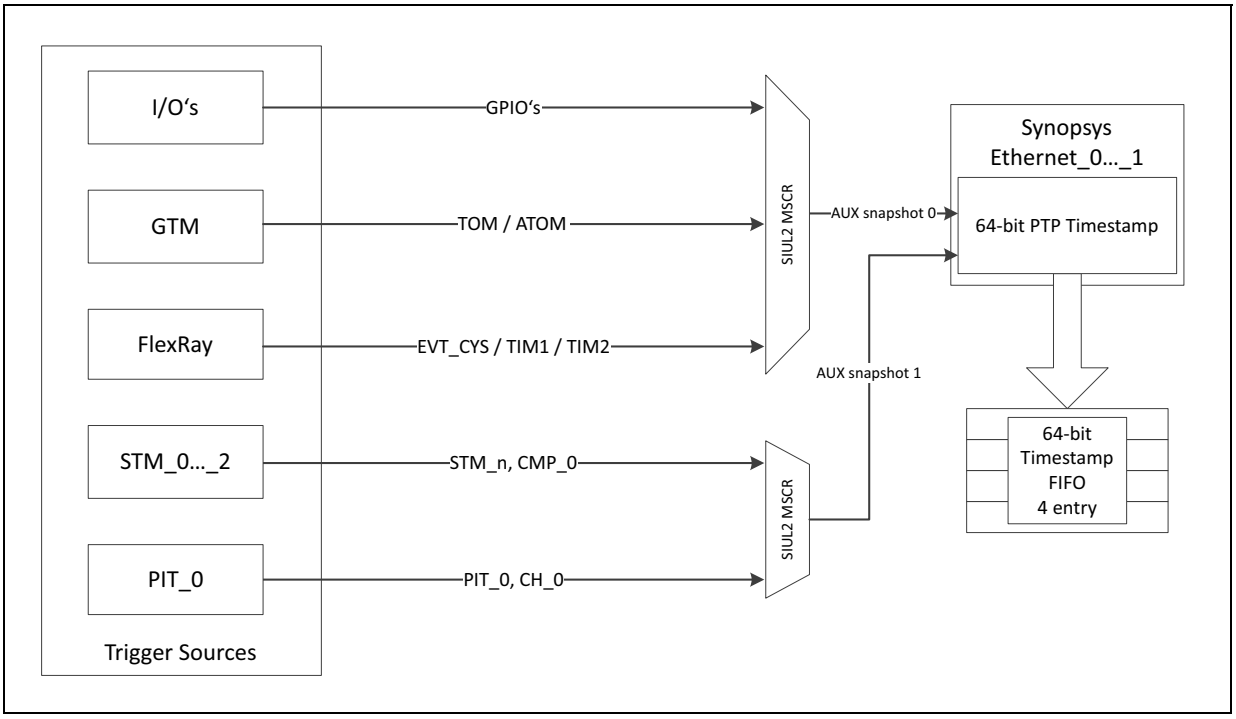
The Ethernet interface includes an internal 64-bit time base counter to support IEEE 1588 Precision Time Protocol (PTP) timestamps.

To synchronize other system events to the Ethernet PTP time base snapshot feature is available to capture the Ethernet time base on different system events into a four-entry-deep FIFO. Details about the snapshot feature is available in section “IEEE 1588 Timestaps” in Ethernet chapter.

The selection of the various trigger sources is done in the SIUL2 module by configuration of the corresponding MSCR registers. The details about the signal configuration is available in I/O Signal table.

The system implementation of the snapshot feature is shown in *Figure 26*.

Figure 26. Ethernet Snapshot feature integration

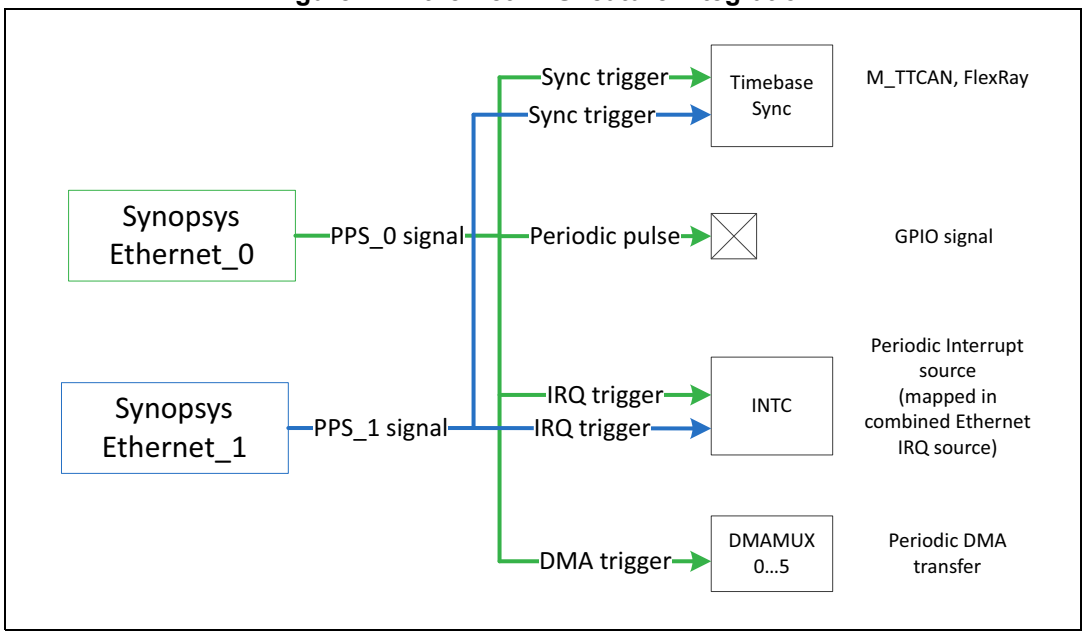


7.8.1.3 Ethernet Pulse-Per-Second output (PPS)

The Ethernet interface module includes a programmable time reference signal which can be used to trigger several system events. Details about the Pulse-Per-Second feature is available in section “Reference Timing Source” in Ethernet chapter.

The system implementation of the PPS feature is shown in *Figure 27*.

Figure 27. Ethernet PPS feature integration



7.8.2 CAN subsystem configuration

Two CAN subsystems are implemented:

- CAN subsystem_0 (M_CAN with M_TTCAN type)
- CAN subsystem_1 (M_CAN type)

Refer to *Chapter 52: CAN Subsystem* for details.

7.8.2.1 CAN nodes

Table 65 lists the different CAN nodes for both CAN subsystems.

Table 65. CAN bus controllers

Subsystem	Peripheral bridge instance	Shared message RAM size (words)	Type	Instance	CAN FD support up to 64 data byte(1)	Debug on CAN support	Clock calibration(2)
CAN subsystem_0	PBRIDGE_2	3968	M_CAN with TT_CAN	CAN_SUB_0_M_TTCAN_0	Yes	No	Yes
				CAN_SUB_0_M_CAN_1	Yes	Yes	Yes
				CAN_SUB_0_M_CAN_2	Yes	Yes	Yes
				CAN_SUB_0_M_CAN_3	Yes	No	Yes
CAN subsystem_1	PBRIDGE_1	3968	M_CAN	CAN_SUB_1_M_CAN_1	Yes	No	No
				CAN_SUB_1_M_CAN_2	Yes	No	No
				CAN_SUB_1_M_CAN_3	Yes	No	No
				CAN_SUB_1_M_CAN_4	Yes	No	No

1. M_CAN revision 3.2.1
2. Clock calibration based on M_CAN_1 messages for CAN subsystem_0.

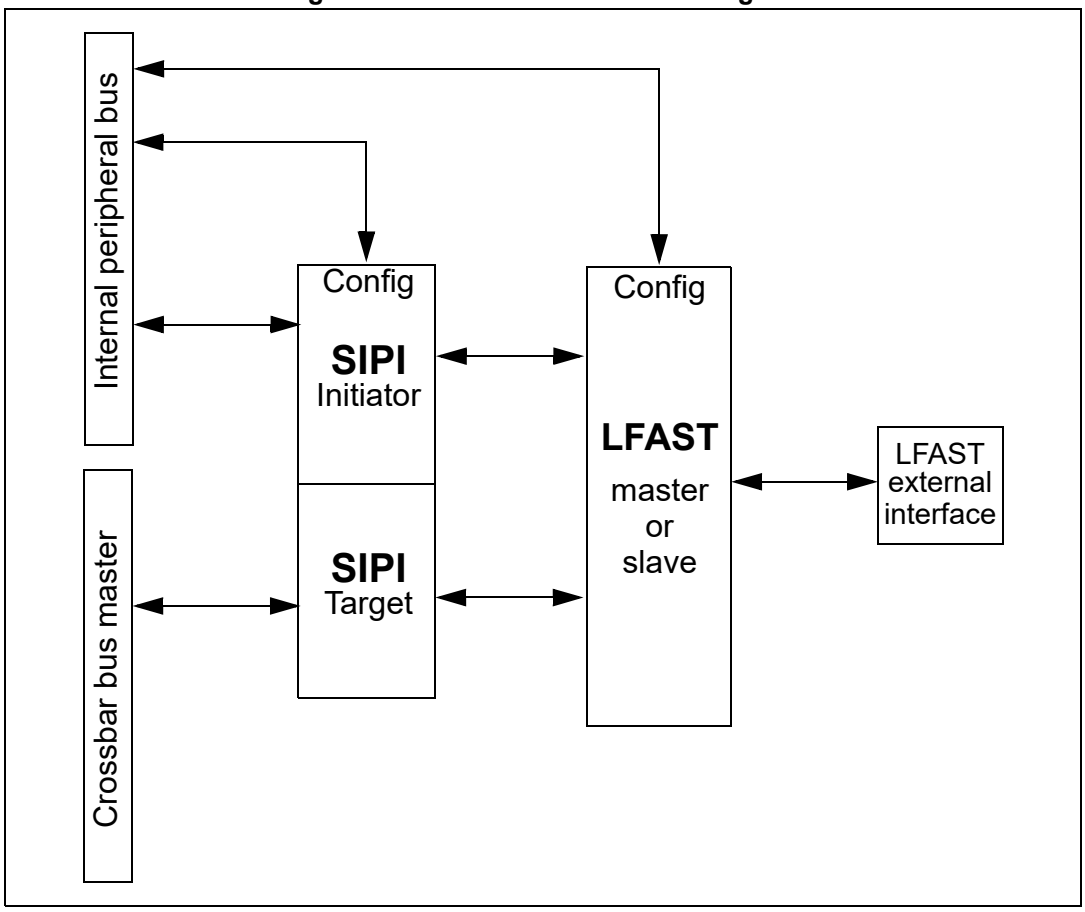
7.8.3 LFAST and SIPI

This section summarizes the module configuration in the controller. For a comprehensive description of SIPI and LFAST, please refer to the dedicated SIPI and LFAST chapters.

7.8.3.1 Instantiation

The SIPI and LFAST are connected to one another and appear as a single unit. The LFAST portion of the two modules allows for high speed inter-device communications. The SIPI allows memory to be shared between devices which have SIPI and LFAST communication modules (see *Figure 28*).

Figure 28. LFAST and SIPI block diagram



7.8.3.2 Device-specific features

The LFAST can operate in either slave or master mode configurations. The node running in master mode controls the serial link, but SIPI, in both master and slave modes, can act as both initiator and target for SIPI commands simultaneously. Please see the SIPI and LFAST chapters for detailed information on module configuration and functionality. The SIPI CHIP ID = 00110041h for SPC58xEx/SPC58xGx

7.8.3.3 Memory map and register description

Table 66. Reference text to related information

Topic	Related module	Reference
LVDS Fast Asynchronous Serial Transmission	LFAST	*Chapter 55: LVDS Fast Asynchronous Serial Transmission – Interprocessor Communications*
Serial Interprocessor Interface	SIPI	*Chapter 53: Serial Interprocessor Interface (SIPI)*
Direct memory access	eDMA	*Chapter 25: Enhanced Direct Memory Access (eDMA)*
System memory map	Memory Map	*Chapter 5: Memory Map*

7.8.4 DSPI configuration

This section summarizes how the module has been configured in the chip. For a comprehensive description of the module itself, see the module's dedicated chapter.

Figure 29. DSPI configuration

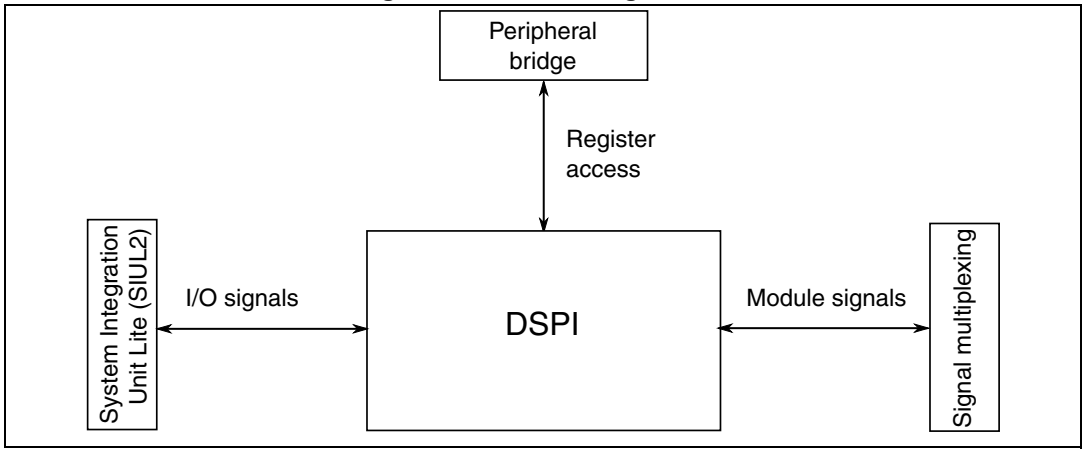


Table 67. Reference links to related information

Topic	Related module	Reference
Full description	DSPI	*Chapter 62: Deserial Serial Peripheral Interface (DSPI)*
System memory map		*Chapter 5: Memory Map*
Clocking		*Chapter 27: Clocking*
Signal Multiplexing	Port control	*Chapter 4: Signal Description*
System Module	System Integration Unit Lite	*Chapter 17: System Integration Unit Lite2 (SIUL2)*
DMA Channel Assignments	Direct memory access multiplexer	*Chapter 26: DMA channel multiplexer (DMACHMUX)*

7.8.4.1 DSPI instantiation

This device contains ten DSPI modules. *Table 68* shows the instantiation information for each module.

Table 68. DSPI instantiation

DSPI Module	CTARs	TX FIFO Depth	RX FIFO Depth	PCS Signals	LVDS Support	Data Serialization Support
DSPI0	8	4	4	Refer to the SPC58xEx/SPC58xGx pinout Excel file attached to the IO_Definition document	Not Available	—
DSPI1	8	4	4		Not Available	—
DSPI2	8	4	4		Available(1)	—
DSPI3	8	4	4		Not Available	—
DSPI4	8	4	4		Available(2)	64
DSPI5	8	4	4		Available(3)	64
DSPI6	8	4	4		Not Available	—
DSPI7	8	4	4		Not Available	—
DSPI8	8	4	4		Not Available	—
DSPI9	8	4	4		Not Available	—

1. Supports Differential DSPI with SCK, SIN and SOUT LVDS signal pairs.
2. Supports Microsecond Bus SCK and SOUT LVDS signal pairs.
3. Supports Microsecond Bus SCK and SOUT LVDS signal pairs; supports Differential DSPI with SCK, SIN and SOUT LVDS signal pairs

7.8.4.2 DSPI half duplex operation

Half duplex operation is not supported within the DSPI module, but it is configured by connecting the SIN and SOUT together within the SIUL2 and operating the combined pad in open drain mode. Software must ensure that the SOUT signal is passive (high) during SIN reception. This can be done by reconfiguring the SIUL2 to disconnect the SOUT or by writing a data frame that is all '1's.

7.8.4.3 Serialized timed I/O

The timer sources and injection pin inputs to be DSPI serialized downstream outputs are selected though the SIUL2 and selected through the MSCR registers. Inversion of these signals is also handled in the SIUL2. Refer to *Chapter 17: System Integration Unit Lite2 (SIUL2)*, for details.

7.8.4.4 High Speed and MicroSecond Channel (MSC) Support

The baud rate generator in the DSPI is clocked from the peripheral clock. The DSPI module can operate at a baud rate of up to half the peripheral clock.

To support microsecond channel functionality, pairs of LVDS signals are implemented for the appropriate DSPI signals. See *Table 68: DSPI instantiation* for details. In addition, SIUL2 MSCR muxing is implemented to mux the SOUT and CLK onto these LVDS signals well as standard CMOS signals. Finally, in the SIUL2 MSCR pin muxing, some input pins can be routed to 64 possible serialization inputs, each selectable for a specific DSPI.

Details of these pin assignments can be found in the SPC58xEx/SPC58xGx I/O Signal Description and Input Multiplexing Tables (Excel file) attached to the IO_Definition document. Locate the paperclip symbol on the left side of the PDF window, and click it. Double-click on the Excel file to open it and select the Input Multiplexing Table tab.

MSC is supported in both TSB and ITSB modes.

7.8.4.5 DSPI HT external trigger

External hardware triggers are not supported.

7.8.4.6 DSPI ITSB mode

For the ITSB trigger, the internal DSPI trigger (TRGPRD in DSICR1) is used. Use of external trigger is not supported.

7.8.4.7 DSPI registers

The absolute addresses of DSPI module registers in this MCU are provided in *Chapter 5: Memory Map*.

7.8.5 FlexRay configuration

The FlexRay communications system is designed to provide high-speed deterministic distributed control for advanced automotive applications. Its dual-channel architecture supporting data rates up to 10 Mbit/s per channel offers system-wide redundancy that supports the reliability requirements of safety system.

7.8.5.1 FlexRay signals pin assignment

This table lists the pin names for the FlexRay signals that are connected to pins on the boundary of the chip.

Table 69. Flexray pin names

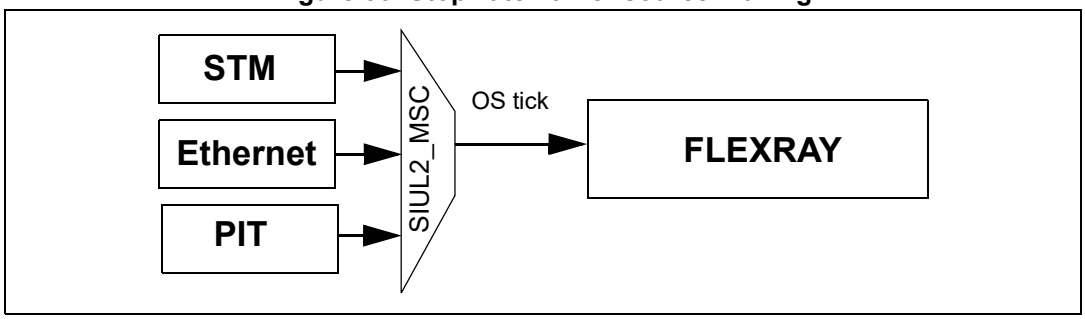
Signal description	Pin name
FR_A_RX (Receive Data Channel A)	RXDA
FR_A_TX (Transmit Data Channel A)	CA_TX
$\overline{\text{FR_A_TX_EN}}$ (Transmit Enable Channel A)	CA_TR_EN
FR_B_RX (Receive Data Channel B)	RXDB
FR_B_TX (Transmit Data Channel B)	CB_TX
$\overline{\text{FR_B_TX_EN}}$ (Transmit Enable Channel B)	CB_TR_EN
FR_DBG[0] (Debug Strobe Signal 0)	DBG[0]
FR_DBG[1] (Debug Strobe Signal 1)	DBG[1]
FR_DBG[2] (Debug Strobe Signal 2)	DBG[2]
FR_DBG[3] (Debug Strobe Signal 3)	DBG[3]

7.8.5.2 Stopwatch timer input

Stopwatch timer input is routed to FlexRay module (selected by SIUL MSCR register) via:

- STM0_CMP0
- STM1_CMP0
- STM2_CMP0
- PIT1_CH0
- Ethernet_0 PTP_PPS_0

Figure 30. Stopwatch timer source muxing



7.8.6 PSI5 controller configuration

The PSI5 chapter has 2 instances. Parameters for the 2 instances are listed below:

Table 70. PSI5 controller generics

Signal name	Description	PSI5_0 value	PSI5_1 value
PSI5_NUM_CH	Number of channels	1	1
PSI5_PDIS_RST	Default State of the IP after reset	1	1

PSI5_0 and PSI5_1 have both a single channel (Channel0). PSI5_0 is a master instance here hence the PSI5_GCR register for this instance controls all the 2channels ; PSI5_0 (Channel 0) and PSI5_1 (Channel0). PSI5_1 instance does not contain the PSI5_GCR register and hence the corresponding PSI5_GCR address location for PSI5_1 is reserved. Further, with respect to the PSI5 base address the unimplemented reserved locations for PSI5_0 extend from 0x01C8 to 0xFFFF and unimplemented reserved locations for PSI5_1 extend from 0x01C8 to 0xFFFF.

7.8.7 SENT Receiver (SRX) configuration

7.8.7.1 SRX channels

Table 71 shows the SENT receiver channels for each instance configured on this chip.

Table 71. SENT channel configuration

Instance	Channels (CH)
SENT 0	8 Channels SENT_0_CH0 SENT_0_CH1 SENT_0_CH2 SENT_0_CH3 SENT_0_CH4 SENT_0_CH5 SENT_0_CH6 SENT_0_CH7
SENT 1	7 Channels SENT_1_CH0 SENT_1_CH1 SENT_1_CH2 SENT_1_CH3 SENT_1_CH4 SENT_1_CH5 SENT_1_CH6

7.8.7.2 SRX registers

Table 72 lists the address offset and memory map for both SENT receiver (SRX) instances for this chip.

Table 72. SENT receiver memory map

Offset	Register name	SENT 0	SENT 1
0x0000	GBL_CTRL— Global Control Register	X	X
0x0004	CHNL_EN—Channel Enable Register	X	X
0x0008	GBL_STATUS—Global Status Register	X	X
0x000C	FMSG_RDY—Fast Message Ready Register	X	X
0x0010	SMSG_RDY—Slow Serial Message Ready Register	X	X
0x0014	Reserved	—	—
0x0018	DATA_CTRL1— Data Control Register 1	X	X
0x001C	Reserved	—	—
0x0020	Reserved	—	—
0x0024	Reserved	—	—
0x0028	FDMA_CTRL—Fast Message DMA Control Register	X	X
0x002C	SDMA_CTRL—Slow Serial Message DMA Control Register	X	X
0x0030	Reserved	—	—
0x0034	FRDY_IE—Fast Message Ready Interrupt Control Register	X	X
0x0038	SRDY_IE—Slow Serial Message Ready Interrupt Control Register	X	X

Table 72. SENT receiver memory map (continued)

Offset	Register name	SENT 0	SENT 1
0x003C	Reserved	—	—
0x0040	DMA_FMSG_DATA—DMA Fast Message Read Register	X	X
0x0044	DMA_FMSG_CRC—DMA Fast Message CRC Read Register	X	X
0x0048	DMA_FMSG_TS—DMA Fast Message Time Stamp Read Register	X	X
0x004C	Reserved	—	—
0x0050	DMA_SMSG_BIT3—DMA Serial Message Read Register (Bit 3)	X	X
0x0054	DMA_SMSG_BIT2—DMA Serial Message Read Register (Bit 2)	X	X
0x0058	DMA_SMSG_TS—DMA Serial Message Time Stamp Read Register	X	X
0x005C	Reserved	—	—
0x0060	CH0_CLK_CTRL—Channel 0 Clock Control Register	X	X
0x0064	CH0_STATUS—Channel 0 Status Register	X	X
0x0068	CH0_CONFIG—Channel 0 Configuration Register	X	X
0x006C	Reserved	—	—
0x0070	CH1_CLK_CTRL—Channel 1 Clock Control Register	X	X
0x0074	CH1_STATUS—Channel 1 Status Register	X	X
0x0078	CH1_CONFIG—Channel 1 Configuration Register	X	X
0x007C	Reserved	—	—
0x0080	CH2_CLK_CTRL—Channel 2 Clock Control Register	X	X
0x0084	CH2_STATUS—Channel 2 Status Register	X	X
0x0088	CH2_CONFIG—Channel 2 Configuration Register	X	X
0x008C	Reserved	—	—
0x0090	CH3_CLK_CTRL—Channel 3 Clock Control Register	X	X
0x0094	CH3_STATUS—Channel 3 Status Register	X	X
0x0098	CH3_CONFIG—Channel 3 Configuration Register	X	X
0x009C	Reserved	—	—
0x00A0	CH4_CLK_CTRL—Channel 4 Clock Control Register	X	X
0x00A4	CH4_STATUS—Channel 4 Status Register	X	X
0x00A8	CH4_CONFIG—Channel 4 Configuration Register	X	X
0x00AC	Reserved	—	—
0x00B0	CH5_CLK_CTRL—Channel 5 Clock Control Register	X	X
0x00B4	CH5_STATUS—Channel 5 Status Register	X	X
0x00B8	CH5_CONFIG—Channel 5 Configuration Register	X	X
0x00BC	Reserved	—	—
0x00C0	CH6_CLK_CTRL—Channel 6 Clock Control Register	X	X
0x00C4	CH6_STATUS—Channel 6 Status Register	X	X

Table 72. SENT receiver memory map (continued)

Offset	Register name	SENT 0	SENT 1
0x00C8	CH6_CONFIG—Channel 6 Configuration Register	X	X
0x00cC	Reserved	—	—
0x00D0	CH7_CLK_CTRL—Channel 7 Clock Control Register	X	—
0x00D4	CH7_STATUS—Channel 7 Status Register	X	—
0x00D8	CH7_CONFIG—Channel 7 Configuration Register	X	—
0x00DC	Reserved	—	—
0x0160	CH0_FMSG_DATA—Channel 0 Fast Message Register	X	X
0x0164	CH0_FMSG_CRC—Channel 0 Fast Message CRC Register	X	X
0x0168	CH0_FMSG_TS—Channel 0 Fast Message Time Stamp Register	X	X
0x016C	CH0_SMSG_BIT3—Channel 0 Serial Message Register (Bit 3)	X	X
0x0170	CH0_SMSG_BIT2—Channel 0 Serial Message Register (Bit 2)	X	X
0x0174	CH0_SMSG_TS—Channel 0 Serial Message Time Stamp Register	X	X
0x0178	CH1_FMSG_DATA—Channel 1 Fast Message Register	X	X
0x017C	CH1_FMSG_CRC—Channel 1 Fast Message CRC Register	X	X
0x0180	CH1_FMSG_TS—Channel 1 Fast Message Time Stamp Register	X	X
0x0184	CH1_SMSG_BIT3—Channel 1 Serial Message Register (Bit 3)	X	X
0x0188	CH1_SMSG_BIT2—Channel 1 Serial Message Register (Bit 2)	X	X
0x018C	CH1_SMSG_TS—Channel 1 Serial Message Time Stamp Register	X	X
0x0190	CH2_FMSG_DATA—Channel 2 Fast Message Register	X	X
0x0194	CH2_FMSG_CRC—Channel 2 Fast Message CRC Register	X	X
0x0198	CH2_FMSG_TS—Channel 2 Fast Message Time Stamp Register	X	X
0x019C	CH2_SMSG_BIT3—Channel 2 Serial Message Register (Bit 3)	X	X
0x01A0	CH2_SMSG_BIT2—Channel 2 Serial Message Register (Bit 2)	X	X
0x01A4	CH2_SMSG_TS—Channel 2 Serial Message Time Stamp Register	X	X
0x01A8	CH3_FMSG_DATA—Channel 3 Fast Message Register	X	X
0x01AC	CH3_FMSG_CRC—Channel 3 Fast Message CRC Register	X	X
0x01B0	CH3_FMSG_TS—Channel 3 Fast Message Time Stamp Register	X	X
0x01B4	CH3_SMSG_BIT3—Channel 3 Serial Message Register (Bit 3)	X	X
0x01B8	CH3_SMSG_BIT2—Channel 3 Serial Message Register (Bit 2)	X	X
0x01BC	CH3_SMSG_TS—Channel 3 Serial Message Time Stamp Register	X	X
0x01C0	CH4_FMSG_DATA—Channel 4 Fast Message Register	X	X
0x01C4	CH4_FMSG_CRC—Channel 4 Fast Message CRC Register	X	X
0x01C8	CH4_FMSG_TS—Channel 4 Fast Message Time Stamp Register	X	X
0x01CC	CH4_SMSG_BIT3—Channel 4 Serial Message Register (Bit 3)	X	X
0x01D0	CH4_SMSG_BIT2—Channel 4 Serial Message Register (Bit 2)	X	X

Table 72. SENT receiver memory map (continued)

Offset	Register name	SENT 0	SENT 1
0x01D4	CH4_SMSG_TS—Channel 4 Serial Message Time Stamp Register	X	X
0x01D8	CH5_FMSG_DATA—Channel 5 Fast Message Register	X	X
0x01DC	CH5_FMSG_CRC—Channel 5 Fast Message CRC Register	X	X
0x01E0	CH5_FMSG_TS—Channel 5 Fast Message Time Stamp Register	X	X
0x01E4	CH5_SMSG_BIT3—Channel 5 Serial Message Register (Bit 3)	X	X
0x01E8	CH5_SMSG_BIT2—Channel 5 Serial Message Register (Bit 2)	X	X
0x01EC	CH5_SMSG_TS—Channel 5 Serial Message Time Stamp Register	X	X
0x01F0	CH6_FMSG_DATA—Channel 6 Fast Message Register	X	X
0x01F4	CH6_FMSG_CRC—Channel 6 Fast Message CRC Register	X	X
0x01F8	CH6_FMSG_TS—Channel 6 Fast Message Time Stamp Register	X	X
0x01FC	CH6_SMSG_BIT3—Channel 6 Serial Message Register (Bit 3)	X	X
0x0200	CH6_SMSG_BIT2—Channel 6 Serial Message Register (Bit 2)	X	X
0x0204	CH6_SMSG_TS—Channel 6 Serial Message Time Stamp Register	X	X
0x0208	CH7_FMSG_DATA—Channel 7 Fast Message Register	X	—
0x0214	CH7_FMSG_CRC—Channel 7 Fast Message CRC Register	X	—
0x0218	CH7_FMSG_TS—Channel 7 Fast Message Time Stamp Register	X	—
0x021C	CH7_SMSG_BIT3—Channel 7 Serial Message Register (Bit 3)	X	—
0x0220	CH7_SMSG_BIT2—Channel 7 Serial Message Register (Bit 2)	X	—
0x0224	CH7_SMSG_TS—Channel 7 Serial Message Time Stamp Register	X	—

7.8.8 LINFlexD

7.8.8.1 LINFlexD – Configurations

SPC58xEx/SPC58xGx has 18 instances of LINFlexD. *Table 73* lists the configuration for each instance.

Table 73. LINFlexD configurations

Description	LINFlexD_0	LINFlexD_1	LINFlexD_2	LINFlexD_3	LINFlexD_4	LINFlexD_5	LINFlexD_6	LINFlexD_7	LINFlexD_8	LINFlexD_9	LINFlexD_10	LINFlexD_11	LINFlexD_12	LINFlexD_13	LINFlexD_14	LINFlexD_15	LINFlexD_16	LINFlexD_17
Number of filters implemented	16	0																
Number of Tx DMA channels	1 (TX_CH_NUM = 0)(1)																	
Number of Rx DMA channels	1 (RX_CH_NUM = 0)(2)																	
LIN operation mode	Master/Slave	Master (slave = 0)																
Value of auto-synchronization	1	0																

1. Number of TX DMA channels = 2**TX_CH_NUM where ** stands for exponentiation.
2. Number of RX DMA channels = 2**RX_CH_NUM where ** stands for exponentiation.

7.8.8.2 LINFlexD implemented registers

Table 74 shows the registers implemented for each LINFlexD instance on the SPC58xEx/SPC58xGx chip.

7.9 Reset and Boot modules

Table 74. LINFlexD register comparison map

Register description	Address offset (hex)																	
	LINFlexD_0	LINFlexD_1	LINFlexD_2	LINFlexD_3	LINFlexD_4	LINFlexD_5	LINFlexD_6	LINFlexD_7	LINFlexD_8	LINFlexD_9	LINFlexD_10	LINFlexD_11	LINFlexD_12	LINFlexD_13	LINFlexD_14	LINFlexD_15	LINFlexD_16	LINFlexD_17
LIN Control register 1 (LINCR1)	00	00																
LIN Interrupt Enable register (LINIER)	04	04																
LIN Status register (LINSR)	08	08																
LIN Error Status register (LINESR)	0C	0C																
UART Mode Control register (UARTCR)	10	10																
UART Mode Status register (UARTSR)	14	14																
LIN Timeout Control Status register (LINTCSR)	18	18																
LIN Output Compare register (LINOCR)	1C	1C																
LIN Timeout Control register (LINTOCR)	20	20																
LIN Fractional Baud Rate register (LINFBRR)	24	24																
LIN Integer Baud Rate register (LINIBRR)	28	28																
LIN Checksum Field register (LINCFR)	2C	2C																
LIN Control register 2 (LINCR2)	30	30																
Buffer Identifier register (BIDR)	34	34																

Table 74. LINFlexD register comparison map (continued)

Register description	Address offset (hex)																	
	LINFlexD_0	LINFlexD_1	LINFlexD_2	LINFlexD_3	LINFlexD_4	LINFlexD_5	LINFlexD_6	LINFlexD_7	LINFlexD_8	LINFlexD_9	LINFlexD_10	LINFlexD_11	LINFlexD_12	LINFlexD_13	LINFlexD_14	LINFlexD_15	LINFlexD_16	LINFlexD_17
Buffer Data Register Least Significant (BDRL)	38	38																
Buffer Data Register Most Significant (BDRM)	3C	3C																
Identifier Filter Enable register (IFER)	40	40																
Identifier Filter Match index (IFMI)	44	44																
Identifier Filter Mode register (IFMR)	48	48																
Identifier Filter Control registers (IFCR0)	4C	—																
Identifier Filter Control registers (IFCR1)	50	—																
Identifier Filter Control registers (IFCR2)	54	—																
Identifier Filter Control registers (IFCR3)	58	—																
Identifier Filter Control registers (IFCR4)	5C	—																
Identifier Filter Control registers (IFCR5)	60	—																
Identifier Filter Control registers (IFCR6)	64	—																

Table 74. LINFlexD register comparison map (continued)

Register description	Address offset (hex)																	
	LINFlexD_0	LINFlexD_1	LINFlexD_2	LINFlexD_3	LINFlexD_4	LINFlexD_5	LINFlexD_6	LINFlexD_7	LINFlexD_8	LINFlexD_9	LINFlexD_10	LINFlexD_11	LINFlexD_12	LINFlexD_13	LINFlexD_14	LINFlexD_15	LINFlexD_16	LINFlexD_17
Identifier Filter Control registers (IFCR7)	68	—																
Identifier Filter Control registers (IFCR8)	6C	—																
Identifier Filter Control registers (IFCR9)	70	—																
Identifier Filter Control registers (IFCR10)	74	—																
Identifier Filter Control registers (IFCR11)	78	—																
Identifier Filter Control registers (IFCR12)	7C	—																
Identifier Filter Control registers (IFCR13)	80	—																
Identifier Filter Control registers (IFCR14)	84	—																
Identifier Filter Control registers (IFCR15)	88	—																
Global Control register (GCR)	8C	8C																
UART Preset Timeout register (UARTPTO)	90	90																
UART Current Timeout register (UARTCTO)	94	94																

Table 74. LINFlexD register comparison map (continued)

Register description	Address offset (hex)																	
	LINFlexD_0	LINFlexD_1	LINFlexD_2	LINFlexD_3	LINFlexD_4	LINFlexD_5	LINFlexD_6	LINFlexD_7	LINFlexD_8	LINFlexD_9	LINFlexD_10	LINFlexD_11	LINFlexD_12	LINFlexD_13	LINFlexD_14	LINFlexD_15	LINFlexD_16	LINFlexD_17
DMA Tx Enable register (DMATXE)	98	98																
DMA Rx Enable register (DMARXE)	9C	9C																

7.9.1 BAF configuration

7.9.1.1 Search boot header and boot options

Refer to *Section 8.3.12.2.3: SW boot record search (BAF search)* in the *Chapter 8: Reset and Boot*.

7.9.1.2 LINFlexD and MCAN pins configuration

Table 75 shows the device pin configurations for LINFlexD_0/2 and CAN_SUB_0_M_CAN_1/2 modules.

Table 75. LINFlexD and MCAN pin configuration

Pads	Initial Serial Boot Mode		Serial Boot Mode after a valid CAN message received	Serial Boot Mode after a valid LINFlexD message received
	Function	Pad Configuration	Function	Function
PA[10]	—	(1) Output: — Push/Pull medium drive no pull-up/pull-down	MCAN_2 Tx	LINFlexD_2 Tx
PA[11]	LINFlexD_2 Rx and MCAN_2 Rx	Input: — pull-up hysteresis TTL voltage level	MCAN_2 Rx	LINFlexD_2 Rx

Table 75. LINFlexD and MCAN pin configuration (continued)

Pads	Initial Serial Boot Mode		Serial Boot Mode after a valid CAN message received	Serial Boot Mode after a valid LINFlexD message received
	Function	Pad Configuration	Function	Function
PB[10]	—	(1) Output:— Push/Pull medium drive no pull-up/pull-down	MCAN_1 Tx	LINFlexD_0 Tx
PB[9]	LINFlexD_0 Rx and MCAN_1 Rx	Input: — pull-up hysteresis TTL voltage level	MCAN_1 Rx	LINFlexD_0 Rx

1. Pad is set to output configuration as described in the table. Output buffer is enabled for the whole duration of the BAF serial boot.

7.9.2 System Status and Configuration Module (SSCM) configuration

This section summarizes the System Status and Configuration Module (SSCM) configuration in the SPC58xEx/SPC58xGx device. For a comprehensive description of the SSCM, please refer to the SSCM's dedicated chapter.

7.9.2.1 SSCM instantiation

There is one SSCM instance in the SPC58xEx/SPC58xGx device. The SSCM is part of the reset and boot sub-system.

7.9.2.2 Device-specific features

The SSCM allows Boot Assist from Flash (BAF), and utilizes PASS security concept.

The device specific reset values for the SSCM_MEMCONFIG register are shown in *Table 76*.

Table 76. SSCM_MEMCONFIG reset values

Field	Field definition	Value
JPIN	JTAG Part ID	10'h110(1)
MREV	Minor Revision	4'h0(2)

1. This value will change for each major revision of the device.
2. This value will change for each minor revision of the device.

Table 77 shows the address locations that the SSCM uses to search for startup information.

Table 77. SSCM startup information addresses

Description	Value
The location at which the SSCM will look for the UTEST DCF records (UTEST_OFFSET)	0x0040_0000
Location of the Boot Assist code in Flash (BAF)	0x0040_4000
Specifies the locations which are searched for a valid HSM boot header (starting from the lowest address)	0x60_C000 0x61_0000 0x62_0000

7.9.2.3 Memory map and register description

Table 78. Reference text to related information

Topic	Related module	Reference
Reset	Reset Generation Module (MC_RGM)	*Chapter 64: Reset Generation Module (MC_RGM)*
Reset/Boot	Reset and Boot	*Chapter 8: Reset and Boot*
Boot	Boot Assist from Flash (BAF)	*Chapter 65: Boot Assist Flash (BAF)*
Register Protection	Register Protection (REG_PROT)	*Chapter 91: Register Protection (REG_PROT)*
Serial Boot	LINFlexD	*Chapter 63: LINFlexD*
Security	PASS	*Chapter 92: Password and Device Security Module (PASS)*

7.10 Safety modules

7.10.1 CRC

Table 79 lists the number of contexts available for the current CRC computation of multiple data streams of CRC.

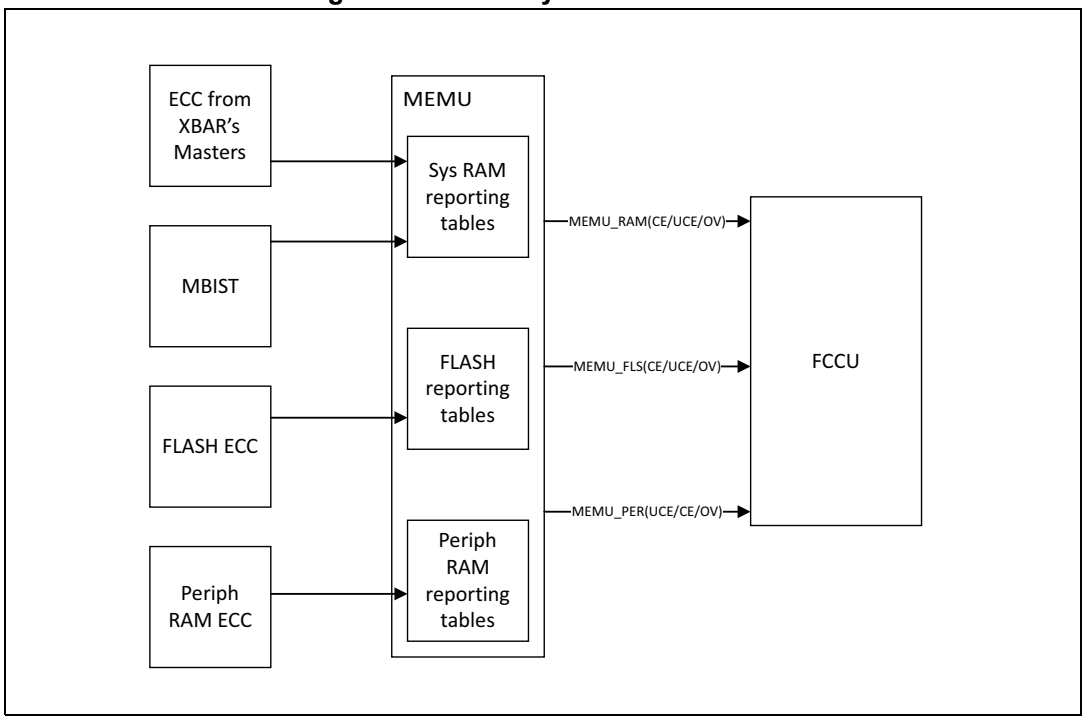
Table 79. CRC contexts

Signal name	Description	CRC_0	CRC_1
CRC_CNTX_NUM	Number of contexts available for the current CRC computation of multiple data streams.	4	4

7.10.2 MEMU configuration

7.10.2.1 MEMU system connections

Figure 31. MEMU system connections



7.10.2.2 MEMU error sources

Table 80 shows the MEMU error sources for each instance configured on this chip.

Table 80. MEMU error sources

Instance	Value
Number of system RAM unique error sources	128
Number of peripheral RAM unique error sources	36
Number of flash memory unique error sources	5

7.10.2.3 Reporting tables implementation

There are 3 categories of error reporting for the MEMU:

- Flash error reporting
- System RAM error reporting
- Peripheral RAM error reporting

Each category of error reporting has two tables associated with it, these are:

- Correctable Error Reporting Table
- Uncorrectable Error Reporting Table

Each entry in a reporting table corresponds to a unique error event. The number of entries supported by each table depends upon the error source.

The table below describes the number of entries in each reporting table for the different error sources.

Table 81. Number of entries in each reporting table

Error Source	Number of entries in correctable error reporting table	Number of entries in un-correctable error reporting table
Flash	20	1
System RAM	10	1
Peripheral RAM	2	1

7.10.2.4 Concurrent overflow register (OFLW) implementation

Errors are reported to the MEMU from one of the three categories, Flash, System RAMs or Peripheral RAMs. In an overflow condition, it is possible to identify the unique error source that caused the overflow. Corresponding error sources for the OFLW register details the corresponding error source to the respective bit-field in the Concurrent Overflow registers.

Note: z4a/b/c stand for Core_0/1/2 respectively in the following tables.

Table 82. Corresponding error sources for system RAM OFLW0 register

Source number	OFLW0 bit-field position (bit)(1)	Error sources	Reporting address range (hex)
0	31	z4a Data AHB + z4A DSMC	Refer to XBAR memory map
1	30	z4a Instruction AHB	Refer to XBAR memory map
2	29	z4a Data TCM (DMEM)	0x50800000–0x5080FFFF
3	28	z4a Instruction TCM (IMEM)	0x50000000–0x5000FFFF
4	27	z4a Data Cache	0x4F010000–0x4F01FFFF
5	26	z4a Instruction Cache	0x4F000000–0x4F00FFFF
6	25	z4b Data AHB + z4b DSMC	Refer to XBAR memory map
7	24	z4b Instruction AHB	Refer to XBAR memory map
8	23	z4b Data TCM (DMEM)	0x51800000–0x51807FFF
9	22	z4b Instruction TCM (IMEM)	0x51000000–0x5100FFFF
10	21	z4b Data Cache	0x4F050000–0x4F05FFFF
11	20	z4b Instruction Cache	0x4F040000–0x4F04FFFF
12	19	z4c Instruction Cache + z4c DSMC	0x4F080000–0x4F08FFFF
13	18	z4c Data AHB	Refer to XBAR memory map
14	17	z4c Instruction AHB	Refer to XBAR memory map
15	16	z4c Data TCM (DMEM)	0x52800000–0x52807FFF
16	15	z4c Instruction TCM (IMEM)	0x52000000–0x52003FFF
17	14	SRAM Memory 0	0x40060000–0x4007FFFF

Table 82. Corresponding error sources for system RAM OFLW0 register (continued)

Source number	OFLW0 bit-field position (bit)(1)	Error sources	Reporting address range (hex)
18	13	SRAM Memory 1	0x40080000–0x400A7FFF
19	12	SRAM Memory 2	0x400A8000–0x400E7FFF
20	11	SRAM Memory 3	0x400E8000–0x400F7FFF
21	10	Concentrator 1 (Ethernet_0 port)	Refer to XBAR memory map
22	9	Concentrator 2 (Ethernet_1 port)	Refer to XBAR memory map
23	8	DMA 0	Refer to XBAR memory map
24	7	SIPI 0	Refer to XBAR memory map
25	6	Concentrator 0 (DMA_1 or HSM port)	Refer to XBAR memory map
26	5	Concentrator 1 (SIPI_1 port)	Refer to XBAR memory map
27	4	Concentrator 1 (Flexray_0 port)	Refer to XBAR memory map
28	3	MBIST DWC 1 TX ODD	0x4F0A9000–0x4F0A97FF
29	2	MBIST z4a Data Cache Instance 4	0x4F010000–0x4F01FFFF
30	1	MBIST z4a Instruction Cache Tag	0x4F020000–0x4F02FFFF
31	0	MBIST z4a Data Cache Tag	0x4F030000–0x4F03FFFF

1. Bit 0 in the table bit-field corresponds to bit 31 in target register.

Table 83. Corresponding error sources for system RAM OFLW1 register

Source number	OFLW1 bit-field position (bit)(1)	Error sources	Reporting address range (hex)
32	31	MBIST z4b Instruction TCM	0x51000000–0x5100FFFF
33	30	MBIST z4b Data TCM Instance 1	0x51800000–0x51807FFF
34	29	MBIST z4b Data TCM Instance 2	0x51800000–0x51807FFF
35	28	MBIST z4b Instruction Cache Instance 1	0x4F040000–0x4F04FFFF
36	27	MBIST z4b Instruction Cache Instance 2	0x4F040000–0x4F04FFFF
37	26	MBIST z4b Instruction Cache Instance 3	0x4F040000–0x4F04FFFF
38	25	MBIST z4b Instruction Cache Instance 4	0x4F040000–0x4F04FFFF
39	24	MBIST z4b Data Cache Instance 1	0x4F050000–0x4F05FFFF
40	23	MBIST z4b Data Cache Instance 2	0x4F050000–0x4F05FFFF
41	22	MBIST z4b Data Cache Instance 3	0x4F050000–0x4F05FFFF
42	21	MBIST z4b Data Cache Instance 4	0x4F050000–0x4F05FFFF
43	20	MBIST z4b Instruction Cache Tag	0x4F060000–0x4F06FFFF
44	19	MBIST z4b Data Cache Tag	0x4F070000–0x4F07FFFF
45	18	MBIST z4c Instruction TCM Instance 1	0x52000000–0x52003FFF
46	17	MBIST z4c Data TCM Instance 1	0x52800000–0x52807FFF

Table 83. Corresponding error sources for system RAM OFLW1 register (continued)

Source number	OFLW1 bit-field position (bit)(1)	Error sources	Reporting address range (hex)
47	16	MBIST z4c Data TCM Instance 2	0x52808000–0x5280FFFF
48	15	MBIST z4c Instruction Cache Instance 1	0x4F080000–0x4F08FFFF
49	14	MBIST z4c Instruction Cache Instance 2	0x4F080000–0x4F08FFFF
50	13	MBIST z4c Instruction Cache Instance 3	0x4F080000–0x4F08FFFF
51	12	MBIST z4c Instruction Cache Instance 4	0x4F080000–0x4F08FFFF
52	11	MBIST z4c Instruction Cache Tag	0x4F090000–0x4F09FFFF
53	10	MBIST RAM 0	0xA0008000–0xA0009FFF
54	9	MBIST RAM 1 Instance 1	0xA0000000–0xA0003FFF
55	8	MBIST RAM 1 Instance 2	0xA0004000–0xA0007FFF
56	7	MBIST Instruction Cache Instance 1	0x4F0B0000–0x4F0B07FF
57	6	MBIST Instruction Cache Instance 2	0x4F0B0800–0x4F0B0FFF
58	5	MBIST Instruction Cache Tag Instance 1	0x4F0C0200–0x4F0C02FF
59	4	MBIST Instruction Cache Tag Instance 2	0x4F0C0300–0x4F0C03FF
60	3	MBIST HSM Crypto Channel Controller (C3)	0xA3F80000–0xA3F83FFF
61	2	MBIST DMA 0	0xFC0A1000–0xFC0A3FFF
62	1	MBIST Flexray 0 Data RAM	0x4F0E0000–0x4F0E00FF
63	0	MBIST Flexray 0 LUT RAM	0xF7E50800–0xF7E50FFF

1. Bit 0 in the table bit-field corresponds to bit 31 in target register.

Table 84. Corresponding error sources for system RAM OFLW2 register

Source number	OFLW2 bit-field position (bit)(1)	Error sources	Reporting address range (hex)
64	31	MBIST Controller Area Network 0	0xF7ED4000–0xF7ED7FFF
65	30	MBIST GTM FIFO	0xF7D19000–0xF7D19FFF
66	29	MBIST GTM RAM0 Instance 1 (S0 core1)	0xF7D40000–0xF7D41FFF
67	28	MBIST GTM RAM0 Instance 2 (S0 core2)	0xF7D48000–0xF7D49FFF
68	27	MBIST GTM RAM0 Instance 3 (S0 core3)	0xF7D50000–0xF7D51FFF
69	26	MBIST GTM RAM0 Instance 4 (S0 core4)	0xF7D58000–0xF7D59FFF
70	25	MBIST GTM RAM1 Instance 1 (S1 core1)	0xF7D42000–0xF7D43FFF
71	24	MBIST GTM RAM1 Instance 2 (S1 core2)	0xF7D4A000–0xF7D4BFFF
72	23	MBIST GTM RAM1 Instance 3 (S1 core3)	0xF7D52000–0xF7D53FFF
73	22	MBIST GTM RAM1 Instance 4 (S1 core4)	0xF7D5A000–0xF7D5BFFF
74	21	MBIST Digital PLL Instance 1a	0xF7D28200–0xF7D283FF
75	20	MBIST Digital PLL Instance 1b	0xF7D28400–0xF7D293FF

Table 84. Corresponding error sources for system RAM OFLW2 register (continued)

Source number	OFLW2 bit-field position (bit)(1)	Error sources	Reporting address range (hex)
76	19	MBIST Digital PLL Instance 2	0xF7D2C000–0xF7D2DFFF
77	18	MBIST SRAM Instance 0_1	0x40068000–0x4006FFFF
78	17	MBIST SRAM Instance 0_2	0x40070000–0x40077FFF
79	16	MBIST Nexus Instance 1	0x4F0D0000–0x4F0D07E0
80	15	MBIST Nexus Instance 2	0x4F0D0010–0x4F0D07F0
81	14	MBIST Nexus Instance 3	0x4F0D0800–0x4F0D0FEF
82	13	MBIST Nexus Instance 4	0x4F0D0810–0x4F0D0FFF
83	12	MBIST AMU	0xF40C0000–0xF40C7FFF
84	11	MBIST DWC 0 RX EVEN	0x4F0A4000–0x4F0A4FFF
85	10	MBIST DWC 0 RX ODD	0x4F0A6000–0x4F0A6FFF
86	9	MBIST Overlay Instance 0_0	0x0D000000–0x0D001FFF
87	8	MBIST Overlay Instance 0_1	0x0D002000–0x0D003FFF
88	7	MBIST z4a Instruction TCM	0x50000000–0x50000000
89	6	MBIST z4a Data TCM Instance 1	0x50800000–0x50807FFF
90	5	MBIST z4a Data TCM Instance 2	0x50808000–0x5080FFFF
91	4	MBIST z4a Instruction Cache Instance 1	0x4F000000–0x4F00FFFF
92	3	MBIST z4a Instruction Cache Instance 2	0x4F000000–0x4F00FFFF
93	2	MBIST MP Flash Krypton RAM	0x4F1F0000–0x4F1F3FFF
94	1	MBIST z4a Instruction Cache Instance 3	0x4F000000–0x4F00FFFF
95	0	MBIST DMA 1	0xF40A5000–0xF40A7FFF

1. Bit 0 in the table bit-field corresponds to bit 31 in target register.

Table 85. Corresponding error sources for system RAM OFLW3 register

Source number	OFLW3 bit-field position (bit)(1)	Error sources	Reporting address range (hex)
96	31	MBIST DWC 1 RX EVEN	0x4F0AC000–0x4F0ACFFF
97	30	MBIST DWC 1 RX ODD	0x4F0AE000–0x4F0AEFFF
98	29	MBIST Controller Area Network 1	0xFBED4000–0xFBED7FFF
99	28	MBIST GTM RAM0 Instance 0 (S0 core0)	0xF7D38000–0xF7D39FFF
100	27	MBIST GTM RAM1 Instance 0 (S1 core0)	0xF7D3A000–0xF7D3BFFF
101	26	MBIST SRAM Instance 0_0	0x40060000–0x40067FFF
102	25	MBIST SRAM Instance 0_3	0x40078000–0x4007FFFF
103	24	MBIST DWC 1 TX EVEN	0x4F0A8000–0x4F0A87FF
104	23	MBIST SRAM Instance 1_0	0x40080000–0x40087FFF

Table 85. Corresponding error sources for system RAM OFLW3 register (continued)

Source number	OFLW3 bit-field position (bit)(1)	Error sources	Reporting address range (hex)
105	22	MBIST SRAM Instance 1_1	0x40088000–0x4008FFFF
106	21	MBIST SRAM Instance 1_2	0x40090000–0x40097FFF
107	20	MBIST SRAM Instance 1_3	0x40098000–0x4009FFFF
108	19	MBIST SRAM Instance 1_4	0x400A0000–0x400A7FFF
109	18	MBIST SRAM Instance 2_0_0	0x400A8000–0x400A9FFF
110	17	MBIST SRAM Instance 2_0_1	0x400AA000–0x400AFFFF
111	16	MBIST SRAM Instance 2_1	0x400B0000–0x400B7FFF
112	15	MBIST SRAM Instance 2_2	0x400B8000–0x400BFFFF
113	14	MBIST SRAM Instance 2_3	0x400C0000–0x400C7FFF
114	13	MBIST SRAM Instance 2_4	0x400C8000–0x400CFFFF
115	12	MBIST SRAM Instance 2_5	0x400D0000–0x400D7FFF
116	11	MBIST SRAM Instance 2_6	0x400D8000–0x400DFFFF
117	10	MBIST SRAM Instance 2_7	0x400E0000–0x400E7FFF
118	9	MBIST SRAM Instance 3_0	0x400E8000–0x400EFFFF
119	8	MBIST SRAM Instance 3_1	0x400F0000–0x400F7FFF
120	7	MBIST DWC 0TX EVEN	0x4F0A0000–0x4F0A07FF
121	6	MBIST DWC 0TX ODD	0x4F0A1000–0x4F0A17FF
122	5	MBIST Overlay Instance 1_0	0x0D004000–0x0D005FFF
123	4	MBIST Overlay Instance 1_1	0x0D006000–0x0D007FFF
124	3	MBIST z4a Instruction Cache Instance 4	0x4F000000–0x4F00FFFF
125	2	MBIST z4a Data Cache Instance 1	0x4F010000–0x4F01FFFF
126	1	MBIST z4a Data Cache Instance 2	0x4F010000–0x4F01FFFF
127	0	MBIST z4a Data Cache Instance 3	0x4F010000–0x4F01FFFF

1. Bit 0 in the table bit-field corresponds to bit 31 in target register.

Table 86. Corresponding error sources for peripheral RAM OFLW0 register

Source number	OFLW0 bit-field position (bit)(1)	Error sources	Reporting address range (hex)
0	31	DMA 0 TCD	0xFC0A1000–0xFC0A3FFF
1	30	Advanced Memory Unit RAM	0xF40C0000–0xF40C7FFF
2	29	Ethernet1 RX ODD	0x4F0AE000–0x4F0AEFFF
3	28	CAN RAM 1	0xFBED4000–0xFBED7FFF
4	27	PBRIDGE_0	0xFC000000–0xFC003FFF
5	26	PBRIDGE_1	0xF8000000–0xF8003FFF

Table 86. Corresponding error sources for peripheral RAM OFLW0 register (continued)

Source number	OFLW0 bit-field position (bit)(1)	Error sources	Reporting address range (hex)
6	25	Ethernet0 TX EVEN	0x4F0A0000–0x4F0A07FF
7	24	Ethernet0 RX ODD	0x4F0A6000–0x4F0A6FFF
8	23	HSM Platform RAM	0xA0000000–0xA0009FFF
9	22	HSM Instruction Cache RAM	0x4F0B0000–0x4F0BFFFF
10	21	Flexray Data RAM 0	0x4F0E0000–0x4F0E00FF
11	20	Flexray LUT RAM 0	0xF7E50800–0xF7E50FFF
12	19	CAN RAM 0	0xF7ED4000–0xF7ED7FFF
13	18	GTM FIFO	0xF7D19000–0xF7D19FFF
14	17	GTM Digital PLL r1a	0xF7D28200–0xF7D283FF
15	16	GTM Digital PLL r1b	0xF7D28400–0xF7D293FF
16	15	GTM Digital PLL r2	0xF7D2C000–0xF7D2DFFF
17	14	GTM MCS0 RAM0	0xF7D38000–0xF7D39FFF
18	13	GTM MCS0 RAM1	0xF7D3A000–0xF7D3BFFF
19	12	GTM MCS1 RAM0	0xF7D40000–0xF7D41FFF
20	11	GTM MCS1 RAM1	0xF7D42000–0xF7D43FFF
21	10	GTM MCS2 RAM0	0xF7D48000–0xF7D49FFF
22	9	GTM MCS2 RAM1	0xF7D4A000–0xF7D4BFFF
23	8	GTM MCS3 RAM0	0xF7D50000–0xF7D51FFF
24	7	GTM MCS3 RAM1	0xF7D52000–0xF7D53FFF
25	6	DMA 1 TCD	0xF40A5000–0xF40A7FFF
26	5	Reserved	
27	4	Reserved	
28	3	PBRIDGE_2	0xF4000000–0xF4003FFF
29	2	Ethernet1 TX EVEN	0x4F0A8000–0x4F0A87FF
30	1	GTM MCS4 RAM0	0xF7D58000–0xF7D59FFF
31	0	GTM MCS4 RAM1	0xF7D5A000–0xF7D5BFFF

1. Bit 0 in the table bit-field corresponds to bit 31 in target register.

Table 87. Corresponding error sources for peripheral memory OFLW1 register

Source number	OFLW1 bit-field position (bit)(1)	Error sources	Reporting address range (hex)
32	31	Ethernet0 TX ODD	0x4F0A1000–0x4F0A17FF
33	30	Ethernet0 RX EVEN	0x4F0A4000–0x4F0A4FFF
34	29	Ethernet1 TX ODD	0x4F0A9000–0x4F0A97FF

Table 87. Corresponding error sources for peripheral memory OFLW1 register (continued)

Source number	OFLW1 bit-field position (bit)(1)	Error sources	Reporting address range (hex)
35	28	Ethernet1 RX EVEN	0x4F0AC000–0x4F0ACFFF
36	27	Reserved	
37	26	Reserved	
38	25	Reserved	
39	24	Reserved	
40	23	Reserved	
41	22	Reserved	
42	21	Reserved	
43	20	Reserved	
44	19	Reserved	
45	18	Reserved	
46	17	Reserved	
47	16	Reserved	
48	15	Reserved	
49	14	Reserved	
50	13	Reserved	
51	12	Reserved	
52	11	Reserved	
53	10	Reserved	
54	9	Reserved	
55	8	Reserved	
56	7	Reserved	
57	6	Reserved	
58	5	Reserved	
59	4	Reserved	
60	3	Reserved	
61	2	Reserved	
62	1	Reserved	
63	0	Reserved	

1. Bit 0 in the table bit-field corresponds to bit 31 in target register.

Table 88. Corresponding error sources for flash memory OFLW0 register

Source number	OFLW0 bit-field position (bit)(1)	Error sources	Reporting address range (hex)
0	31	Flash 0 port 0	Refer to FLASH memory map
1	30	Flash 0 port 1	
2	29	Flash 1 port 0	
3	28	Flash 1 port 1	
4	27	PFlash AMU (IPMX)	
5	26	Reserved	
6	25	Reserved	
7	24	Reserved	
8	23	Reserved	
9	22	Reserved	
10	21	Reserved	
11	20	Reserved	
12	19	Reserved	
13	18	Reserved	
14	17	Reserved	
15	16	Reserved	
16	15	Reserved	
17	14	Reserved	
18	13	Reserved	
19	12	Reserved	
20	11	Reserved	
21	10	Reserved	
22	9	Reserved	
23	8	Reserved	
24	7	Reserved	
25	6	Reserved	
26	5	Reserved	
27	4	Reserved	
28	3	Reserved	
29	2	Reserved	
30	1	Reserved	
31	0	Reserved	

1. Bit 0 in the table bit-field corresponds to bit 31 in target register.

7.10.2.5 MEMU registers

Table 89 shows the address offsets and implementation of the MEMU registers in this chip.

Table 89. MEMU registers

Offset	Register
0x0000	Control register (CTRL)
0x0004	Error flag register (ERR_FLAG)
0x000C	Debug register (DEBUG)
0x0020	System RAM correctable error reporting table status register (SYS_RAM_CERR_STS0)
0x0024	System RAM correctable error reporting table address register (SYS_RAM_CERR_ADDR0)
0x0028	System RAM correctable error reporting table status register (SYS_RAM_CERR_STS1)
0x002C	System RAM correctable error reporting table address register (SYS_RAM_CERR_ADDR1)
0x0030	System RAM correctable error reporting table status register (SYS_RAM_CERR_STS2)
0x0034	System RAM correctable error reporting table address register (SYS_RAM_CERR_ADDR2)
0x0038	System RAM correctable error reporting table status register (SYS_RAM_CERR_STS3)
0x003C	System RAM correctable error reporting table address register (SYS_RAM_CERR_ADDR3)
0x0040	System RAM correctable error reporting table status register (SYS_RAM_CERR_STS4)
0x0044	System RAM correctable error reporting table address register (SYS_RAM_CERR_ADDR4)
0x0048	System RAM correctable error reporting table status register (SYS_RAM_CERR_STS5)
0x004C	System RAM correctable error reporting table address register (SYS_RAM_CERR_ADDR5)
0x0050	System RAM correctable error reporting table status register (SYS_RAM_CERR_STS6)
0x0054	System RAM correctable error reporting table address register (SYS_RAM_CERR_ADDR6)
0x0058	System RAM correctable error reporting table status register (SYS_RAM_CERR_STS7)
0x005C	System RAM correctable error reporting table address register (SYS_RAM_CERR_ADDR7)
0x0060	System RAM correctable error reporting table status register (SYS_RAM_CERR_STS8)
0x0064	System RAM correctable error reporting table address register (SYS_RAM_CERR_ADDR8)
0x0068	System RAM correctable error reporting table status register (SYS_RAM_CERR_STS9)
0x006C	System RAM correctable error reporting table address register (SYS_RAM_CERR_ADDR9)
0x0070	System RAM uncorrectable error reporting table status register (SYS_RAM_UNCERR_STS)

Table 89. MEMU registers (continued)

Offset	Register
0x0074	System RAM uncorrectable error reporting table address register (SYS_RAM_UNCERR_ADDR)
0x0078	System RAM concurrent overflow register (SYS_RAM_OFLW0)
0x007C	System RAM concurrent overflow register (SYS_RAM_OFLW1)
0x0080	System RAM concurrent overflow register (SYS_RAM_OFLW2)
0x0084	System RAM concurrent overflow register (SYS_RAM_OFLW3)
0x0620	Peripheral RAM correctable error reporting table status register (PERIPH_RAM_CERR_STS0)
0x0624	Peripheral RAM correctable error reporting table address register (PERIPH_RAM_CERR_ADDR0)
0x0628	Peripheral RAM correctable error reporting table status register (PERIPH_RAM_CERR_STS1)
0x062C	Peripheral RAM correctable error reporting table address register (PERIPH_RAM_CERR_ADDR1)
0x0630	Peripheral RAM uncorrectable error reporting table status register (PERIPH_RAM_UNCERR_STS)
0x0634	Peripheral RAM uncorrectable error reporting table address register (PERIPH_RAM_UNCERR_ADDR)
0x0638	Peripheral RAM concurrent overflow register (PERIPH_RAM_OFLW0)
0x0C20	Flash memory correctable error reporting table status register (FLASH_CERR_STS0)
0x0C24	Flash memory correctable error reporting table address register (FLASH_CERR_ADDR0)
0x0C28	Flash memory correctable error reporting table status register (FLASH_CERR_STS1)
0x0C2C	Flash memory correctable error reporting table address register (FLASH_CERR_ADDR1)
0x0C30	Flash memory correctable error reporting table status register (FLASH_CERR_STS2)
0x0C34	Flash memory correctable error reporting table address register (FLASH_CERR_ADDR2)
0x0C38	Flash memory correctable error reporting table status register (FLASH_CERR_STS3)
0x0C3C	Flash memory correctable error reporting table address register (FLASH_CERR_ADDR3)
0x0C40	Flash memory correctable error reporting table status register (FLASH_CERR_STS4)
0x0C44	Flash memory correctable error reporting table address register (FLASH_CERR_ADDR4)
0x0C48	Flash memory correctable error reporting table status register (FLASH_CERR_STS5)
0x0C4C	Flash memory correctable error reporting table address register (FLASH_CERR_ADDR5)
0x0C50	Flash memory correctable error reporting table status register (FLASH_CERR_STS6)

Table 89. MEMU registers (continued)

Offset	Register
0x0C54	Flash memory correctable error reporting table address register (FLASH_CERR_ADDR6)
0x0C58	Flash memory correctable error reporting table status register (FLASH_CERR_STS7)
0x0C5C	Flash memory correctable error reporting table address register (FLASH_CERR_ADDR7)
0x0C60	Flash memory correctable error reporting table status register (FLASH_CERR_STS8)
0x0C64	Flash memory correctable error reporting table address register (FLASH_CERR_ADDR8)
0x0C68	Flash memory correctable error reporting table status register (FLASH_CERR_STS9)
0x0C6C	Flash memory correctable error reporting table address register (FLASH_CERR_ADDR9)
0x0C70	Flash memory correctable error reporting table status register (FLASH_CERR_STS10)
0x0C74	Flash memory correctable error reporting table address register (FLASH_CERR_ADDR10)
0x0C78	Flash memory correctable error reporting table status register (FLASH_CERR_STS11)
0x0C7C	Flash memory correctable error reporting table address register (FLASH_CERR_ADDR11)
0x0C80	Flash memory correctable error reporting table status register (FLASH_CERR_STS12)
0x0C84	Flash memory correctable error reporting table address register (FLASH_CERR_ADDR12)
0x0C88	Flash memory correctable error reporting table status register (FLASH_CERR_STS13)
0x0C8C	Flash memory correctable error reporting table address register (FLASH_CERR_ADDR13)
0x0C90	Flash memory correctable error reporting table status register (FLASH_CERR_STS14)
0x0C94	Flash memory correctable error reporting table address register (FLASH_CERR_ADDR14)
0x0C98	Flash memory correctable error reporting table status register (FLASH_CERR_STS15)
0x0C9C	Flash memory correctable error reporting table address register (FLASH_CERR_ADDR15)
0x0CA0	Flash memory correctable error reporting table status register (FLASH_CERR_STS16)
0x0CA4	Flash memory correctable error reporting table address register (FLASH_CERR_ADDR16)
0x0CA8	Flash memory correctable error reporting table status register (FLASH_CERR_STS17)
0x0CAC	Flash memory correctable error reporting table address register (FLASH_CERR_ADDR17)
0x0CB0	Flash memory correctable error reporting table status register (FLASH_CERR_STS18)
0x0CB4	Flash memory correctable error reporting table address register (FLASH_CERR_ADDR18)
0x0CB8	Flash memory correctable error reporting table status register (FLASH_CERR_STS19)

Table 89. MEMU registers (continued)

Offset	Register
0x0CBC	Flash memory correctable error reporting table address register (FLASH_CERR_ADDR19)
0x0CC0	Flash memory uncorrectable error reporting table status register (FLASH_UNCERR_STS)
0x0CC4	Flash memory uncorrectable error reporting table address register (FLASH_UNCERR_ADDR)
0x0CC8	Flash memory concurrent overflow register (FLASH_OFLW0)

7.10.2.6 FlexRay lookup table (LUT) RAM alignment

Table 90. FlexRay LUT RAM alignment

IMA row select (decimal)	Peripheral RAM start address	Peripheral RAM end address	Peripheral register names	MEMU error reporting (PERIPH_RAM_*_ADDR)
0	F7E5_0800	F7E5_080C	FR_MBCCFR0 FR_MBFIDR0 FR_MBIDXR0 FR_MBCCFR1 FR_MBFIDR1 FR_MBIDXR1	F7E5_0802
1	F7E5_0810	F7E5_081C	FR_MBCCFR2 FR_MBFIDR2 FR_MBIDXR2 FR_MBCCFR3 FR_MBFIDR3 FR_MBIDXR3	F7E5_0812
2	F7E5_0820	F7E5_082C	FR_MBCCFR4 FR_MBFIDR4 FR_MBIDXR4 FR_MBCCFR5 FR_MBFIDR5 FR_MBIDXR5	F7E5_0822
3	F7E5_0830	F7E5_083C	FR_MBCCFR6 FR_MBFIDR6 FR_MBIDXR6 FR_MBCCFR7 FR_MBFIDR7 FR_MBIDXR7	F7E5_0832
4	F7E5_0840	F7E5_084C	FR_MBCCFR8 FR_MBFIDR8 FR_MBIDXR8 FR_MBCCFR9 FR_MBFIDR9 FR_MBIDXR9	F7E5_0842

Table 90. FlexRay LUT RAM alignment (continued)

IMA row select (decimal)	Peripheral RAM start address	Peripheral RAM end address	Peripheral register names	MEMU error reporting (PERIPH_RAM_*_ADDR)
5	F7E5_0850	F7E5_085C	FR_MBCCFR10 FR_MBFIDR10 FR_MBIDXR10 FR_MBCCFR11 FR_MBFIDR11 FR_MBIDXR11	F7E5_0852
6	F7E5_0860	F7E5_086C	FR_MBCCFR12 FR_MBFIDR12 FR_MBIDXR12 FR_MBCCFR13 FR_MBFIDR13 FR_MBIDXR13	F7E5_0862
7	F7E5_0870	F7E5_087C	FR_MBCCFR14 FR_MBFIDR14 FR_MBIDXR14 FR_MBCCFR15 FR_MBFIDR15 FR_MBIDXR15	F7E5_0872
8	F7E5_0880	F7E5_088C	FR_MBCCFR16 FR_MBFIDR16 FR_MBIDXR16 FR_MBCCFR17 FR_MBFIDR17 FR_MBIDXR17	F7E5_0882
9	F7E5_0890	F7E5_089C	FR_MBCCFR18 FR_MBFIDR18 FR_MBIDXR18 FR_MBCCFR19 FR_MBFIDR19 FR_MBIDXR19	F7E5_0892
...	...	...	...	...
62	F7E5_0BE0	F7E5_0BEC	FR_MBCCFR125 FR_MBFIDR125 FR_MBIDXR125 FR_MBCCFR124 FR_MBFIDR124 FR_MBIDXR124	F7E5_0BE2
63	F7E5_0BF0	F7E5_0BFC	FR_MBCCFR127 FR_MBFIDR127 FR_MBIDXR127 FR_MBCCFR126 FR_MBFIDR126 FR_MBIDXR126	F7E5_0BF2

7.10.2.7 FlexRay lookup table (LUT) RAM misalignment

Table 91. FlexRay LUT RAM misalignment

IMA row select (decimal)	Peripheral RAM start address	Peripheral RAM end address	Peripheral register names	MEMU error reporting (PERIPH_RAM_*_ADDR)
64	F7E5_1000	F7E5_100A	MBDOR0..5	F7E5_1000
65	F7E5_100C	F7E5_1016	MBDOR6..11	F7E5_1010
66	F7E5_1018	F7E5_1022	MBDOR12..17	F7E5_1020
67	F7E5_1024	F7E5_102E	MBDOR18..23	F7E5_1030
68	F7E5_1030	F7E5_103A	MBDOR24..29	F7E5_1040
69	F7E5_103C	F7E5_1046	MBDOR30..35	F7E5_1050
70	F7E5_1048	F7E5_1052	MBDOR36..41	F7E5_1060
71	F7E5_1054	F7E5_105E	MBDOR42..47	F7E5_1070
72	F7E5_1060	F7E5_106A	MBDOR48..53	F7E5_1080
73	F7E5_106C	F7E5_1076	MBDOR54..59	F7E5_1090
74	F7E5_1078	F7E5_1082	MBDOR60..65	F7E5_10A0
75	F7E5_1084	F7E5_108E	MBDOR66..71	F7E5_10B0
76	F7E5_1090	F7E5_109A	MBDOR72..77	F7E5_10C0
77	F7E5_109C	F7E5_10A6	MBDOR78..83	F7E5_10D0
78	F7E5_10A8	F7E5_10B2	MBDOR84..89	F7E5_10E0
79	F7E5_10B4	F7E5_10BE	MBDOR90..95	F7E5_10F0
80	F7E5_10C0	F7E5_10CA	MBDOR96..101	F7E5_1100
81	F7E5_10CC	F7E5_10D6	MBDOR102..107	F7E5_1110
82	F7E5_10D8	F7E5_10E2	MBDOR108..113	F7E5_1120
83	F7E5_10E4	F7E5_10EE	MBDOR114..119	F7E5_1130
84	F7E5_10F0	F7E5_10FA	MBDOR120..125	F7E5_1140
85	F7E5_10FC	F7E5_1106	MBD0R126..131	F7E5_1150
86	F7E5_1108	F7E5_1112	LEETR0..5	F7E5_1160

7.10.3 Indirect Memory Access (IMA) configuration

The IMA refers to the alteration of memory data values during system development and test activities. This section provides details about what memory locations can be accessed—either via processor core access or via the IMA module—and how they are configured. For details on IMA see *Chapter 88: Indirect Memory Access (IMA)*.

Note: See the SPC58xEx/SPC58xGx Microcontroller Security Reference Manual for the HSM IMA details.

7.10.3.1 SRAM Arrays accessible by processor cores

All the SRAM arrays in *Table 92* can be accessed by all main Cores (Core_0, Core_1, Core_2).

Table 92. SRAM with processor Core access

Block	Function	SPC58xEx/SPC58xGx		
		Words	Bits	ECC Bits
Core_0	IMEM	2048	72	—
	DMEM	8192	40	—
		8192	40	—
	Icache	256	72	—
		256	72	—
		256	72	—
		256	72	—
	Dcache	128	80	—
		128	80	—
		128	80	—
		128	80	—
	ITAG	128	65	—
	DTAG	64	71	—
Core_1	IMEM	2048	72	—
	DMEM	8192	40	—
		8192	40	—
	Icache	256	72	—
		256	72	—
		256	72	—
		256	72	—
	Dcache	128	80	—
		128	80	—
		128	80	—
		128	80	—
	ITAG	128	65	—
	DTAG	64	71	—

Table 92. SRAM with processor Core access (continued)

Block	Function	SPC58xEx/SPC58xGx		
		Words	Bits	ECC Bits
Core_2	IMEM	2048	72	—
	DMEM	4096	40	—
		4096	40	—
	Icache	256	72	—
		256	72	—
		256	72	—
		256	72	—
	ITAG	128	65	—
System RAM 0	System RAM	4096	72	—
		4096	72	—
		4096	72	—
		4096	72	—
System RAM 1	System RAM	4096	72	—
		4096	72	—
		4096	72	—
		4096	72	—
		4096	72	—
System RAM 2	System RAM	1024	72	—
		3072	72	—
		4096	72	—
		4096	72	—
		4096	72	—
		4096	72	—
		4096	72	—
		4096	72	—
		4096	72	—
System RAM 3	System RAM	4096	72	—
		4096	72	—
Overlay 0	Overlay	1024	72	—
		1024	72	—
Overlay 1	Overlay	1024	72	—
		1024	72	—

7.10.3.2 Memory accessible via the IMA module

The SRAM arrays in *Table 93* are accessible via the IMA Module.

Table 93. SRAM arrays accessible via IMA

IMA ARRAY SELECT	Block	Function	SPC58xEx/SPC58xGx		
			Words	Bits	ECC Bits
0	None	IMA Disabled	—	—	—
1	DMA	DMA_0	128	72	71–64
2		DMA_1	256	72	71–64
3	FlexRay	DRAM_0	128	26	25–16
4		Reserved	—	—	—
5		LRAM_0	96	126	125–96
6		Reserved	—	—	—
7		Reserved	—	—	—
8		Reserved	—	—	—
9	CAN Subsystem 0	Shared message RAM	3968	39	38–32
10	GTM	FIFO	1024	36	35–29
11		Reserved	—	—	—
12		MCS0 RAM0	2048	39	38–32
13		MCS1 RAM0	2048	39	38–32
14		MCS2 RAM0	2048	39	38–32
15		MCS3 RAM0	2048	39	38–32
16		MCS4 RAM0	2048	39	38–32
17		Reserved	—	—	—
18		MCS0 RAM1	1024	39	38–32
19		MCS1 RAM1	1024	39	38–32
20		MCS2 RAM1	1024	39	38–32
21		MCS3 RAM1	1024	39	38–32
22		MCS4 RAM1	1024	39	38–32
23		Reserved	—	—	—
24		DPLL RAM1a	96	31	30–24
25		DPLL RAM1b	384	31	30–24
26		DPLL RAM2	2048	31	30–24

Table 93. SRAM arrays accessible via IMA (continued)

IMA ARRAY SELECT	Block	Function	SPC58xEx/SPC58xGx		
			Words	Bits	ECC Bits
27	AMU	AMU	2048	144	143–136, 71–64
28	Ethernet_0	TX FIFO	256	76	75–68
29					
30		RX FIFO	512	76	75–68
31					
32	CAN Subsystem 1	Shared message RAM	3968	39	38–32
33	Ethernet_1	TX FIFO	256	76	75–68
34					
35		RX FIFO	512	76	75–68
36					

Memory with inaccessible ECC bits

The following table shows memory locations that cannot be accessed.

Table 94. No ECC bit access

IMA ARRAY SELECT	Block	Function	SPC58xEx/SPC58xGx		
			Words	Bits	ECC Bits
No IMA Access Required - Debug function					
—	NAR	NAR	64	128	—
—			64	128	—
—			64	128	—
—			64	128	—

7.10.4 FCCU configuration

7.10.4.1 Available FCCU output signals

Table 95 shows the available FCCU output signals implemented on this chip.

Table 95. FCCU output signals

FCCU output signal name	Chip-top signal name
EOUT[0], FI[0], ERROR0	F0
EOUT[1], FI[1], ERROR1	F1

7.10.4.2 FCCU register reset values

Table 96 shows the reset values of selected FCCU registers.

Table 96. Reset values of selected FCCU registers

Register	Reset value
FCCU_RF_CFG0	0xFFFF_FFFF
FCCU_RF_CFG1	0xFFFF_FFFF
FCCU_RF_CFG2	0xFFFF_FFFF
FCCU_RF_CFG3	0x0000_00FF
FCCU_RFS_CFG0	0x0000_2000
FCCU_RFS_CFG1	0x0000_0020
FCCU_RFS_CFG2	0x0000_0000
FCCU_RFS_CFG3	0x0000_0000
FCCU_RFS_CFG4	0x0000_0000
FCCU_RFS_CFG5	0x0000_0000
FCCU_RFS_CFG6	0x0000_0000
FCCU_RFS_CFG7	0x0000_0000
FCCU_RF_S0	0x0000_0000
FCCU_RF_S1	0x0000_0000
FCCU_RF_S2	0x0000_0000
FCCU_RF_S3	0x0000_0000
FCCU_RF_E0	0x0004_0040
FCCU_RF_E1	0x0000_0000
FCCU_RF_E2	0x0000_0000
FCCU_RF_E3	0x0000_0000
FCCU_RF_TOE0	0x0000_0000
FCCU_RF_TOE1	0x0000_0000
FCCU_RF_TOE2	0x0000_0000
FCCU_RF_TOE3	0x0000_0000
FCCU_EOUT_SIG_EN0	0x0000_0000
FCCU_EOUT_SIG_EN1	0x0000_0000
FCCU_EOUT_SIG_EN2	0x0000_0000
FCCU_EOUT_SIG_EN3	0x0000_0000
FCCU_CTRL	0x0000_00C0
FCCU_CTRLK	0x0000_0000
FCCU_CFG	0x0000_0000
FCCU_RFK	0x0000_0000
FCCU_RF_T0	0x0000_FFFF
FCCU_CFG_T0	0x0000_0006
FCCU_EINOUT	0x0000_0030

Table 96. Reset values of selected FCCU registers (continued)

Register	Reset value
FCCU_STAT	0x0000_0010
FCCU_NAFS	0x0000_0000
FCCU_AFFS	0x0000_0000
FCCU_NFFS	0x0000_0000
FCCU_FAFS	0x0000_0000
FCCU_SCFS	0x0000_0000
FCCU_RFF	0x0000_0000
FCCU_IRQ_STAT	0x0000_0000
FCCU_IRQ_EN	0x0000_0000
FCCU_XTMR	0x0000_0000
FCCU_MCS	0x0080_8384
FCCU_TRANS_LOCK	0x0000_0000
FCCU_PERMNT_LOCK	0x0000_0000
FCCU_DELTA_T	0x0000_0000
FCCU_IRQ_ALARM_EN0	0x0000_0000
FCCU_IRQ_ALARM_EN1	0x0000_0000
FCCU_IRQ_ALARM_EN2	0x0000_0000
FCCU_IRQ_ALARM_EN3	0x0000_0000
FCCU_NMI_EN0	0x0000_0000
FCCU_NMI_EN1	0x0000_0000
FCCU_NMI_EN2	0x0000_0000
FCCU_NMI_EN3	0x0000_0000

7.10.4.3 FOSU parameters

The FOSU_COUNT, or FOSU timeout response time, is 1000 safe clock cycles.

7.10.5 Register protection (REG_PROT) configuration

Some modules on this device include a built-in register protection mechanism that can be used to override the normal write permissions associated with individual registers in the module. This mechanism is software-controlled and can be dynamically reconfigured at runtime or made effective until device reset. For a detailed explanation, see the *Chapter 91: Register Protection (REG_PROT)*.

Table 97. Reference links to related information

Topic	Related module	Reference
System memory map	—	*Chapter 5: Memory Map*
System Integration Unit Lite2 registers	SIUL2	*Chapter 17: System Integration Unit Lite2 (SIUL2)*
Clock monitor Unit registers	CMU_1 CMU_0	*Chapter 29: Clock Monitor Unit (CMU)*
PMC Digital Interface registers	PMC_DIG	*Chapter 67: Power management controller digital interface (PMC_Dig)*
PLL Digital Interface registers	PLL_DIG	*Chapter 28: Dual PLL digital interface (PLLDIG)*
Oscillator Digital Interface registers	OSC40M_DIG	*Chapter 31: OSC digital interface (XOSC)*
IRCOSC Digital Interface registers	RC16M_DIG	*Chapter 32: IRCOSC digital interface*
Memory Error Management Unit registers	MEMU	*Chapter 87: Memory Error Management Unit (MEMU)*
Mode Entry Module registers	MC_ME	*Chapter 69: Mode Entry Module (MC_ME)*
Clock Generation Module registers	MC_CGM	*Chapter 30: Clock Generation Module (MC_CGM)*
Reset Generation Module registers	MC_RGM	*Chapter 64: Reset Generation Module (MC_RGM)*
Power Control Unit registers	MC_PCU	*Chapter 68: Power Control Unit (MC_PCU)*
System Status and Configuration Module registers	SSCM	*Chapter 66: System Status and Configuration Module (SSCM)*
Indirect Memory Access registers	IMA	*Chapter 88: Indirect Memory Access (IMA)*
JTAG Master registers	JTAGM	*Chapter 77: Buddy Device—JTAG Master (JTAGM)*

7.10.5.1 SIUL2 protected registers

Table 98 lists the SIUL2 registers that can be protected.

Table 98. Protected SIUL2 registers

Register	Register size (bits)	Offset from module base address	Protected size (bits)
SIUL_DIRER0	32	0x0018	32
SIUL_DIRSR0	32	0x0020	32
SIUL_IREER0	32	0x0028	32
IFEER0	32	0x0030	32
IFER0	32	0x0038	32
SCR0	32	0x0101	8
MSCR0–15	32	0x0240	32 (×16)
MSCR24–27	32	—(1)	32 (×4)
MSCR32–58	32	—(1)	32 (×27)
MSCR60–76	32	—(1)	32 (×17)

Table 98. Protected SIUL2 registers (continued)

Register	Register size (bits)	Offset from module base address	Protected size (bits)
MSCR80–96	32	—(1)	32 (×17)
MSCR109–127	32	—(1)	32 (×19)
MSCR136–148	32	—(1)	32 (×13)
MSCR149–159	32	—(1)	32 (×11)
MSCR172–183	32	—(1)	32 (×12)
MSCR185–202	32	—(1)	32 (×18)
MSCR506–519	32	—(1)	32 (×14)
MSCR520–527	32	—(1)	32 (×8)
MSCR528–535	32	—(1)	32 (×8)
MSCR536–543	32	—(1)	32 (×8)
MSCR544–575	32	—(1)	32 (×32)
MSCR576–577	32	—(1)	32 (×2)
MSCR578–579	32	—(1)	32 (×2)
MSCR580–581	32	—(1)	32 (×2)
MSCR582–583	32	—(1)	32 (×2)
MSCR584–585	32	—(1)	32 (×2)
MSCR586–587	32	—(1)	32 (×2)
MSCR594–607	32	—(1)	32 (×14)
MSCR608	32	—(1)	32
MSCR609	32	—(1)	32
MSCR610	32	—(1)	32
MSCR611	32	—(1)	32
MSCR616	32	—(1)	32
MSCR617	32	—(1)	32
MSCR619	32	—(1)	32
MSCR624	32	—(1)	32
MSCR625	32	—(1)	32
MSCR626	32	—(1)	32
MSCR627	32	—(1)	32
MSCR632	32	—(1)	32
MSCR633	32	—(1)	32
MSCR635	32	—(1)	32
MSCR638	32	—(1)	32
MSCR639	32	—(1)	32

Table 98. Protected SIUL2 registers (continued)

Register	Register size (bits)	Offset from module base address	Protected size (bits)
MSCR640	32	—(1)	32
MSCR641	32	—(1)	32
MSCR642	32	—(1)	32
MSCR643	32	—(1)	32
MSCR644	32	—(1)	32
MSCR645	32	—(1)	32
MSCR656–687	32	—(1)	32 (×32)
MSCR688–719	32	—(1)	32 (×32)
MSCR720–751	32	—(1)	32 (×32)
MSCR752–754	32	—(1)	32 (×3)
MSCR758	32	—(1)	32
MSCR759	32	—(1)	32
MSCR760	32	—(1)	32
MSCR764–767	32	—(1)	32 (×4)
MSCR772–774	32	—(1)	32 (×3)
MSCR786–802	32	—(1)	32 (×17)
MSCR808–812	32	—(1)	32 (×5)
MSCR816–819	32	—(1)	32 (×4)
MSCR821–824	32	—(1)	32 (×4)
MSCR843–846	32	—(1)	32 (×4)
MSCR848–865	32	—(1)	32 (×18)
MSCR880–909	32	—(1)	32 (×30)
MSCR922–934	32	—(1)	32 (x13)
MSCR936–967	32	—(1)	32 (×32)
MSCR968–978	32	—(1)	32 (×11)
MSCR980–981	32	—(1)	32 (×2)
MSCR986–1017	32	—(1)	32 (×32)
GPDO0–15	8	—(1)	8 (×16)
GPDO24–27	8	—(1)	8 (×4)
GPDO32–58	8	—(1)	8 (×27)
GPDO60–76	8	—(1)	8 (×17)
GPDO80–96	8	—(1)	8 (×17)
GPDO109–127	8	—(1)	8 (×19)
GPDO136–159	8	—(1)	8 (×24)

Table 98. Protected SIUL2 registers (continued)

Register	Register size (bits)	Offset from module base address	Protected size (bits)
GPDO172–183	8	—(1)	8(×12)
GPDO185–202	8	—(1)	8 (×18)
PGPDO0–12	16	—(1)	16 (×13)
MPGPDO0–12	32	—(1)	32 (×13)

1. See the module memory map in the SIUL2 chapter for address offset.

7.10.5.2 Core Clock Monitor Unit (CMU_1) protected registers

Table 99 lists the registers that can be protected for the fast crossbar CMU (CMU_1)

Table 99. Protected CMU_1 (CORE_XBAR_BD) registers

Register	Register size (bits)	Offset from module base address	Protected size (bits)
CSR	32	0x0003	8
HFREFR	32	0x000A	16
LFREFR	32	0x000E	16
ISR	32	0x0013	8

7.10.5.3 I/O Processor Clock Monitor Unit (CMU_0) protected registers

Table 100 lists the registers that can be protected for PLL CMU (CMU_0)

Table 100. Protected CMU (PLL0_XOSC_IRCOSC) registers

Register	Register size (bits)	Offset from module base address	Protected size (bits)
CSR	32	0x0003	8
HFREFR	32	0x000A	16
LFREFR	32	0x000E	16
ISR	32	0x0013	8

7.10.5.4 PMC Digital Interface (PMC_DIG) protected registers

Table 101 lists the PMC_DIG registers that can be protected.

Table 101. Protected PMC_DIG registers

Register	Register size (bits)	Offset from module base address	Protected size (bits)
REE_LV0	32	0x0004	8
RES_LV0	32	0x0008	8
FEE_LV0	32	0x0010	8

Table 101. Protected PMC_DIG registers (continued)

Register	Register size (bits)	Offset from module base address	Protected size (bits)
REE_LV1	32	0x0025	8
RES_LV1	32	0x0029	8
FEE_LV1	32	0x0031	8
REE_HV0	32	0x0044	8
RES_HV0	32	0x0048	8
FEE_HV0	32	0x0050	8
REE_HV1	32	0x0064	32
RES_HV1	32	0x0068	32
FEE_HV1	32	0x0070	32
VSIO	32	0x0107	8
PNREG_S	32	0x0243	8
MISC_CTRL_REG	32	0x024B	8
REE_TD	32	0x0307	8
RES_TD	32	0x030B	8
CTL_TD	32	0x030C	32
FEE_TD	32	0x031B	8
BIST_CTRL	32	0x03D4	32

7.10.5.5 PLL Digital Interface (PLL_DIG) protected registers

Table 102 lists the PLL_DIG registers that can be protected.

Table 102. Protected PLL_DIG registers

Register	Register size (bits)	Offset from module base address	Protected size (bits)
PLL0_CR	32	0x0000	32
PLL0_SR	32	0x0004	32
PLL0_DIVR	32	0x0008	32
PLL1_CR	32	0x0020	32
PLL1_SR	32	0x0024	32
PLL1_DIVR	32	0x0028	32
PLL1_FMR	32	0x002C	32
PLL1_FDR	32	0x0030	32

7.10.5.6 Oscillator Digital Interface (OSC40M_DIG) protected registers

Table 103 lists the XOSC registers that can be protected.

Table 103. Protected OSC40M_DIG registers

Register	Register size (bits)	Offset from module base address	Protected size (bits)
CTL	32	0x0000	32

7.10.5.7 IRCOSC Digital interface (RC16M_DIG) protected registers

Table 104 lists the IRCOSC registers that can be protected.

Table 104. Protected RC16M_DIG registers

Register	Register size (bits)	Offset from module base address	Protected size (bits)
CTL	8	0x0001	8

7.10.5.8 Memory Error Management Unit (MEMU) protected registers

Table 105 lists the MEMU registers that can be protected.

Table 105. Protected MEMU registers

Register	Register size (bits)	Offset from module base address	Protected size (bits)
CTRL	32	0x0002	8
DEBUG	32	0x000C	32

7.10.5.9 Mode Entry (MC_ME) protected registers

Table 106 lists the MC_ME registers that can be protected.

Table 106. Protected MC_ME registers

Register	Register size (bits)	Offset from module base address	Protected size (bits)
MCTL	32	0x0004	32
ME	32	0x000A	16
IS	32	0x000F	8
IM	32	0x0013	8
IMTS	32	0x0017	8
SAFE_MC	32	0x0028	32
DRUN_MC	32	0x002C	32
RUN0–3_MC	32	0x0030–0x003C	32 (×4)
HALT0_MC	32	0x0040	32
STOP0_MC	32	0x0048	32
RUN_PC0–7	32	0x0083–0x009F	8 (×8)

Table 106. Protected MC_ME registers (continued)

Register	Register size (bits)	Offset from module base address	Protected size (bits)
STANDBY0_MC	32	0x0054	32
LP_PC0–7	32	0x00A2–0x00BE	8 (×8)
PCTL9	8	0x00C9	8
PCTL11	8	0x00CB	8
PCTL15	8	0x00CF	8
PCTL30	8	0x00DE	8
PCTL33	8	0x00E1	8
PCTL36	8	0x00E4	8
PCTL38	8	0x00E6	8
PCTL58–60	8	0x00FA–0x00FC	8 (×3)
PCTL62	8	0x00FE	8
PCTL67–69	8	0x0103–0x0105	8 (×3)
PCTL72	8	0x0108	8
PCTL74	8	0x010A	8
PCTL84–92	8	0x0114–0x011C	8 (×10)
PCTL95–99	8	0x011F–0x0123	8 (×5)
PCTL101	8	0x0125	8
PCTL104	8	0x0128	8
PCTL107	8	0x12B	8
PCTL111–112	8	0x012F–0x0130	8 (×2)
PCTL114	8	0x0132	8
PCTL119	8	0x0137	8
PCTL126–130	8	0x013E–0x0142	8 (×11)
PCTL134–135	8	0x0146–0x0147	8 (×2)
PCTL140–143	8	0x014C–0x014F	8 (×4)
PCTL158	8	0x015E	8
PCTL164	8	0x0164	8
PCTL166	8	0x0166	8
PCTL186–188	8	0x017A–0x017C	8 (×3)
PCTL195–198	8	0x0183–0x0186	8 (×4)
PCTL202	8	0x018A	8
PCTL212–220	8	0x0194–0x019C	8 (×9)
PCTL223–227	8	0x019F–0x01A3	8 (×5)
PCTL232	8	0x01A8	8

Table 106. Protected MC_ME registers (continued)

Register	Register size (bits)	Offset from module base address	Protected size (bits)
PCTL234–235	8	0x01AA–0x01AB	8 (×2)
PCTL239–243	8	0x01AF–0x01B3	8 (×5)
PCTL247	8	0x01B7	8
PCTL254–255	8	0x01BE–0x01BF	8 (×2)
CCTL0–1	16	0x01C4–0x01C6	16 (×2)
CCTL3–4	16	0x01CA–0x01CC	16 (×2)
CADDR0–1	32	0x01E0–0x01E4	32 (×2)
CADDR3–4	32	0x01EC–0x01F0	32 (×2)

7.10.5.10 Indirect Memory Access (IMA) protected registers

Table 107 lists the IMA register that can be protected.

Table 107. Protected IMA registers

Register	Register size (bits)	Offset from module base address	Protected size (bits)
ENABLE	32	0x0004	32

7.10.5.11 Clock Generation Module (MC_CGM) protected registers

Table 108 lists the MC_CGM registers that can be protected.

Table 108. Protected MC_CGM registers

Register	Register size (bits)	Offset from module base address	Protected size (bits)
SC_DIV_RC	32	0x01D3	8
DIV_UPD_TYPE	32	0x01D4	32
DIV_UPD_TRIG	32	0x01D8	32
DIV_UPD_STAT	32	0x01DC	32
SC_DC0–4	32	0x01E8–0x01F8	16 (×5)
AC0–9_SC	32	0x0200–0x0320	8 (×10)
AC11–12_SC	32	0x0360–0x0380	8 (×2)
AC0_DC0–3	32	0x0208–0x0214	16 (×4)
AC1_DC0	32	0x0228	16
AC2_DC0–1	32	0x0248–0x024C	16 (×2)
AC5_DC0	32	0x02A8	32
AC5_DC1–2	32	0x02AC–0x02B0	16 (×2)
AC6_DC0	32	0x02C8	16

Table 108. Protected MC_CGM registers (continued)

Register	Register size (bits)	Offset from module base address	Protected size (bits)
AC7_DC0	32	0x02E8	16
AC8_DC0	32	0x0308	16
AC9_DC0	32	0x0328	16
AC11_DC0	32	0x0368	16
AC12_DC0	32	0x0388	32
AC12_DC1–4	32	0x038C–0x0398	16 (×5)

7.10.5.12 Reset Generation Module (MC_RGM) protected registers

Table 109 lists the MC_RGM registers that can be protected.

Table 109. Protected MC_RGM registers

Register	Register size (bits)	Offset from module base address	Protected size (bits)
DES	32	0x0000	32
DERD	32	0x0010	32
DEAR	32	0x0020	32
DBRE	32	0x0030	32
FES	32	0x0300	32
FERD	32	0x0310	32
FEAR	32	0x0320	32
FBRE	32	0x0330	32
FESS	32	0x0340	32
FRET	32	0x0604	8
DRET	32	0x0608	8
EROEC	32	0x060C	8
PRST0–7	32	0x0610–0x062C	32 (×8)

7.10.5.13 Power Control Unit (MC_PCU) protected registers

Table 110 lists the MC_RGM registers that can be protected.

Table 110. Protected MC_PCU registers

Register	Register size (bits)	Offset from module base address	Protected size (bits)
PCONF0	32	0x0002	16
PCONF1	32	0x0006	16

Table 110. Protected MC_PCU registers (continued)

Register	Register size (bits)	Offset from module base address	Protected size (bits)
PCONF2	32	0x000A	16
PCONF3	32	0x000E	16

7.10.5.14 System Status and Control Module (SSCM) protected registers

Table 111 lists the SSCM registers that can be protected.

Table 111. Protected SSCM registers

Register	Register size (bits)	Offset from module base address	Protected size (bits)
STATUS	16	0x0000	8
ERROR	16	0x0006	16

7.10.5.15 JTAG Master (JTAGM) protected registers

Table 112 lists the JTAGM registers that can be protected.

Table 112. Protected JTAGM registers

Register	Register size (bits)	Offset from module base address	Protected size (bits)
MCR	32	0x00	32

7.10.6 STCU2 configuration

The number of memory cuts (NMCUT) available in this device is 102.

Table 113. MBIST partition

M_BIST_i	Location	Partition	Addressable	Start_Address	End_Adress
0	ROM	BAR	Yes	not addressable	not addressable
1		MPFLASH	No	not addressable	not addressable
2	FLASH	MPFLASH_1	No	0x4F1F_0000	0x4F1F_3FFF
3	Core_2	I-MEM	Yes	0x5200_0000	0x5200_3FFF
4		D-MEM (1)	Yes	0x5280_0000	0x5280_FFFF
5		D-MEM (0)	No		
6		I_Cache (3)	No	0x4F08_0000	0x4F08_1FFF
7		I_Cache (2)	No		
8		I_Cache (1)	No		
9		I_Cache (0)	No		
10		ITAG	No	0x4F09_0000	0x4F09_03FF

Table 113. MBIST partition (continued)

M_BIST_i	Location	Partition	Addressable	Start_Address	End_Adress
11	DMA	RAM DMA_1	Yes	0xF40A_5000	0xF40A_57FF
12	Platform	System RAM 2	Yes	0x400A_8000	0x400C_7FFF
13			Yes		
14			Yes		
15			Yes		
16			Yes		
17	CAN	CAN_1	Yes	0xFBE_D4000	0xFBED_7FFF
18		CAN_0	Yes	0xF7ED_4000	0xF7ED_7FFF
19	Core_0	I-MEM	Yes	0x5000_0000	0x5000_3FFF
20		D-MEM (1)	Yes	0x5080_0000	0x5080_FFFF
21		D-MEM (0)	Yes		
22		I_Cache (3)	No	0x4F00_0000	0x4F00_1FFF
23		I_Cache (2)	No		
24		I_Cache (1)	No		
25		I_Cache (0)	No		
26		ITAG	No	0x4F02_0000	0x4F02_03FF
27		D_Cache (3)	No	0x4F01_0000	0x4F01_FFFF
28		D_Cache (2)	No		
29		D_Cache (1)	No		
30		D_Cache (0)	No		
31		DTAG	No	0x4F03_0000	0x4F03_01FF
32	HSM	pram_0	Yes	0xA000_8000	0xA000_9FFF
33		pram_1_1	Yes	0xA000_0000	0xA000_7FFF
34		pram_1_0	Yes		
35		I_Cache (0,1)	No	0x4F0B_0000	0x4F0B_0FFF
36			No		
37		ITAG (0,1)	No	0x4F0C_0200	0x4F0C_02FF
38			No	0x4F0C_0300	0x4F0C_03FF
39		C3	Yes	0xA3F8_0000	0xA3F8_01FF
40	FlexRay	DRAM	No	0x4F0E_0000	0x4F0E_00FF
41		LRAM	Yes	0xF7E5_0800	0xF7E5_13FF

Table 113. MBIST partition (continued)

M_BIST_i	Location	Partition	Addressable	Start_Address	End_Adress
42	ETHERNET_0	FEC_tx_0_odd	No	0x4F0A_1000	0x4F0A_17FF
43		FEC_tx_0_even	No	0x4F0A_0000	0x4F0A_07FF
44		FEC_rx_0_odd	No	0x4F0A_6000	0x4F0A_6FFF
45		FEC_tx_0_even	No	0x4F0A_4000	0x4F0A_4FFF
46	Platform	System RAM 2	Yes	0x400C_8000	0x400E_7FFF
47			Yes		
48			Yes		
49			Yes		
50	Platform	System RAM 1	Yes	0x4008_0000	0x400A_7FFF
51			Yes		
52			Yes		
53			Yes		
54			Yes		
55	Platform	System RAM 3	Yes	0x400E_8000	0x400F_7FFF
56			Yes		
57	Platform	System RAM 0	Yes	0x4000_6000	0x4007_FFFF
58			Yes		
59			Yes		
60			Yes		
61	Core_1	I-MEM	Yes	0x5100_0000	0x5100_FFFF
62		D-MEM(1)	Yes	0x5180_0000	0x5180_FFFF
63		D-MEM(0)	Yes		
64		I_Cache (3)	No	0x4F04_0000	0x4F04_1FFF
65		I_Cache (2)	No		
66		I_Cache (1)	No		
67		I_Cache (0)	No		
68		D_Cache (3)	No	0x4F05_0000	0x4F05_FFFF
69		D_Cache (2)	No		
70		D_Cache (1)	No		
71		D_Cache (0)	No		
72		ITAG	No	0x4F06_0000	0x4F06_03FF
73		DTAG	No	0x4F07_0000	0x4F07_01FF
74	DMA	RAM DMA_0	Yes	0xFC0A_1000	0xFC0A_13FF

Table 113. MBIST partition (continued)

M_BIST_i	Location	Partition	Addressable	Start_Address	End_Adress
75	ETHERNET_1	FEC_tx_1_odd	No	0x4F0A_9000	0x4F0A_97FF
76		FEC_tx_1_even	No	0x4F0A_8000	0x4F0A_87FF
77		FEC_rx_1_odd	No	0x4F0A_E000	0x4F0A_EFFF
78		FEC_tx_1_even	No	0x4F0A_C000	0x4F0A_CFFF
79	Calibration	ram_overlay _0_1	Yes	0x0D00_2000	0x0D00_3FFF
80		ram_overlay _0_0	Yes	0x0D00_0000	0x0D00_1FFF
81	AMU	Ram Amu	Yes	0xF40C_0000	0xF40C_7FFF
82	Nexus	ram_nexus_nar_3	No	0x4F0D_0810	0x4F0D_0FFF
83		ram_nexus_nar_2	No	0x4F0D_0800	0x4F0D_0FEF
84		ram_nexus_nar_1	No	0x4F0D_0010	0x4F0D_07F0
85		ram_nexus_nar_0	No	0x4F0D_0000	0x4F0D_07E0
86	GTM	ram_gtm_fifo	Yes	0xF7D1_9000	0xF7D1_9FFF
87		ram_gtm_ram0_0	Yes	0xF7D3_8000	0xF7D3_9FFF
88		ram_gtm_ram0_1	Yes	0xF7D4_0000	0xF7D4_1FFF
89		ram_gtm_ram0_2	Yes	0xF7D4_8000	0xF7D4_9FFF
90		ram_gtm_ram0_3	Yes	0xF7D5_0000	0xF7D5_1FFF
91		ram_gtm_ram0_4	Yes	0xF7D5_8000	0xF7D5_9FFF
92		ram_gtm_ram1_0	Yes	0xF7D3_A000	0xF7D3_AFFF
93		ram_gtm_ram1_1	Yes	0xF7D4_2000	0xF7D4_2FFF
94		ram_gtm_ram1_2	Yes	0xF7D4_A000	0xF7D4_AFFF
95		ram_gtm_ram1_3	Yes	0xF7D5_2000	0xF7D2_83FF
96		ram_gtm_ram1_4	Yes	0xF7D5_A000	0xF7D2_8BFF
97		ram_gtm_dpll_ram1a	Yes	0xF7D2_8200	0xF7D2_83FF
98		ram_gtm_dpll_ram1b	Yes	0xF7D2_8400	0xF7D2_8BFF
99		ram_gtm_dpll_ram2	Yes	0xF7D2_C000	0xF7D2_DFFF
100	Calibration	ram_overlay _1_1	Yes	0x0D00_6000	0x0D00_7FFF
101		ram_overlay _1_0	Yes	0x0D00_4000	0x0D00_5FFF

Note: HSM and FLASH online MBIST are not accessible when life cycle is customer delivered/OEM/IN FIELD (SSCM_LCSTAT[LC]=CUST_DELIV/PROD_OEM/IN_FIELD).

Refer to the Chapter 90: Self-Test Control Unit (STCU2) for a description of the registers programming and to the DCF client table for the description of the available DCF.
The STCU setup has to be defined based on the application-specific safety, boot time and consumption requirements. By default, STCU2 is not bypassed.
This means that if offline self tests are not programed, the hard-coded WDG of the STCU2 will be anyway activated. As a consequence, the startup time of the device will be increased according to the WDG timeout value.

Table 114 lists the MBIST registers in STCU2 with their available fields.

Table 114. MBIST registers in STCU2 with available fields

MBIST register	Fields available
STCU_MBS1	MBS31 – MBS0
STCU_MBS2	MBS63 – MBS32
STCU_MBS3	MBS95 – MBS64
STCU_MBS4	MBS101 – MBS96
STCU_MBE1	MBE31 – MBE0
STCU_MBE2	MBE63– MBE32
STCU_MBE3	MBE95– MBE64
STCU_MBE4	MBE101– MBE96
STCU_MBS1SW	MBSSW31 – MBSSW0
STCU_MBS2SW	MBSSW63 – MBSSW32
STCU_MBS3SW	MBSSW95 – MBSSW64
STCU_MBS4SW	MBSSW101 – MBSSW96
STCU_MBE1SW	MBESW31 – MBESW0
STCU_MBE2SW	MBESW63 – MBESW32
STCU_MBE3SW	MBESW95 – MBESW64
STCU_MBE4SW	MBESW101 – MBESW96
STCU_MBUFM1	MBUFM31 – MBUFM0
STCU_MBUFM2	MBUFM63 – MBUFM32
STCU_MBUFM3	MBUFM95 – MBUFM64
STCU_MBUFM4	MBUFM101 – MBUFM96

Table 115. LBIST partition

lbist_partition_0(1)	lbist_partition_1	lbist_partition_2	lbist_partition_3	lbist_partition_4	lbist_partition_5	lbist_partition_6
acp_comp_bridge_S6_lmen	gtm	hsm	cansubsys_1	psi5_1	platform	cansubsys_0
acp_comp_bridge_S7_lmen	—	flash_0	flexray_0	ethernet_0	—	dma_che_mux_1_checker
acp_comp_bridge_dma_m4	—	sent_0	memu	adcsar_dig_2	—	dma_che_mux_0_checker
acp_comp_bridge_sipi_m5	—	sent_1	emios_1	linflex_0	—	dma_che_mux_2_checker
acp_comp_iahb_gskt_aips0	—	emios_0	psi5_s_0	adcsar_dig_3	—	dma_che_mux_3_checker
acp_periph_bridge_concentrator_0	—	psi5_0	fccu	adcsar_seq_1	—	dma_che_mux_4_checker
acp_periph_bridge_concentrator_1	—	dspi_4	ethernet_1	adcsar_seq_2	—	dma_che_mux_5_checker
acp_periph_bridge_concentrator_2	—	dspi_5	adcsd_ana_1	adcsar_seq_b	—	dma_che_mux_0
acp_periph_bridge_m5	—	—	adcsd_ana_3	adcsar_seq_0	—	dma_che_mux_1
acp_periph_bridge_S6	—	—	adcsd_ana_2	adcsar_seq_3	—	dma_che_mux_2
acp_periph_bridge_S7_tcm_gskt	—	—	adcsd_ana_5	iic_0	—	dma_che_mux_3
acp_periph_iahb_gskt_aips1	—	—	adcsd_ana_4	dspi_0	—	dma_che_mux_4
acp_periph_iahb_gskt_aips2	—	—	adcsd_ana_0	dspi_9	—	dma_che_mux_5
ahb_edc_after_ecc_mstr_gskt_sipi	—	—	crc_0	dspi_1	—	—
ahb_smpu2_mon	—	—	crc_1	dspi_2	—	—
ahb_smpu2_mon_0	—	—	body_ctu_0	dspi_3	—	—
aic0	—	—	fosu	dspi_8	—	—
aic1	—	—	cmu_2_hpbm	dspi_7	—	—
aic2	—	—	cmu_13_pfbridge	dspi_6	—	—
comp_checker_rccu_intc_or_glue_1	—	—	cmu_7_sent	adcsar_dig_1	—	—
comp_checker_rccu_intc_or_glue_2	—	—	cmu_11_fbridge	linflex_13	—	—
comp_chka_mirq_checker_delay	—	—	bam	linflex_10	—	—

Table 115. LBIST partition (continued)

lbist_partition_0[1]	lbist_partition_1	lbist_partition_2	lbist_partition_3	lbist_partition_4	lbist_partition_5	lbist_partition_6
comp_chkc_mirq_checker_delay	—	—	adcsd_dig_2	linflex_4	—	—
dmc_zxb_dbus	—	—	adcsd_dig_4	linflex_6	—	—
ips_ e2e_edc_slv_gskt_aips0	—	—	adcsd_dig_5	linflex_1	—	—
ips_ e2e_edc_slv_gskt_aips1	—	—	adcsd_dig_1	linflex_14	—	—
ips_ e2e_edc_slv_gskt_aips1	—	—	adcsd_dig_0	linflex_15	—	—
ib0_interface_dummy	—	—	adcsd_dig_3	linflex_16	—	—
periph_shell_xbic_ahb	—	—	—	linflex_17	—	—
pfalsh_cal_overlay	—	—	—	linflex_7	—	—
pfalsh_cal_overlay_0	—	—	—	linflex_5	—	—
pflash_fl_ecc	—	—	—	linflex_3	—	—
pflash_fl_ecc_0	—	—	—	linflex_9	—	—
pram_e2e_ecc_edc	—	—	—	linflex_12	—	—
pram_e2e_ecc_edc_0	—	—	—	linflex_11	—	—
pram_e2e_ecc_edc_1	—	—	—	linflex_2	—	—
pram_e2e_ecc_edc_2	—	—	—	linflex_8	—	—
pram_e2e_ecc_safety_mon	—	—	—	adcsar_dig_0	—	—
pram_e2e_ecc_safety_mon_0	—	—	—	adcsar10_dig_0	—	—
pram_e2e_ecc_safety_mon_1	—	—	—	adcsar10_dig_1	—	—
pram_e2e_ecc_safety_mon_2	—	—	—	pit_rti_0	—	—
rccu_aips0_ecc	—	—	—	fr_rom_1	—	—
rccu_aips1_ecc	—	—	—	fr_rom_0	—	—
rccu_aips2_ecc	—	—	—	ips_read_mux_2	—	—
rccu_dma0	—	—	—	adcsar_10bit_seq_0	—	—
rccu_dma1	—	—	—	adcsar_10bit_seq_1	—	—
rccu_dma_ch_mux_0	—	—	—	ima	—	—
rccu_dma_ch_mux_1	—	—	—	ips_read_mux_1	—	—
rccu_dma_ch_mux_2	—	—	—	intg_ima_read mux_top	—	—
rccu_dma_ch_mux_3	—	—	—	pit_rti_1	—	—

Table 115. LBIST partition (continued)

lbist_partition_0(1)	lbist_partition_1	lbist_partition_2	lbist_partition_3	lbist_partition_4	lbist_partition_5	lbist_partition_6
rccu_dma_ch_mux_4	—	—	—	cmu_0_pll0_xoc_ircosc	—	—
rccu_dma_ch_mux_5	—	—	—	cmu_4_gtm	—	—
rccu_dmc_zxa	—	—	—	cmu_12_emios	—	—
rccu_dmc_zxc	—	—	—	cmu_10_psi5_1us	—	—
rccu_intc	—	—	—	cmu_5_sdadc	—	—
rccu_zxd	—	—	—	cmu_3_pbridge	—	—
rccu_zxd_dtcm	—	—	—	cmu_9_psi5_f125	—	—
rccu_zxd_intc_if	—	—	—	cmu_8_psi5f189	—	—
rccu_zxf	—	—	—	cmu_6_saradc	—	—
rccu_zxf_dtcm	—	—	—	cmu_1_core_xbar_bd	—	—
rccu_zxf_intc_if	—	—	—	adcsar_ana_wrap_2	—	—
u_zxb_dahb_masterid_mon	—	—	—	adcsar_ana_wrap_3	—	—
u_zxb_iahb_masterid_mon	—	—	—	adcsar_ana_wrap_0	—	—
xbic_ahb_comp	—	—	—	adcsar_ana_wrap_1	—	—
zxa_dc_data0_fb_chk	—	—	—	adcsar_ana_wrap_b	—	—
zxa_dc_data1_fb_chk	—	—	—	adcsar_ana_wrap_comp_0	—	—
zxa_dc_data2_fb_chk	—	—	—	adcsar_ana_wrap_comp_1	—	—
zxa_dc_data3_fb_chk	—	—	—	adcsar_bias	—	—
zxa_dc_tag_fb_chk	—	—	—	adcsd_bias	—	—
zxa_dtcm0_fb_chk	—	—	—	adcsar_dig_b	—	—
zxa_dtcm1_fb_chk	—	—	—	—	—	—
zxa_ic_data0_fb_chk	—	—	—	—	—	—
zxa_ic_data1_fb_chk	—	—	—	—	—	—
zxa_ic_data2_fb_chk	—	—	—	—	—	—
zxa_ic_data3_fb_chk	—	—	—	—	—	—
zxa_ic_tag_fb_chk	—	—	—	—	—	—

Table 115. LBIST partition (continued)

lbist_partition_0[1]	lbist_partition_1	lbist_partition_2	lbist_partition_3	lbist_partition_4	lbist_partition_5	lbist_partition_6
zxa_itcm_fb_chk	—	—	—	—	—	—
zxb_dc_data0_fb_chk	—	—	—	—	—	—
zxb_dc_data1_fb_chk	—	—	—	—	—	—
zxb_dc_data2_fb_chk	—	—	—	—	—	—
zxb_dc_data3_fb_chk	—	—	—	—	—	—
zxb_dc_tag_fb_chk	—	—	—	—	—	—
zxc_dtcm0_fb_chk	—	—	—	—	—	—
zxc_dtcm1_fb_chk	—	—	—	—	—	—
zxc_ic_data0_fb_chk	—	—	—	—	—	—
zxc_ic_data1_fb_chk	—	—	—	—	—	—
zxc_ic_data2_fb_chk	—	—	—	—	—	—
zxc_ic_data3_fb_chk	—	—	—	—	—	—
zxc_ic_tag_fb_chk	—	—	—	—	—	—
zxc_itcm_fb_chk	—	—	—	—	—	—

1. SPC58xGx devices have only lbist_partition_0.

7.11 Security modules

7.11.1 Password and Device Security Module (PASS) configuration

The PASS module is used to implement password-based read and write protection for flash blocks on the SPC58xEx/SPC58xGx MCU. Up to four levels of password protection can be implemented for each flash block.

Note: The registers discussed in this section are initialized via DCF records, so the reset values depend on user settings.

See *Chapter 92: Password and Device Security Module (PASS)* for further details.

7.11.1.1 PASS_LOCK*n*_PG*n* register bit mapping

Each password group defined in the DCF records has a set of four LOCKn registers association with it: PASS_LOCK*n*_PG*n*. The bits of these registers are associated with specific flash blocks. The mapping is shown in the following figures.

Note: The mapping of various LOCK register bits to flash blocks in the SPC58xEx/SPC58xGx microcontroller is very similar to the LOCKx registers implemented in the FLASH, but not identical. The PASS module has additional mapping in the LOCK3 register and defines read locking regions. See the PASS chapter for details.

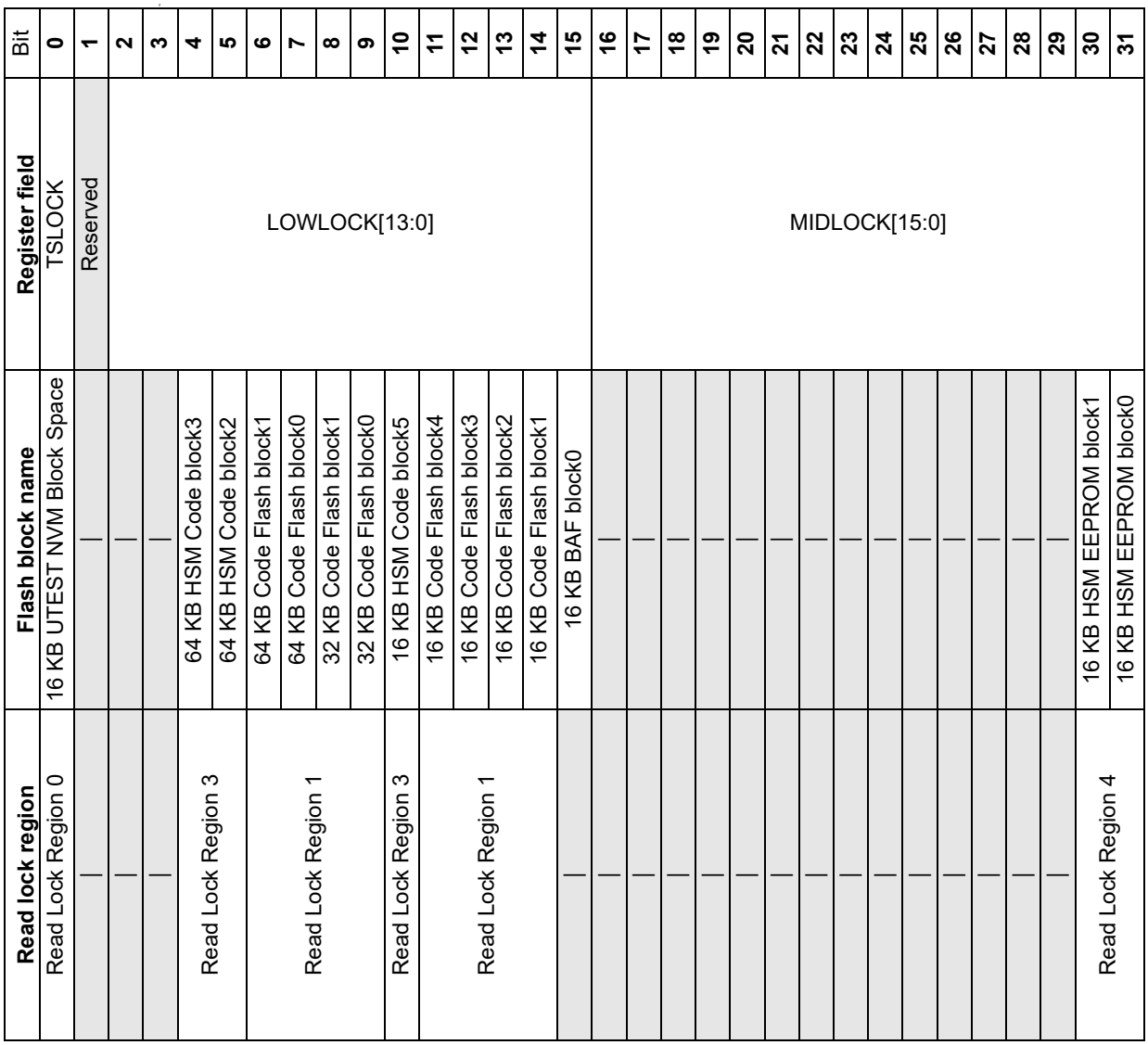


Figure 32. PASS_LOCK0_PGn Registers

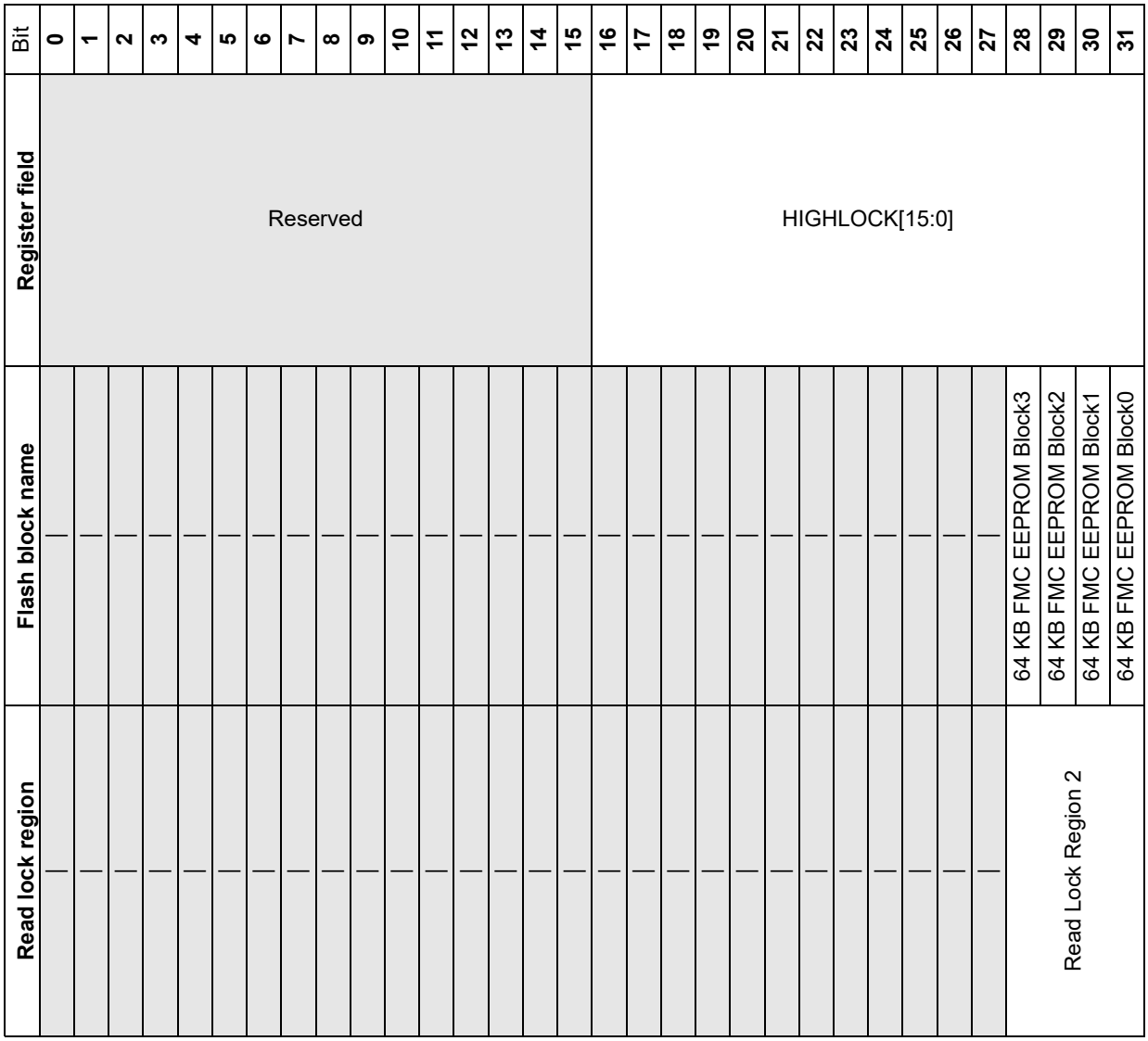


Figure 33. PASS_LOCK1_PGn registers

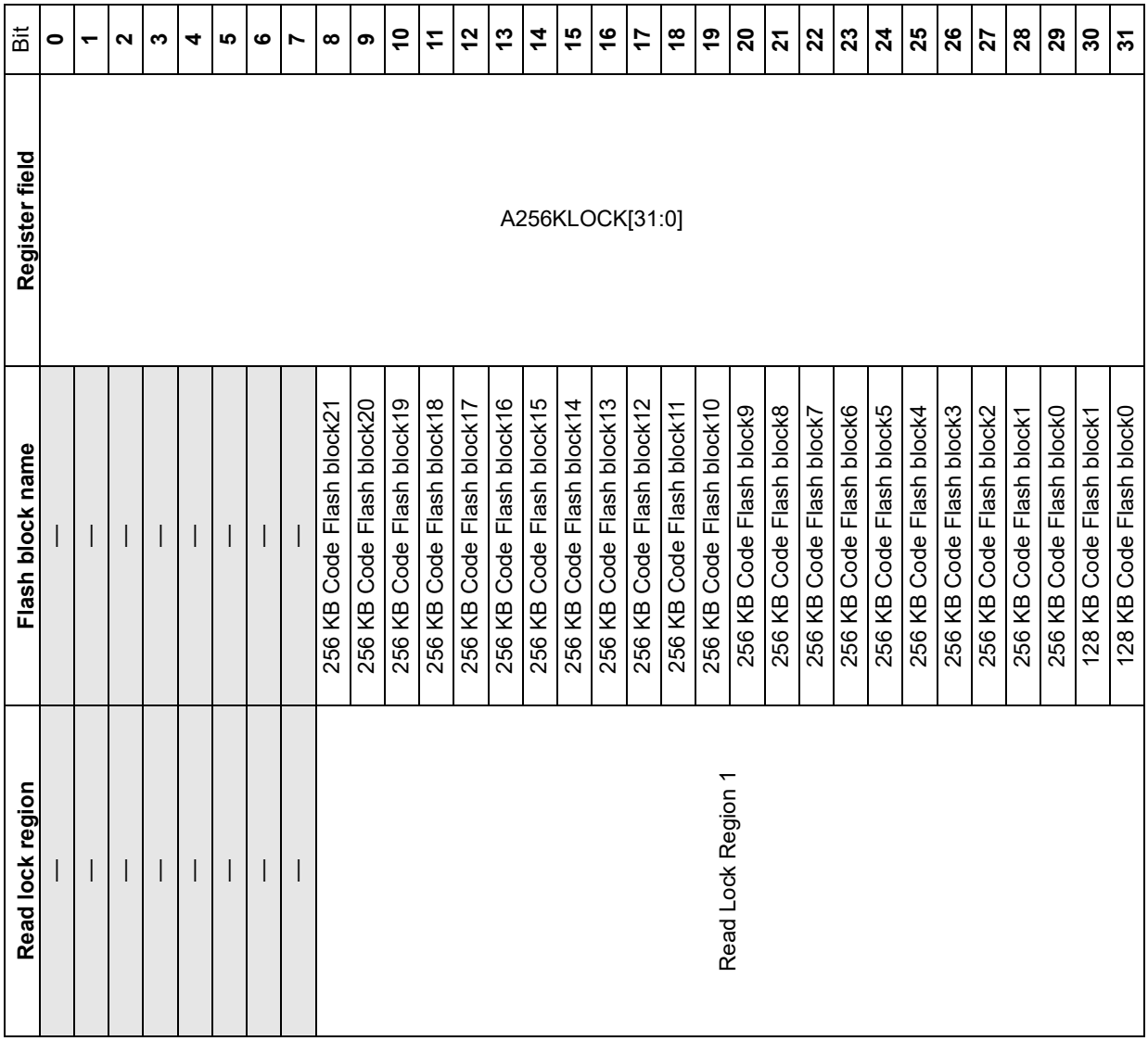


Figure 34. PASS_LOCK2_PGn registers

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Register field	PGL	DBL	Master Only	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	RL4	RL3	RL2	RL1	RL0	A256KLOCK[[47:32]															
Flash block name	Password Group Lock	Debug Interface Lock	—	—	—	—	—	—	—	—	—	HSM EEPROM Data Flash Read Lock	HSM Code Flash Read Lock	EEPROM Data Flash Read Lock	Code Flash Read Lock	UTEST Read Lock	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—

Figure 35. PASS_LOCK3_PGn registers

7.11.1.2 PASS Module DCF Records (includes lifecycle state and censorship)

The records required to configure the initial values for the PASS module password group registers are shown in *Table 116*. These values provide the following capabilities:

- Set flash memory to censored/uncensored state (censorship also depends on lifecycle)
- Override the basic flash write protection for flash memory blocks
- Set levels of password-based write protection (up to four 256-bit passwords can be required) for individual blocks of flash memory
- Set levels of password-based read protection (up to four 256-bit passwords can be required) for up to five flash memory Read Locking Regions
- Set levels of password-based protection (up to four 256-bit passwords can be required) for the debug part

Table 116. PASS DCF records

DCF CS[14:0]	DCF address (binary)		DCF client description	Number of	
	[16:10]	[9:2]		Valid bits	DCF records
000_0000_0000_1000	0000000	00101000–00101011	Reserved		
000_0000_0000_1000	0000000	00101100	Censorship(1)	16	1

Table 116. PASS DCF records (continued)

DCF CS[14:0]	DCF address (binary)		DCF client description	Number of	
	[16:10]	[9:2]		Valid bits	DCF records
000_0000_0000_1000	0000000	00101101–00101111	Reserved		
000_0000_0000_1000	0000000	00110000	Production Disable	2	1
000_0000_0000_1000	0000000	00110001–00111111	Reserved		
000_0000_0000_1000	0000000	01000000	LOCK0_PG0(2)	32	1
000_0000_0000_1000	0000000	01000001	LOCK1_PG0(2)	32	1
000_0000_0000_1000	0000000	01000010	LOCK2_PG0(2)	32	1
000_0000_0000_1000	0000000	01000011	LOCK3_PG0(2)	32	1
000_0000_0000_1000	0000000	01000100	LOCK0_PG1(2)	32	1
000_0000_0000_1000	0000000	01000101	LOCK1_PG1(2)	32	1
000_0000_0000_1000	0000000	01000110	LOCK2_PG1(2)	32	1
000_0000_0000_1000	0000000	01000111	LOCK3_PG1(2)	32	1
000_0000_0000_1000	0000000	01001000	LOCK0_PG2(2)	32	1
000_0000_0000_1000	0000000	01001001	LOCK1_PG2(2)	32	1
000_0000_0000_1000	0000000	01001010	LOCK2_PG2(2)	32	1
000_0000_0000_1000	0000000	01001011	LOCK3_PG2(2)	32	1
000_0000_0000_1000	0000000	01001100	LOCK0_PG3(2)	32	1
000_0000_0000_1000	0000000	01001101	LOCK1_PG3(2)	32	1

1. See *Section 92.6.3: Censoring and uncensoring the device* for values.
2. See *Section 7.11.1.1: PASS_LOCKn_PGn register bit mapping* for values.

7.11.2 Tamper Detection Module (TDM) configuration

The Tamper Detection Module provides a type of flash memory write protection mechanism that forces software to write a record associated with one or more blocks in a Tamper Detection Region (TDR) before the block(s) can be erased. An additional feature is provided to enable customers to override the protection mechanism for any TDR so that no record is required to be written before a block within the TDR can be erased.

Additionally, the TDM provides a mechanism for configuring individual flash memory blocks as One Time Programmable (OTP), that is, write once, blocks.

The following sections provide details on the TDM configuration registers and DCF clients.

Warning: **See the Tamper Detection Module (TDM) chapter for further details on the TDM, including the DCF clients and two additional memory-mapped registers.**

7.11.2.1 Diary Base Address (DBA) DCF client

DBA DCF client holds the base address of the diary, which is the region of flash that contains records corresponding to erase operations. This DCF client is described in the TDM chapter.

7.11.2.2 Tamper Region Override (TO) DCF client

Tamper detection can be overridden by writing to enable bits (one per region) in the Tamper Region Override (TO) DCF client. This DCF client is described in the TDM chapter. The mapping is shown below.

DCF Client Address: 0x0001 Access: User read/write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R																
W	0	0	0	0	0	0	0	0	0	0	TOE5	TOE4	TOE3	TOE2	TOE1	TOE0
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 36. Tamper Region Override (TO) DCF client

The mapping of the Tamper Region Override Enable (TOE*n*) DCF client is as follows:

Table 117. TO[TOE*n*] field mapping

Field	TDR
TOE0	0
TOE1	1
TOE2	2
TOE3	3
TOE4	4
TOE5	5

This DCF client is more fully described in the TDM chapter.

7.11.2.3 OTPEN*n* DCF clients

Configuring individual flash blocks as OTP blocks is accomplished by setting the appropriate values in the TDM_OTPEN*n* DCF clients.Each mapped bit is associated with a specific flash block. A “1” bit value indicates a block is OTP.

The mapping is shown in *Figure 37* - *Figure 40*.

Bit Number	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Field	TSLOCK	Reserved	LOWLOCK[13:0]													
Flash Block Name	16 KB UTEST NVM Block Space	—	—	—	64 KB HSM Code block3	64 KB HSM Code block2	64 KB Code Flash block1	64 KB Code Flash block0	32 KB Code Flash block1	32 KB Code Flash block0	16 KB HSM Code block5	16 KB Code Flash block4	16 KB Code Flash block3	16 KB Code Flash block2	16 KB Code Flash block1	16 KB BAF block0

Bit Number	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Field	MIDLOCK[15:0]															
Flash Block Name	—	—	—	—	—	—	—	—	—	—	—	—	—	—	16 KB HSM EEPROM block1	16 KB HSM EEPROM block0

Figure 37. TDR*x*_LOCK0/OTPEN0 DCF client

Bit Number	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Field	Reserved															
Flash Block Name	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—

Bit Number	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Field	HIGHLOCK[15:0]															
Flash Block Name	—	—	—	—	—	—	—	—	—	—	—	—	64 KB EEPROM Block3	64 KB EEPROM Block2	64 KB EEPROM Block1	64 KB EEPROM Block0

Figure 38. TDR*x*_LOCK1/OTPEN1 DCF client

Bit Number	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Bit Name	A256KLOCK[31:16]															
Flash block Name	—	—	—	—	—	—	—	—	256 KB Code Flash block21	256 KB Code Flash block20	256 KB Code Flash block19	256 KB Code Flash block18	256 KB Code Flash block17	256 KB Code Flash block16	256 KB Code Flash block15	256 KB Code Flash block14

Bit Number	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Bit Name	A256KLOCK[15:0]															
Flash block Name	256 KB Code Flash block13	256 KB Code Flash block12	256 KB Code Flash block11	256 KB Code Flash block10	256 KB Code Flash block9	256 KB Code Flash block8	256 KB Code Flash block7	256 KB Code Flash block6	256 KB Code Flash block5	256 KB Code Flash block4	256 KB Code Flash block3	256 KB Code Flash block2	256 KB Code Flash block1	256 KB Code Flash block0	128 KB Code Flash block1	128 KB Code Flash block0

Figure 39. TDR*x*_LOCK2/OTPEN2 DCF client

Bit Number	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Bit Name	A256KLOCK[63:48]															
Flash block Name	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—

Bit Number	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Bit Name	A256KLOCK[47:32]															
Flash block Name	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—

Figure 40. TDR*x*_LOCK3/OTPEN3 DCF client

7.11.2.4 TDR*x*_LOCK*n* DCF clients

Configuration of the 6 tamper regions is accomplished by setting field values in LOCK DCF clients (4 for each TDR).

Each set of 4 LOCKn DCF clients acts as a map of the MCU's flash memory blocks. A "1" in a LOCKn bit indicates that software must write a record before the corresponding block can be modified. The mapping of TDM_LOCKn DCF client bits to flash blocks is identical to the OTPEN*n* mapping.and is shown in *Figure 37* - *Figure 40*. Any available flash block can be mapped to any tamper protection region.

7.11.2.5 DCF client details

Table 118 contains the DCF client information.

Table 118. Tamper Detection Module (TDM) DCF clients

DCF CS[14:0]	DCF Address [16:10]	DCF Address [9:2] (Binary)	DCF Client Description	No. of Valid Bits	No. of DCF Records
000_0000_0001_0000	0000000	00000000	Diary Base Address	32	1
000_0000_0001_0000	0000000	00000001	Tamper Region Override	6	1
000_0000_0001_0000	0000000	00000010 - 00000111	Reserved		
000_0000_0001_0000	0000000	00001000	OTPEN0(1)	15	1
000_0000_0001_0000	0000000	00001001	OTPEN1(1)	4	1
000_0000_0001_0000	0000000	00001010	OTPEN2(1)	14	1
000_0000_0001_0000	0000000	00001011	OTPEN3(2)	0	1
000_0000_0001_0000	0000000	00001100- 00010011	Reserved		
000_0000_0001_0000	0000000	00010100	TDR0_LOCK0	32	1
000_0000_0001_0000	0000000	00010101	TDR0_LOCK1	32	1
000_0000_0001_0000	0000000	00010110	TDR0_LOCK2	32	1
000_0000_0001_0000	0000000	00010111	TDR0_LOCK3	32	1
000_0000_0001_0000	0000000	00011000	TDR1_LOCK0	32	1
000_0000_0001_0000	0000000	00011001	TDR1_LOCK1	32	1
000_0000_0001_0000	0000000	00011010	TDR1_LOCK2	32	1
000_0000_0001_0000	0000000	00011011	TDR1_LOCK3	32	1
000_0000_0001_0000	0000000	00011100	TDR2_LOCK0	32	1
000_0000_0001_0000	0000000	00011101	TDR2_LOCK1	32	1
000_0000_0001_0000	0000000	00011110	TDR2_LOCK2	32	1
000_0000_0001_0000	0000000	00011111	TDR2_LOCK3	32	1
000_0000_0001_0000	0000000	00100000	TDR3_LOCK0	32	1
000_0000_0001_0000	0000000	00100001	TDR3_LOCK1	32	1
000_0000_0001_0000	0000000	00100010	TDR3_LOCK2	32	1
000_0000_0001_0000	0000000	00100011	TDR3_LOCK3	32	1
000_0000_0001_0000	0000000	00100100	TDR4_LOCK0	32	1
000_0000_0001_0000	0000000	00100101	TDR4_LOCK1	32	1
000_0000_0001_0000	0000000	00100110	TDR4_LOCK2	32	1
000_0000_0001_0000	0000000	00100111	TDR4_LOCK3	32	1
000_0000_0001_0000	0000000	00101000	TDR5_LOCK0	32	1

Table 118. Tamper Detection Module (TDM) DCF clients (continued)

DCF CS[14:0]	DCF Address [16:10]	DCF Address [9:2] (Binary)	DCF Client Description	No. of Valid Bits	No. of DCF Records
000_0000_0001_0000	0000000	00101001	TDR5_LOCK1	32	1
000_0000_0001_0000	0000000	00101010	TDR5_LOCK2	32	1
000_0000_0001_0000	0000000	00101011	TDR5_LOCK3	32	1
000_0000_0001_0000	0000000	00101100 - 1111111	Reserved		

1. OTP DCF clients have the same structure and mapping as the write-protect bits of the flash and PASS module LOCK DCF clients. A value of '1' in an OTP bit indicates a block is programmable only once. Note that a block cannot be reconfigured as non-OTP after it has been configured as OTP.
2. There are no flash memory blocks mapped to the OTPEN3 DCF client.

8 Reset and Boot

8.1 Introduction

This chapter describes the reset and boot phase, from the time a power on or reset is applied, till the completion of the Boot Assist Flash (BAF) code.

After reset, Core_2 is woken-up, Checker Core_2 and HSM may be woken-up (refer to details in *Chapter 9: Device Configuration Format (DCF) Records*).

The following memory elements are involved in the boot-up process:

- UTEST flash memory
- TEST flash memory
- Boot Assist Flash (BAF)

8.1.1 TEST flash memory block

The TEST flash memory block contains Device Configuration Format (DCF) records used to hold trim values and other variables, as well as general device configuration information.

The trim values are for:

- Adjusting low-voltage and high-voltage detect circuit trip points
- Temperature sensor adjustments
- Power-on reset voltage trip point
- Analog-to-digital adjustments
- Internal RC oscillator (IRCOSC) trim values
- Power management configuration, for the devices supporting different regulation schemes

Note: The DCF records are written by the device manufacturer and programmed into TEST flash memory during production testing. Further programming of the TEST flash is disabled at the end of the factory test cycle.

8.1.2 UTEST flash memory block

The UTEST flash memory block also contains DCF records. Some UTEST DCF records are written by the factory and programmed during production testing. Others are written by the end user and programmed at the same time application code is programmed into the flash memory. UTEST DCF records are used to set up various control and configuration registers, including the Self-Test Control Unit (STCU2) and default memory configuration.

8.1.3 Boot Assist Flash

The Boot Assist Flash (BAF) contains factory code to facilitate the device boot-up procedure.

Refer to *Chapter 65: Boot Assist Flash (BAF)* for further details on this procedure.

8.2 Modules used in reset sequence

The modules involved in the SPC58xEx/SPC58xGx reset sequence are:

- Power Management Controller (PMC)
- Reset Generation Module (MC_RGM)
- Mode Entry Module (MC_ME)
- System Status and Configuration Module (SSCM)
- Self-Test Control Unit (STCU2)

8.2.1 Power Management Controller

The Power Management Controller (PMC) controls and monitors:

- Its own supply voltage
- The supply voltages to all the high- and low-voltage detect circuits
- The trip points for all the high- and low-voltage detect circuits
- The power supplies and reference voltages to the Analog-to-Digital Converter
- The major power supplies to the SPC58xEx/SPC58xGx

8.2.2 Reset Generation Module

The Reset Generation Module (MC_RGM) is a complex state machine that begins sequencing the SPC58xEx/SPC58xGx through the initial steps of the reset process. The MC_RGM does not execute program code, it is a state machine that centralizes the different reset sources and manages the reset sequence. Reset sources are organized into two categories: destructive and functional.

Refer to *Figure 41* for more information on the MC_RGM reset sequence.

8.2.2.1 Destructive resets

A destructive reset source is related to a critical error or dysfunction, usually caused by a hardware malfunction. When a destructive reset event occurs, software recovery is not possible and the contents of memory are assumed unknown. A full device reset sequence starting from PHASE0 is therefore applied to the device, ensuring a safe startup state for both digital and analog modules.

Examples of destructive resets are:

- Power-on reset
- Low voltage detection
- Reset escalation (when too many functional resets occur simultaneously)

8.2.2.2 Functional resets

A functional reset source is related to a less critical error or dysfunction usually not associated with hardware malfunction. When a functional reset event occurs, a partial reset sequence is applied to the device starting with PHASE1[FUNC]: most digital modules are reset normally and the state of analog modules, specific digital modules (for example, debug and flash memory modules) and system memory content is preserved.

Examples of functional resets are:

- External reset
- Machine check
- Software reset from mode entry
- Boundary scan instructions

8.2.3 Mode Entry module

The Mode Entry module (MC_ME) is responsible for configuring the execution mode, including the delivery of reset vectors to all the SPC58xEx/SPC58xGx cores.

8.2.4 System Status and Configuration Module

During the reset phase, the SSCM reads the DCF records in the TEST and UTEST flash memory areas in order to distribute within the device the pre-assigned configuration.

The DCF records are primarily for setting up the memory, configuring the Self-Test Control Unit (STCU2), and providing initial device configuration values.

The SSCM continues with the reset or boot-up sequence by locating boot vectors for the boot core and, if enabled for the HSM CPU. The start addresses are written into the Mode Entry Module (MC_ME) registers, which, in turns, will pass them to the proper CPU at the end of the reset phase.

8.2.5 Self-Test Control Unit

The Self-Test Control Unit (STCU2) is a self-contained module that runs a Logic Built-In Self-Test (LBIST) and a Memory Built-In Self-Test (MBIST). The DCF record can be used to select which individual tests are run for LBIST and MBIST and disable tests for modules and memory elements that are not going to be used in user applications, thus shortening device boot-up time.

8.3 Reset sequence

The power-up reset sequence always begins with the application of power and follows different sequences depending on the condition of the SPC58xEx/SPC58xGx device and whether various modes are enabled from settings in the DCF records. For instance, the SPC58xEx/SPC58xGx device enters Serial Boot mode (the Serial Boot Loader receives startup code and begins program execution) if a valid boot header file is not found (conditioned by the Life Cycle value).

If an emulation and debug device is detected during the reset sequence, the device enters the calibration sequence. The calibration sequence is not a user mode and is only used for code development.

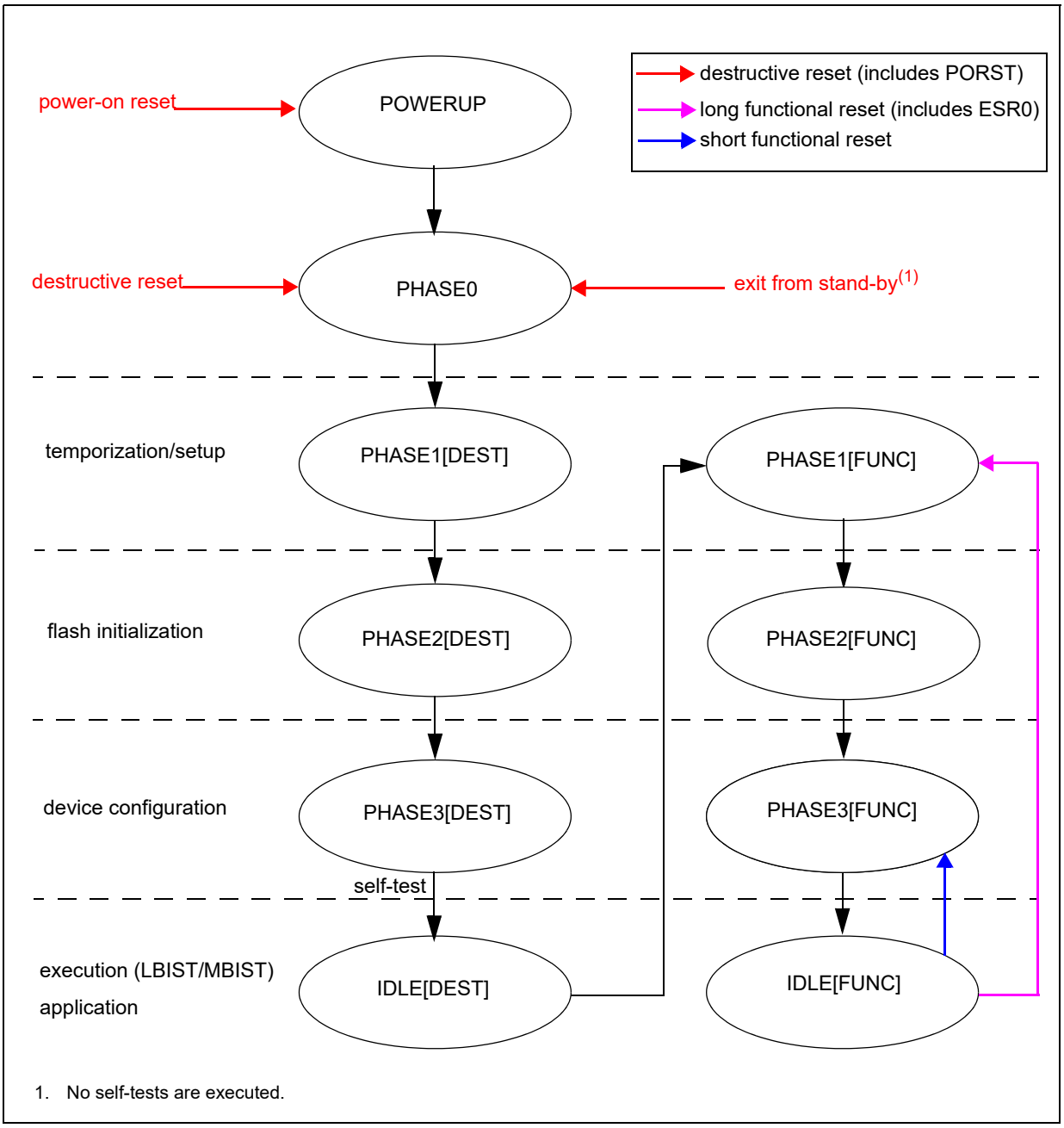
8.3.1 Power-on and the Reset Generation Module

When power is applied to the SPC58xEx/SPC58xGx, the Reset Generation Module (MC_RGM) advances the device through a series of steps shown in *Figure 41*.

The Reset Generation Module (MC_RGM) starts the System Status and Control Module (SSCM), which continues the boot-up process after the MC_RGM enters the IDLE state.

The STCU is reset and offline self-test (if enabled) is triggered on any power-on, destructive, or long external reset (external reset in this case is ESR0).

Figure 41. Reset Generation Module Reset Sequence



1. No self-tests are executed.

Table 119. Module status during reset phases

Module	PHASE								
	P0	P1 [DEST]	P2 [DEST]	P3 [DEST]	IDLE [DEST]	P1 [FUNC]	P2 [FUNC]	P3 [FUNC]	IDLE [FUNC]
IRCOSC	ON	ON	ON	ON	ON	ON	ON	ON	ON
PMC	ON	ON	ON	ON	ON	ON	ON	ON	ON
FUSE	ON	ON	ON	ON	ON	ON	ON	ON	ON
MC_RGM	ON(1)	ON	ON	ON	ON	ON	ON	ON	ON
System watchdog	RESET	RESET	ON	RESET	LBIST(2)	RESET	ON	RESET	ON
Flash module	RESET	RESET	ON	ON	LBIST	RESET	ON	ON	ON
XOSC	RESET	RESET	RESET	ON(3)	ON(3)	ON(3)	ON(3)	ON(3)	ON(3)
DEVICE CONFIG	RESET	RESET	RESET	LOAD	HOLD	HOLD	HOLD	HOLD	HOLD
SSCM	RESET	RESET	RESET	ON	ON	RESET	RESET	ON	ON
FCCU	RESET	RESET	CONFIG	CONFIG	LBIST	RESET/ ON(4)	CONFIG/ ON	CONFIG/ ON	ON
STCU	RESET	RESET	CONFIG	CONFIG	ON	RESET/ IDLE(5)	CONFIG/ IDLE	CONFIG/ IDLE	ON/ IDLE(6)
MEMU	RESET	RESET	ON	ON	LBIST	RESET/ ON(7)	ON	ON	ON
PASS	RESET	ON	ON	ON	ON	ON	ON	ON	ON
HSM	RESET	RESET	RESET	RESET	LBIST(8)	RESET	RESET	RESET	ON
Boot CPU	RESET	RESET	RESET	RESET	LBIST	RESET	RESET	RESET	ON
PIT	RESET	RESET	RESET	RESET	LBIST	IDLE/ON	IDLE/ON	IDLE/ON	IDLE/ON
Other modules including cores	RESET	RESET	RESET	RESET	LBIST	RESET	RESET	RESET	ON(9)
								Short functional reset	
						functional reset			
	destructive reset								

1. RESET only during POR.
2. STCU time-out monitoring LBIST/MBIST execution.
3. Depending on flash configuration.
4. FCCU remains ON during functional reset; it is instead reset in PHASE1[FUNC] if coming from IDLE[DEST].
5. The STCU remains idle during functional reset; it is instead reset on long external reset (external reset here is ESR0).
6. Can be triggered by SW. Please refer to *Chapter 90: Self-Test Control Unit (STCU2).*
7. MEMU is reset whenever STCU is reset.
8. LBIST is optionally triggered depending on safety execution directives.

9. LBIST is optionally triggered depending on safety execution directives after enabling from IOP. Please refer to *Chapter 65: Boot Assist Flash (BAF).*

8.3.2 Power-up phase: power stabilization

When a power-on reset event occurs (initial application of power), the reset generation module (MC_RGM) causes the device to enter the power-up phase of the reset sequence (refer to *Figure 41*). The only method to enter the power-up phase is from a power-on reset event.

The device remains in the power-up phase until the Power Management Controller module (PMC) determines that all power supplies are in their respective operational ranges.

The supplies for initial configuration are:

- VDD_HV_CORE: high voltage supply for core module, including power management unit, internal RC oscillator (IRCOSC).
- VDD_LV_CORE: low voltage supply for digital module.
- VDD_HV_FLA: high voltage supply for flash module.
- VDD_LV_FLA: low voltage supply for flash module.
- VDD_HV_IO_IM: high voltage supply associated with system pins (PORST, ESR0, TEST Mode).

Peripheral supplies such as VDD_HV_IO_JTAG, VDD_HV_IO_FLEX or VDD_HV_ADC do not gate the power-up process. The low- and high-voltage detect circuits for these and other power supplies are managed by the PMC, which provides enable and status bits for these voltage detection circuits.

The PMC sends a signal to the Reset Generation Module when power stabilization has been achieved. The Reset Generation Module then advances to the PHASE0 state (refer to *Figure 41*). During the power-up phase, the PMC does the following:

- Drives to a logic ‘0’ all VDs (voltage detectors) signals monitored by the Power Management Control (PMC) module.
- Monitors its own power supply voltage (VDD_HV_PMC) to determine that the applied voltage is within a specified range.
- Monitors the core power supply voltage (VDD_LV_CORE) to determine that the applied voltage is within a specified range.
- Monitors the low- and high-voltage detect circuit power supply voltage to determine that the applied voltage is within a specified range.
- Holds the outputs of the internal low- and high-voltage detect circuits at a ground state until the PMC determines that all power supplies needed for correct device initialization and configuration are within their respective functional voltage ranges.
- Determines that all power supplies needed for correct device initialization and configuration are within their respective functional voltage ranges.
- Enables all low- and high-voltage detect circuits once all power supplies are at their functional levels.
- Begins to monitor the various power supplies using the high- and low-voltage detect circuits.
- Provides status information to the Reset Generation Module (MC_RGM) indicating whether or not power is properly applied.

To exit power-up and enter PHASE0:

- All enabled destructive resets must be processed.
- All power supplies must be at their functional voltages.
- The PMC must signal the MC_RGM that all power supplies are at the required levels.

8.3.3 PHASE0 Phase: analog supply initial configuration

This phase is entered:

- On exit from power-up phase.
- On PORST pin falling edge detection except for the initial power-up sequence.
- Immediately from PHASE2[DEST], PHASE3[DEST], IDLE[DEST], PHASE1[FUNC], PHASE2[FUNC], PHASE3[FUNC] or IDLE[FUNC], on the occurrence of a PORST falling edge event other than power-on reset.

During PHASE0:

- All digital modules are reset, including safety, security, and test modules.
- All trimming bits for analog modules are reset.
- Startup of the internal analog modules (IRCOSC, FUSE, Flash memory, and I/Os) begins:
  - PMC determines that proper voltages are applied.
  - I/O pins are ensured correctly configured: outputs are driven to known levels, inputs ignored.

To exit PHASE0 and enter PHASE1[DEST]:

- All enabled destructive resets must be processed.
- Gating PHASE0 must be released.
- All processes initiated in PHASE0 must be completed. This generally includes startup of the internal analog modules: PMC, IRCOSC, and I/Os.

Note: The core voltage must rise above LVD100 upper threshold to exit PHASE0. This ensures significant hysteresis on low voltage supply avoiding spurious low voltage detection during the power-up process.

8.3.4 PHASE1[DEST] Phase: temporization and monitoring setup

This phase is entered on exit from PHASE0.

During PHASE1[DEST]:

- Initial configuration of the watchdog to monitor the reset sequence
- Temporization for reset signal propagation in case of reset

Test modules are available during PHASE1[DEST], but their functionality is limited by the security module.

All PHASE1[DEST] tasks must be completed before entering PHASE2[DEST].

8.3.5 PHASE2[DEST] Phase: flash initial configuration

This phase is entered on exit from PHASE1[DEST].

During PHASE2[DEST]:

- Reset is released to the flash memory module by the Reset Generation Module.
- The flash memory module starts its initial configuration process.
- The flash memory initialization is performed by a state machine internal to the flash module.
- The system watchdog starts monitoring flash memory configuration execution so that a reset occurs in the event of configuration failure.

Note: System Watchdog is hardware configured to generate destructive reset in case of flash memory failure during flash boot process. In this case the software can read the cause of the reset by the MC_RGM_DES.F_FIF flag. On boot completion, the watchdog event is converted to generate an FCCU fault only.

All PHASE2[DEST] tasks must be completed before entering PHASE3[DEST].

8.3.6 PHASE3[DEST] Phase: device configuration

This phase is entered on exit from PHASE2[DEST].

During PHASE3[DEST], the System Status and Configuration Module (SSCM) starts by retrieving the Device Configuration Format (DCF) record from the TEST and UTEST flash memory areas, and uses the configuration and initialization information therein to:

- Configure the Self-Test Control Unit (STCU2) and the initial memory map of the device.
- Transfer reset vectors for the various cores to the Mode Entry Module (MC_ME). This information enables tests that the STCU runs and also provides the memory map setup information so that the STCU can test memory.
- Write trim values for analog modules including the Analog-to-Digital Converter, low- and high-voltage detect circuits, and the temperature sensor to their respective registers.

The DCF record programmed into the TEST flash memory area is written by the manufacturer. The portion of the DCF record programmed into the UTEST flash memory is usually written by the customer, although there is a default DCF record programmed into both the TEST and the UTEST flash memory areas during device production testing.

The DCF record also provides configuration information for the following:

- Trimming of the analog modules (for example, voltage regulator, voltage detectors, I/Os, and oscillator)
  - Trim values are in DCF records in the TEST flash memory area
  - Trim values are determined during production test and programmed into the TEST flash memory area
  - Security configuration
- Application configuration bits:
  - Watchdog configuration
  - Safety execution directives (self-test, LBIST/MBIST)
  - Boot Assist Flash options
  - Oscillator startup during reset sequence configuration

To exit PHASE3[DEST]: All processes that need to be completed in PHASE3[DEST] must be finished.

This phase depends on actual trim values, especially those used for safety execution directives. This device configuration is maintained until the next internal power-on reset. Each time a destructive reset event occurs, the SSCM takes values from the DCF record and writes them to the applicable registers.

During this phase, the internal STCU watchdog is started and is only cleared after completion of STCU configuration.

If self-test execution is the configuration information from the DCF record requests, the device moves to IDLE[DEST] and the STCU2 begins self-test. At the end of the test, the STCU2 asserts a functional interrupt causing the Reset Generation Module to move to PHASE1[FUNC]. If self-test execution is not requested, the device moves directly to IDLE[FUNC]. The Reset Generation Module determines whether self-test execution is enabled or not by examining an entry in the DCF record stored in the UTEST flash memory.

Note: When self-test execution is not requested, it is possible to maintain the device in PHASE3[DEST] by forcing either the PORST or ESR0 pin low. This can be used for debugging purposes by allowing external test/development equipment to connect through JTAG to the internal debugger.

8.3.7 IDLE[DEST] Phase: self-test execution

This phase is entered upon exit from PHASE3[DEST] if the execution of self-test is enabled.

During the IDLE[DEST] phase:

- Dedicated watchdog is configured to monitor self-test execution completion within expected time. The length of this phase is variable depending on the self-test requirements.
- The Self-Test Control Unit (STCU2) executes all tests specified by the DCF information retrieved in PHASE3[DEST].
- The self-test engine updates the self-test completion flag and triggers an associated functional reset on completion of self-test.

The length of this phase is variable depending on the self-test requirements.

Note: When the SSCM is decoding the Device Configuration field, it sets and clears various flags in STCU registers and memory that can later be examined by the SSCM to determine which self-test directives must be run.

Exiting IDLE[DEST] and moving to PHASE1[FUNC]:

- The IDLE[DEST] phase is automatically exited when the self-test code generates a functional reset on completion.
- The functional reset causes the reset generation module to move to the PHASE1[FUNC] phase.

Note: In the case of unrecoverable fault detected during the execution of the offline BIST, the STCU requests a destructive reset and device re-enters PHASE0[DEST].

In the case of a permanent fault, when the reset escalation is reached, the hardware keeps the device in PHASE0[DEST] until a Power On Reset occurs.

Whereas in the case of a transient fault, the software can read the cause of the reset by reading the MC_RGM_DES.F_SUF flag.

8.3.8 PHASE1[FUNC] Phase: temporization and monitoring setup

This phase is entered either:

- On exit from IDLE[DEST].
- Immediately from PHASE2[FUNC], PHASE3[FUNC], or on IDLE[FUNC] when a non-masked external or functional reset event occurs, providing the source of the reset event has not been configured to trigger a ‘short’ reset sequence.

During PHASE1[FUNC]:

- Initial configuration of the watchdog to monitor reset process.
- Temporization for reset signal propagation in case of reset.

To exit PHASE1[FUNC] and enter PHASE2 [FUNC]:

- All enabled, non-shortened functional resets must be processed.
- All processes started in PHASE1[FUNC] must be completed.
  - Initial configuration of the watchdog to monitor reset process.
  - Temporization for reset signal propagation in case of reset.

During this phase, test modules are available.

8.3.9 PHASE2[FUNC] Phase: flash initial configuration

This phase is entered on exit from PHASE1[FUNC].

During the PHASE2[FUNC] phase:

- The reset signal is released to the flash memory.
- Flash initialization and configuration begins using a state machine that is internal to the flash memory module.
- A watchdog monitors flash initialization to ensure a reset in case of flash memory configuration failure.

The reset state machine exits PHASE2[FUNC] and enters PHASE3[FUNC] on verification that all processes started in PHASE2[FUNC] are completed.

8.3.10 PHASE3[FUNC] Phase: device configuration monitoring

PHASE3[FUNC] is entered:

- On exit from PHASE2[FUNC].
- Immediately from IDLE[FUNC] when an enabled, short functional reset event occurs.

During PHASE3[FUNC], configuration information contained within the UTEST flash memory sector is checked against information extracted during PHASE3[DEST] and written to the Self-Test Control Unit (STCU2). In case of mismatches, a dedicated boot fault is triggered within the FCCU which, if enabled, can cause a system reset. The purpose of PHASE3[FUNC] is to verify that the DCF information transferred to the STCU, MC_ME and other modules, in PHASE3[DEST], was performed correctly and has not been corrupted.

The Reset Generation Module exits PHASE3[FUNC] and enters the IDLE[FUNC] on verification of the following:

- All processes started in PHASE3[FUNC] are completed.
- PORST and ESR0 pins are not forced low externally.
- A minimum number of cycles have elapsed since the last enabled, shortened functional reset event.

After all PHASE3[FUNC] internal actions have been completed, it is possible to maintain the device within PHASE3[FUNC] by forcing either the PORST or ESR0 pin low. This can be used for debugging purposes in order to connect through the JTAG port to the internal debugger.

8.3.11 IDLE[FUNC] Phase

This is the Reset Generation Module's final phase of the reset sequence. It is entered either on exit from PHASE3[DEST] (if self-test was not enabled) or at the completion of PHASE3[FUNC]. When the IDLE[FUNC] phase is reached, the MC_RGM releases control of the system to the System Status and Control Module. The SSCM, which started in PHASE3[FUNC], (or in PHASE3[DEST] in case no self-test was enabled) runs and waits for a signal from the MC_RGM indicating that the MC_RGM is now in the IDLE state. The MC_RGM then waits for new reset events that can trigger a reset sequence. The SSCM continues with the system boot-up sequence.

8.3.12 System start-up

At the start-up, only one processor (from now on called Boot CPU) and, if available and enabled, the Security Module CPU wake up.

8.3.12.1 Boot CPU reset vector

After the reset phase is completed, the Boot CPU starts executing at the value forced onto its reset vector. The reset vector is driven by the ME peripheral, according to the value stored into the CADDR register.

The value of the CADDR register depends on whether the reset is generated as a consequence of a wake-up reset event (restart from Stand-by) or any other event. The Flash status (powered on or off) also concurs to select the reset vector value.

In case the reset is not due to a wake-up event, the CADDR registers for the Boot CPU and, if existing, the Security Module CPU are written by the SSCM, according to the system startup configuration, which is stored in Flash and read by the SSCM from the Flash during the reset.

In case the reset is due to a wake-up event, the ME reads the CADDR reset vectors from the RGM, which had saved a copy of the CADDR registers in its internal latches, which resides in the low-power domain. The same is valid (if embedded) for the HSM Module (refer to *Section 8.3.12.4*).

8.3.12.2 Boot CPU start-up

The Boot CPU startup configuration comprises of the following settings:

- BAF configuration DCF. It is used to decide whether the BAF code (stored by the manufacturer in the Flash BAF block) will be skipped or executed. The BAF start address is also set within the same DCF.
- Application boot records (programmed by the customer). It is used to set the customer application entry point after reset or after the BAF execution (if enabled).

The target for this device is to let the customer choose whether either the BAF code shall be started after the reset phase or directly the customer application, using the same sales part number.

The BAF configuration DCF shall be permanently written into the UTest block.

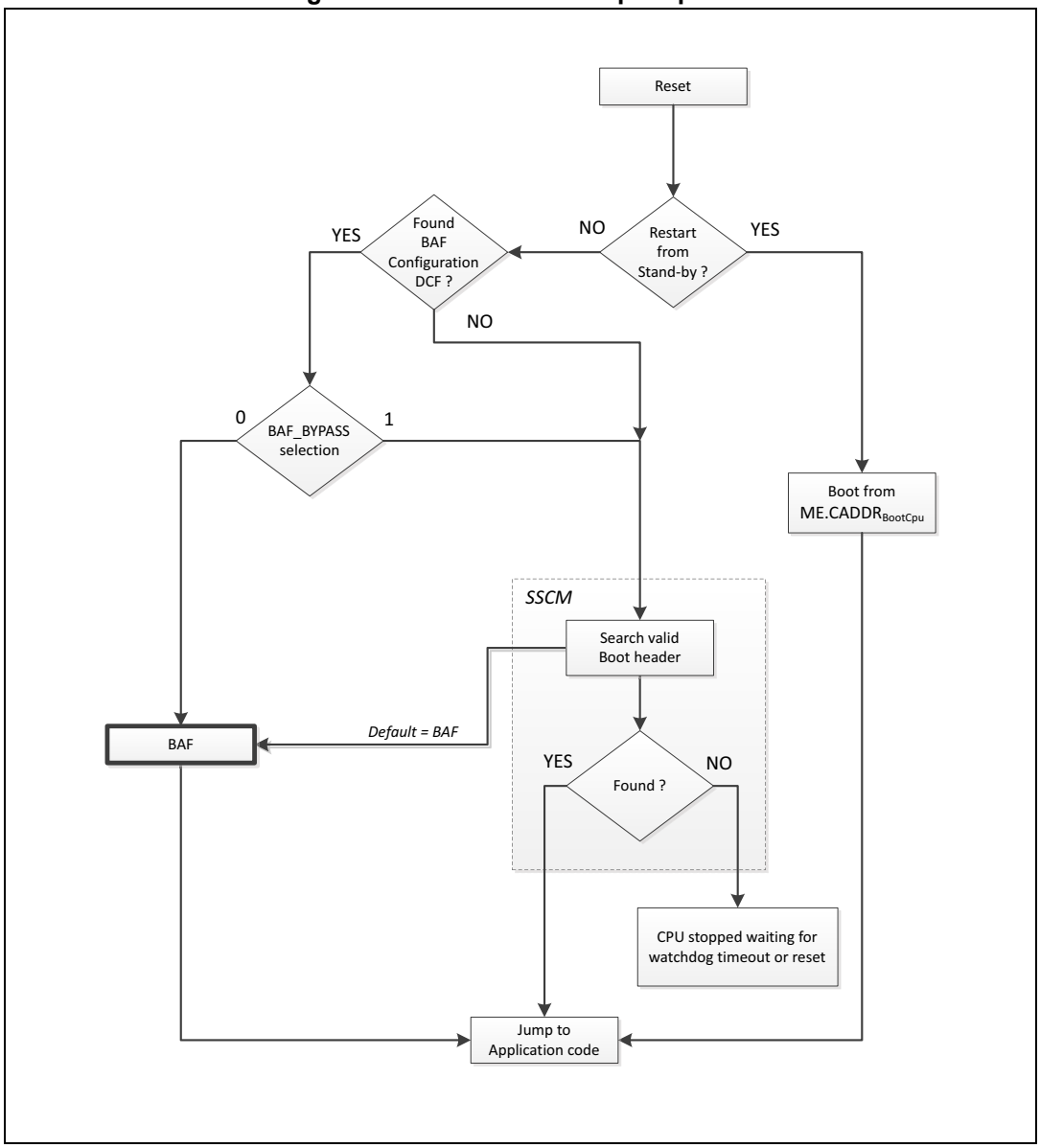
If the BAF configuration DCF is not programmed, the default value for the reset start address depends on the Life Cycle.

If the BAF configuration DCF is programmed, its value determines whether the CPU will start directly from the BAF code (programmed by ST in the Flash BAF block) or at the application entry point defined by the customer.

The application entry point can be retrieved both in HW (by the SSCM or ME, depending on the fact the system restarts from Stand-by or not (refer to *Section 8.3.12.1*)) or in SW (by the BAF). In both cases, the application entry point is retrieved by searching for a valid Boot Record, which shall be programed by the customer in one of the pre-determined Flash memory locations looked at by the SSCM or the BAF code. Next sections provide more details about the Boot Record structure and search operation.

8.3.12.2.1 Start-up flow chart

Figure 42. Boot CPU start-up sequence



8.3.12.2.2 HW boot record search (SSCM search)

If the programmed value of the BAF configuration DCF skips the BAF, then the SSCM module looks for a valid software entry point, called SSCM boot record. The SSCM boot record is organized as shown in the *Table 120*. The “Application start address” value will be passed to the CADDR register of MC_ME, which is used to set the CPU start address (reset vector) after the reset phase.

Table 120. Boot record structure searched by the SSCM

Offset	Field description	
0x4	Application start address	
0x0	SSCM Boot Record Tag (0x5A)	Reserved

The SSCM searches for a valid boot record at the addresses specified in *Table 121*. A boot record is considered valid if it contains the value 0x5A in the “Boot Record Tag” field.

The “Search Order” column specifies which location is searched first.
The SSCM stops the search as soon as a valid boot record is found.

Table 121. SSCM boot record search locations

Search order	Address	Location	SSCM configuration parameter
0 (First)	0x00FC_0000	Flash 16 KB Code Block 3	RCHW_LOC_0
1	0x00FC_4000	Flash 16 KB Code Block 1	RCHW_LOC_1
2	0x00FC_8000	Flash 16 KB Code Block 4	RCHW_LOC_2
3	0x00FC_C000	Flash 16 KB Code Block 2	RCHW_LOC_3
4	0x0100_0000	Flash 128 KB Code Block 0	RCHW_LOC_4
5	0x0102_0000	Flash 128 KB Code Block 1	RCHW_LOC_5
6 (Last)	0x0040_4000	Flash 16 KB BAF block	RCHW_LOC_6

Depending upon the fact that a valid SSCM boot record is found and the content of the SSCM boot record, three scenarios can happen:

1. If a valid Boot Record is programmed by the customer, then the application is executed after the reset from the specified entry point address.
2. If no valid Boot Record is programmed by the customer, then the default BAF Boot Record is retrieved and the BAF code is executed after the reset.
3. If no valid Boot Record is found, then the BAR code is executed after the reset. This situation happens only during ST production stage.

Note: When the device is shipped to the customer, the slot number 6 is programmed with a valid SSCM Boot Record, which points to the BAF code. This is the default start address for the SSCM Boot CPU when no other valid Boot Record are programmed or found. The content of the slot number 6 cannot be changed.

8.3.12.2.3 SW boot record search (BAF search)

The BAF is executed if:

- the BAF configuration DCF does not bypass the BAF code.
- the BAF configuration DCF does bypass the BAF code and no valid Boot Record is programmed by the customer.

The latter case happens because before the unit is delivered to the customer a valid SSCM boot record pointing to the BAF code is programmed at the location number 6. This boot record will point to the BAF entry point. Because the boot record number 6 is the last one being searched by the SSCM, this means that, if no other boot record is programmed, the

software will start from the BAF. Notice that the location of the boot record number 6 is placed within the BAF block itself, making it not erasable and not over-programmable (the BAF block is OTP).

When the BAF code is executed after the reset sequence is completed, a valid application Boot Record for the application entry point is searched. The mechanism is the same executed by the SSCM: a set of fixed locations are analyzed and, if a valid boot record is found, then, at the end of the BAF code, the CPU will start executing from the application entry point.

Note that the valid boot record is identified by a different tag: 5Ah for the SSCM; A5H for the BAF. This will make it possible to program the same boot record locations and launch the application directly after the reset or after the BAF code execution. *Table 122* shows the BAF boot record structure.*Table 124* shows the addresses of the location searched by the BAF code. *Figure 43* shows a flow chart of the whole reset sequence, including the BAF search for a valid boot header.

Table 122. Boot record structure searched by the BAF

Offset	Field description	
0xC	CPU_1 Reset vector	
0x8	CPU_0 Reset vector	
0x4	Boot CPU (CPU_2) Reset vector	
0x0	BAF Boot Record Tag (0xA5)	CPU enable flags (refer to *Table 123*)

Table 123. CPU enable flags

16-19	20-23	24-27	28	29	30	31
0000	0000	0000	0	CPU_1	CPU_0	CPU_2

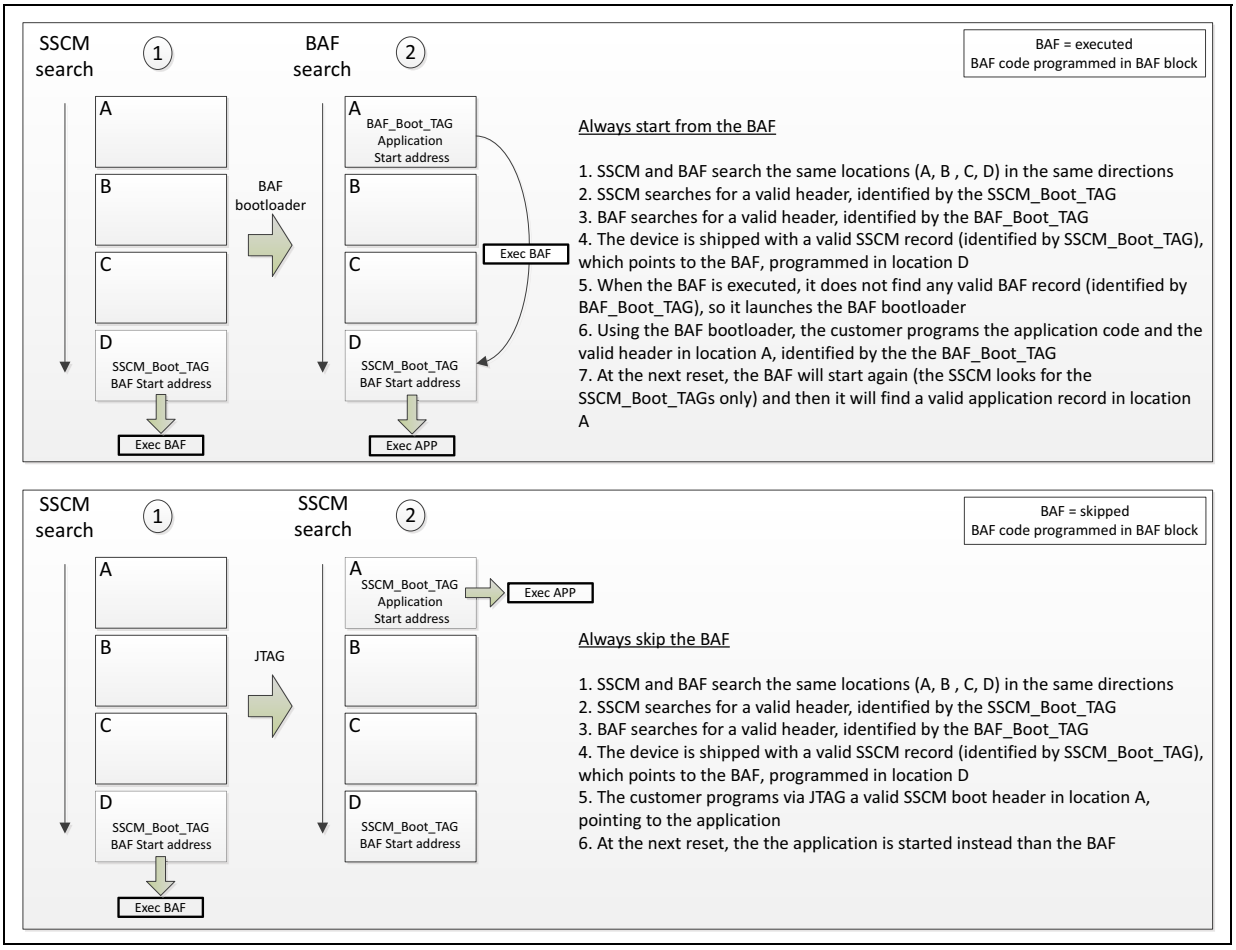
Table 124. BAF boot record search locations

Search order	Address	Location
0 (First)	0x00FC_0000	Flash 16 KB Code Block 3
1	0x00FC_4000	Flash 16 KB Code Block 1
2	0x00FC_8000	Flash 16 KB Code Block 4
3	0x00FC_C000	Flash 16 KB Code Block 2
4	0x0100_0000	Flash 128 KB Code Block 0
5 (Last)	0x0102_0000	Flash 128 KB Code Block 1

8.3.12.2.4 HW and SW boot header search summary

The following picture summarizes the mechanism adopted to search for a valid boot header with the SCCM or the BAF code, or both.

Figure 43. Valid boot header search



Note: for sake of simplicity, the picture shows only 4 boot record positions.

8.3.12.3 Security Module CPU start-up

The Security Module CPU is also woken up after a reset. The Security Module CPU receives the start address as described in *Section 8.3.12.2.2: HW boot record search (SSCM search)*.

The SSCM looks for a valid boot header record and then direct the Security Module CPU to the application start address specified within the record.

8.3.12.3.1 HSM restart from Stand-by

In case of restart from stand-by, then the application entry point is conditioned by the Flash status (powered on or off) and by two DCF:

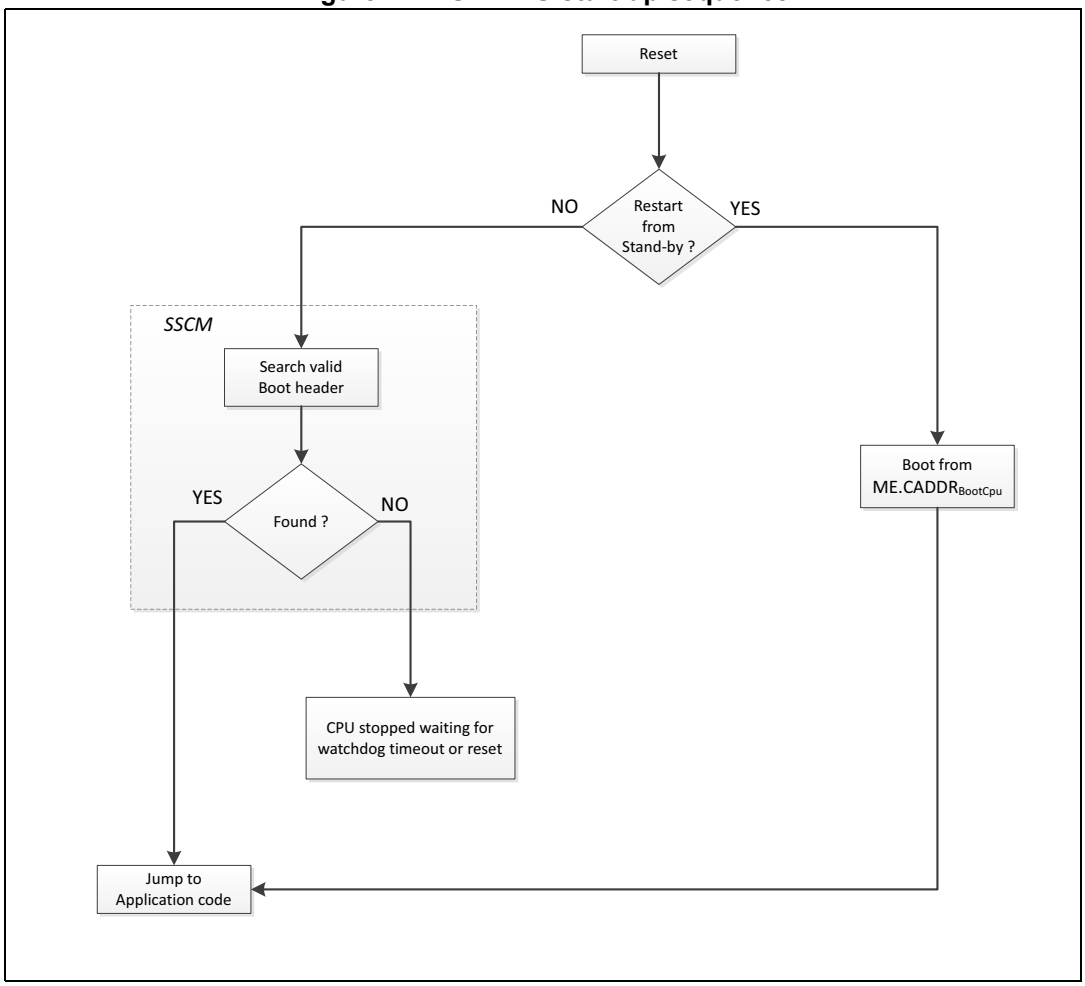
1. a DCF (to be placed in UTest under the customer control) determines whether the security module should also restart after Stand-by(6).

6. If the HSM is flagged by this DCF to not restart, then also the HSM Standby RAM can be turned off during Stand-by

2. a DCF (to be placed in UTest under the customer control) is used to inhibit the possibility to let the Flash powered off after a restart from Stand-by. If the restart is from Stand-by, the Flash is powered off, and the DCF client allows this scenario, then the Boot CPU restarts from the address stored into the corresponding ME.CADDR register. Otherwise, the Boot CPU restarts following the same flow as per a non stand-by reset.

8.3.12.3.2 Start-up flow chart

Figure 44. HSM CPU start-up sequence



8.3.12.3.3 Boot vector search

The Security Module start vector is searched by the SSCM in a similar way as described in the chapter for the Boot CPU (HW SSCM search).

The SSCM stops the search as soon as a valid boot record is found within one of these locations.

Table 125. SSCM boot record search locations

Search order	Address	Location
0 (First)	0x0060_C000	Flash 16 KB HSM Code Block 5
1	0x0061_0000	Flash 64 KB HSM Code Block 2
2	0x0062_0000	Flash 64 KB HSM Code Block 3

8.3.12.4 Restart from Stand-by RAM with the Flash powered off

This feature makes it possible to wake-up the micro-controller from the Stand-By mode and let the Boot CPU and the Security Module CPU restart directly from the RAM, while the Flash is powered off.

Note: For security reason, the possibility to keep the Flash powered off in any execution modes shall be gated by a DCF.

In case of restart from stand-by with the Flash powered-off, only the Boot CPU shall be woken up. Other CPUs shall be kept under reset. The restart after Stand-by from RAM is made possible by using the Mode Entry registers:

- ME_CCTL*n* => to set up the processor status (ON/OFF) for each mode.
- ME_CADDR*n* => to set up the processor reset vector (note: the register associated to the Security Module CPU can be written by the Security Module CPU only).
- ME_<mode>_MC => to set up whether the Flash is ON or OFF.

Note: The registers related to the HSM CPU can be modified by the HSM only.

8.3.12.5 Start-up summary

Table 126. Start-up summary

Start from	Flash status	Security Module CPU (1)	Boot CPU	Other CPUs (2)
Power-on / Destructive reset	ON	It starts from the address provided by SSCM	Depending on the BAF configuration DCF setting, it starts from the BAF or from the address provided by SSCM	They can be woken-up with a ME mode change
Functional reset	ON	It starts from the address provided by SSCM	Depending on the BAF configuration DCF setting, it starts from the BAF or from the address provided by SSCM	They can be woken-up with a ME mode change

Table 126. Start-up summary (continued)

Start from	Flash status	Security Module CPU (1)	Boot CPU	Other CPUs (2)
From Stand-by	ON	It starts from the address provided by SSCM	It starts from the address saved in the ME CADDR register for the Boot CPU (which can be both in Flash or in the low power portion of the system RAM)	They can be woken-up with a ME mode change
	OFF	It starts from the address is saved in the ME CADDR register for the Security Module CPU (in case the Security Module RAM has a portion in the low-power domain, it is advisable for security reason to set the restart address within it)	It starts from the address saved in the ME CADDR register for the Boot CPU (which shall be in the low power portion of the system RAM)	They shall not be woken-up (3)

1. If existing and enabled.
2. For MCUs with multiple application processors.
3. DCFs for TCM memory repair will not be preserved during Stand-by.

8.3.13 Waking-up other CPUs

The process of waking-up the other CPU is handled by writing some registers within the ME module and triggering a Mode Change.

If the BAF code is executed, it reads form the Boot Header record the information regarding which CPU shall be woken up and at which address shall they start from, and then applied a Mode change to enable them.

If the BAF code is not executed, it is responsibility of the user code to properly write the ME module registers and trigger a mode change in order to wake up the other CPUs form the intended addresses.

8.3.14 BAF Bootloader

The BAF code includes a way to upload via CAN or UART some code into the system RAM and execute it. The possibility to execute the Bootloader is conditioned by the Life Cycle status, which regulates what can or cannot be done according to the stage of the micro-controller. If the Bootloader is allowed by the Life Cycle, then it is executed whenever the Boot CPU, executing code in the BAF, does not find a valid boot header file in flash memory. This mode uses a defined protocol to receive a program over a serial port via LINFlexD or CAN. The Boot CPU then jumps to the first instruction of the uploaded program and begins execution.

The BAF bootloader code includes the following steps:

1. Configure LINFlexD or M_CAN module and external pins
2. Wait for START WORD
3. Receive START WORD
4. Receive START ADDRESS
5. Receive DOWNLOAD SIZE
6. Receive CODE
7. Branch to START ADDRESS

Note: There is a software watchdog timer that is started by the BAF code. If a program is not received before the watchdog timer timeout, a reset occurs.

8.3.14.1 Calibration Sequence

The micro-controller detects the presence of an emulation and debug device (ED) normally used in place of a regular production SPC58xEx/SPC58xGx (PD). The purpose of the emulation and debug device is to facilitate the use of external software development tools, such as CodeWarrior. The emulation and debug device (ED) consists of a regular production SPC58xEx/SPC58xGx (PD) and a special Buddy Device (BD) in a single package.

The I/O Processor (IOP) executing BAF code writes a special sequence to the emulation and debug device. If no emulation and debug device is detected, the IOP steps to decision point E.

A discussion of software development and software emulation tools is beyond the scope of this explanation of SPC58xEx/SPC58xGx reset.

In this example, it is assumed that an emulation and debug device (ED) is not being used in place of a regular production SPC58xEx/SPC58xGx.

9 Device Configuration Format (DCF) Records

9.1 Introduction

Device Configuration Format (DCF) records are used to configure certain registers in the device during system boot while the reset signal is asserted. An individual DCF record is 64 bits long. It consists of a pointer to the location of a register internal to the device and the respective data.

Note: Because the Flash memory programming resolution is 128 bits then 2 DCF records must be written within 1 Flash programming operation.

There are two broad categories of DCF records: TEST DCF Records and UTEST DCF Records.

- TEST DCF records are programmed by the MCU manufacturer. They are used mainly to:
  - Program registers involved in trimming trip points for voltage comparators
  - Adjusting analog to digital voltage supplies
  - Trim oscillator frequencies
  - Enable RAM repair

Note: The TEST DCF Records are programmed into TEST Flash during production and cannot be modified. TEST Flash is not visible to the user.

- UTEST DCF records can be either:
  - Factory programmed during production testing (see “UTEST Flash memory map” table in the Memory Map chapter).
  - User programmed when application code is written into the Flash memory. User-supplied UTEST DCF records start at the next location in the UTEST memory map following the factory UTEST DCF records. The user-defined DCF records set up the initial memory map, define which tests the Self-Test Control Unit (STCU2) runs during the boot sequence, enable the HSM, assign Flash memory blocks to specific tamper detect regions, assign memory blocks as OTP and assign Flash blocks to be associated with specific password groups.

Note: DCF records are HW processed by SSCM during reset.

Soft DCF records are SW processed by BAF code execution. Refer to Section 65.3.5.1: ‘Soft’ DCF clients in Chapter 65: Boot Assist Flash (BAF).

System boot is a complex process requiring a considerable amount of initialization to take place before releasing reset. Before the device can be used in an application, the user application code, reset vectors for all of the CPUs and the DCF records must be properly programmed into their respective Flash memories.

When power is applied to a properly programmed device:

1. The Power Management Controller (PMC) takes control of the device until the system power supplies have reached predefined levels.
2. The PMC then signals the Reset Generation Module (RGM) to begin the boot sequence.
3. During boot, the RGM enables the System Status and Control Module (SSCM) to read the device configuration records and write information to specific registers.
4. After registers are initialized, the STCU2 starts its Self-Test sequence as per its device configuration registers initialized through SSCM.

9.2 DCF clients

DCF clients are 32-bit hardware registers inside a module that receive and store the data from a DCF record. This stored data is used to initialize registers and to configure features. DCF Clients have a default value before any DCF Records are written and may have special writing constraints like ‘Write Once’ or only allowing bits to be written from ‘1’ to ‘0’ or vice versa.

A list of DCF clients is shown in *Table 129*. (TEST DCF Record Only clients are not shown as these are not accessible by the user).

The DCF Records select the target DCF client via a 30-bit field in the DCF Record consisting of:

- A 15-bit Chip-Select field: each module that includes DCF clients is assigned a Chip Select during chip definition.
- A 15-bit Address field: the address field is only relevant to the address decoding within that module and may not necessarily relate to the address of a register visible to software.

9.3 DCF records

DCF records appear as contiguous double-word (64 bits) entries programmed in a reserved area of OTP UTEST Flash memory beginning at 0x0040_0300. Take care to write 128 bits which is the ECC granularity (complete with a dummy DCF if needed). Address alignment is also on 128 bits.

Warning: Over Program Protection will generate an unrecoverable ECC error if only one DCF record (64 bits) is written.

Caution: Once a stop record is detected, the SSCM will stop scanning for further records.

The structure of a DCF Record is shown in *Figure 45*.

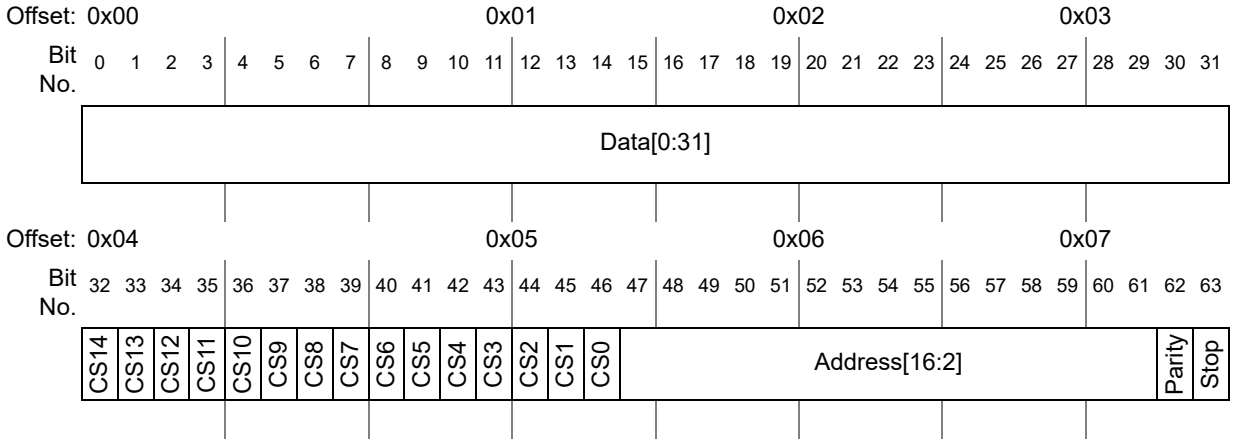


Figure 45. DCF record structure

Table 127. DCF record field description

Field	Description
0:31 Data[0:31]	32 bits of data to be written to the DCF Client.
32:46 CS*n*	Chip Select n. **Note:** Only assert one chip per DCF record to select the target module for the DCF client. All other Chip Selects should be negated. 1 Chip select is asserted. 0 Chip Select is negated.
47:61 Address[16:2]	Address of the DCF client within the selected module. **Note:** Address decoding for DCF clients may not match the standard software address map decoding. Details of DCF Client addresses are defined in each module chapter.
62 Parity	Parity Bit for the DCF Record. **Note:** This bit is NOT implemented for DCF Clients written from UTEST.
63 Stop	This bit indicates the end of the list of DCF Records. **Note:** The Erased state of Flash is 0xFFFF_FFFF_FFFF_FFFF. Therefore the list ends with the first unprogrammed double word. This location can be programmed with a new record to extend the list. 1 End of the list. 0 NOT the end of the list.

The following structure must be present:

- The first record must be a start record.
- DCF records containing configuration data must immediately follow the start record with no blank records between an unprogrammed record is interpreted as a stop record and no DCF records following that record are processed.
- The end of the configuration records are indicated by the presence of a bit in the DCF record (stop record).

Each record is 64 bits long. *Figure 46* shows the DCF start record.

Caution: The start record must be placed at the beginning of the DCF area in UTEST Flash memory to indicate that the following data records must be processed.

Figure 46. DCF start record

0x00 0:31	0x04 32:63
0x05AA55AF	0x00000000

During system boot, while the reset signal is asserted, the SSCM reads the device configuration records and writes the data to the appropriate internal module registers. Many operational aspects of the device are therefore already configured when the reset signal is released and normal device operation begins.

Some device configuration records are calculated during production testing and programmed into the TEST Flash memory area. Other DCF records are developed by the user and programmed into the UTEST Flash area.

The stop bit designates the last device configuration record in a set of records. If it is set to ‘1’ within a DCF, then this DCF is the last one being read during the reset phase. A DCF can have the stop bit set to ‘1’ in two modes:

a) The Flash memory location of a DCF is not programmed (that is all the bits are set to ‘1’).

b) A DCF is programmed and the stop bit is set to ‘1’.

When the device is shipped from the factors, some DCF records are already programmed and the Flash memory location after the last DCF is left unprogrammed to make it possible to add additional DCF records.

Due to the Flash programming memory resolution, which is 2-times the length of a DCF record, in case an odd number of DCF shall be written but additional DCF should be programmed later on, then a dummy DCF should be written to pad the last 64 bits of the programming data.

A dummy DCF can be one of the following:

a) The same DCF record written within the 128 bits (that is the DCF record is written two times).

b) A DCF record with an invalid Chips Select (refer to the *Table 129* for a list of used Chip Selects).

The padded DCF must have the stop bit set to ‘0’ in order to be able to add new DCF records.

The general format of the stop record is shown in *Figure 47*. Only the stop bit needs to be ‘1’ in order to form a stop record, all other bits are ignored. An unprogrammed location in UTEST Flash is interpreted as a stop record.

Figure 47. DCF stop record

0:31	32:62	63
Reserved	Reserved	1

If *n* data records are to be stored in UTEST, the data structure must be as shown in *Table 128*.

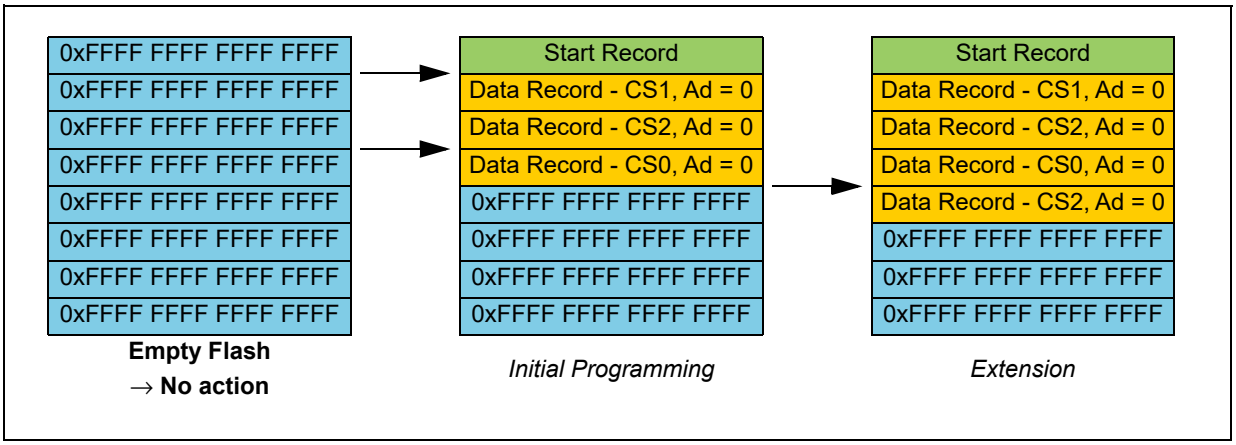
Table 128. Series of DCF records in UTEST Flash memory

ADDR offset	DATA			
0x00	0x05AA55AF			
0x04	0x00000000			STOP=0
0x08	WDATA[31:0]			
0x0C	CS[14:0]	ADDR[16:2]	PRTY	STOP=0
0x10	WDATA[31:0]			
0x14	CS[14:0]	ADDR[16:2]	PRTY	STOP=0
...	...			
8*n* + 0x0	Reserved			
8*n* + 0x4	Reserved			STOP=1
8(*n*+1) + 0x0	—			
8(*n*+1) + 0x4				

Caution: There must never be an unprogrammed record in the DCF data structure, as it is interpreted as a stop record and subsequent records are ignored.

Records programmed in several sessions is shown in *Figure 48*, appending new records to the end of the list each time.

Figure 48. Appending DCF records



It is possible for several DCF Records to write to the same DCF client. The later record usually overrides a DCF client value set by a previous record. However, not all DCF clients allow overwrites; this depends on the DCF client implementation.

9.3.1 UTEST DCF records

UTEST DCF records are located in the UTEST Flash memory area (refer to UTEST Flash memory map in the Memory map chapter). Some UTEST DCF records are programmed at the factory and the user may add USER DCF records from the first unprogrammed location following the factory programmed UTEST DCF records.

When programming UTEST DCF records, the records must start at the first address in the UTEST Flash memory area and be continuous: one UTEST DCF record must immediately follow the previous UTEST DCF record. An unprogrammed location in the UTEST Flash memory area is interpreted by the SSCM as the end of the UTEST DCF records. When this happens, the SSCM passes control of the boot sequence to the Reset Generation Module.

9.4 DCF client table

A list of DCF clients is shown in *Table 129*. Various attributes of the DCF clients are listed in these tables:

- DCF Chip Select [14:0]: the CS field is a 15-bit field with each bit acting as a module select signal for the internal modules. Only one bit in this field should be set in a DCF record.
- DCF address [16:2]: the 15-bit ADDR field specifies an internal address for a DCF client. These addresses are only used by the SSCM when interpreting DCF records and writing data to specified modules.
- DCF Client Description: the name of the DCF clients.
- Reset value of DCF client: the reset value of DCF client before being written by the SSCM with the associated DCF record.
- IPS read: this column specifies whether or not a DCF client can be read by user software after the release of reset.
- DCF client special strategy: registers are designated as DCF clients if they need to be written with a DCF record. Other features that must be designated for DCF clients include different write strategies listed below. The following list defines the special strategies:
  - None: no special DCF strategy is used.
  - Write Once: a DCF client can only be written once. The DCF client ignores subsequent writes.
  - Triple Voted: DCF clients have three copies of the register. The SSCM writes to all three registers in a single write cycle. The outputs of the three registers are majority voted together to determine the correct data value. Triple voting allows for ‘bit-flip’ error tolerance without changing the DCF client output data.
  - Write 0 only: a bit in a DCF client can only be written from a logic 1 to a logic 0. An attempt to write a logic 1 is ignored.
  - Write 1 only: a bit in a DCF client can only be written from a logic 0 to a logic 1. An attempt to write a logic 0 is ignored.

- DCF order dependency in Flash memory
  - DCF order dependency in Flash memory: this column details whether or not there are special conditions on the location of a particular DCF record in Flash memory.

9.4.1 DCF client list

Table 129. DCF client list

DCF CS [14:0] (Binary)	DCF Address [16:10] (Binary)	DCF Address [9:2] (Binary)	DCF client description In case of peripheral belonging to STANDBY domain, the system must trigger a destructive reset to reload the related DCF record	Reset value of DCF client	IPS Read direct access	DCF Client Special Strategy	DCF Order Dependency in Flash
SSCM							
000_0000_0000_0010	0000000	00000101	BAF Configuration (Bypass mode) (see *Figure 49* and *Table 130*)	0xF7FFC002	yes	None	None
STCU - Support for 7 LBIST partitions (NLBIST); 102 Memory Cuts (NMCUT); 32-bit PRPG; 64-bit MISR							
000_0000_0000_0100	0000000	00000010	STCU_SKC	0x00000000	yes	None	First record among STCU, Key1/Key2 DCF record sequence; Write Key2 every (1024-8) clock edges
000_0000_0000_0100	0000000	00000000	STCU_RUN	0x00000000	yes	None	None
000_0000_0000_0100	0000000	00000011	STCU_CFG	0x00000000	yes	None	None
000_0000_0000_0100	0000000	00000100	STCU_PLL_CFG	0x00000000	yes	None	None
000_0000_0000_0100	0000000	00000101	STCU_WDG	0x0000FFFF	yes	None	None
000_0000_0000_0100	0000000	00000111	STCU_CRCE	0x00000000	yes	None	None
000_0000_0000_0100	0000000	00001010	STCU_ERR_FM	0x00000000	yes	None	None
000_0000_0000_0100	0000000	00100001	STCU_MBUFM1[31:0]	0x00000000	yes	None	None
000_0000_0000_0100	0000000	00100010	STCU_MBUFM2[31:0]	0x00000000	yes	None	None
000_0000_0000_0100	0000000	00100011	STCU_MBUFM3[31:0]	0x00000000	yes	None	None
000_0000_0000_0100	0000000	00100100	STCU_MBUFM4[31:0]	0x00000000	yes	None	None

Table 129. DCF client list (continued)

DCF CS [14:0] (Binary)	DCF Address [16:10] (Binary)	DCF Address [9:2] (Binary)	DCF client description In case of peripheral belonging to STANDBY domain, the system must trigger a destructive reset to reload the related DCF record	Reset value of DCF client	IPS Read direct access	DCF Client Special Strategy	DCF Order Dependency in Flash
000_0000_0000_0100	0000000	01000000	STCU_LB_CTRL_0	0x00000000	yes	None	None
000_0000_0000_0100	0000000	01000001	STCU_LB_PCS_0	0x00000000	yes	None	None
000_0000_0000_0100	0000000	01000010	STCU_LB_PRPGL_0	0xFFFFFFFF	yes	None	None
000_0000_0000_0100	0000000	01000011	STCU_LB_PRPGH_0	0xFFFFFFFF	yes	None	None
000_0000_0000_0100	0000000	01000100	STCU_LB_MISREL_0	0xFFFFFFFF	yes	None	None
000_0000_0000_0100	0000000	01000101	STCU_LB_MISREH_0	0xFFFFFFFF	yes	None	None
000_0000_0000_0100	0000000	01001000	STCU_LB_MISRELSW_0	0xFFFFFFFF	yes	None	None
000_0000_0000_0100	0000000	01001001	STCU_LB_MISREHSW_0	0xFFFFFFFF	yes	None	None
000_0000_0000_0100	0000000	01010000	STCU_LB_CTRL_1	0x00000000	yes	None	None
000_0000_0000_0100	0000000	01010001	STCU_LB_PCS_1	0x00000000	yes	None	None
000_0000_0000_0100	0000000	01010010	STCU_LB_PRPGL_1	0xFFFFFFFF	yes	None	None
000_0000_0000_0100	0000000	01010011	STCU_LB_PRPGH_1	0xFFFFFFFF	yes	None	None
000_0000_0000_0100	0000000	01010100	STCU_LB_MISREL_1	0xFFFFFFFF	yes	None	None
000_0000_0000_0100	0000000	01010101	STCU_LB_MISREH_1	0xFFFFFFFF	yes	None	None
000_0000_0000_0100	0000000	01011000	STCU_LB_MISRELSW_1	0xFFFFFFFF	yes	None	None
000_0000_0000_0100	0000000	01011001	STCU_LB_MISREHSW_1	0xFFFFFFFF	yes	None	None
000_0000_0000_0100	0000000	01100000	STCU_LB_CTRL_2	0x00000000	yes	None	None
000_0000_0000_0100	0000000	01100001	STCU_LB_PCS_2	0x00000000	yes	None	None
000_0000_0000_0100	0000000	01100010	STCU_LB_PRPGL_2	0xFFFFFFFF	yes	None	None
000_0000_0000_0100	0000000	01100011	STCU_LB_PRPGH_2	0xFFFFFFFF	yes	None	None
000_0000_0000_0100	0000000	01100100	STCU_LB_MISREL_2	0xFFFFFFFF	yes	None	None
000_0000_0000_0100	0000000	01100101	STCU_LB_MISREH_2	0xFFFFFFFF	yes	None	None
000_0000_0000_0100	0000000	01101000	STCU_LB_MISRELSW_2	0xFFFFFFFF	yes	None	None
000_0000_0000_0100	0000000	01101001	STCU_LB_MISREHSW_2	0xFFFFFFFF	yes	None	None
000_0000_0000_0100	0000000	01110000	STCU_LB_CTRL_3	0x00000000	yes	None	None
000_0000_0000_0100	0000000	01110001	STCU_LB_PCS_3	0x00000000	yes	None	None
000_0000_0000_0100	0000000	01110010	STCU_LB_PRPGL_3	0xFFFFFFFF	yes	None	None
000_0000_0000_0100	0000000	01110011	STCU_LB_PRPGH_3	0xFFFFFFFF	yes	None	None
000_0000_0000_0100	0000000	01110100	STCU_LB_MISREL_3	0xFFFFFFFF	yes	None	None

Table 129. DCF client list (continued)

DCF CS [14:0] (Binary)	DCF Address [16:10] (Binary)	DCF Address [9:2] (Binary)	DCF client description In case of peripheral belonging to STANDBY domain, the system must trigger a destructive reset to reload the related DCF record	Reset value of DCF client	IPS Read direct access	DCF Client Special Strategy	DCF Order Dependency in Flash
000_0000_0000_0100	0000000	01110101	STCU_LB_MISREH_3	0xFFFFFFFF	yes	None	None
000_0000_0000_0100	0000000	01111000	STCU_LB_MISRELSW_3	0xFFFFFFFF	yes	None	None
000_0000_0000_0100	0000000	01111001	STCU_LB_MISREHSW_3	0xFFFFFFFF	yes	None	None
000_0000_0000_0100	0000000	10000000	STCU_LB_CTRL_4	0x00000000	yes	None	None
000_0000_0000_0100	0000000	10000001	STCU_LB_PCS_4	0x00000000	yes	None	None
000_0000_0000_0100	0000000	10000010	STCU_LB_PRPGL_4	0xFFFFFFFF	yes	None	None
000_0000_0000_0100	0000000	10000011	STCU_LB_PRPGH_4	0xFFFFFFFF	yes	None	None
000_0000_0000_0100	0000000	10000100	STCU_LB_MISREL_4	0xFFFFFFFF	yes	None	None
000_0000_0000_0100	0000000	10000101	STCU_LB_MISREH_4	0xFFFFFFFF	yes	None	None
000_0000_0000_0100	0000000	10001000	STCU_LB_MISRELSW_4	0xFFFFFFFF	yes	None	None
000_0000_0000_0100	0000000	10001001	STCU_LB_MISREHSW_4	0xFFFFFFFF	yes	None	None
000_0000_0000_0100	0000000	10010000	STCU_LB_CTRL_5	0x00000000	yes	None	None
000_0000_0000_0100	0000000	10010001	STCU_LB_PCS_5	0x00000000	yes	None	None
000_0000_0000_0100	0000000	10010010	STCU_LB_PRPGL_5	0xFFFFFFFF	yes	None	None
000_0000_0000_0100	0000000	10010011	STCU_LB_PRPGH_5	0xFFFFFFFF	yes	None	None
000_0000_0000_0100	0000000	10010100	STCU_LB_MISREL_5	0xFFFFFFFF	yes	None	None
000_0000_0000_0100	0000000	10010101	STCU_LB_MISREH_5	0xFFFFFFFF	yes	None	None
000_0000_0000_0100	0000000	10011000	STCU_LB_MISRELSW_5	0xFFFFFFFF	yes	None	None
000_0000_0000_0100	0000000	10011001	STCU_LB_MISREHSW_5	0xFFFFFFFF	yes	None	None
000_0000_0000_0100	0000000	10100000	STCU_LB_CTRL_6	0x00000000	yes	None	None
000_0000_0000_0100	0000000	10100001	STCU_LB_PCS_6	0x00000000	yes	None	None
000_0000_0000_0100	0000000	10100010	STCU_LB_PRPGL_6	0xFFFFFFFF	yes	None	None
000_0000_0000_0100	0000000	10100011	STCU_LB_PRPGH_6	0xFFFFFFFF	yes	None	None
000_0000_0000_0100	0000000	10100100	STCU_LB_MISREL_6	0xFFFFFFFF	yes	None	None
000_0000_0000_0100	0000000	10100101	STCU_LB_MISREH_6	0xFFFFFFFF	yes	None	None
000_0000_0000_0100	0000000	10101000	STCU_LB_MISRELSW_6	0xFFFFFFFF	yes	None	None
000_0000_0000_0100	0000000	10101001	STCU_LB_MISREHSW_6	0xFFFFFFFF	yes	None	None
000_0000_0000_0100	0000001	10000000	STCU_MB_CTRL_0	0x00000000	yes	None	None
000_0000_0000_0100	0000001	10000001	STCU_MB_CTRL_1	0x00000000	yes	None	None

Table 129. DCF client list (continued)

DCF CS [14:0] (Binary)	DCF Address [16:10] (Binary)	DCF Address [9:2] (Binary)	DCF client description In case of peripheral belonging to STANDBY domain, the system must trigger a destructive reset to reload the related DCF record	Reset value of DCF client	IPS Read direct access	DCF Client Special Strategy	DCF Order Dependency in Flash
000_0000_0000_0100	0000001	10000010	STCU_MB_CTRL_2	0x00000000	yes	None	None
000_0000_0000_0100	0000001	10000011	STCU_MB_CTRL_3	0x00000000	yes	None	None
000_0000_0000_0100	0000001	10000100	STCU_MB_CTRL_4	0x00000000	yes	None	None
000_0000_0000_0100	0000001	10000101	STCU_MB_CTRL_5	0x00000000	yes	None	None
000_0000_0000_0100	0000001	10000110	STCU_MB_CTRL_6	0x00000000	yes	None	None
000_0000_0000_0100	0000001	10000111	STCU_MB_CTRL_7	0x00000000	yes	None	None
000_0000_0000_0100	0000001	10001000	STCU_MB_CTRL_8	0x00000000	yes	None	None
000_0000_0000_0100	0000001	10001001	STCU_MB_CTRL_9	0x00000000	yes	None	None
000_0000_0000_0100	0000001	10001010	STCU_MB_CTRL_10	0x00000000	yes	None	None
000_0000_0000_0100	0000001	10001011	STCU_MB_CTRL_11	0x00000000	yes	None	None
000_0000_0000_0100	0000001	10001100	STCU_MB_CTRL_12	0x00000000	yes	None	None
000_0000_0000_0100	0000001	10001101	STCU_MB_CTRL_13	0x00000000	yes	None	None
000_0000_0000_0100	0000001	10001110	STCU_MB_CTRL_14	0x00000000	yes	None	None
000_0000_0000_0100	0000001	10001111	STCU_MB_CTRL_15	0x00000000	yes	None	None
000_0000_0000_0100	0000001	10010000	STCU_MB_CTRL_16	0x00000000	yes	None	None
000_0000_0000_0100	0000001	10010001	STCU_MB_CTRL_17	0x00000000	yes	None	None
000_0000_0000_0100	0000001	10010010	STCU_MB_CTRL_18	0x00000000	yes	None	None
000_0000_0000_0100	0000001	10010011	STCU_MB_CTRL_19	0x00000000	yes	None	None
000_0000_0000_0100	0000001	10010100	STCU_MB_CTRL_20	0x00000000	yes	None	None
000_0000_0000_0100	0000001	10010101	STCU_MB_CTRL_21	0x00000000	yes	None	None
000_0000_0000_0100	0000001	10010110	STCU_MB_CTRL_22	0x00000000	yes	None	None
000_0000_0000_0100	0000001	10010111	STCU_MB_CTRL_23	0x00000000	yes	None	None
000_0000_0000_0100	0000001	10011000	STCU_MB_CTRL_24	0x00000000	yes	None	None
000_0000_0000_0100	0000001	10011001	STCU_MB_CTRL_25	0x00000000	yes	None	None
000_0000_0000_0100	0000001	10011010	STCU_MB_CTRL_26	0x00000000	yes	None	None
000_0000_0000_0100	0000001	10011011	STCU_MB_CTRL_27	0x00000000	yes	None	None
000_0000_0000_0100	0000001	10011100	STCU_MB_CTRL_28	0x00000000	yes	None	None
000_0000_0000_0100	0000001	10011101	STCU_MB_CTRL_29	0x00000000	yes	None	None
000_0000_0000_0100	0000001	10011110	STCU_MB_CTRL_30	0x00000000	yes	None	None

Table 129. DCF client list (continued)

DCF CS [14:0] (Binary)	DCF Address [16:10] (Binary)	DCF Address [9:2] (Binary)	DCF client description In case of peripheral belonging to STANDBY domain, the system must trigger a destructive reset to reload the related DCF record	Reset value of DCF client	IPS Read direct access	DCF Client Special Strategy	DCF Order Dependency in Flash
000_0000_0000_0100	0000001	10011111	STCU_MB_CTRL_31	0x00000000	yes	None	None
000_0000_0000_0100	0000001	10100000	STCU_MB_CTRL_32	0x00000000	yes	None	None
000_0000_0000_0100	0000001	10100001	STCU_MB_CTRL_33	0x00000000	yes	None	None
000_0000_0000_0100	0000001	10100010	STCU_MB_CTRL_34	0x00000000	yes	None	None
000_0000_0000_0100	0000001	10100011	STCU_MB_CTRL_35	0x00000000	yes	None	None
000_0000_0000_0100	0000001	10100100	STCU_MB_CTRL_36	0x00000000	yes	None	None
000_0000_0000_0100	0000001	10100101	STCU_MB_CTRL_37	0x00000000	yes	None	None
000_0000_0000_0100	0000001	10100110	STCU_MB_CTRL_38	0x00000000	yes	None	None
000_0000_0000_0100	0000001	10100111	STCU_MB_CTRL_39	0x00000000	yes	None	None
000_0000_0000_0100	0000001	10101000	STCU_MB_CTRL_40	0x00000000	yes	None	None
000_0000_0000_0100	0000001	10101001	STCU_MB_CTRL_41	0x00000000	yes	None	None
000_0000_0000_0100	0000001	10101010	STCU_MB_CTRL_42	0x00000000	yes	None	None
000_0000_0000_0100	0000001	10101011	STCU_MB_CTRL_43	0x00000000	yes	None	None
000_0000_0000_0100	0000001	10101100	STCU_MB_CTRL_44	0x00000000	yes	None	None
000_0000_0000_0100	0000001	10101101	STCU_MB_CTRL_45	0x00000000	yes	None	None
000_0000_0000_0100	0000001	10101110	STCU_MB_CTRL_46	0x00000000	yes	None	None
000_0000_0000_0100	0000001	10101111	STCU_MB_CTRL_47	0x00000000	yes	None	None
000_0000_0000_0100	0000001	10110000	STCU_MB_CTRL_48	0x00000000	yes	None	None
000_0000_0000_0100	0000001	10110001	STCU_MB_CTRL_49	0x00000000	yes	None	None
000_0000_0000_0100	0000001	10110010	STCU_MB_CTRL_50	0x00000000	yes	None	None
000_0000_0000_0100	0000001	10110011	STCU_MB_CTRL_51	0x00000000	yes	None	None
000_0000_0000_0100	0000001	10110100	STCU_MB_CTRL_52	0x00000000	yes	None	None
000_0000_0000_0100	0000001	10110101	STCU_MB_CTRL_53	0x00000000	yes	None	None
000_0000_0000_0100	0000001	10110110	STCU_MB_CTRL_54	0x00000000	yes	None	None
000_0000_0000_0100	0000001	10110111	STCU_MB_CTRL_55	0x00000000	yes	None	None
000_0000_0000_0100	0000001	10111000	STCU_MB_CTRL_56	0x00000000	yes	None	None
000_0000_0000_0100	0000001	10111001	STCU_MB_CTRL_57	0x00000000	yes	None	None
000_0000_0000_0100	0000001	10111010	STCU_MB_CTRL_58	0x00000000	yes	None	None
000_0000_0000_0100	0000001	10111011	STCU_MB_CTRL_59	0x00000000	yes	None	None

Table 129. DCF client list (continued)

DCF CS [14:0] (Binary)	DCF Address [16:10] (Binary)	DCF Address [9:2] (Binary)	DCF client description In case of peripheral belonging to STANDBY domain, the system must trigger a destructive reset to reload the related DCF record	Reset value of DCF client	IPS Read direct access	DCF Client Special Strategy	DCF Order Dependency in Flash
000_0000_0000_0100	0000001	10111100	STCU_MB_CTRL_60	0x00000000	yes	None	None
000_0000_0000_0100	0000001	10111101	STCU_MB_CTRL_61	0x00000000	yes	None	None
000_0000_0000_0100	0000001	10111110	STCU_MB_CTRL_62	0x00000000	yes	None	None
000_0000_0000_0100	0000001	10111111	STCU_MB_CTRL_63	0x00000000	yes	None	None
000_0000_0000_0100	0000001	11000000	STCU_MB_CTRL_64	0x00000000	yes	None	None
000_0000_0000_0100	0000001	11000001	STCU_MB_CTRL_65	0x00000000	yes	None	None
000_0000_0000_0100	0000001	11000010	STCU_MB_CTRL_66	0x00000000	yes	None	None
000_0000_0000_0100	0000001	11000011	STCU_MB_CTRL_67	0x00000000	yes	None	None
000_0000_0000_0100	0000001	11000100	STCU_MB_CTRL_68	0x00000000	yes	None	None
000_0000_0000_0100	0000001	11000101	STCU_MB_CTRL_69	0x00000000	yes	None	None
000_0000_0000_0100	0000001	11000110	STCU_MB_CTRL_70	0x00000000	yes	None	None
000_0000_0000_0100	0000001	11000111	STCU_MB_CTRL_71	0x00000000	yes	None	None
000_0000_0000_0100	0000001	11001000	STCU_MB_CTRL_72	0x00000000	yes	None	None
000_0000_0000_0100	0000001	11001001	STCU_MB_CTRL_73	0x00000000	yes	None	None
000_0000_0000_0100	0000001	11001010	STCU_MB_CTRL_74	0x00000000	yes	None	None
000_0000_0000_0100	0000001	11001011	STCU_MB_CTRL_75	0x00000000	yes	None	None
000_0000_0000_0100	0000001	11001100	STCU_MB_CTRL_76	0x00000000	yes	None	None
000_0000_0000_0100	0000001	11001101	STCU_MB_CTRL_77	0x00000000	yes	None	None
000_0000_0000_0100	0000001	11001110	STCU_MB_CTRL_78	0x00000000	yes	None	None
000_0000_0000_0100	0000001	11001111	STCU_MB_CTRL_79	0x00000000	yes	None	None
000_0000_0000_0100	0000001	11010000	STCU_MB_CTRL_80	0x00000000	yes	None	None
000_0000_0000_0100	0000001	11010001	STCU_MB_CTRL_81	0x00000000	yes	None	None
000_0000_0000_0100	0000001	11010010	STCU_MB_CTRL_82	0x00000000	yes	None	None
000_0000_0000_0100	0000001	11010011	STCU_MB_CTRL_83	0x00000000	yes	None	None
000_0000_0000_0100	0000001	11010100	STCU_MB_CTRL_84	0x00000000	yes	None	None
000_0000_0000_0100	0000001	11010101	STCU_MB_CTRL_85	0x00000000	yes	None	None
000_0000_0000_0100	0000001	11010110	STCU_MB_CTRL_86	0x00000000	yes	None	None
000_0000_0000_0100	0000001	11010111	STCU_MB_CTRL_87	0x00000000	yes	None	None
000_0000_0000_0100	0000001	11011000	STCU_MB_CTRL_88	0x00000000	yes	None	None

Table 129. DCF client list (continued)

DCF CS [14:0] (Binary)	DCF Address [16:10] (Binary)	DCF Address [9:2] (Binary)	DCF client description In case of peripheral belonging to STANDBY domain, the system must trigger a destructive reset to reload the related DCF record	Reset value of DCF client	IPS Read direct access	DCF Client Special Strategy	DCF Order Dependency in Flash
000_0000_0000_0100	0000001	11011001	STCU_MB_CTRL_89	0x00000000	yes	None	None
000_0000_0000_0100	0000001	11011010	STCU_MB_CTRL_90	0x00000000	yes	None	None
000_0000_0000_0100	0000001	11011011	STCU_MB_CTRL_91	0x00000000	yes	None	None
000_0000_0000_0100	0000001	11011100	STCU_MB_CTRL_92	0x00000000	yes	None	None
000_0000_0000_0100	0000001	11011101	STCU_MB_CTRL_93	0x00000000	yes	None	None
000_0000_0000_0100	0000001	11011110	STCU_MB_CTRL_94	0x00000000	yes	None	None
000_0000_0000_0100	0000001	11011111	STCU_MB_CTRL_95	0x00000000	yes	None	None
000_0000_0000_0100	0000001	11100000	STCU_MB_CTRL_96	0x00000000	yes	None	None
000_0000_0000_0100	0000001	11100001	STCU_MB_CTRL_97	0x00000000	yes	None	None
000_0000_0000_0100	0000001	11100010	STCU_MB_CTRL_98	0x00000000	yes	None	None
000_0000_0000_0100	0000001	11100011	STCU_MB_CTRL_99	0x00000000	yes	None	None
000_0000_0000_0100	0000001	11100100	STCU_MB_CTRL_100	0x00000000	yes	None	None
000_0000_0000_0100	0000001	11100101	STCU_MB_CTRL_101	0x00000000	yes	None	None
PASS							
000_0000_0000_1000	0000000	00101000 -00101011	Reserved				
000_0000_0000_1000	0000000	00101100	Censorship	0x0	yes	None	None
000_0000_0000_1000	0000000	00101101	Password Slot Access	0x0	yes	wr_once	None
000_0000_0000_1000	0000000	00101110-00101111	Reserved				
000_0000_0000_1000	0000000	00110000	Production Disable	0x0	yes	wr_once	None
000_0000_0000_1000	0000000	00110001-00110111	Reserved				
000_0000_0000_1000	0000000	00111000	Fuse Bypass Enable	0x1	yes	wr_once	None
000_0000_0000_1000	0000000	00111001-00111111	Reserved				
000_0000_0000_1000	0000000	01000000	LOCK0_PG0	0xFFFFFFFF	no	None	None
000_0000_0000_1000	0000000	01000001	LOCK1_PG0	0xFFFFFFFF	no	None	None
000_0000_0000_1000	0000000	01000010	LOCK2_PG0	0xFFFFFFFF		None	None
000_0000_0000_1000	0000000	01000011	LOCK3_PG0	0xFFFFFFFF	no	None	None

Table 129. DCF client list (continued)

DCF CS [14:0] (Binary)	DCF Address [16:10] (Binary)	DCF Address [9:2] (Binary)	DCF client description In case of peripheral belonging to STANDBY domain, the system must trigger a destructive reset to reload the related DCF record	Reset value of DCF client	IPS Read direct access	DCF Client Special Strategy	DCF Order Dependency in Flash
000_0000_0000_1000	0000000	01000100	LOCK0_PG1	0xFFFFFFFF	no	None	None
000_0000_0000_1000	0000000	01000101	LOCK1_PG1	0xFFFFFFFF	no	None	None
000_0000_0000_1000	0000000	01000110	LOCK2_PG1	0xFFFFFFFF	no	None	None
000_0000_0000_1000	0000000	01000111	LOCK3_PG1	0xFFFFFFFF	no	None	None
000_0000_0000_1000	0000000	01001000	LOCK0_PG2	0xFFFFFFFF	no	None	None
000_0000_0000_1000	0000000	01001001	LOCK1_PG2	0xFFFFFFFF	no	None	None
000_0000_0000_1000	0000000	01001010	LOCK2_PG2	0xFFFFFFFF	no	None	None
000_0000_0000_1000	0000000	01001011	LOCK3_PG2	0xFFFFFFFF	no	None	None
000_0000_0000_1000	0000000	01001100	LOCK0_PG3	0xFFFFFFFF	no	None	None
000_0000_0000_1000	0000000	01001101	LOCK1_PG3	0xFFFFFFFF	no	None	None
000_0000_0000_1000	0000000	01001110	LOCK2_PG3	0xFFFFFFFF	no	None	None
000_0000_0000_1000	0000000	01001111	LOCK3_PG3	0xFFFFFFFF	no	None	None
000_0000_0000_1000	0000000	01001100-11111111	Reserved				
Tamper Detect							
000_0000_0001_0000	0000000	00000000	Diary Base Address	0xFFFFFFFF	no	wr_once + wr0_only	None
000_0000_0001_0000	0000000	00000001	Tamper Region Override	0x00	no	wr_once + wr1_only	None
000_0000_0001_0000	0000000	00000010	Software Tamper Region Override Disable	0x00	no	wr1_only	None
000_0000_0001_0000	0000000	00000011-00000111	Reserved				
000_0000_0001_0000	0000000	00001000	OTP_EN0	0x00000000	no	wr1_only	None
000_0000_0001_0000	0000000	00001001	OTP_EN1	0x00000000	no	wr1_only	None
000_0000_0001_0000	0000000	00001010	OTP_EN2	0x00000000	no	wr1_only	None
000_0000_0001_0000	0000000	00001011	OTP_EN3	0x00000000	no	wr1_only	None
000_0000_0001_0000	0000000	00001100-00010011	Reserved				

Table 129. DCF client list (continued)

DCF CS [14:0] (Binary)	DCF Address [16:10] (Binary)	DCF Address [9:2] (Binary)	DCF client description In case of peripheral belonging to STANDBY domain, the system must trigger a destructive reset to reload the related DCF record	Reset value of DCF client	IPS Read direct access	DCF Client Special Strategy	DCF Order Dependency in Flash
000_0000_0001_0000	0000000	00010100	TDR0_LOCK0	0x00000000	no	wr_once + wr1_only	None
000_0000_0001_0000	0000000	00010101	TDR0_LOCK1	0x00000000	no	wr_once + wr1_only	None
000_0000_0001_0000	0000000	00010110	TDR0_LOCK2	0x00000000	no	wr_once + wr1_only	None
000_0000_0001_0000	0000000	00010111	TDR0_LOCK3	0x00000000	no	wr_once + wr1_only	None
000_0000_0001_0000	0000000	00011000	TDR1_LOCK0	0x00000000	no	wr_once + wr1_only	None
000_0000_0001_0000	0000000	00011001	TDR1_LOCK1	0x00000000	no	wr_once + wr1_only	None
000_0000_0001_0000	0000000	00011010	TDR1_LOCK2	0x00000000	no	wr_once + wr1_only	None
000_0000_0001_0000	0000000	00011011	TDR1_LOCK3	0x00000000	no	wr_once + wr1_only	None
000_0000_0001_0000	0000000	00011100	TDR2_LOCK0	0x00000000	no	wr_once + wr1_only	None
000_0000_0001_0000	0000000	00011101	TDR2_LOCK1	0x00000000	no	wr_once + wr1_only	None
000_0000_0001_0000	0000000	00011110	TDR2_LOCK2	0x00000000	no	wr_once + wr1_only	None
000_0000_0001_0000	0000000	00011111	TDR2_LOCK3	0x00000000	no	wr_once + wr1_only	None

Table 129. DCF client list (continued)

DCF CS [14:0] (Binary)	DCF Address [16:10] (Binary)	DCF Address [9:2] (Binary)	DCF client description In case of peripheral belonging to STANDBY domain, the system must trigger a destructive reset to reload the related DCF record	Reset value of DCF client	IPS Read direct access	DCF Client Special Strategy	DCF Order Dependency in Flash
000_0000_0001_0000	0000000	00100000	TDR3_LOCK0	0x00000000	no	wr_once + wr1_only	None
000_0000_0001_0000	0000000	00100001	TDR3_LOCK1	0x00000000	no	wr_once + wr1_only	None
000_0000_0001_0000	0000000	00100010	TDR3_LOCK2	0x00000000	no	wr_once + wr1_only	None
000_0000_0001_0000	0000000	00100011	TDR3_LOCK3	0x00000000	no	wr_once + wr1_only	None
000_0000_0001_0000	0000000	00100100	TDR4_LOCK0	0x00000000	no	wr_once + wr1_only	None
000_0000_0001_0000	0000000	00100101	TDR4_LOCK1	0x00000000	no	wr_once + wr1_only	None
000_0000_0001_0000	0000000	00100110	TDR4_LOCK2	0x00000000	no	wr_once + wr1_only	None
000_0000_0001_0000	0000000	00100111	TDR4_LOCK3	0x00000000	no	wr_once + wr1_only	None
000_0000_0001_0000	0000000	00101000	TDR5_LOCK0	0x00000000	no	wr_once + wr1_only	None
000_0000_0001_0000	0000000	00101001	TDR5_LOCK1	0x00000000	no	wr_once + wr1_only	None
000_0000_0001_0000	0000000	00101010	TDR5_LOCK2	0x00000000	no	wr_once + wr1_only	None
000_0000_0001_0000	0000000	00101011	TDR5_LOCK3	0x00000000	no	wr_once + wr1_only	None
000_0000_0001_0000	0000000	00101100-11111111	Reserved				

Table 129. DCF client list (continued)

DCF CS [14:0] (Binary)	DCF Address [16:10]	DCF Address [9:2]	DCF client description In case of peripheral belonging to STANDBY domain, the system must trigger a destructive reset to reload the related DCF record	Reset value of DCF client	IPS Read direct access	DCF Client Special Strategy	DCF Order Dependency in Flash
MISCELLANEOUS							
000_0000_1000_0000	0000000	00000000	UTEST Miscellaneous (See *Figure 50* and *Table 131* for bit position details.)	0x00049E01	no	Triple Voted	None
000_0000_1000_0000	0000000	00000001	UTEST Miscellaneous_2 (See *Figure 51* and *Table 132* for bit position details.)	0x00000000	no	Triple Voted + wr_once	None
000_0000_1000_0000	0000000	00000010 -00000110	Reserved				
000_0000_1000_0000	0000000	00000111	JTAG Pin Configuration	0x00000001	no	None	None
000_0000_1000_0000	0000000	00001000 - 01001000	Reserved				
000_0000_1000_0000	0000000	01001001	PMC REE_BUS	0x00000000	no	Triple Voted	None
000_0000_1000_0000	0000000	01001010	PNREG_ENB (See PNREG_S register section in PMC Digital Interface chapter)	0x00000000	no	Triple Voted	None
000_0000_1000_0000	0000000	01001011- 11111111	Reserved				
000_0000_1000_0000	0000001 -1111111	00000000 -11111111	Reserved				
SOFT DCF CLIENTS (BAF)							
100_0000_0000_0000	0000000	00000000	Security Watchdog Control Register	NA			Refer to *Chapter 65: Boot Assist Flash (BAF)*

Table 129. DCF client list (continued)

DCF CS [14:0] (Binary)	DCF Address [16:10] (Binary)	DCF Address [9:2] (Binary)	DCF client description In case of peripheral belonging to STANDBY domain, the system must trigger a destructive reset to reload the related DCF record	Reset value of DCF client	IPS Read direct access	DCF Client Special Strategy	DCF Order Dependency in Flash
100_0000_0000_0000	0000000	00000001	Security Watchdog Timeout	NA			Refer to *Chapter 65: Boot Assist Flash (BAF)*
100_0000_0000_0000	0000000	00000010	Security Watchdog Service Address	NA			Refer to *Chapter 65: Boot Assist Flash (BAF)*
100_0000_0000_0000	0000000	00000011	Security Watchdog CPU select	NA			Refer to *Chapter 65: Boot Assist Flash (BAF)*
100_0000_0000_0000	0000000	00000100 -11111111	Reserved				
100_0000_0000_0000	0000001	00000000 -11111111	Reserved				
100_0000_0000_0000	0000010	00000000	Address	NA			None
100_0000_0000_0000	0000010	00000001	32-bit Data	NA			None
100_0000_0000_0000	0000010	00000010	16-bit Data	NA			None
100_0000_0000_0000	0000010	00000011	8-bit Data	NA			None
100_0000_0000_0000	0000010	00000100 -00001111	Reserved				
100_0000_0000_0000	0000010	00010000	Callback address	NA			None
100_0000_0000_0000	0000010	00010001	User Serial Boot	NA			
100_0000_0000_0000	0000010	00010010 - 00010100	Reserved				
100_0000_0000_0000	0000010	00010101	Boot CPU IVPR	NA			

Table 129. DCF client list (continued)

DCF CS [14:0] (Binary)	DCF Address [16:10] (Binary)	DCF Address [9:2] (Binary)	DCF client description In case of peripheral belonging to STANDBY domain, the system must trigger a destructive reset to reload the related DCF record	Reset value of DCF client	IPS Read direct access	DCF Client Special Strategy	DCF Order Dependency in Flash
100_0000_0000_0000	0000010	00010110-11111111	Reserved				
100_0000_0000_0000	0000011-1111111	00000000-11111111	Reserved				

Note: DCF programming should start from address 0x00400208.

9.4.2 BAF configuration DCF register

Offset: 00000000 Access: DCF Client

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	BAF_ADDRESS															
W																
Reset	1	1	1	1	0	1	1	1	1	1	1	1	1	1	1	1

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	BAF_ADDRESS														Reserved	BAF_BYPASS
W																
Reset	1	1	0	0	0	0	0	0	0	0	0	0	0	0	1	0

Figure 49. BAF configuration DCF Client

Table 130. BAF configuration DCF Client field descriptions

Field	Description
0:29 BAF_ADDRESS	BAF start address
31 BAF_BYPASS	BAF bypass 0 BAF is executed 1 BAF is bypassed

9.4.3 Miscellaneous DCF registers

Offset: 00000000 Access: DCF Client

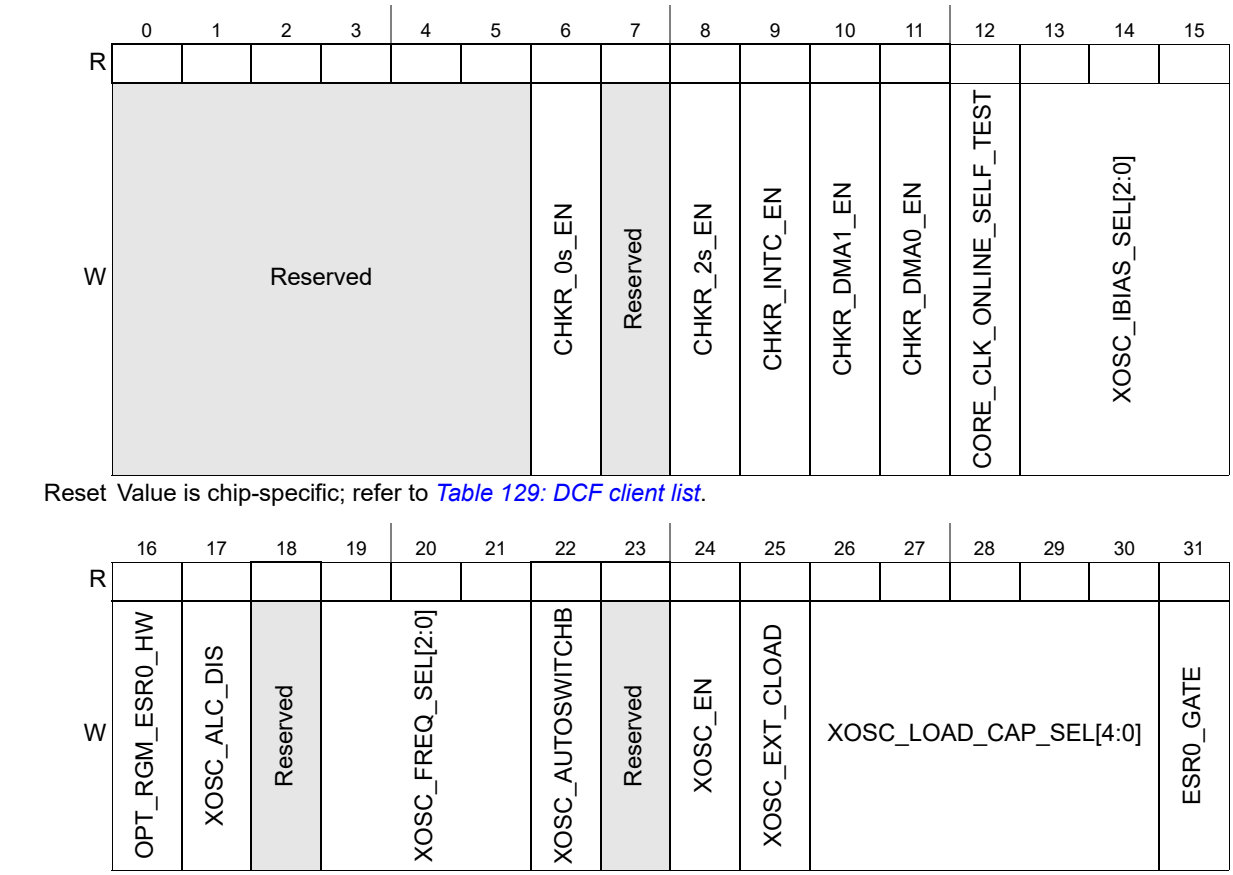


Figure 50. UTEST Miscellaneous DCF Client

Table 131. UTEST Miscellaneous DCF Client field descriptions

Field	Description
6 CHKR_0s_EN	Allows to enable/disable the checker core 0. 0 Checker disabled 1 (reset value) Checker enabled
8 CHKR_2s_EN	Allows to enable/disable the checker core 2. 0 Checker disabled 1 (reset value) Checker enabled
9 CHKR_INTC_EN	Allows to disable the checker INTC. 0 Checker disabled 1 Checker enabled **Note:** If any CHKR_x_EN is set, then CHKR_INTC_EN must be set to 1. If all CHKR_x_EN are disabled, then CHKR_INTC_EN must be set to 0.
10 CHKR_DMA1_EN	Allows to disable the checker DMA_1. 0 Checker disabled 1 Checker enabled

Table 131. UTEST Miscellaneous DCF Client field descriptions (continued)

Field	Description
11 CHKR_DMA0_EN	Allows to disable the checker DMA_0. 0 Checker disabled 1 Checker enabled
12 CORE_CLK_ONLINE_SELF_TEST	Core clock configuration during online self test. 0 Core Clock enabled. It allows to avoid RCCU alarms from checker cores during MBIST execution. 1 Core clock disabled during online LBIST.
13:15 XOSC_IBIAS_SEL[2:0]	Allows to change oscillation margin (default “100”) - User to keep default value of “100”
16 OPT_RGM_ESR0_HW	Active low to enable the SW control of release of ESR0. 0 Allows MC_RGM to continue to assert ESR0 until RGM_EROEC[EROEC] bit has been cleared by software 1 Does not allow MC_RGM to continue to assert ESR0 until RGM_EROEC[EROEC] bit has been cleared by software
17 XOSC_ALC_DIS	Active high to disable Automatic Level Controller
19:21 XOSC_FREQ_SEL[2:0]	This will select the different frequency settings of XOSC. 000 4 MHz – 8 MHz 001 5 MHz – 10 MHz 010 10 MHz – 15 MHz 011 15 MHz – 20 MHz 100 20 MHz – 25 MHz 101 25 MHz – 30 MHz 110 30 MHz – 35 MHz 111 35 MHz – 40 MHz
22 XOSC_AUTOSWITCHB	Allows to enable or disable the autoswitch 0 Reserved 1 Autoswitch mode disabled
24 XOSC_EN	Enable XOSC 0 XOSC disabled 1 XOSC enabled
25 XOSC_EXT_CLOAD	XOSC EXT CLOAD 0 Selects XOSC internal cap (recommended for 20 MHz – 40 MHz operation) 1 Selects XOSC external cap (recommended for 4 MHz – 20 MHz operation) **Note:** It is always recommended to verify the required size of the capacitances with the crystal supplier.

Table 131. UTEST Miscellaneous DCF Client field descriptions (continued)

Field	Description
26:30 XOSC_LOAD_CAP_SEL[0:4]	XOSC LOAD CAP SEL [0:4] Five trim bits program the load capacitance. The formula to calculate the capacitance offered is $C_{tot} = C_{fix}+n.C_u$ where: Fixed Capacitance C_{fix} = 11.0 pF (variation ±15 %) Unit capacitance C_u = 0.49 pF (variation ±15 %) n = load_cap_sel[4] * 2^4 + load_cap_sel[3] * 2^3 + load_cap_sel[2] * 2^2 + load_cap_sel[1] * 2^1 + load_cap_sel[0] * 2^0
31 ESR0_GATE	Gate ESR0 Input 0 ESR0 only asserted as an output from the device 1 Assertion of the ESR0 pin causes a hard system reset.

Offset: 00000001 Access: DCF Client

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R																
W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R																
W	Reserved													CFLA_ON_BYPASS_MC_ME	CFLA_ON_AFTER_STANDBY	BAF_EXEC_DURING_FA
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 51. UTEST2 Miscellaneous DCF Client

Table 132. UTEST2 Miscellaneous DCF Client field descriptions

Field	Description
29 CFLA_ON_BYPASS_MC_ME	1 Flash memory cannot be switched-off by MC_ME.CFLAON bits in RUNx modes 0 Flash memory can be switched-off by MC_ME.CFLAON bits in RUNx modes
30 CFLA_ON_AFTER_STANDBY	1 Flash memory will be turn-on after the STDBY exit 0 Flash memory can be switched (on/off) by MC_ME.CFLAON bits
31 BAF_EXEC_DURING_FA	1 BAF does not stop into an endless loop when LifeCycle = FA 0 BAF stops into an endless loop when LifeCycle = FA

Offset: 00000111 Access: DCF Client

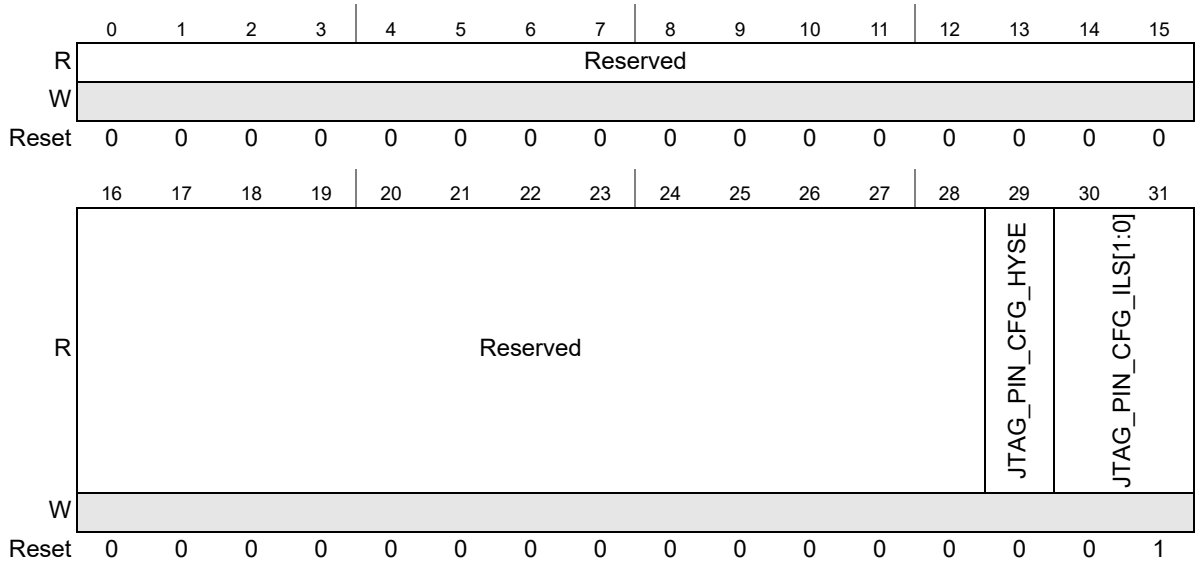


Figure 52. JTAG Pin Configuration

Table 133. JTAG Pin Configuration field descriptions

Field	Description
29 JTAG_PIN_CFG_HYSE	JTAG pins hysteresis 0 JTAG pins hysteresis not enabled. 1 JTAG pins hysteresis enabled. Default: JTAG pins hysteresis not enabled.
30:31 JTAG_PIN_CFG_ILS	JTAG pins mode 00 JTAG pins configured for AUTO mode. 01 JTAG pins configured for TTL mode. 10 JTAG pins configured for TTL mode. 11 JTAG pins configured for CMOS mode. Default: JTAG pins configured for TTL mode.

Offset: 01001001 Access: DCF Client

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0														LVD14_AD	LVD14_AS
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	LVD14_IF	LVD14_IJ	LVD14_IM	HVD13_IF	HVD13_IJ	HVD13_C	LVD11_AD	LVD11_AS	LVD11_IF	LVD11_IJ	LVD11_FL	LVD11_C	HVD6_C	LVD3_SB	LVD3F_L	LVD3_C
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 53. PMC_REE_BUS

Table 134. PMC_REE_BUS field descriptions

Field	Description
14 LVD14_AD	This bit defines whether an LVD assertion on the first supply of the SAR ADC generates a system reset. 0 Disabled. LVD assertion on the first supply of the SAR ADC does not cause system reset. 1 Enabled. LVD assertion on the first supply of the SAR ADC causes system reset.
15 LVD14_AS	This bit defines whether an LVD assertion on the first supply of the ADC generates a system reset. 0 Disabled. LVD assertion on the first supply of the ADC does not cause system reset. 1 Enabled. LVD assertion on the first supply of the ADC causes system reset.
16 LVD14_IF	This bit defines whether an LVD assertion on the first supply of the IO FlexRay/Ethernet generates a system reset. 0 Disabled. LVD assertion on the first supply of the IO FlexRay/Ethernet does not cause system reset. 1 Enabled. LVD assertion on the first supply of the IO FlexRay/Ethernet causes system reset.
17 LVD14_IJ	This bit defines whether an LVD assertion on the first supply of the IO JTAG generates a system reset. 0 Disabled. LVD assertion on the first supply of the IO JTAG does not cause system reset. 1 Enabled. LVD assertion on the first supply of the IO JTAG causes system reset.
18 LVD14_IM	This bit defines whether an LVD assertion on the first supply of the IO mains generates a system reset. 0 Disabled. LVD assertion on the first supply of the IO mains does not cause system reset. 1 Enabled. LVD assertion on the first supply of the IO mains causes system reset.

Table 134. PMC_REE_BUS field descriptions (continued)

Field	Description
19 HVD13_IF	This bit defines whether an HVD assertion on the first supply of the IO FlexRay/Ethernet generates a system reset. 0 Disabled. HVD assertion on the first supply of the IO FlexRay/Ethernet does not cause system reset. 1 Enabled. HVD assertion on the first supply of the IO FlexRay/Ethernet causes system reset.
20 HVD13_IJ	This bit defines whether an HVD assertion on the first supply of the IO JTAG generates a system reset. 0 Disabled. HVD assertion on the first supply of the IO JTAG does not cause system reset. 1 Enabled. HVD assertion on the first supply of the IO JTAG causes system reset.
21 HVD13_C	This bit defines whether an HVD assertion on the first supply of the core generates a system reset. 0 Disabled. HVD assertion on the first supply of the core does not cause system reset. 1 Enabled. HVD assertion on the first supply of the core causes system reset.
22 LVD11_AD	This bit defines whether an LVD assertion on the first supply of the Sigma Delta ADC generates a system reset. 0 Disabled. LVD assertion on the first supply of the Sigma Delta ADC does not cause system reset. 1 Enabled. LVD assertion on the first supply of the Sigma Delta ADC causes system reset.
23 LVD11_AS	This bit defines whether an LVD assertion on the first supply of the SARADC generates a system reset. 0 Disabled. LVD assertion on the first supply of the SARADC does not cause system reset. 1 Enabled. LVD assertion on the first supply of the SARADC causes system reset.
24 LVD11_IF	This bit defines whether an LVD assertion on the first supply of the IO FlexRay/Ethernet generates a system reset. 0 Disabled. LVD assertion on the first supply of the IO FlexRay/Ethernet does not cause system reset. 1 Enabled. LVD assertion on the first supply of the IO FlexRay/Ethernet causes system reset.
25 LVD11_IJ	This bit defines whether an LVD assertion on the first supply of the IO JTAG generates a system reset. 0 Disabled. LVD assertion on the first supply of the IO JTAG does not cause system reset. 1 Enabled. LVD assertion on the first supply of the IO JTAG causes system reset.
26 LVD11_FL	This bit defines whether an LVD assertion on the first supply of the Flash generates a system reset. 0 Disabled. LVD assertion on the first supply of the Flash does not cause system reset. 1 Enabled. LVD assertion on the first supply of the Flash causes system reset.

Table 134. PMC_REE_BUS field descriptions (continued)

Field	Description
27 LVD11_C	This bit defines whether an LVD assertion on the first supply of the core generates a system reset. 0 Disabled. LVD assertion on the first supply of the core does not cause system reset. 1 Enabled. LVD assertion on the first supply of the core causes system reset.
28 HVD6_C	This bit defines whether an HVD assertion on the first supply of the core generates a system reset. 0 Disabled. HVD assertion on the first supply of the core does not cause system reset. 1 Enabled. HVD assertion on the first supply of the core causes system reset.
29 LVD3_SB	This bit defines whether an LVD assertion on the first supply of the IO ring generates a system reset. 0 Disabled. LVD assertion on the first supply of the IO ring does not cause system reset. 1 Enabled. LVD assertion on the first supply of the IO ring causes system reset.
30 LVD3_FL	This bit defines whether an LVD assertion on the first supply of the Flash generates a system reset. 0 Disabled. LVD assertion on the first supply of the Flash does not cause system reset. 1 Enabled. LVD assertion on the first supply of the Flash causes system reset.
31 LVD3_C	This bit defines whether an LVD assertion on the first supply of the core generates a system reset. 0 Disabled. LVD assertion on the first supply of the core does not cause system reset. 1 Enabled. LVD assertion on the first supply of the core causes system reset.

Offset: 01001010 Access: DCF Client

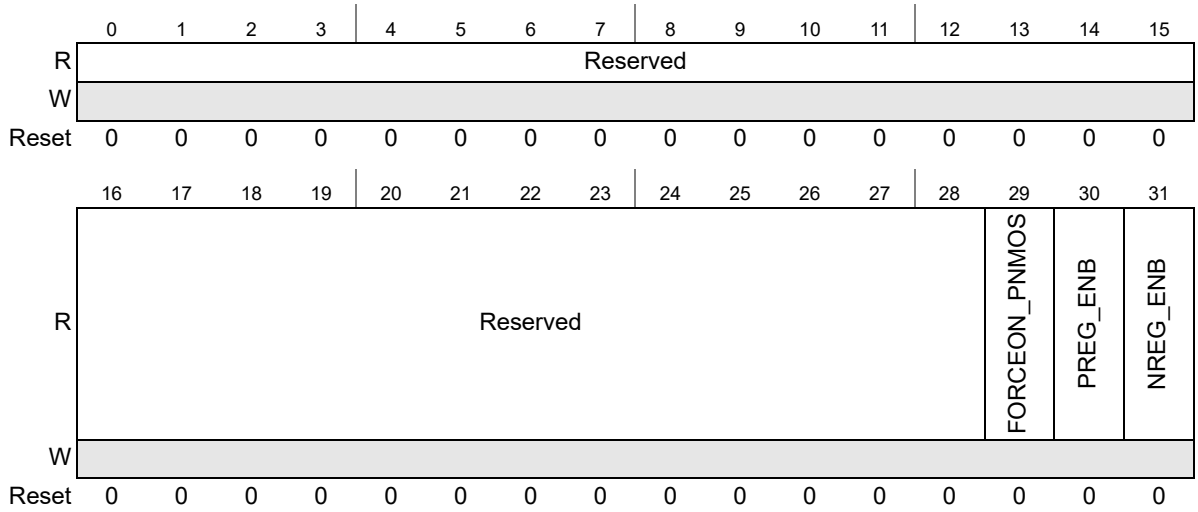


Figure 54. PNREG_ENB

Table 135. PNREG_ENB field descriptions

Field	Description
31 NREG_ENB	This bit is used to enable/disable of NREG. 0 NREG is enabled (depending on FORCEON_PMOS_NMOS value). 1 NREG regulator is off.
30 PREG_ENB	This bit is used to enable/disable of PREG. 0 PREG is enabled (depending on FORCEON_PMOS_NMOS value). 1 PREG is off.
29 FORCEON_PNMOS	This bit is used for enabling of PMOS and NMOS regulator. 0 Both PMOS and NMOS regulator are off. There is no effect of PREG_ENB and NREG_ENB. 1 Both can be switched on depending on enabling of PMOS and NMOS enables.

Offset: 000000000000000 Access: DCF Client

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	Reserved															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	Reserved														eFE	FTMPL
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0

Figure 55. Flash test mode protection

Table 136. Flash test mode protection field descriptions

Field	Description
30 eFE	0 Efuse system is disabled. 1 Efuse system is enabled. Default: Efuse system is enabled.
31 FTMPL	0 FTMP is not locked. 1 FTMP is locked. Default: FTMP is not locked.

10 Power management

10.1 Overview

SPC58xEx/SPC58xGx microcontrollers include a robust power management infrastructure that enables applications to select among various operational and low-power modes and to monitor internal voltages for high- and low-voltage conditions. The monitoring capability is also used to ensure supply voltages and internal voltages are within the required ranges before the microcontroller can leave reset. This chapter gives an overview of the built-in power management features.

The power management infrastructure comprises the following modules:

- Power Control Unit (MC_PCU)
- Clock Generation Module (MC_CGM)
- Mode Entry Module (MC_ME)
- Power Management Controller (PMC)
- Power Management Controller Digital Interface (PMC_DIG)
- Smart Standby Wake-up Unit (SSWU)

This chapter describes these modules in brief. For complete details, refer to the respective module chapters.

The MC_PCU provides a status register and a submodule, the PMC_dig (Power Management Control Digital Interface), provides an interface to configure and control internal voltages.

The MC_CGM generates reference clocks for all the chip blocks. It works in conjunction with internal clock gating logic to implement low power modes.

The MC_ME module controls chip operational modes and mode transition sequences. It also contains configuration, control and status registers accessible for the application.

The PMC module contains all the internal voltage regulator and monitors of the device : refer to *Section 10.1.3: Power management controller overview*.

The PMC_DIG module is the digital interface of the PMC, and contains all the registers to control the PMC during reset and power mode changes.

The SSWU Unit provides services to schedule sequence of tasks executed without CPU intervention. This is aimed to reduce further power consumption during standby.

The SPC58xEx/SPC58xGx has two types of modes, SYSTEM modes and USER modes.

SYSTEM modes:

- SYSTEM modes are entered automatically on system events or on application request.
- The SYSTEM modes are: RESET, SAFE, TEST and DRUN.

USER modes:

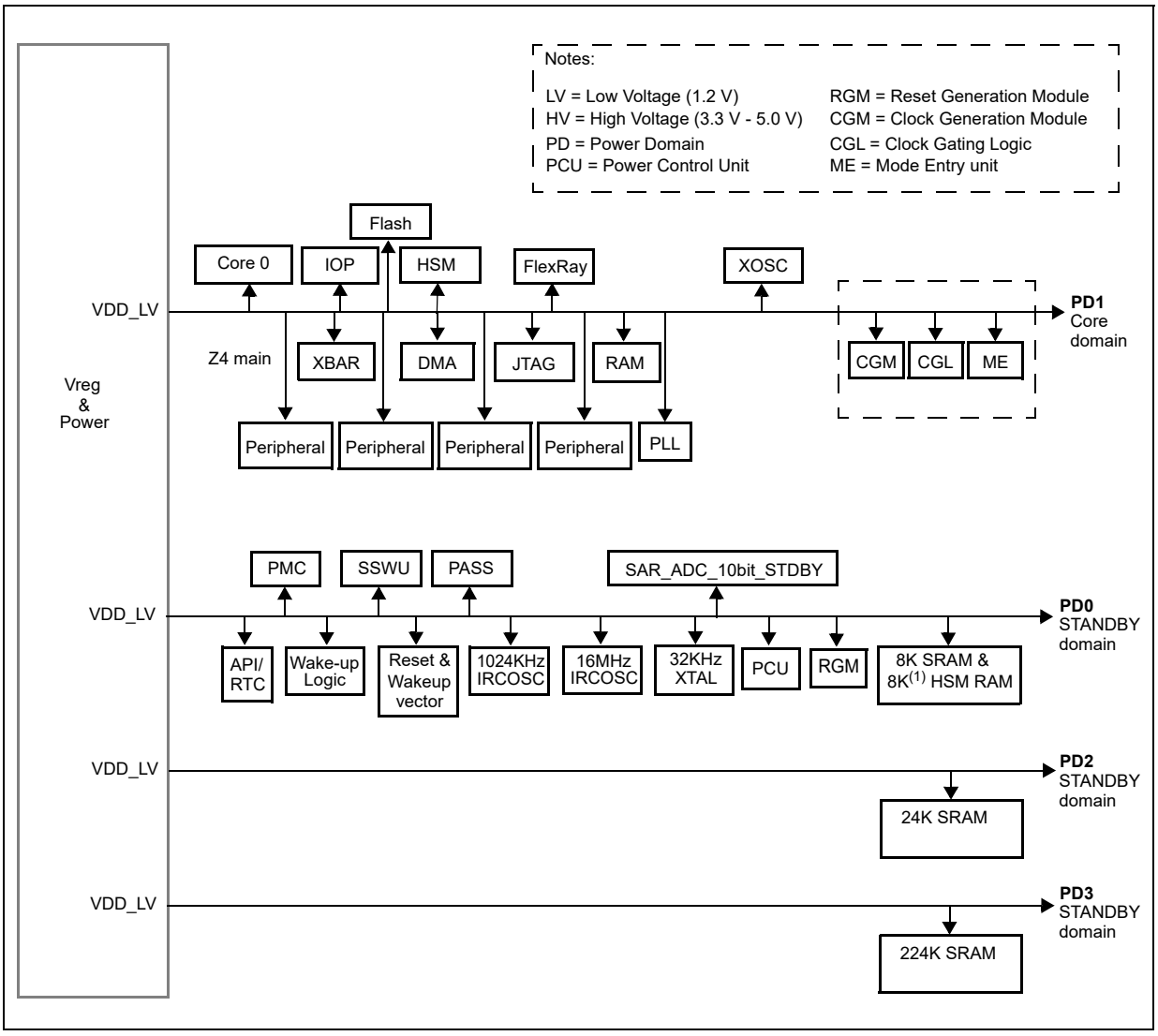
- USER modes control performance and consumption during normal application operation. The modes are organized according to the performance provided
- The USER modes are RUN0-3, HALT, STOP and STANDBY.
- Each mode is entered by writing a dedicated register protected by a key, to avoid unwanted transition

Please refer to the Magic Carpet Mode Entry Module chapter for modes details.

10.1.1 Power management framework

The power management framework supports a variety of configuration options. Refer to *Figure 56* for more details. SPC58xEx/SPC58xGx implements a unique core voltage power domain. There are no separate functional power domains.

Figure 56. Device power management framework



1. Enabled during stand-by depending on a DCF bit

10.1.2 Power management supply description

Table 137 provides an overview of the PMC supply signals.

Table 137. Device power management controller external signals

Name	Type	Description
VDD_HV_ADR_D	Reference	Voltage reference of ADC Σ/Δ module
VDD_HV_ADR_S	Reference	Voltage reference of ADC SAR module
VDD_HV_ADV	Supply	High voltage supply for the ADC modules
VDD_HV_FLA	Supply	Supply pin for Flash
VDD_HV_IO_MAIN	Supply	High voltage power supply for I/O
VDD_HV_IO_BD	Supply	High voltage power supply for buddy die I/O
VDD_HV_IO_FLEX	Supply	FlexRay/Ethernet I/O supply
VDD_HV_IO_JTAG	Supply	JTAG pin voltage supply
VDD_HV_OSC	Supply	Oscillator Voltage Supply
VDD_LV	Supply	External low voltage power supply for the core area
VDD_LV	Supply	Low voltage power supply for the core area generated by internal regulator
VDD_LV_BD	Supply	Low voltage power supply for the buddy device
VSS	Ground	Ground supply for the device / I/O. This is covering both VSS_LV and VSS_HV in case of exposed pad device.
VSS_HV_ADR_D	Reference	Ground reference of ADC Σ/Δ module
VSS_HV_ADR_S	Reference	Ground reference of ADC SAR module
VSS_HV_ADV	Supply	Ground supply for the ADC modules
VSS_HV_OSC	Ground	Oscillator Ground Supply

Caution: There are constraints on some input voltages relative to other input voltages. Refer to the device data sheet for detailed information.

10.1.3 Power management controller overview

The power management controller unit (PMC) contains all the device internal voltage regulators and all the voltage monitors. The supported regulators are:

- External regulator
  - External VDD_LV supply is provided from outside and no internal regulator is required.
- Internal regulator
  - An internal high power regulator (HPREG) using an external NPN bipolar as the ballast device (Refer to the SPC58xEx/SPC58xGx datasheet for the bipolar part number).
  - An internal Switching Mode Power Supply (SMPS) regulator, using external PMOS, NMOS and inductance.
- Low-power regulator
  - When the device is in STANDBY, the supply is provided by the internal LPREG and the HPREG is off, with a reduced maximum driving capability.
  - STANDBY mode is not supported in conjunction with External regulator mode.
- Auxiliary and clamp regulators
  - For fast current transients, auxiliary and clamp regulators can be enabled through DCF records (refer to the dedicated chapter) to improve the load regulation by compensating voltage overshoots and under-shoots. Auxiliary and clamp regulators cannot be enabled in SMPS regulator mode.

Please refer to the data sheet for the electrical characteristics of the regulators and the monitors and the list of external components required. The list and functionality of the monitors is described further in this chapter.

Figure 57 describes the top level block diagram of power management unit.

Figure 57. Power management controller block diagram overview

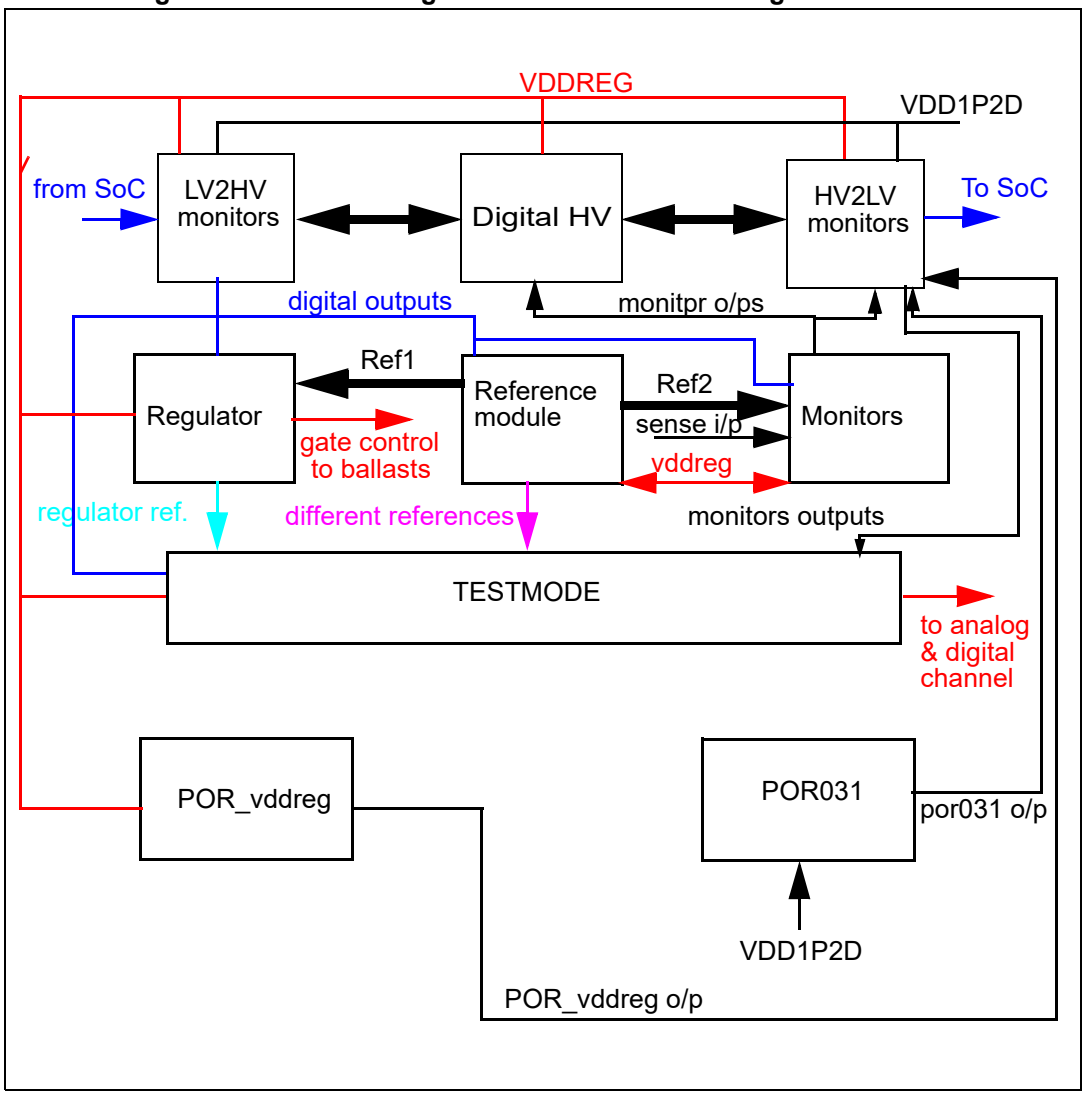
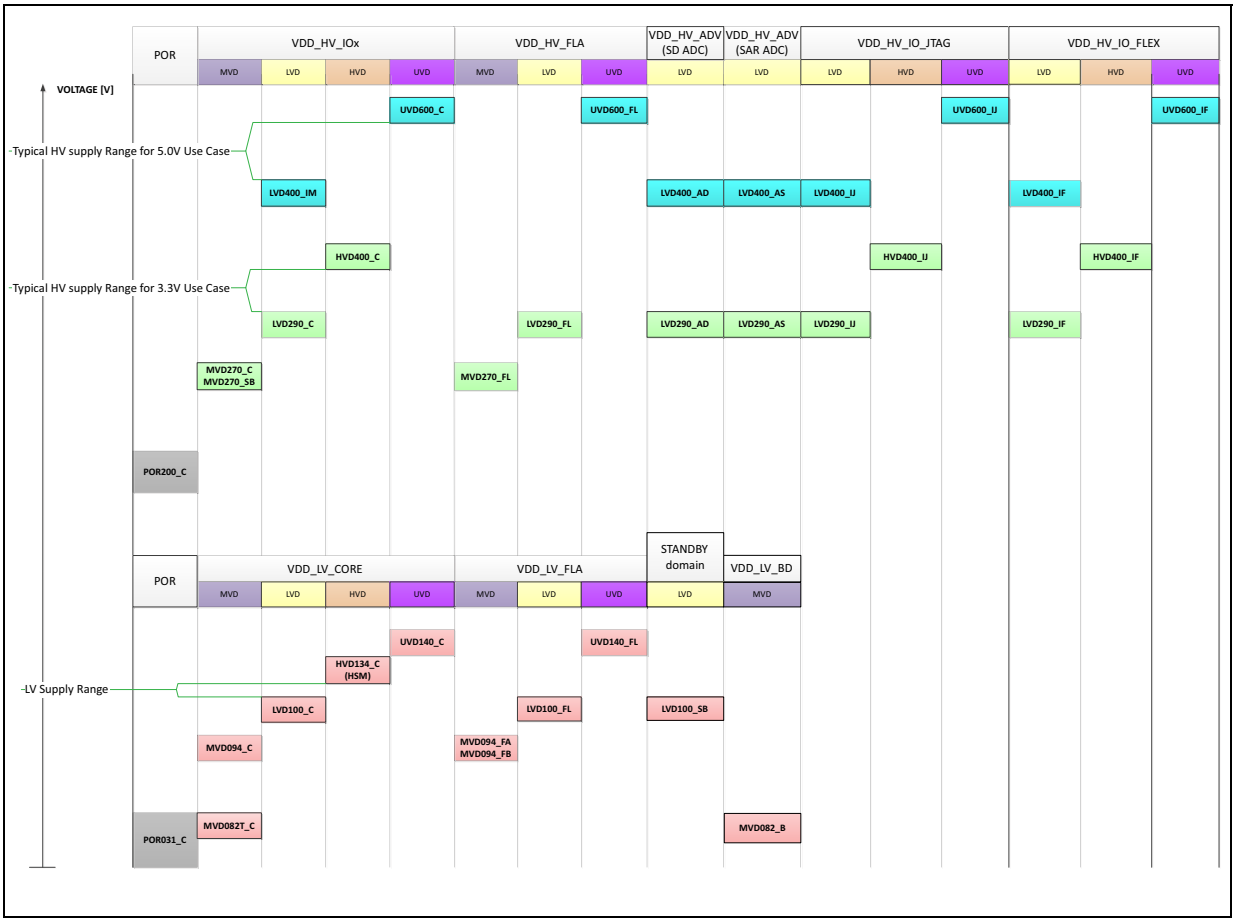


Figure 58. Voltage Monitor Coverage(1)(2)



1. Refer to *Table 138* for VDs descriptions
2. Voltage levels for all VDs are given in the data sheet

VDD_HV_IO is redundant, ensuring that single pin failure does not prevent device from functioning correctly. Full specification may not be guaranteed.

VDD_LV is redundant, ensuring that single pin failure does not prevent the device from functioning correctly.

10.1.3.1 1.2 V safety/spike regulator specification

SPC58xEx/SPC58xGx implements a 1.2 V safety/spike regulator. This regulator can be used to handle fast application current variation that cannot be fully handled by external regulator.

10.2 Low power mode support

The SPC58xEx/SPC58xGx microcontroller supports low power modes.

Dedicated logic prevents uncontrolled activation of low-power mode. This is implemented using protected sequence from ME.

Inadvertent entry into low power mode is prevented by triple voting of the control flops, both SW readable and otherwise.

10.2.1 Low power mode (HALT/STOP/STANDBY)

Low power mode may be used to control the ECU during the IDLE phase, as in a partial networking system or when the key has been turned off and periodic tasks, like contact monitoring, must be maintained.

Note: STANDBY mode behaves very differently from HALT/STOP modes. For more details, refer to Section 68.4.4.1: STANDBY Mode transition and Section 69.4.2.8: STANDBY0 Mode.

The device implements the equivalent of an idle or power-down mode through STANDBY, STOP and HALT mode configuration.

The entry into a low power mode is requested by the application code while the exit is triggered by a hardware event (internal, for instance RTC/API, or external, for instance Wake-up signal) or by the SSWU which detected a wake-up condition.

A combination of mode control and clock control (clock source selection but also peripheral clock gating) allows the user to configure the low power modes according to his application needs.

In STOP/HALT mode, peripheral modules can be enabled/disabled using the control described in the ME section. For example, it is possible to keep two CAN modules, eMIOS module and a few requested pins active. For alternate configurations, other modules can be enabled/disabled using the control described in the ME section. External interrupt request channels and external wake-up channels can be made active to wake-up the device in alternative to peripherals interrupts.

All pins associated to the active modules are functional.

Control of the pins is defined by user during run time. A wake-up line is associated to each selectable module.

One of the available modules can be configured to perform the wake-up task. For example, with all but one CAN modules on the chip set completely to power down mode and only one is configured for wake-up message receive (any of the modules can be configured to perform this task).

All CAN modules can be independently configured to receive messages and respond to wake-up messages.

Basically the low power modes allow to control the clock and the state of all peripherals (analog and digital) but also select the mode of the flash module as well as select and enable/disable clock sources like PLL or oscillator.

10.2.1.1 After-run implementation

After-run mode is implemented mainly making use of clock gating, module disabling, reduced frequency, mode switching.

Assuming that the selected CAN is kept enabled (clocked and active) then it can maintain its synchronization with the CAN network. Assuming also that the supplied CAN module can buffer the received (first) message, the CAN is then able to interrupt the IOP. The IOP would execute from TCM (such as, Flash is disabled) and use the internal RC oscillator (IRCOSC) or the external oscillator (XOSC). To support this would keep the PMC, LVD, temperature sensor, internal FIRC, XOSC and pads active. The PLL, Flash, and ADC would be disabled, with rest of part static and clock gated off.

10.2.1.2 Checker core disabling

The checker core is always in lock step with the main core: hence if the main core is clocked, the checker core is clocked (unless user intentionally disables checker clock, such as afterrun mode). Also if the main core is gated off, then the checker core is also gated off.

It is possible to disable the safety checker core to reduce current consumption.

The checker core can be placed in idle mode throughout the afterrun mode. By default the checker core mimics the main core, but it can be clock gated off during afterrun mode whilst continuing to clock the main core.

The checker core is implemented in such a way that its entire clock tree can be gated off while the core logic remains supplied.

The checker core (part of the Checker Logic) and the corresponding ‘checked main’ core share the same cache SRAM arrays and local SRAM arrays. Once the checker logic is disabled (such as,its clock is gated) the SRAM array of the cache and the local memory of the main core are still usable by the main core.

10.3 Flash power requirements

During start-up operation, the flash module operates from an untrimmed regulator, which has a slightly larger variation than the ideal case. During this start-up phase, the flash can still be read, but it must be read at a slower rate. While operating in this slower read mode, the flash can operate at a much reduced voltage.

Following completion of the start-up phase the regulator is in its trimmed condition, and subsequently the flash module reference current has also been trimmed. Therefore when entering normal operation the flash can be read in the normal access manner and support the full flash specification.

10.4 Device trimming

During the initialization phase the device defaults to a pre-determined state for each of the LVDs, HVDs and the internal regulators. As the flash becomes available the differential read process allows the trimmed data to be available for trimming the internal LVDs, HVDs and regulators.

Please refer to *Section 10.6* for further details.

10.5 Supply monitoring (POR and LVDs)

The function of the POR and LVD circuits is to hold the device in reset regardless of how slow the supply voltage rise is, until the point at which the POR and LVDs are released.

The POR and LVD circuits function correctly even if the input voltage is non monotonic.

10.5.1 Power-on reset (POR)

The PMC implements two internal power-on reset circuits:

- POR031_C monitors the voltage on the 1.2 V input supply. It is monitoring the VDD_LV_CORE pin. The POR031_C asserts a reset when the input supply is below defined values.
- POR200_C monitors the voltage on the 5.0 V - 3.3 V input supply. It is monitoring the VDD_HV_IO_MAIN pin. The POR trip point is high enough to make sure all the LVD circuits are functional.

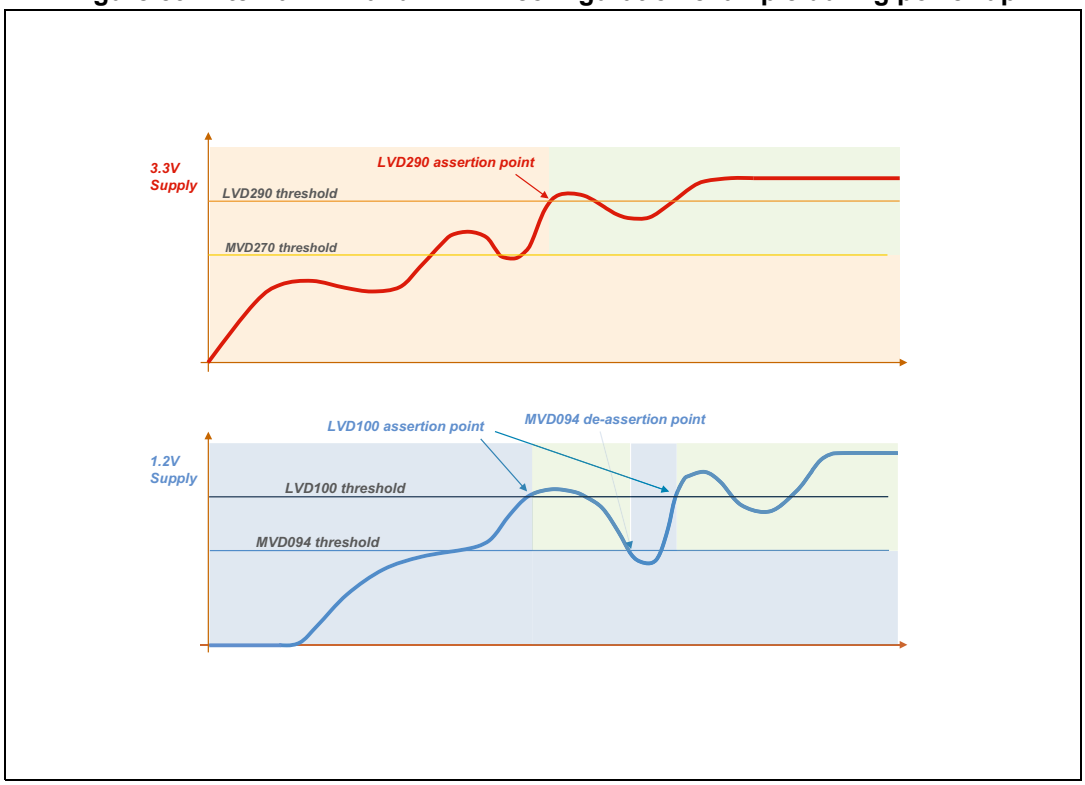
Refer to *Chapter 8: Reset and Boot* for PORST/ESR0 pin functionality.

10.5.2 Behavior of device LVD / HVD

Although there is an option to disable the LVD and HVD following reset (refer to *Section 10.5.3*), they are capable of being used in a 'monitor' only mode and also capable of generating a safe / interrupt event. *Figure 59* illustrates the behavior of these internal LVD / HVD circuits during the power on phase, through device initialization and subsequently in 'normal' run / operating mode.

The LVDs/HVDs can also be configured after device initialization preventing reset to happen when supply crosses the LVD threshold, effectively providing a higher voltage operating range. It is the responsibility of the application to ensure that the device remains in the functional range.

Figure 59. Internal LVD and HVD in configuration example during power-up



10.5.3 Voltage detections (MVDs, LVDs, HVDs, UVDs)

The internal LVD circuits, monitor when the voltage on the corresponding supply is below the defined values and either assert a reset or an interrupt. The LVDs also support hysteresis in the falling and rising trip points.

10.5.3.1 LVD and HVD implementation

- All LVDs and HVDs are capable of generating either an RGM and/or FCCU event and/or interrupt event.
- All LVDs and HVD configured for reset generation can cause functional or destructive reset. MC_RGM PHASE0 is not exited until all destructive reset conditions are cleared.
- The appropriate bits in the PMC registers are set by LVD and HVD events. Please refer to Power management controler digital interface chapter (PMC_Dig).
- The LVDs and HVDs control is protected by the register protection scheme. Therefore it is configurable as long as the scheme is followed. Refer to *Chapter 91: Register Protection (REG_PROT)*.
- SPC58xEx/SPC58xGx implements REE (Reset event enable) DCF records to allow to associate “configurable” LVDs/HVDs to a RESET event. This is a write mechanism managed by SSCM during device initialization. When RESET event is selected through REE DCF record, it cannot be disabled by software programming of PMC REE register. When RESET event is not selected through REE DCF record, LVDs/HVDs trigger event reaction depends on the programming of the PMC REE/RES/IE/FEE registers. Refer to *Chapter 9: Device Configuration Format (DCF) Records*.
- When the LVD or the HVD is enabled for destructive reset generation and a subsequent trigger event is detected, the external PORST pin is driven low.

Note: The full list of voltage monitors is provided in Table 138. The functionality and configurability are reported in Figure 60.

Note: MVDs/UVDs are not configurable.

Table 138. Voltage monitors description

Monitor name	Voltage reference number	Description	Monitor sensing
POWER MANAGEMENT UNIT VOLTAGE MONITORS: LOW VOLTAGE SUPPLY			
POR031_C	—	Low voltage supply power-on reset voltage monitor	
MVD082T_C	VD1	LV supply core minimum voltage detector	VDD_LV_CORE
MVD082_B	VD1	LV supply for buddy device minimum voltage detector	VDD_LV_BD
MVD094_C	VD2	LV supply core minimum voltage detector	VDD_LV_CORE
MVD094_FA	VD2	LV supply flash minimum voltage detector	VDD_LV_FLA
MVD094_FB	VD2	LV supply flash minimum voltage monitor	VDD_LV_FLA
LVD100_C	VD3	LV supply core low voltage detector	VDD_LV_CORE
LVD100_FL	VD3	LV supply flash low voltage detector	VDD_LV_FLA
LVD100_SB	VD3	LV supply STANDBY low voltage detector	VDD_LV_STBY

Table 138. Voltage monitors description (continued)

Monitor name	Voltage reference number	Description	Monitor sensing
HVD134_C	VD6	LV supply core high voltage detector	VDD_LV_CORE
UVD140_C	VD7	LV supply core upper voltage detector	VDD_LV_CORE
UVD140_FL	VD7	LV supply flash upper voltage detector	VDD_LV_FLA
POWER MANAGEMENT UNIT VOLTAGE MONITORS: HIGH VOLTAGE SUPPLY			
POR200_C	—	High voltage supply power-on reset voltage monitor	
MVD270_C	VD10	HV supply core minimum voltage monitor	VDD_HV_IOx
MVD270_FL	VD10	HV supply flash minimum voltage monitor	VDD_HV_FLA
MVD270_SB	VD10	HV supply STANDBY minimum voltage monitor	VDD_HV_IOx (in Standby)
LVD290_C	VD11	HV supply core low voltage monitor	VDD_HV_IOx
LVD290_FL	VD11	HV supply flash low voltage monitor	VDD_HV_FLA
LVD290_AD	VD11	HV supply ADC low voltage monitor	VDD_HV_ADV (SD ADC)
LVD290_AS	VD11	HV supply ADC low voltage monitor	VDD_HV_ADV (SAR ADC)
LVD290_IJ	VD11	HV supply I/O low voltage monitor	VDD_HV_IO_JTAG
LVD290_IF	VD11	HV supply I/O low voltage monitor	VDD_HV_IO_FLEX
HVD400_C	VD13	HV supply core high voltage monitor	VDD_HV_IOx
HVD400_IF	VD13	HV supply I/O high voltage monitor	VDD_HV_IO_FLEX
HVD400_IJ	VD13	HV supply I/O high voltage monitor	VDD_HV_IO_JTAG
LVD400_IM	VD14	HV supply core low voltage monitor	VDD_HV_IOx
LVD400_AD	VD14	HV supply ADC range low voltage detector	VDD_HV_ADV (SD ADC)
LVD400_AS	VD14	HV supply ADC low voltage detector	VDD_HV_ADV (SAR ADC)
LVD400_IF	VD14	HV supply I/O main low voltage detector	VDD_HV_IO_FLEX
LVD400_IJ	VD14	HV supply I/O main low voltage detector	VDD_HV_IO_JTAG
UVD600_C	VD15	HV supply core upper voltage monitor	VDD_HV_IOx
UVD600_FL	VD15	HV supply flash upper voltage monitor	VDD_HV_IO_FLASH
UVD600_IF	VD15	HV supply I/O upper voltage monitor	VDD_HV_IO_FLEX
UVD600_IJ	VD15	HV supply I/O upper voltage monitor	VDD_HV_IO_JTAG

Figure 60 provides the functionality of monitors depending on configuration.

Figure 60. Voltage monitors functionality

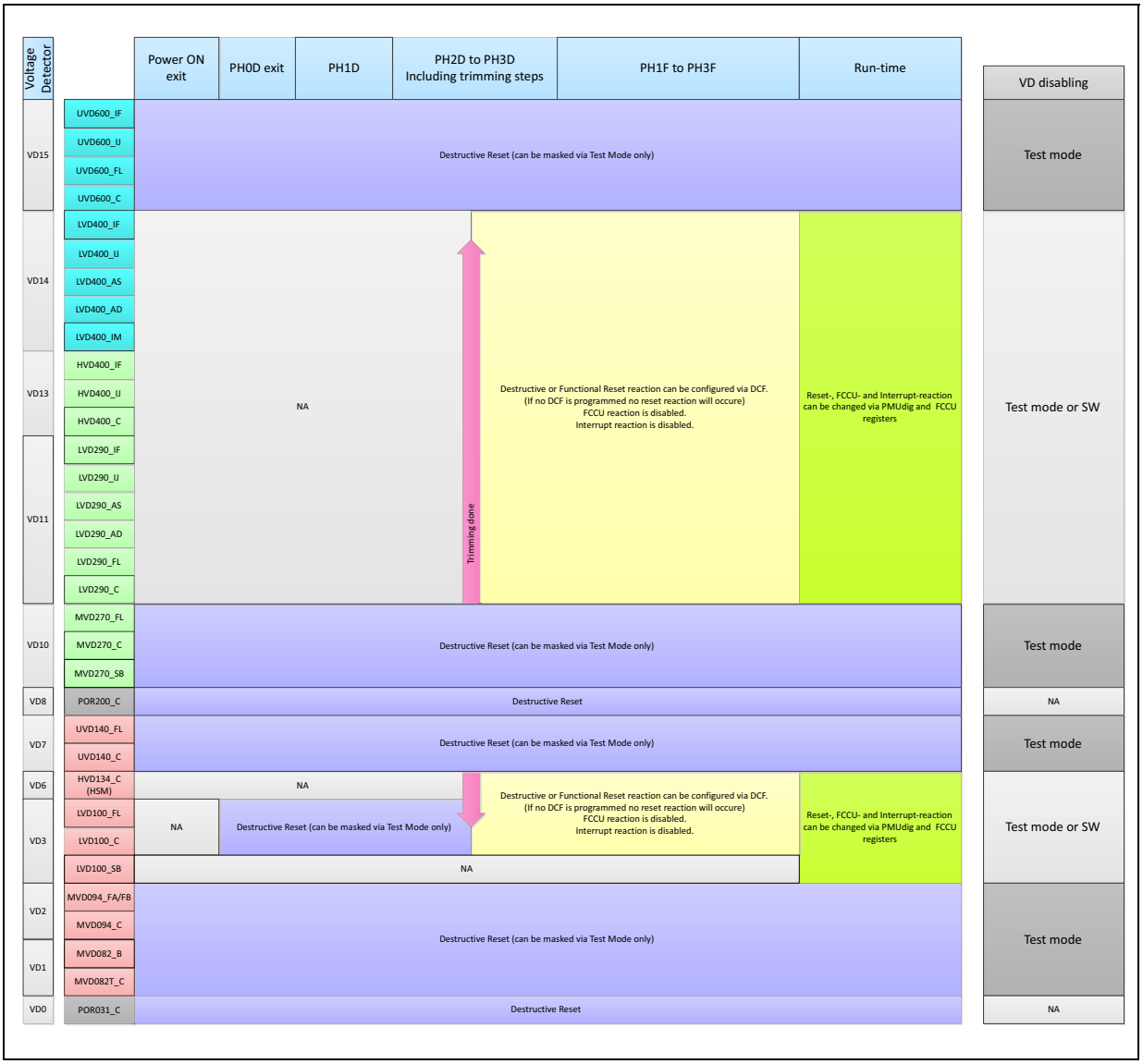


Table 139 provides the monitor status depending on configuration.

Table 139. Voltage monitors configurability

Monitor type	Reset Event Enable(1)	Reset Event Select(1)	Event Pending Register(1)	Interrupt Enable(1)	FCCU Event Enable(1)	Reset Event Enable DCF(1)
MVDs	No	No	No	No	No	No
LVDs	Yes	Yes	Yes	Yes	Yes	Yes
HVDs	Yes	Yes	Yes	Yes	Yes	Yes
UVDs	No	No	No	No	No	No

1. Refer to the Power management controller digital interface (PMC_Dig) chapter.

When using an external regulator for low voltage generation, the voltage regulator system shall have an internal LVD to satisfy the safety requirement.

10.6 Power sequence

The following sections describe the power sequence and the relation between the different supplies during power-up and power-down.

The device is considered to be in a power sequence (or POWERUP state) when it is either not supplied or partially supplied. An internal power-on signal is used to identify POWERUP state. This signal is released high on exit of power sequence. The power-on signal is a combination of the LVDs monitoring the following supplies:

- VDD_LV
- VDD_HV_IO
- VDD_HV_FLA

The actual threshold use for each LVD is dependent on the configuration of the device. This is configurable by hardware (flash option bits content) or by software (LVD event configuration through register interface). Once power-on signal has been asserted, device configuration is reset to default power-up configuration.

10.6.1 Power-up sequence

In this section it is assumed that all supplies are low when entering the power-up sequence. Brown-out and power down sequences are managed in specific sections.

10.6.1.1 POWERUP state

At the beginning of power-up, the internal power-on signal remains low due to internal parasitics diodes. As soon as minimum threshold is reached on VDD_LV and VDD_HV supply, the internal power-on signal is forced low until power-up LVDs reach upper not trimmed threshold (refer to SPC58xEx/SPC58xGx datasheet).

The different supplies can rise independently as long as the constraints described in the SPC58xEx/SPC58xGx datasheet are met.

During power-up, all functional terminals are maintained into a known state as described in the SPC58xEx/SPC58xGx datasheet.

10.6.1.2 POWERUP exit

The POWERUP state can be exited when both VDD_LV and VDD_HV are above the internal LVD thresholds.

10.6.1.2.1 VDD_LV conditions and LVD trimming/enabling sequence

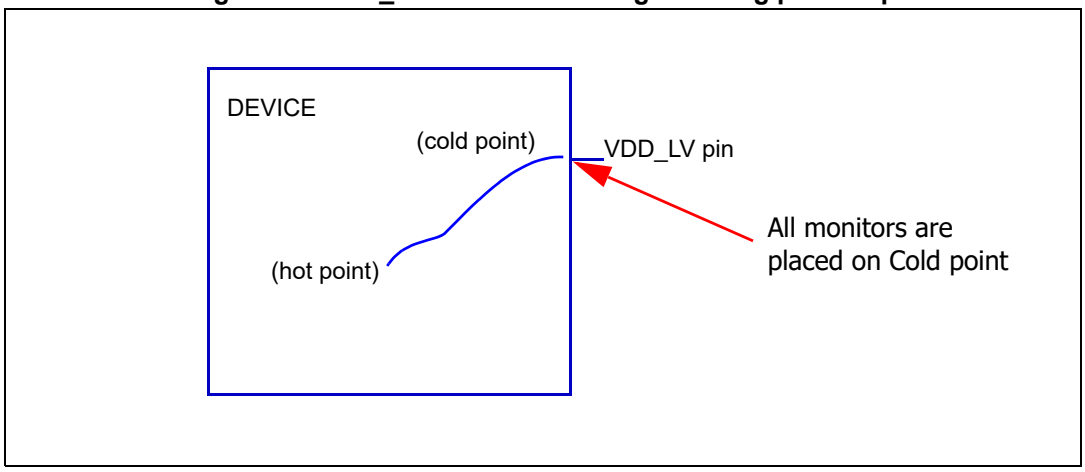
During POWERUP, only the following low voltage VDs are enabled:

- MVD082T_C , MVD094_C , MVD094_FA, MVD094_FB , UVD140_C , UDV140_FL

The VDD_LV conditions to exit POWERUP are the following:

- MVD094_C, MVD094_FA, MVD094_FB upper threshold is crossed

Figure 61. VDD_LV monitored voltages during power-up



After the POWERUP exit conditions (low voltage, high voltage conditions) have been verified, the internal power-on signal is released to all analog modules.

The internal RC oscillator module (IRCOSC) starts initialization and provides clock to the system after $t_{RCSTARTUP}$. PMC digital interface reset is released after two RC clock cycles and LVD100_C is masked. Only MVD094_C remains active.

MVD094_C and LVD100_C share the same reference. This difference provides margin with respect to maximum internal resistive drop and hysteresis addressing external supply.

The device proceeds through the reset sequence through RGM phase PHASE0, PHASE1[DEST], PHASE2[DEST] and PHASE3[DEST].

Voltage detector (LVD/HVD) modules are trimmed at the beginning of PHASE3[DEST]. Trimming of LVDs/HVDs are stored as internal DCF records. They are read by SSCM at low voltage, using the differential flash read accesses and applied to each LVDs/HVDs module. After analog delay ($t_{LVDTRIM}$) has elapsed, the PMC acknowledges that all LVDs/HVDs have been trimmed and supply is above threshold, the SSCM proceeds with the reset sequence, eventually running full speed accesses to the flash to read the option bits required to complete device configuration.

The configurable LVD/HVD modules can optionally be unmasked at the end of PHASE3[DEST]. Mask information is read as DCF record from the flash UTEST option bits.

When LVD100_C and LVD290_C are masked by the application using the flash user option bits, the device should rely on PORST signal to detect a voltage failure during power-up. The device must wait for PORST to be released high before proceeding with the power-up sequence.This may increase the amount of time necessary to complete the reset sequence.

Figure 62 provides an example of VDD_LV and VDD_HV power-up sequence before and after the monitor trimming.

Figure 63 provides an example of VDD_LV power-up sequence before and after the monitor trimming including UVD and HVD monitors.

Figure 62. Example of VDD_LV power-up sequencing

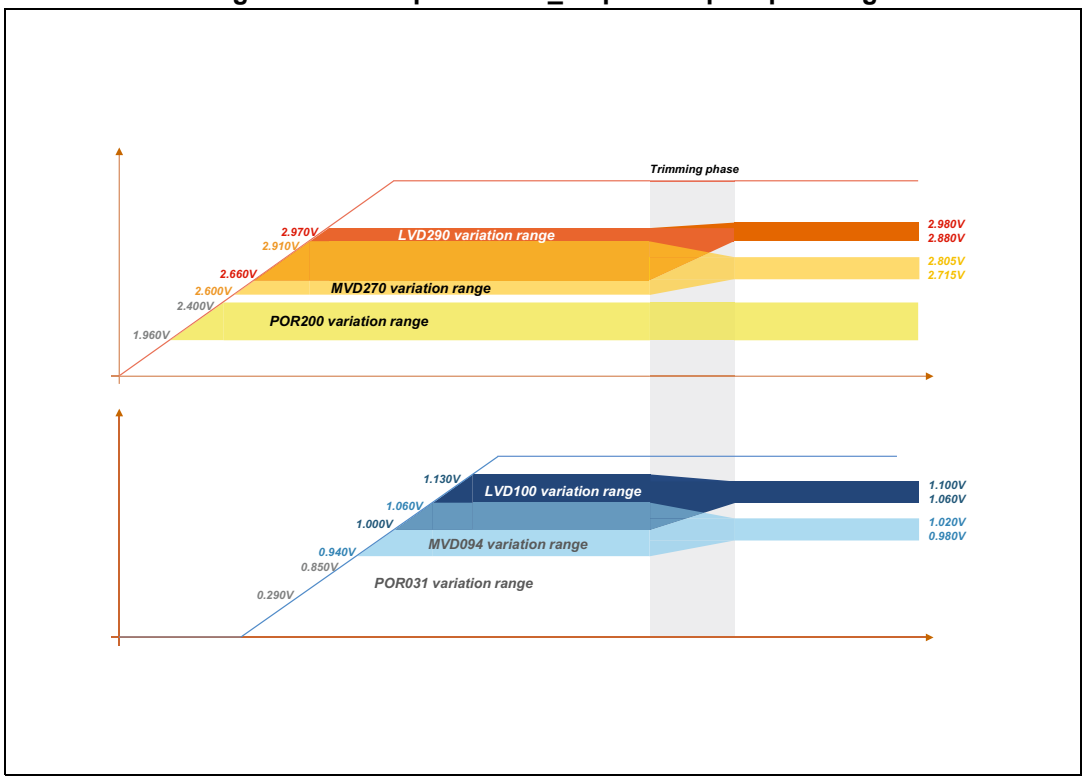
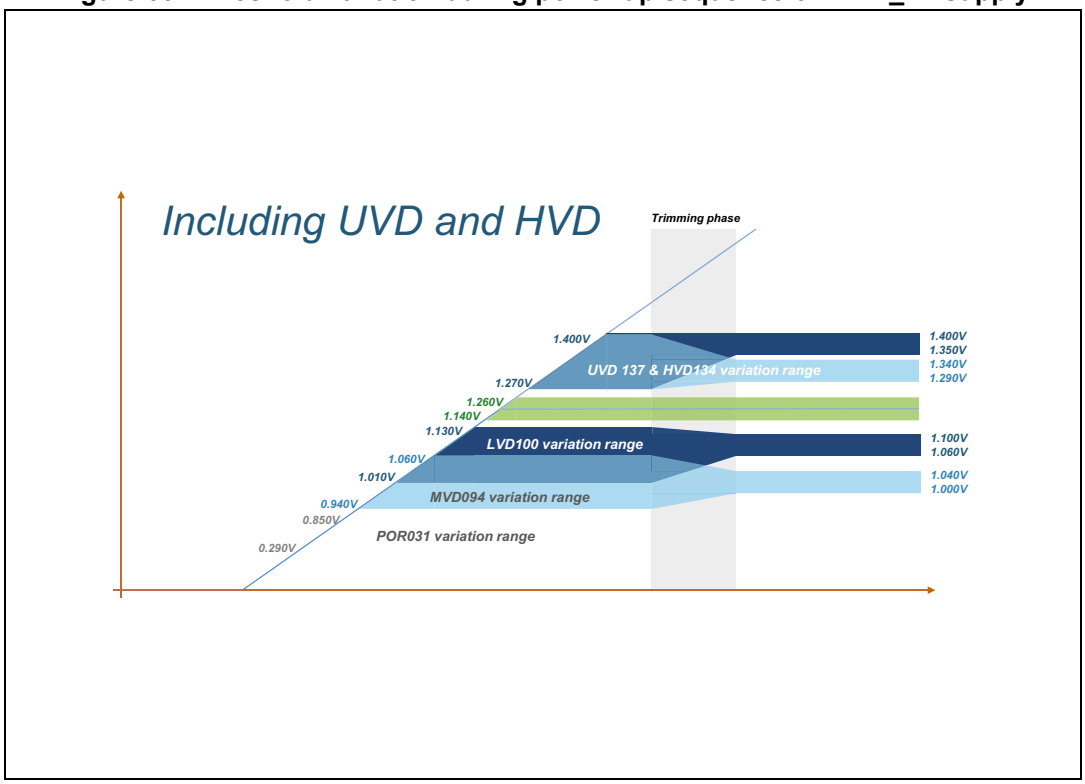


Figure 63. Threshold variation during power-up sequence on VDD_LV supply



10.6.1.2.2 VDD_HV conditions and LVD trimming/enabling sequence

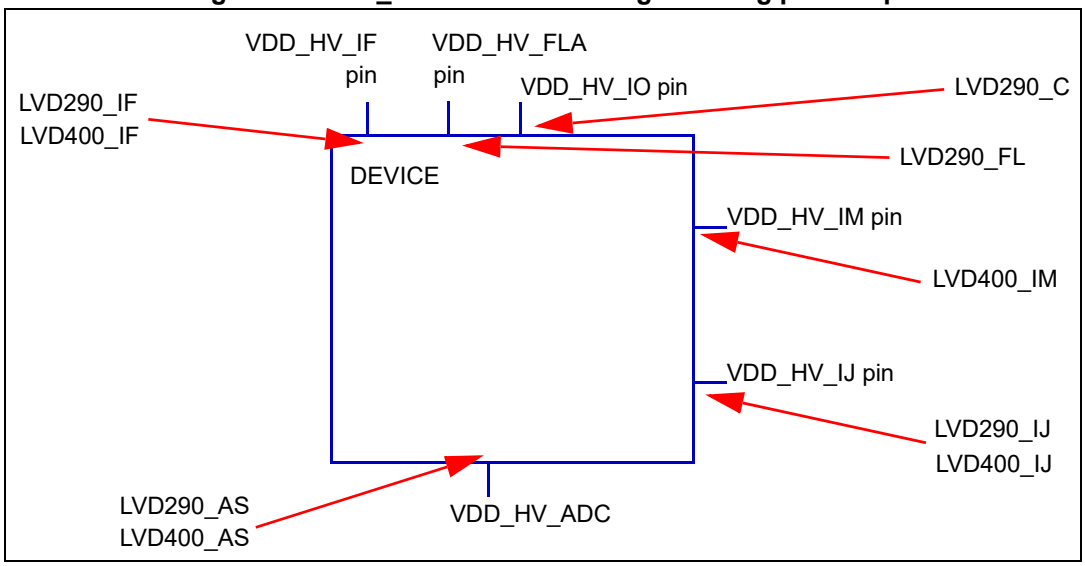
During POWERUP, the following high voltage MVDs are enabled:

- MVD270_C
- MVD270_FL

The VDD_HV conditions to exit POWERUP are the following:

- LVD290_C
- LVD290_FL

Figure 64. VDD_HV monitored voltages during power-up



The IRCOSC module starts initialization and provides the clock to the system after $t_{RCSTARTUP}$. PMC digital interface reset is released after two RC clock cycles.

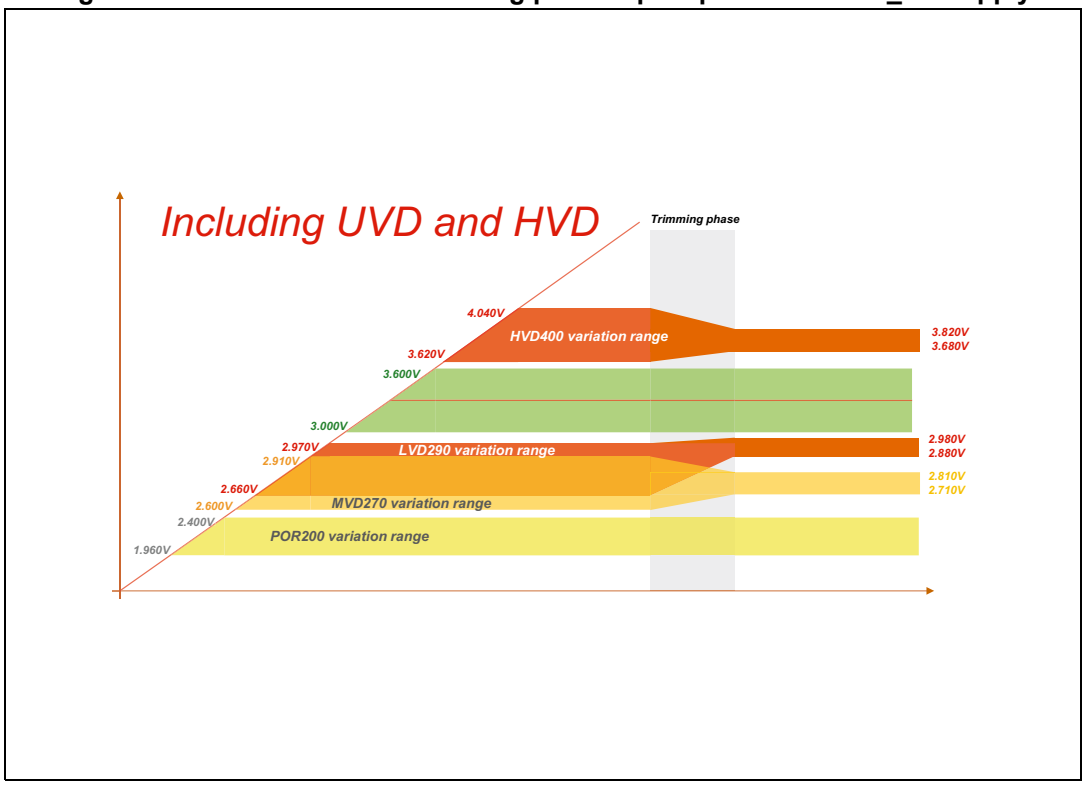
Device proceeds through the reset sequence through RGM phase PHASE0, PHASE1[DEST], PHASE2[DEST], PHASE3[DEST].

LVD/HVD modules are trimmed at the beginning of PHASE3[DEST]. Trimming is done by SSCM at low voltage, using the differential flash read accesses. After trimming is completed, SSCM waits for PMC acknowledgement before proceeding with reset sequence.

Configurable LVD/HVD modules are optionally enabled at the end of PHASE3[DEST].

Figure 65 provides the threshold variation of VDD_HV LVDs monitor during power-up sequence from POWERUP before and after trimming.

Figure 65. Threshold variation during power-up sequence on VDD_HV supply



The device needs to fulfill all conditions described in this section to recover.

10.6.2 Power-down sequence

In case LVDs/HVDs are configured to generate destructive reset , either LVD is below threshold or HVD is above threshold, the device enters PHASE0 phase.

Power-down sequence is actually entered as soon as the threshold of MVD094 and MVD270 is crossed. The device enters the POWERUP state. The internal power-on signal is asserted.

On power-on signal assertion, all modules reach their safe state.

Device supplies may then proceed to drop down to ground either through device leakage or external pull-down. During power-down, all supplies should comply with supply constraints.

10.6.3 Brown-out management

During brown-out, the device re-enters the POWERUP phase as soon as the threshold of either MVD094 or MVD270 is crossed. The internal power-on signal is asserted.

Note: The device will correctly start independently from any residual voltage. Below MVD270 *threshold the device is under POWERUP state. Above the* MVD270 *threshold, the device is into a RESET state as defined by Voltage monitor or external PORST pin or active state when all have been released.*

On internal power-on signal assertion, all analog modules reach their safe state.

10.6.4 Low voltage requirement during crank

The device can continue operation to the minimum input voltage during crank.

In order to proceed with execution during cranking and prevent device reset, it is important to correctly configure the high voltage LVDs.

During device switch-off the external LVDs monitoring may not be activated quickly enough (external assertion of the PORST pin) and hence the −10% level is crossed before POR is asserted. Malfunction is possible in this case, but reliability is not impacted. In particular the flash is not corrupted.

Internal LVDs that monitor the supply ensure that the flash content is protected.

11 Smart Stand-by Wake-up Unit (SSWU)

11.1 Overview

The Smart Standby Wake-Up Unit (SSWU) is aimed at further reducing the device power consumption by scheduling, on an RTC time basis, a sequence of user-programmable tasks, which are executed in STANDBY Mode without the CPU intervention.

These tasks are programmed in order to allow the following operations in STANDBY:

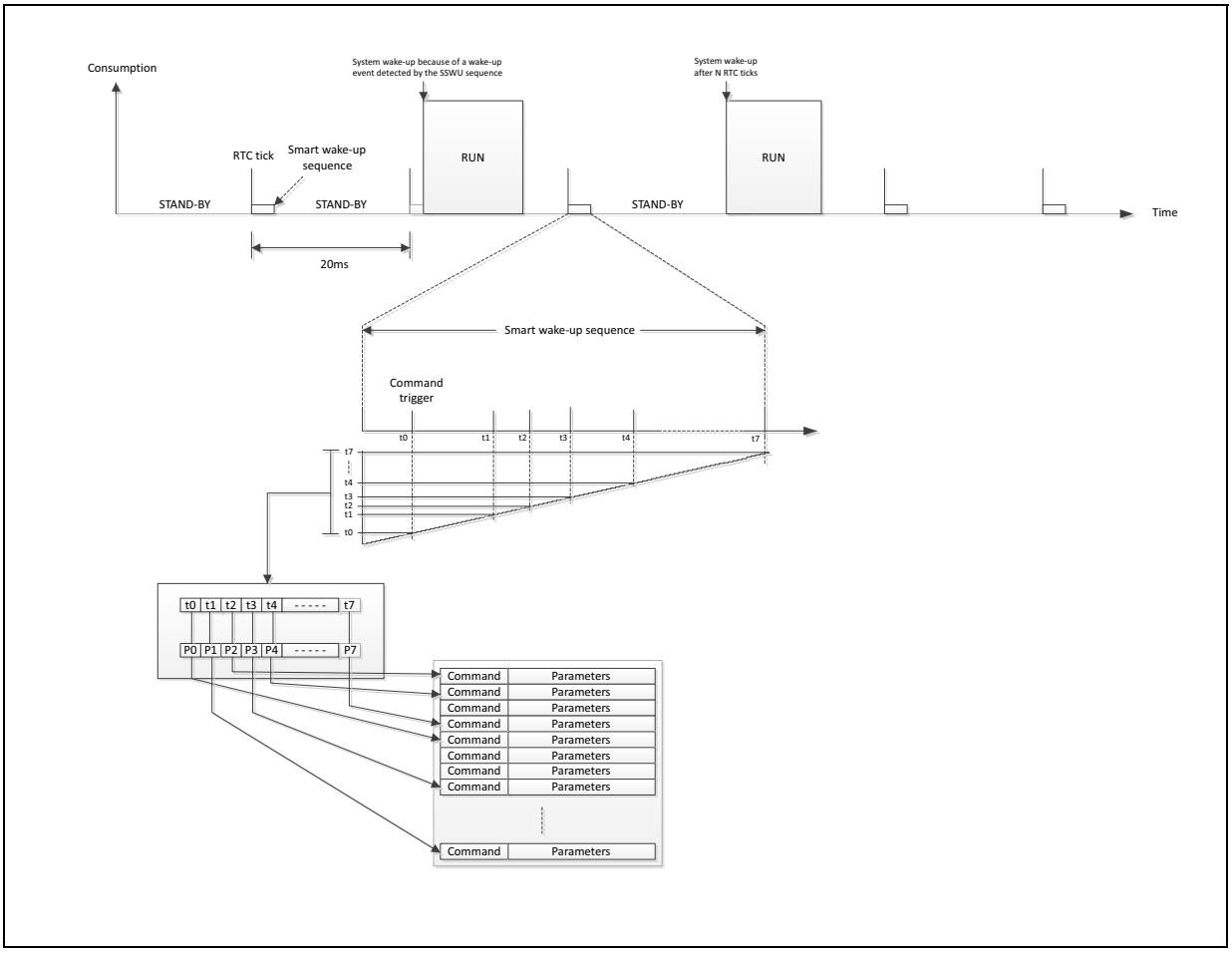
- Control output level
- Analog comparator versus voltage threshold or range
- Digital input state detector: In addition, it is possible to schedule a system wake-up based on a user-defined time interval.

The next sections explain the whole concept in details.

11.2 Functional description

Figure 66 shows the SSWU functionality.

Figure 66. Smart Wake-Up Sequence Functionality



The module is able to maintain the system is mostly in stand-by and eventually woken up based on the acknowledge of predifined wake-up event, which basically can be of two types:

- a time based wake-up event (WKUP1 line event)
- an asynchronous wake-up event (WKUP0 line event) detected during the SSWU sequence, which is periodically started based on the RTC tick (programming API period)

The time based wake-up event type is generated by the API/RTC IP, which can be programmed to generate it after a certain number of timer ticks.
The second asynchronous wake-up event type is generated by the SSWU block itself, as explained in the next sections.

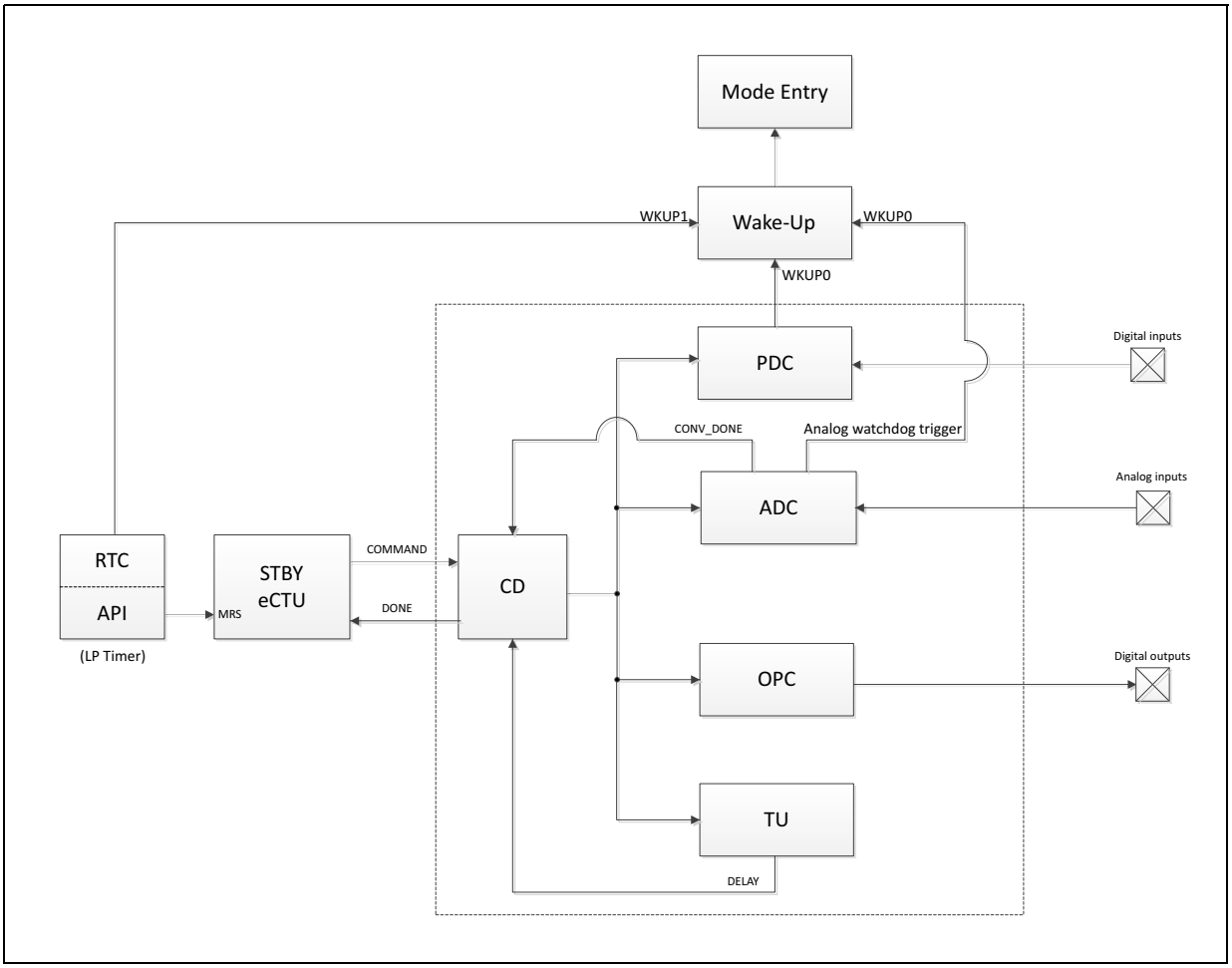
The SSWU sequence is a series of "commands" (plus some command-dependent parameters) generated by the Stand-by eCTU block based on a certain pre-programmed time schedule (t0, t1, etc. in the figure). For each scheduled time event, it is possible to assign a command to be executed. As explained later on, it is possible to assign a sub-sequence of commands, which can be displaced with a programmable time delay. Each command is then interpreted by a command decoder, which is dispatching them to the target client.

11.3 Architecture

Once the device is in STANDBY Mode, a minimum set of IPs will be periodically woken-up. This allows a pre-programmed sequence of measurements to run, limiting the power consumption to the minimum requested for a body application.

Figure 67 shows the block diagram of the SSWU architecture.

Figure 67. Smart Wake-up Architecture



Hereafter is a list of the blocks in the picture.

- RTC = Real Time Clock
- STBY eCTU = Stand-by Enhanced Cross Triggering Unit
- CD = Command Decoder
- ADC = Analog Digital Converter (client)
- PDC = Port Digital Comparator (client)
- OPC = Output Pin Control block (client)
- TU = Timer Unit (client)

Hereafter is a brief description of each block, which is explained in details in the following chapters.

The RTC/API timers block is always powered-on in STANDBY and generates a time tick based on the low power clock source (LPRC or OSC32K) selected.

The RTC timer (mapped on the WKUP1 line) allows to periodically exit from STANDBY. The API timer (mapped on the WKUP0 line), instead, drives the starting of SSWU mechanism (refer to *Section 11.4*).

The Stand-by eCTU (Enhanced Cross Triggering Unit) is used to schedule the execution of a set of pre-programmed commands: it is the "engine" of the SSWU sequence.

The CD (Command Decoder) is used to decode the command generated by the Stand-by eCTU and dispatch them to the following client blocks:

- The ADC (Analog Digital Converter) is a dedicated, low-power analog to digital converter, used to generate an API wake-up event (WKUP0) in case the converted value is above or below a certain threshold.
- The OPC (Output Pin Control block) block is used to set the output level of a digital port pin, in order to let the customer open/close some external switches.
- The PDC is used to trigger an API wake-up event (WKUP0) when the level of a pin is a certain value for a specific time.
- The TU is used to generate a programmable delay between different Stand-by eCTU command executed within the same command sequence.

The Wake-up unit collects the wake-up triggers (eventually provided by the ADC or PDC client) and generates a wake-up event (WKUP0) for the Mode Entry block.

The Mode Entry block is then responsible to change the system execution mode (from STANDBY to Run).

In STANDBY mode, only RTC is powered on, while all other blocks are off. At the RTC tick, the hardware switches on the 16 MHz IRCOSC and wakes up the other blocks.

11.4 Clocking

Figure 68 shows the SSWU clocking architecture.

The RTC evolves with the low power clock, which can be selectable between 32 kHz or 128 kHz.

Note: The 1024 kHz clock signal from slow internal RC oscillator (LPRC) is divided by 8 before applied as 128 kHz clock to RTC.

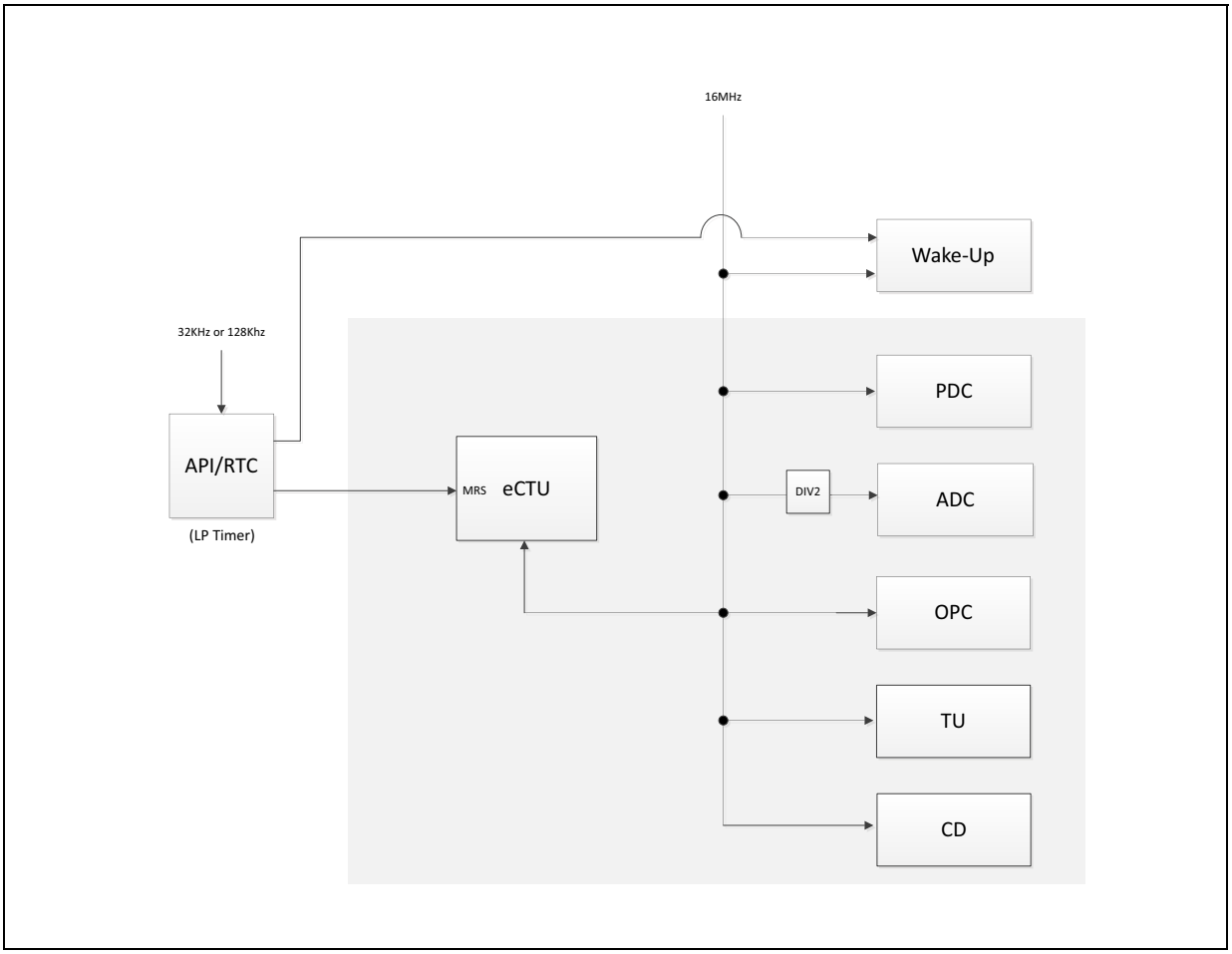
The SSWU IPs evolves with the internal 16 MHz IRCOSC.

A prescaler is embedded in the TU block in order to let the customer program the delay resolution (refer to *Section 11.6.6: TU*).

Note: Even if IRCOSC is configured off in STANDBY through the ME_STANDBY0_MC register, it will be automatically turned-on once SSWU is activated. The IRCOSC will be automatically turned off after SSWU Stop command is executed.

The ADC IP runs at 8MHz (IRCOSC clock divided by 2).

Figure 68. SSWU clock architecture



11.5 Memory map and register descriptions

11.5.1 Memory map

This section provides the memory map of the SSWU. For double buffering and synchronization refer to *Section 11.5.15: Double buffered registers*.

11.5.1.1 TGS registers

Table 140. TGS registers

Address offset	Register	Double buffered	Synchronization	Reset value
0x0000	Trigger Generator Subunit Input Selection Register (TGSISR)	Yes	TGSISR_RE	0x0000 0000
0x0004	Trigger Generator Subunit Control Register (TGSCR)	Yes	MRS	0x0000 0000
0x0008	Trigger Compare Register 0 (TCR0)	Yes	MRS	0x0000 0000

Table 140. TGS registers (continued)

Address offset	Register	Double buffered	Synchronization	Reset value
0x000C	Trigger Compare Register 1 (TCR1)	Yes	MRS	0x0000 0000
0x0010	Trigger Compare Register 2 (TCR2)	Yes	MRS	0x0000 0000
0x0014	Trigger Compare Register 3 (TCR3)	Yes	MRS	0x0000 0000
0x0018	Trigger Compare Register 4 (TCR4)	Yes	MRS	0x0000 0000
0x001C	Trigger Compare Register 5 (TCR5)	Yes	MRS	0x0000 0000
0x0020	Trigger Compare Register 6 (TCR6)	Yes	MRS	0x0000 0000
0x0024	Trigger Compare Register 7 (TCR7)	Yes	MRS	0x0000 0000
0x0028 - 0x0047	Reserved			
0x0048	TGS Counter Compare Register (TGSCCR)	Yes	MRS	0x0000 0000
0x004C	TGS Counter Reload Register (TGSCRR)	Yes	MRS	0x0000 0000

11.5.1.2 SU registers

Table 141. SU registers

Address offset	Register	Double buffered	Synchronization	Reset value
0x0100	Commands List Control Register 1 (CLCR1)	Yes	MRS	0x0000 0000
0x0104	Commands List Control Register 2 (CLCR2)	Yes	MRS	0x0000 0000
0x0108 - 0x010F	Reserved			
0x0110	Trigger Handler Control Register 1 (THCR1)	Yes	MRS	0x0000 0000
0x0114	Trigger Handler Control Register 2 (THCR2)	Yes	MRS	0x0000 0000
0x0118 - 0x011F	Reserved			
0x0120 - 0x019C	Command List Register 1-32 (CLR1-CLR32)	Yes	MRS	0x0000 0000
0x01A0 - 0x031F	Reserved			

11.5.1.3 Other eCTU registers

Table 142. Other eCTU registers

Address offset	Register	Double buffered	Synchronization	Reset value
0x0500	CTU Error Flag Register (CTUEFR)	NO	handshake	0x0000 0000
0x0504	CTU Interrupt Flag Register (CTUIFR)	NO	handshake	0x0000 0000
0x0508 - 0x050F	Reserved			
0x0510	CTU Control Register (CTUCR)	NO	handshake	0x0000 0000
0x0514 - 0x054B	Reserved			

11.5.2 Trigger generator subunit input selection register (TGSISR)

Offset: 0x000 Access: User Read/Write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	I0_FE	I0_RE
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 69. Trigger generator subunit input selection register (TGSISR)

Table 143. TGSISR field descriptions

Field	Description
30 I0_FE	ctu_trg_in[0] Falling edge Enable 0 disabled 1 enabled
31 I0_RE	ctu_trg_in[0] Rising edge Enable 0 disabled 1 enabled

Note: The IO_RE bit should be set in order to enable the SSWU mechanism.

11.5.3 Trigger generator subunit control register (TGSCR)

Offset: 0x0004 Access: User Read/Write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	PRES		Reserved					TGS_M
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 70. Trigger generator subunit control register (TGSCR)

Table 144. TGSCR field descriptions

Field	Description
24:25 PRES	TGS prescaler selection bits 00 1 01 2 10 3 11 4
31 TGS_M	Trigger Generator Subunit Mode 0 Triggered Mode 1 Sequential Mode **Note:** the only mode for eCTU STANDBY is the triggered mode, so the user has to keep TGS_M = 0.

11.5.4 Trigger compare register 0–7 (TCR0–TCR7)

There are 8 TCRx registers to support 8 triggers.

Offset: 0x008–0x0024 Access: User Read/Write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	TxCR_value															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 71. Trigger compare register 0–7 (TCR0–TCR7)

Table 145. TCR0–TCR7 field descriptions

Field	Description
16:31 TxCR_value	Value to be compared with the counter value to generate the trigger.

11.5.5 TGS counter compare register (TGSCCR)

Offset: 0x0048 Access: User Read/Write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	TGSCCR_value															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 72. TGS counter compare register (TGSCCR)

Table 146. TGSCCR field descriptions

Field	Description
16:31 TGSCCR_value	This register is used to stop the counter when TGSCCR_value is reached.

11.5.6 TGS counter reload register (TGSCRR)

Offset: 0x004C Access: User Read/Write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	TGSCRR_value															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 73. TGS counter reload register (TGSCRR)

Table 147. TGSCRR field descriptions

Field	Description
16:31 TGSCRR_value	This register is used to reload the counter when MRS is one.

11.5.7 Commands list control register 1 (CLCR1)

Offset: 0x0100 Access: User Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	T3_INDEX					0	0	0	T2_INDEX				
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	T1_INDEX					0	0	0	T0_INDEX				
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 74. Commands list control register 1 (CLCR1)

Table 148. CLCR1 field descriptions

Field	Description
3:7 T3_INDEX	Points to the (1st command address-1) for Trigger 3.
11:15 T2_INDEX	Points to the (1st command address-1) for Trigger 2.
19:23 T1_INDEX	Points to the (1st command address-1) for Trigger 1.
27:31 T0_INDEX	Points to the (1st command address-1) for Trigger 0.

Note: First command for any trigger will start from CLR < Tx_INDEX + 1 > register.

11.5.8 Commands list control register 2 (CLCR2)

Offset: 0x0104 Access: User Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	T7_INDEX					0	0	0	T6_INDEX				
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	T5_INDEX					0	0	0	T4_INDEX				
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 75. Commands list control register 2 (CLCR2)

Table 149. CLCR2 field descriptions

Field	Description
3:7 T7_INDEX	Points to the (Ist command address-1) for Trigger 7.
11:15 T6_INDEX	Points to the (Ist command address-1) for Trigger 6.
19:23 T5_INDEX	Points to the (Ist command address-1) for Trigger 5.
27:31 T4_INDEX	Points to the (Ist command address-1) for Trigger 4.

Note: First command for any trigger will start from CLR < Tx_INDEX + 1 > register.

11.5.9 Trigger handler control register 1 (THCR1)

Offset: 0x0110 Access: User Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	T3_E	0	0	0	T3_CMDE	0	0	0	T2_E	0	0	0	T2_CMDE
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	T1_E	0	0	0	T1_CMDE	0	0	0	T0_E	0	0	0	T0_CMDE
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 76. Trigger handler control register 1 (THCR1)

Table 150. THCR1 field descriptions

Field	Description
3 T3_E	Trigger 3 enable 0 Disabled 1 Enabled
7 T3_CMDE	Trigger 3 Command output enable 0 Disabled 1 Enabled
11 T2_E	Trigger 2 enable 0 Disabled 1 Enabled
15 T2_CMDE	Trigger 2 Command output enable 0 Disabled 1 Enabled

Table 150. THCR1 field descriptions (continued)

Field	Description
19 T1_E	Trigger 1 enable 0 Disabled 1 Enabled
23 T1_CMDE	Trigger 1 Command output enable 0 Disabled 1 Enabled
27 T0_E	Trigger 0 enable 0 Disabled 1 Enabled
31 T0_CMDE	Trigger 0 Command output enable 0 Disabled 1 Enabled

11.5.10 Trigger handler control register 2 (THCR2)

Offset: 0x0114 Access: User Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0		0	0	0		0	0	0		0	0	0	
W				T7_E				T7_CMDE				T6_E				T6_CMDE
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0		0	0	0		0	0	0		0	0	0	
W				T5_E				T5_CMDE				T4_E				T4_CMDE
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 77. Trigger handler control register 2 (THCR2)

Table 151. THCR2 field descriptions

Field	Description
3 T7_E	Trigger 7 enable 0 Disabled 1 Enabled
7 T7_CMDE	Trigger 7 Command output enable 0 Disabled 1 Enabled
11 T6_E	Trigger 6 enable 0 Disabled 1 Enabled

Table 151. THCR2 field descriptions (continued)

Field	Description
15 T6_CMDE	Trigger 6 Command output enable 0 Disabled 1 Enabled
19 T5_E	Trigger 5 enable 0 Disabled 1 Enabled
23 T5_CMDE	Trigger 5 Command output enable 0 Disabled 1 Enabled
27 T4_E	Trigger 4 enable 0 Disabled 1 Enabled
31 T4_CMDE	Trigger 4 Command output enable 0 Disabled 1 Enabled

11.5.11 Commands list register 1–32 (CLR1–CLR32)

The number of CLRx supported is 32.

Offset: 0x011C + n*0x4 (n = 1 to 32) Access: User Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	LC	EC	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	PARAMS							0	0	0	0	0	0	CLIENT	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 78. Commands list register 1–32 (CLR1–CLR32)

Table 152. CLR1–CLR32 field descriptions

Field	Description
1 LC	1 Implies Last sequence for the trigger.
2 EC	1 Implies end of concurrent conversion. **Note:** This bit should be always set to 1.

Table 152. CLR1–CLR32 field descriptions (continued)

Field	Description
17:23 PARAMS	Client parameters Its structure varies according to selected client.
30:31 CLIENT	Target client of the command 00 ADC 01 OPC 10 PDC 11 TU

11.5.11.1 Commands list register 1–32 (CLR1–CLR32) - ADC client

The number of CLRx supported is 32.

Offset: 0x011C + n*0x4 (n = 1 to 32) Access: User Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	LC	EC	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	EXTCH_SEL	0	0	0	CHANNEL			0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 79. Commands list register 1–32 (CLR1–CLR32)

Table 153. CLR1–CLR32 field descriptions

Field	Description
1 LC	1 Implies Last sequence for the trigger.
2 EC	1 Implies end of concurrent conversion. **Note:** This bit should be always set to 1.
17 EXTCH_SEL	Internal / External channel selection bit 0 Internal channel 1 External channel
21:23 CHANNEL	Channel number Refer to *Table 161: ADC channel selection* for details.
30:31 CLIENT	Target client of the command 00 ADC

11.5.11.2 Commands list register 1–32 (CLR1–CLR32) - OPC client

The number of CLRx supported is 32.

Offset: 0x011C + n*0x4 (n = 1 to 32) Access: User Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	LC	EC	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	PATTERN				GROUP			0	0	0	0	0	0	0	1
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 80. Commands list register 1–32 (CLR1–CLR32)

Table 154. CLR1–CLR32 field descriptions

Field	Description
1 LC	1 Implies Last sequence for the trigger.
2 EC	1 Implies end of concurrent conversion. **Note:** This bit should be always set to 1.
17:20 PATTERN	Pattern to be applied to selected group of four OPC channels.
21:23 GROUP	OPC channels selection 000 Output OPC channels 0-3 001 Output OPC channels 4-7 01x Reserved
30:31 Client	Target client 01 OPC

11.5.11.3 Commands list register 1–32 (CLR1–CLR32) - PDC client

The number of CLRx supported is 32.

Offset: 0x011C + n*0x4 (n = 1 to 32) Access: User Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	LC	EC	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	COMMAND			PDC_CHANNEL				0	0	0	0	0	0	1	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 81. Commands list register 1–32 (CLR1–CLR32)

Table 155. CLR1–CLR32 field descriptions

Field	Description
1 LC	1 Implies Last sequence for the trigger.
2 EC	1 Implies end of concurrent conversion. **Note:** This bit should be always set to 1.
17:18 COMMAND	Command to execute on selected channel 00 CMP_0 (compare input channel with level ‘0’) 01 CMP_1 (compare input channel with level ‘1’) 10 CHECKOUT (evaluate comparison results) 11 Reserved
19:23 PDC_CHANNEL	PDC channel number
30:31 CLIENT	Target client of the command 10 PDC

11.5.11.4 Commands list register 1–32 (CLR1–CLR32) - TU client

The number of CLRx supported is 32. Refer to *Section 11.6.6* for more details about TU function.

Offset: 0x011C + n*0x4 (n = 1 to 32) Access: User Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	LC	EC	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	DELAY							0	0	0	0	0	0	1	1
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 82. Commands list register 1–32 (CLR1–CLR32)

Table 156. CLR1–CLR32 field descriptions

Field	Description
1 LC	1 Implies Last sequence for the trigger.
2 EC	1 Implies end of concurrent conversion. **Note:** This bit should be always set to 1.
17:23 DELAY	Programmable delay before next command starts Time delay = DELAY * $T_{RC16MHz}$* TU_{PRESC}(1)
30:31 CLIENT	Target client of the command 11 TU

1. TU_{PRESC} is defined in the SSWU_CTRL_REG register. Please refer to *Chapter 67: Power management controller digital interface (PMC_Dig)*, *Section 67.5.28: SSWU CTRL Register (SSWU_CTRL_REG)*.

11.5.12 Cross triggering unit error flag register (CTUEFR)

Offset: 0x0500 Access: User Read/Write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	MAX_CON	0	Error_compare	0	0	CMD_OE	TGS_OSM	MRS_O	ICE	SM_TO	MRS_RE
W						w1c		w1c			w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 83. Cross triggering unit error flag register (CTUEFR)

Table 157. CTUEFR field descriptions

Field	Description
21 MAX_CON	Maximum concurrent command error flag 1 If number of concurrent commands exceeds static design parameter that is 4. 0 No error
23 Error_compare	This bit is set to one if the counter reach the TGSCCR register. It is cleared writing one in this field.
26 CMD_OE	Command generation Overrun Error
27 TGS_OSM	TGS Overrun in Sequential Mode

Table 157. CTUEFR field descriptions (continued)

Field	Description
28 MRS_O	Master Reload Signal Overrun
29 ICE	Invalid Command Error
30 SM_TO	Trigger Overrun (more than 8 EV) in TGS Sequential Mode
31 MRS_RE	Master Reload Signal Reload Error

11.5.13 Cross triggering unit interrupt flag register (CTUIFR)

Offset: 0x0504 Access: User Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	T7_I	T6_I	T5_I	T4_I	T3_I	T2_I	T1_I	T0_I
W									w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	ADC_I	0	0	0	0	0	MRS_I	0
W									w1c						w1c	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 84. Cross triggering unit interrupt flag register (CTUIFR)

The design supports up to 8 triggers. The bits for unsupported triggers will always be read as ‘0’.

Table 158. CTUIFR field descriptions

Field	Description
8 T7_I	Trigger 7 interrupt flag
9 T6_I	Trigger 6 interrupt flag
10 T5_I	Trigger 5 interrupt flag
11 T4_I	Trigger 4 interrupt flag
12 T3_I	Trigger 3 interrupt flag

Table 158. CTUIFR field descriptions (continued)

Field	Description
13 T2_I	Trigger 2 interrupt flag
14 T1_I	Trigger 1 interrupt flag
15 T0_I	Trigger 0 interrupt flag
24 ADC_I	ADC command interrupt flag
30 MRS_I	MRS Interrupt flag

11.5.14 Cross triggering unit control register (CTUCR)

Offset: 0x0510 Access: User Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	Reserved(1)	Reserved(1)	Reserved(1)	Reserved(1)
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	CTU_FSM_RESET	Reserved	0	CGRE	FGRE	Reserved(1)	GRE	TGSISR_RE
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. Never write 1 to this bit.

Figure 85. Cross triggering unit control register (CTUCR)

Table 159. CTUCR field descriptions

Field	Description
24 CTU_FSM_RESET	CTU state machine reset when written '1'. Writing a one has effect on the module, but it always reads as zero. **Note:** It is possible that CTU state machine is waiting for end of SSWU command, and due to any fault, it does not arrive. Therefore, this bit should be used to reset the CTU state machine before each standby entry.
27 CGRE	Clear GRE (1)
28 FGRE	Flag GRE (1)
30 GRE	General Reload Enable (1) 0 disabled 1 enabled
31 TGSISR_RE	TGS Input Selection Register Reload Enable. Is set by the user and reset automatically when its reloaded.

1. Refer to *Section 11.5.15* for details.

11.5.15 Double buffered registers

Some registers of the SSWU are double buffered. When MRS is set, the data programmed in a register on register interface clock domain is transferred to the SSWU kernel clock domain. To avoid error condition due to a MRS signal that arrives when a write operation is not fully completed, the following 2 bits are used:

1. GRE (General Reload Enable)
2. FGRE (Flag General Reload disable)

The reload of all double-buffered registers is enabled by setting the GRE bit in the CTU Control Register in order to guarantee the consistency of all double-buffered registers. The user must ensure that the trigger command list is updated before MRS occurs. If the update is not performed, the previous values of all double-buffered registers remain active.

All the double-buffered registers use the same bit (GRE) to enable the reload when the MRS occurs. This bit is used to signal that the user has updated the registers and so a reload can be performed without any problem. The MRS bit is R/S (Read/Set). If the bit is 1 the reload can be performed, while if this bit is 0, the reload is not performed. A correct reload resets this bit. It is also possible to reset this bit by software setting CGRE (Clear GRE) bit. This bit can be reset in a software way but, anyway, it is reset when GRE bit is at zero.

In order to verify if a reload error occurs, FGRE (Flag GRE bit in the CTU Control Register) bit is used. When one of the double-buffered registers is written, FGRE flag is set to 1 and it is reset by a correct reload. When the MRS occurs while FGRE = 1 and GRE = 1, a correct

reload is performed (because all intended registers have been updated before the MRS occurs).

- If FGRE = 1 and GRE = 0, a reload is not performed, the error flag (MRS_RE) is set (in this case at least one register was written but the update has not been ended before the MRS occurrence).
- If FGRE = 0, it is not necessary to perform a reload because all the double-buffered registers are unchanged.
- If GRE = 1 with FGRE = 0, a MRS_RE is generated on the next master reload.

TGSISR_RE bit is used to synchronize TGSISR register. When the user set TGSISR_RE, the value written in the TGSISR register is used.

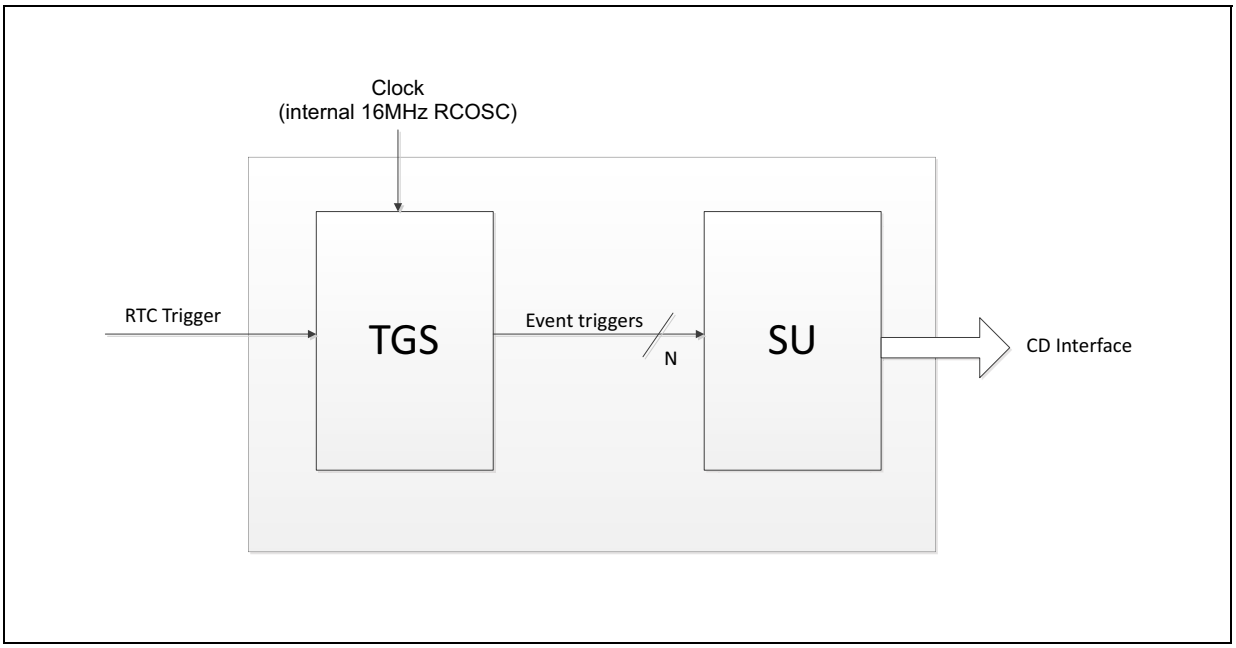
11.6 SSWU module descriptions

11.6.1 Stand-by eCTU

Figure 86 shows the Stand-by eCTU block diagram. Basically it is composed of two blocks:

- TGS (Trigger Generator Sub-unit), which is responsible to generate a set of triggers based on some pre-programmed time schedules
- SU (Scheduler Unit), which receives the trigger and starts the associated command sequence

Figure 86. Stand-by eCTU Block Diagram



The figure shows that the output of the Stand-by eCTU is directed to the CD interface. The CD block of the SSWU decodes the CTU command and dispatches it to the various clients (OPC, PDC, ADC, TU).

11.6.1.1 TGS block

Figure 87 shows the Stand-by eCTU architecture and functionality.

As shown in the figure, the TGS contains a 16-bit counter unit (evolving with the internal slow clock) and a set of comparators (TxCRV) [7]. Each comparator determines one time event point (tx in *Figure 66*).

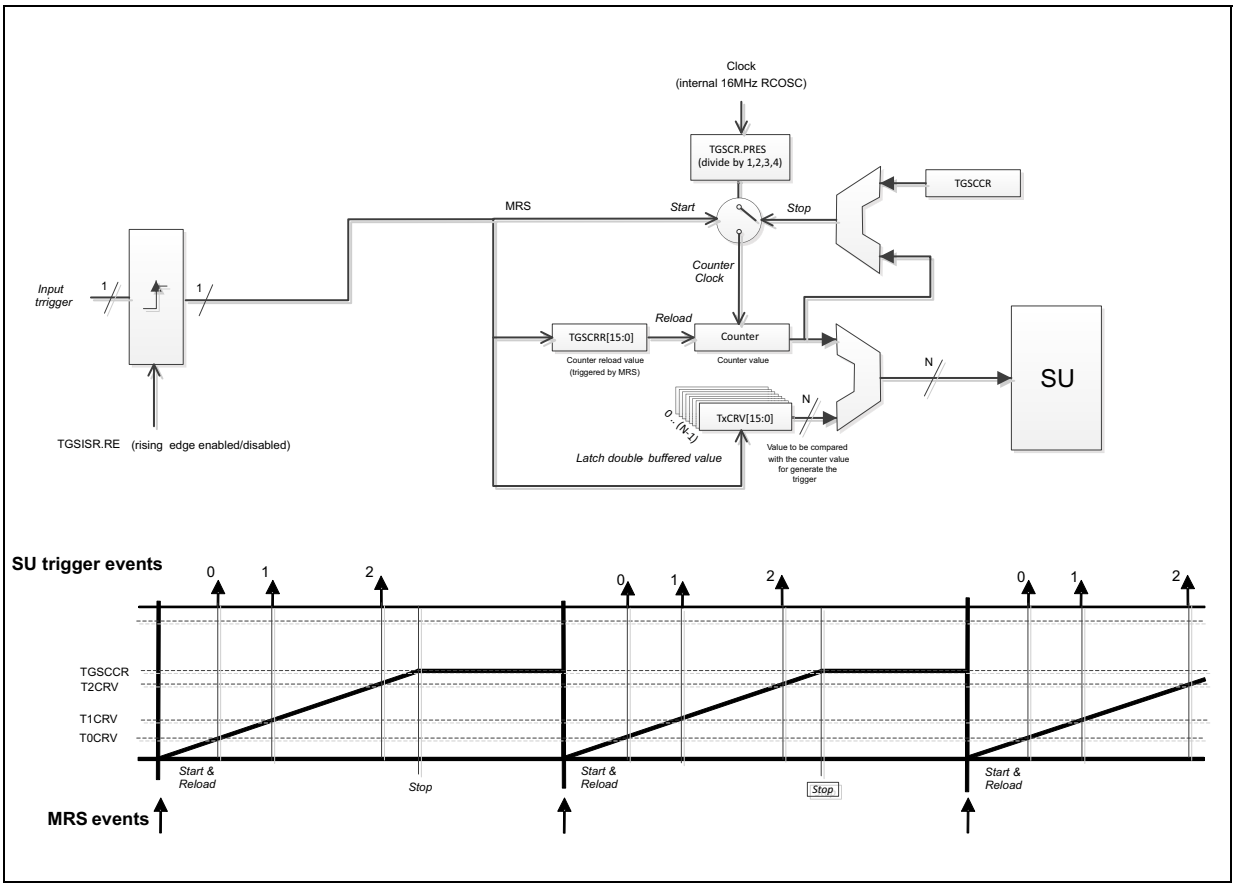
The MSR (Master Reload Signal) is used to reset and start the counter. The reset value is stored in the TGSCRR register. The MRS is generated via HW trigger (RTC tick).

When the counter reaches one of the values programed in the TxCRV comparators, an internal trigger is generated to the SU block. Note that is it possible to program TGSCCR register so that the counter will be stopped as soon as it reaches a certain value.

The waveform in *Figure 87* shows an example of three SU trigger generations, based on T0CRV, T1CRV, and T2CRV comparators. Note that, the TGS internal counter is started by a MRS event and reloaded with the '0' value. Also the TGSCCR register is used to stop the counter when it reaches the value programmed in this register.

Caution: When the SSWU sequence is generated with several SU triggers, the user must insure the SSWU command is finished before triggering a new one.

Figure 87. TGS Architecture and Functionality (triggered mode)



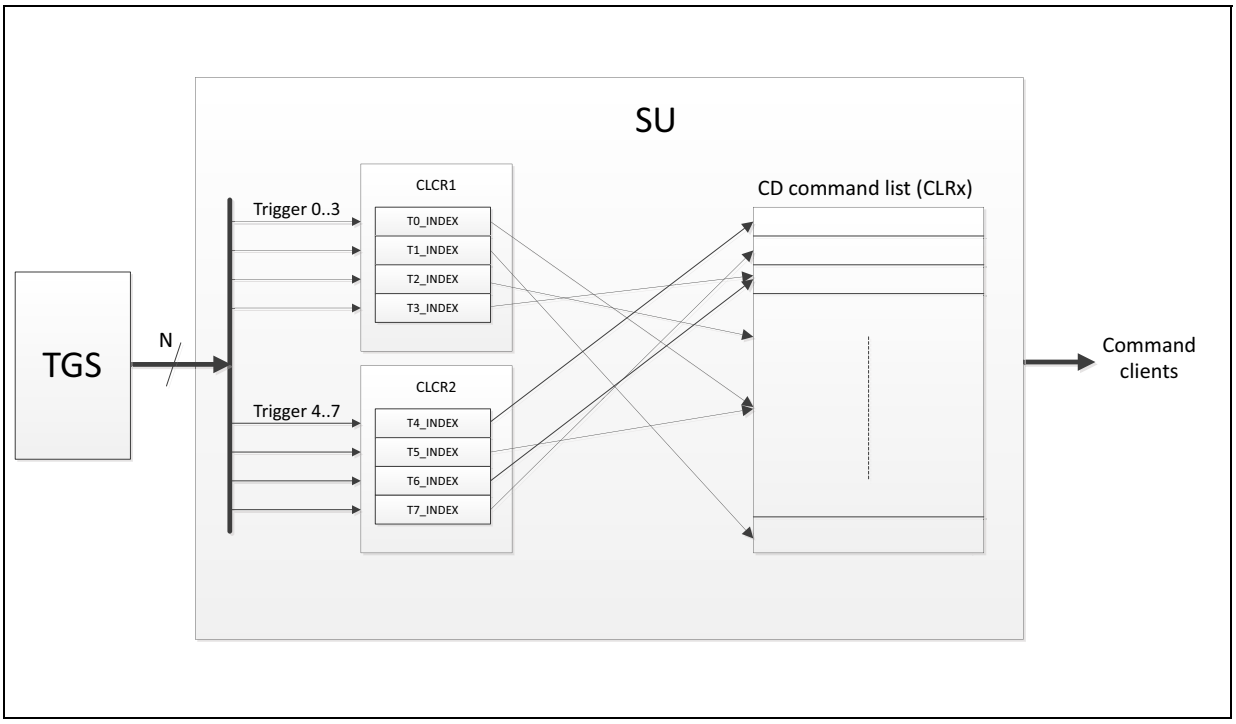
7. The number of comparators is 8

11.6.1.2 SU block

Figure 88 shows the SU block functionality. Its function is to receive the internal SU trigger and activate a sequence of commands, pre-programmed in a command table [8].

In fact, each trigger from the TGS block is associated to a programmable pointer, which determines, within the SU command table, the begin of the sequence of commands.

Figure 88. SU Functionality

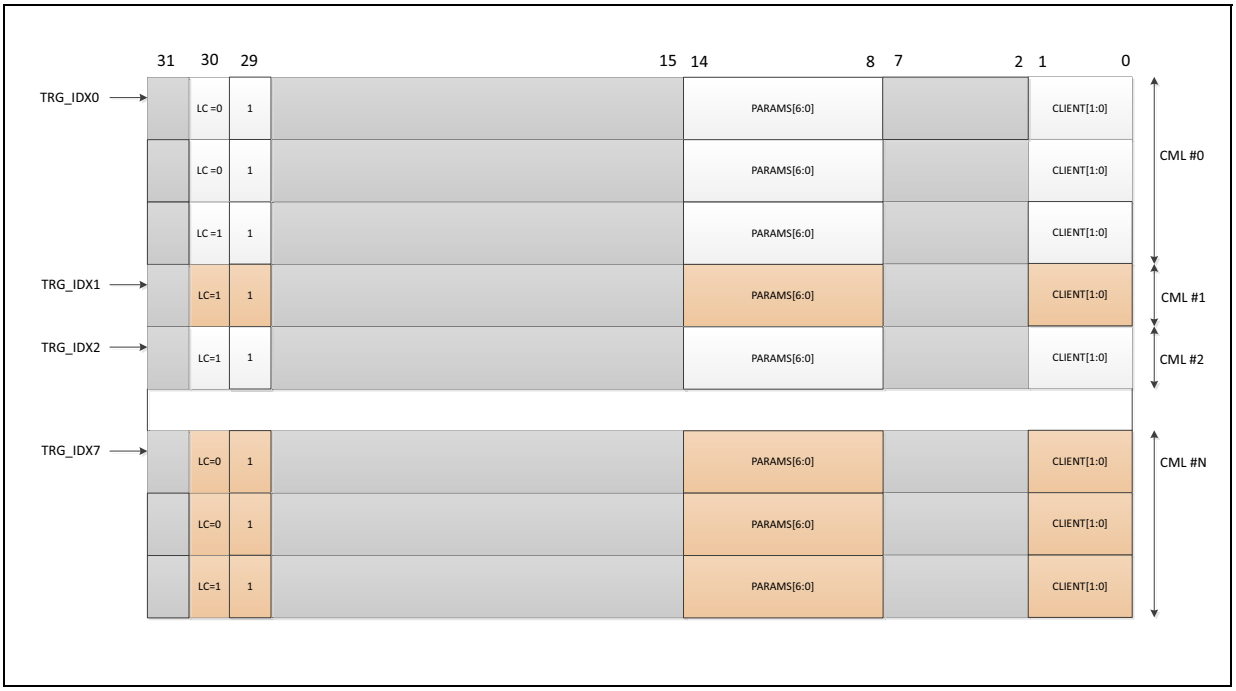


11.6.1.3 Stand-by eCTU command structure

The command structure is specified in *Figure 89*.

8. The command table length is 32

Figure 89. SU Command Structure



Hereafter is a description of the command fields.

- LC (last Command): this flag is used to specify whether the command being executed is the last of the sequence started by a SU trigger. *Figure 89* shows several examples: the command sequence triggered by TRG_IDX0 is composed of 3 commands, while the sequence triggered by TRG_IDX1 is actually one single command.
- EC (bit 29): always set to 1
- PARAMS: specify the command parameters
- CLIENT: specify the client ID

Starting from the first command to be executed, as programmed in the pointer associated to a specific SU trigger, each command is executed after the previous one is completed. The "LC" command flag determines whether the current command is the last one to be executed for the sequence of commands being executed.

11.6.1.4 Stand-by eCTU command sequence summary

As a summary, the command sequence is driven by:

- the TGS sequence of triggers, where each SU trigger is generated on a programmable time-basis,
- the SU command sequence, activated by a SU trigger.

The Stand-by eCTU commands are executed in a row till the one marked as "LC = 1". The Stand-by eCTU executes the next command as soon as it is acknowledged that the previous command was completed. This is represented in *Figure 67* by the DONE feedback signal from the CD block to the Stand-by eCTU block.

The execution time between each command depends on the command itself.

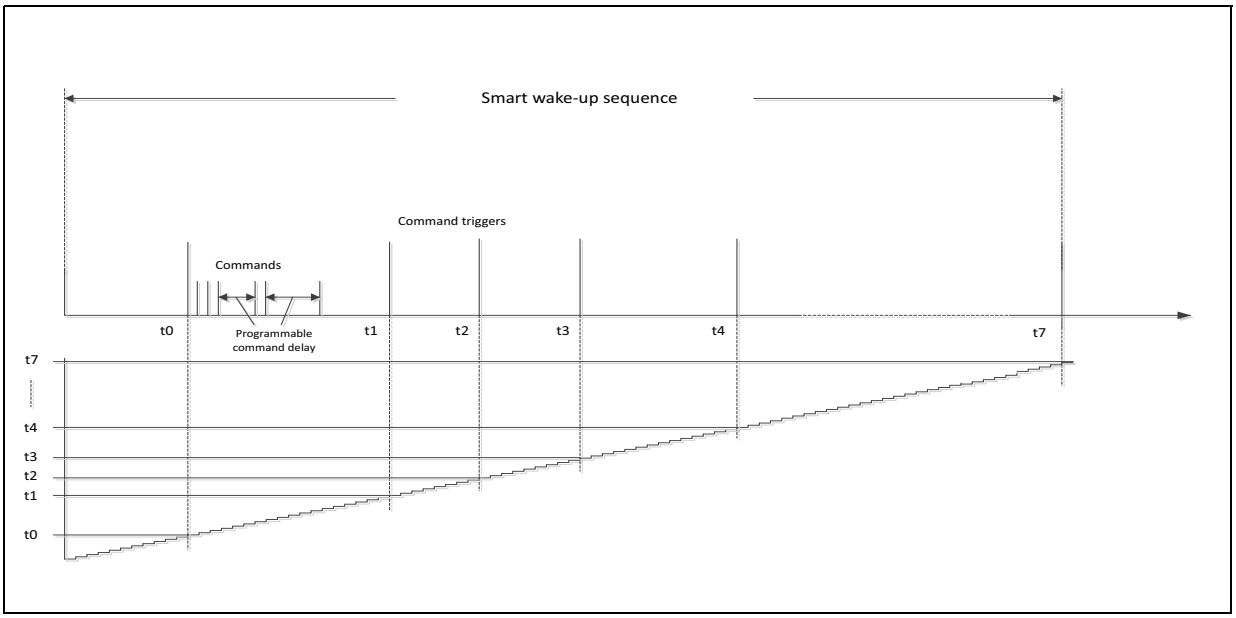
A command associated with an ADC conversion lasts till the ADC has acknowledged the Stand-by eCTU that the conversion was completed. This is represented in *Figure 67* by the CONV_DONE feedback signal from the ADC block to the CD block.

Other commands (for example, the one used to set a digital port pin level) have no "intrinsic" duration. Because of this, the "TU" block has been introduced, in order to be able to set a programmable delay between "immediate" commands. This is represented in *Figure 67* by the DELAY feedback signal from the TU block to the CD block.

As a result, shown in *Figure 90*, the time when a command shall be executed can be controlled in two ways:

- the value of the TxCVR comparators (t0, t1, etc. in *Figure 90*)
- the TU delay times (“programmable command delay” in *Figure 90*)

Figure 90. Programmable Command Delay



11.6.2 CD (Command Dispatcher)

The Stand-by eCTU commands contain two main fields:

- CLIENT
- PARAMS

The information in these two fields is decoded by the CD block and dispatched to the target clients (refer to *Table 160*).

The PARAMS field of the command is interpreted according to the type of client being activated.

Table 160. Smart Wake-up Sequence command

CLIENT[1:0]	Block	PARAMS[6:0]
0	ADC	Refer to *Table 161*
1	OPC	Refer to *Table 162*

Table 160. Smart Wake-up Sequence command (continued)

CLIENT[1:0]	Block	PARAMS[6:0]
2	PDC	Refer to *Table 163*
3	TU	Wait time

Table 161. ADC channel selection(1)

PARAMS[6] EXTCH_SEL	Selected ADC channel Internal / External	PARAMS[2:0] CHANNEL	ADC channel number
0	Internal	000	88
		001	89
		010	90
		011	91
		100	92
		101	93
		110	94
		111	95
1	External	000	128
		001	129
		010	130
		011	131
		100	132
		101	133
		110	134
		111	135

1. PARAMS[5:3] are Reserved

Table 162. OPC Command

PARAMS bits	Field name	Value	Description
[6:3]	PATTERN	Any	Pattern to be applied to the selected 4 channel group
[2:0]	GROUP	000	Select channels 0..3
		001	Select channels 4..7
		010–111	Reserved

Table 163. PDC Commands(1)

PARAMS bits	Field name	Value	Description
[6:5]	COMMAND	00	CMP 0
		01	CMP 1
		10	CHECKOUT
		11	Reserved
[4:0]	PDC channel	Any	PDC channel number

1. Only four PDC inputs are implemented: PDC0 to PDC3

Table 164. TU Commands

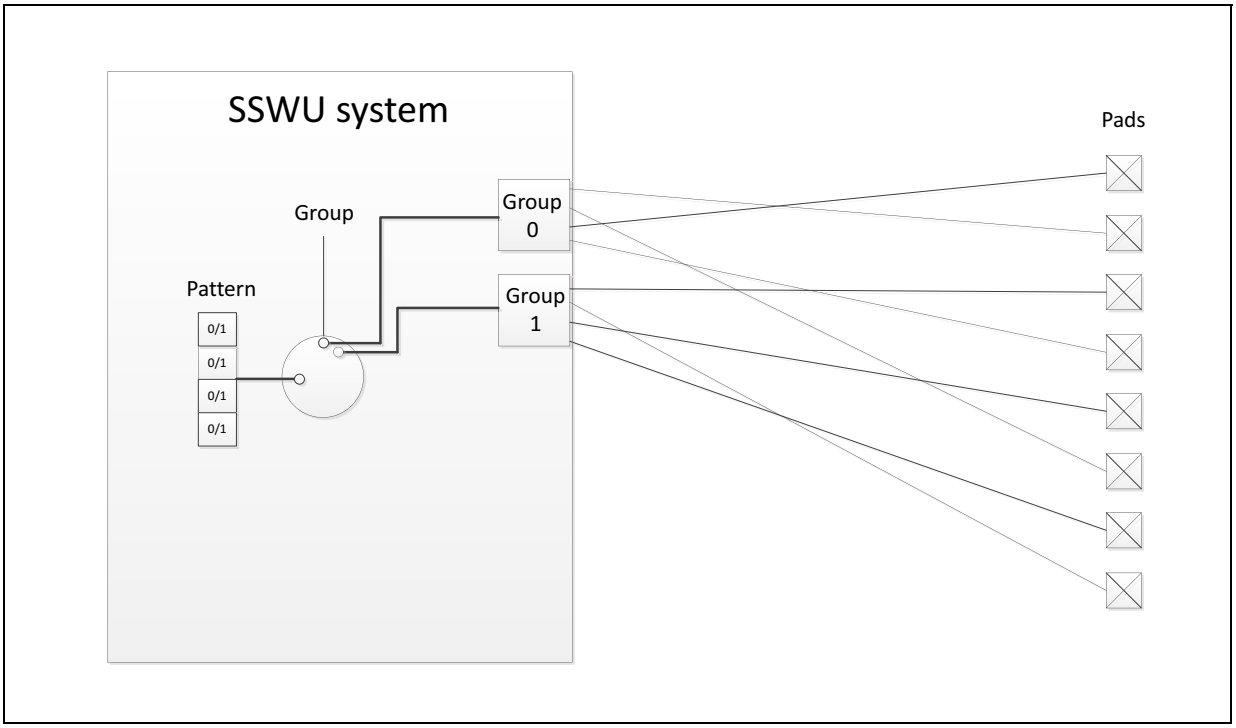
PARAMS bits	Field name	Value	Description
[6:0]	DELAY	Any	Programmable delay

11.6.2.1 OPC channel handling

The SSWU is able to drive up to 8 OPC channels.

The channels are divided into 2 groups of 4 channels each. The “GROUP” command parameter is used to select the group of channels to be operated (refer to *Table 162*). The 4 channels belonging to the selected group are driven in parallel: the 4 channels will be set according to the value specified in the “PATTERN” command parameter. The connection between each OPC channel and the corresponding pad is hard-wired.

Figure 91. SSWU OPC connections



11.6.3 ADC

The ADC block is a reduced, low-power version of the Successive Approximation Register Analog-to-Digital Converter called 10-bit SARADC_STDBY. Its main functionality is to compare the converted value towards a watchdog, and trigger a wake-up event in case the value is out of the intended range.

The watchdog thresholds (low and upper limits) are linked (that is pre-programmed during the RUN mode) to each ADC channel, so that a specific channel has its own threshold limit.

The ADC configuration comprises:

- 8 internal channels
- 4 different watchdog threshold values
- 8 external channels

More details about the ADC configuration are given in the *Chapter 40: Analog-to-Digital Converters (ADC) Configuration*.

11.6.3.1 ADC Channel Selection

The selection of internal or external channel group is done through the PARAMS[6] bit.

In case the internal channel group is selected (PARAMS[6] = 0), the internal channel number depends on the value specified by the PARAMS[2:0] bits. Because the internal channels are statically connected to a specific analog input pin, the channel selection is also a selection vs. which analog input pins will be used.

In case the external channel group is selected (PARAMS[6] = 1), the value of PARAM[2:0] represents the external channel number. The selected external channel number is then re-mapped on an internal channel through the ECMICR registers, which contains a set of fields, one for each external channel, where the linked internal channel can be specified. This association has to be done before entering into the stand-by mode. This way it is possible to link different external channels to the same internal channel.

11.6.3.2 ADC Watchdog Threshold Selection

A voltage measurement threshold (upper and lower limit) can be selected for each channel. The WTHRHRLx registers define a pool of watchdog threshold values, which can be referenced for each channel through the ICWSELx and ECWselx registers for the internal or external channel respectively.

In case the same analog input pin is used to measure the same voltage range with the same voltage threshold, then an internal channel is used (PARAMS[6] = 0).In this scenario, the configuration is:

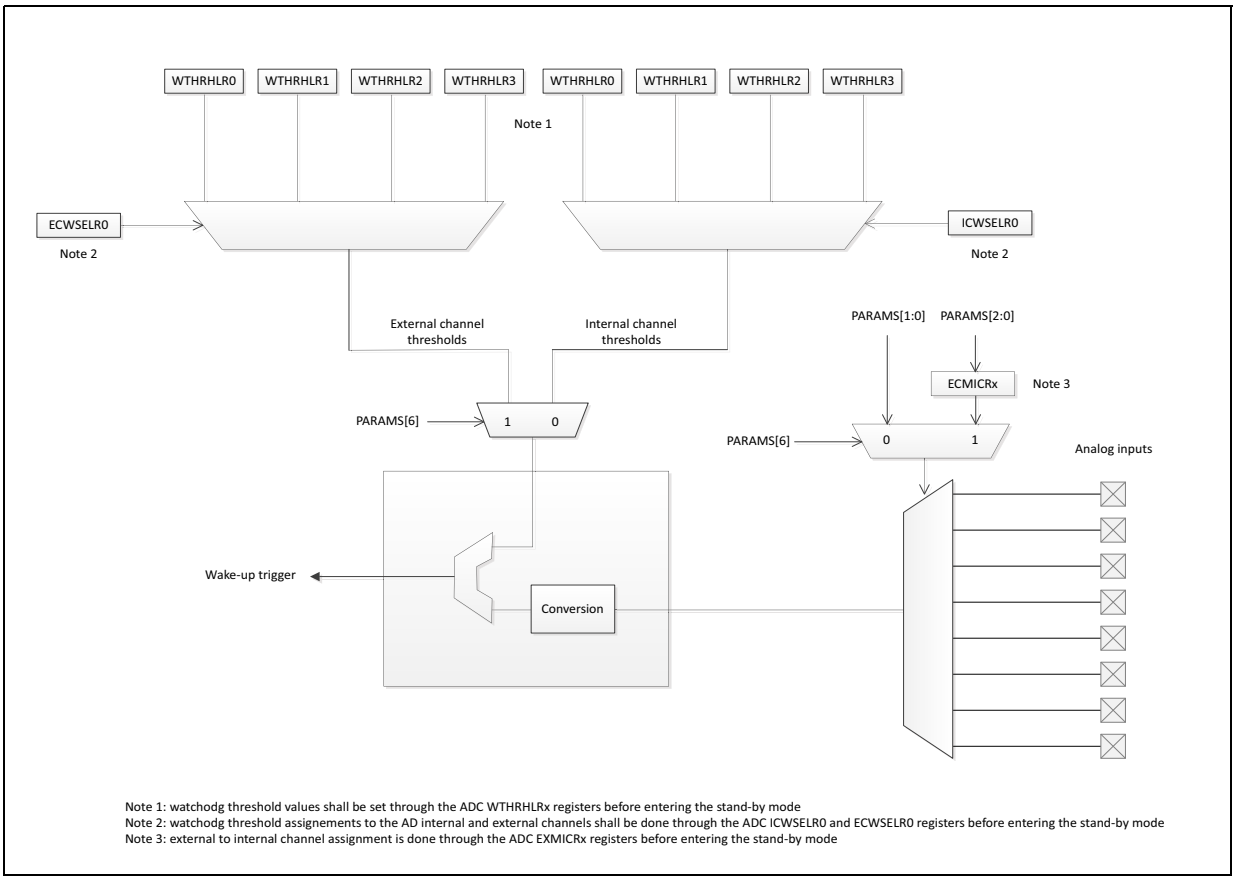
- The internal channel value is fixed by the PARAMS[2:0] value
- The threshold is chosen by setting the ICWSEL register before entering the stand-by mode

In case the same analog input pin is used to measure the different voltage range with the same voltage threshold, then an external channel is used (PARAMS[6] = 1). In this scenario, the configuration is:

- The external channel value is fixed by the PARAMS[2:0] value
- The threshold is chosen by setting the ECWSEL register before entering the stand-by mode
- The internal channel (the pin to be used) linked to the chosen external channel is defined by the ECMICR register

Figure 92 represents in graphical form the above description.

Figure 92. ADC channel selection



11.6.4 OPC

The OPC block is used to drive the 8 digital output pins to a desired level.

The intended functionality is to open/close external switches.

The Stand-by eCTU command parameter contains the OPC channel and level.

The OPC channels are controlled in groups of 4 channels, but, it is possible to mask any one of these channels, in order to retain the value set in RUN mode on any pad not intended to be used for the SSWU.

The mechanism to achieve this is to use the SIUL SCR bits to mask the OPC channel driving to the pad (refer to *Section 17.2.2.11: Soc Configuration Register 0 (SIUL2_SCR0)*). In order to drive an OPC channel, the SW shall configure the associated SCR bit before entering in stand-by mode.

The set of pads used to control external logic during the Stand-by have the following behavior:

- Upon entering the stand-by mode, the pad values corresponding to OPC channels are maintained.
- That means if the pad is used for OPC functionality during STANDBY mode, then on the subsequent STANDBY mode entry, the PAD value is restored to the value during the previous STANDBY mode exit.
- However, if the PAD is not used for OPC functionality during STANDBY, then the same value as in RUN mode is maintained upon entering STANDBY mode.
- SIUL_SCR0.OPC_MASK configuration for any OPC channel should remain same during the application. Otherwise, the value on OPC channel on STANDBY entry may have some undesired transitions.
- During stand-by mode, it is possible to change the value of those pads thanks to OPC programming.
- Upon exiting from stand-by, the value of the pads connected to OPC channels can also be maintained. This avoids glitches during the transition between stand-by and run mode. For this purpose, the SIUL2_SCR0 register must be programmed before entering stand-by.

Note: Note that the pads value and direction can still be changed during the RUN mode, but this requests to write a bit within the SIUL_SCR0 register to “1”. Before entering the standby mode the SIUL_SCR0 bit shall be programmed to "0".

11.6.5 PDC

The PDC block is targeted to measure the level of a set of 4 pins and to generate a system wake-up event when the level is acknowledged at a specified level and, optionally, for a certain time.

Two commands are need for the PDC:

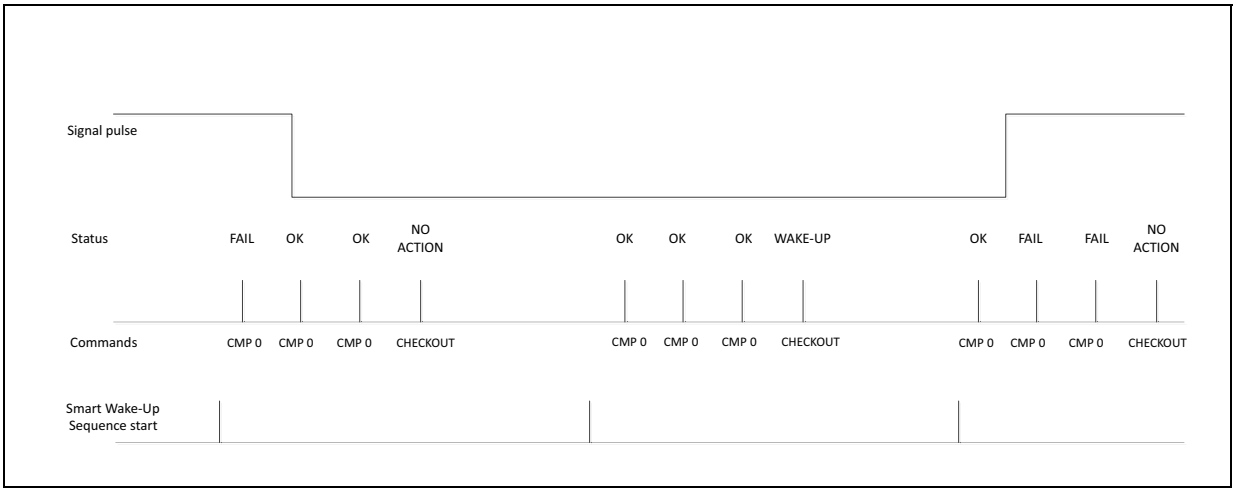
- CMP 0/CMP 1: to compare a pin level versus a 0 or 1
- CHECKOUT: to trigger a wake-up in case the result of the CMP commands for that specific pin executed from the beginning of the Smart Wake-up Sequence is always OK

For each pin, the PDC will latch the comparison results, which will report whether the target value was always matched (whenever a PDC, CMP0/1 command is launched) or not.

When checkout command is launched for a specific pin, it will trigger a wake-up event only if the target level was steadily acknowledged along all the comparisons done for that pin.

Figure 93 summarizes the above description.

Figure 93. Smart Wake-up Sequence PDC



11.6.6 TU

The TU block is used to create a programmable delay between the command execution. It needs only one command with the delay value programmed as a parameter.

The delay parameter is intended as number of IRCOSC periods (so that when TU = 1, then the delay is 1 clock cycle; TU = 2, then the delay is 2 clock cycles; and so on). The clock cycle can be adjusted with a linear prescaler, which is programmed though SSWU_CTRL_REG register of PMC_Dig module during the RUN mode, and whose value is preserved through the stand-by mode. The prescaler has a division factor from 1 to 128 (7 bits). Combining the TU parameter and the prescaler, the delay ranges from 62.5 ns (Clock period = 62.5 ns, PARAMS[6:0] = 1, prescaler = 1) up to 1.016 ms (Clock period = 62.5 ns, DELAY = 127, TU_{PRESC} = 128).

Note: TU_{PRESC} is defined in the SSWU_CTRL_REG. Please refer to Chapter 67: Power management controller digital interface (PMC_Dig).

Caution: The two commands which can cause wakeup (ADC-ADC or ADC-PDC or PDC-PDC) shall not be executed one after another without inserting a TU delay command.

11.6.6.1 SSWU STOP Command

In case the TU parameter (delay time) is set to 0, this is decoded as a command request to stop the whole SSWU sequence.

11.7 SSWU Configuration example

11.7.1 Description

In this scenario, the device stays in STANDBY and, every 32 msec, starts following SSWU operations sequence:

- Suddenly set OPC1 pin to '1'
- After 200 µsec, set OPC1 pin to '0'
- Perform an ADC conversion (configuring an ADC threshold to wakeup the device)
- Set a TU command for stopping the SSWU sequence

Device wakes-up from STANDBY only in case of ADC threshold overcoming.

11.7.2 Registers configuration

Following a pseudo-code that explains how to configure all the SSWU registers and all others IPs that are interconnected with it in order to get the overall STANDBY scenario

11.7.2.1 SSWU registers Configuration

```
//First command list
SSTDBY_CTU_0.CLR[0].R = 0x60001001; //set OPC1(PB9) pins to '1'

//Second command list
STDBY_CTU_0.CLR[2].R = 0x20000001;   //set OPC1(PB9) pins to '0'

STDBY_CTU_0.CLR[3].R = 0x20000C00; //convert internal channel  (AN82/PB10)

STDBY_CTU_0.CLR[4].R = 0x60000003; //add a delay of 0usec to stop the
sequence

STDBY_CTU_0.CLCR1.B.T0_INDEX = 0;   // trigger0 points to CLRN[0]
STDBY_CTU_0.CLCR1.B.T1_INDEX = 2;   // trigger0 points to CLRN[2]

STDBY_CTU_0.THCR1.R = 0x00001111;   // trigger0-1 enable
STDBY_CTU_0.TCRN[0].R = 0x00000001; // trigger0 generated quite immediately
STDBY_CTU_0.TCRN[1].R = 0x00000C80; // trigger1 generated 200us after the
API event

STDBY_CTU_0.TGSISR.R = 1;  // ctu_trg_in[0] (API) Rising edge Enable

STDBY_CTU_0.CTUCR.B.TGSISR_RE = 1;   // TGS Input Selection Register Reload
Enable

STDBY_CTU_0.CTUCR.B.GRE = 1;     // General Reload Enable
STDBY_CTU_0.CTUCR.B.CTU_FSM_reset = 1; // reset CTU FSM before
                                       // next stand-by entry
```

11.7.2.2 Other IP registers configuration

Following IPs should also be configured in order to complete the entire STANDBY scenario:

```
-----------------------------------------------
Clock and Mode Entry configuration
-----------------------------------------------

MC_ME.ME.R = 0x0000FFFF;          // Enable all modes

// Set RUN Configuration Registers
MC_ME.RUN_PCN[1].R=0x000000FE;       // Peripheral ON in every running mode
MC_ME.LP_PCN[1].R=0x00002500;

MC_ME.PCTL15.B.RUN_CFG=1;         // SIUL2
MC_ME.PCTL114.B.RUN_CFG=1;        // ADC_STDBY
MC_ME.PCTL242.B.RUN_CFG=1;        // STDBY_CTU_0

MC_CGM.AC0_DC2.R = 0x80000000;    // enable ADC_CLK
MC_CGM.AC0_DC4.R = 0x80000000;    // enable ECTU_CLK

//Enter DRUN to act the previous configurations
MC_ME.MCTL.R = 0x30005AF0;        // DRUN Mode & Key
MC_ME.MCTL.R = 0x3000A50F;        // DRUN Mode & Key

while(MC_ME.GS.B.S_MTRANS);       // Waiting for end of transaction
while(MC_ME.GS.B.S_CURRENT_MODE != 0x3);   // ME_GS Check DRUN mode has
successfully been entered

----------------------------
WKPU configuration
----------------------------

//Configure wakeup line 0 (SSWU wakeup event)
WKPU.WRER.R = 1;                  // Wakeup Request Enabled
WKPU.WIREER.R = 1;                // Rising Edge Event Enabled

-----------------------------------------------
SAR_ADC_10bit_STDBY configuration
-----------------------------------------------

SAR_ADC_10BIT_STDBY.MCR.B.OWREN = 1; // SAR ADC 0 overwrite enable set
SAR_ADC_10BIT_STDBY.MCR.B.MODE = 0; // Normal Mode
SAR_ADC_10BIT_STDBY.ICNCMR2.B.NCE_CH82 = 1; //sets channels 82 to be
converted in SARADC_10bit_standby (see table in ADC Config)
SAR_ADC_10BIT_STDBY.MCR.B.CTU_MODE = 1; // Trigger mode is selected
SAR_ADC_10BIT_STDBY.MCR.B.CTUEN = 1;    // CTU enabled
```

```
SAR_ADC_10BIT_STDBY.MCR.B.PWDN = 1; // SAR ADC 0 automatic power
                                    // up by SSWU during stand-by

//ADC_STANDBY_CH0 = Internal Channel 82:
SAR_ADC_10BIT_STDBY.WTHRHLRN[0].B.THRH = 0x333; // select high THRHLR0
treshold = 4V
SAR_ADC_10BIT_STDBY.ICWSELR11.B.WSEL_CH82 = 0;  // select THRHLR0 treshold
for CH88
SAR_ADC_10BIT_STDBY.ICWENR2.B.WEN_CH82 = 1;  // enable watchdog for CH82
SAR_ADC_10BIT_STDBY.WTIMR.B.MSKWDG0H = 1;   // enable high THRHLR0 treshold
interrupt

----------------------------
STANDBY configuration
----------------------------

PMCDIG.MISC_CTRL_REG.B.LPRCREG_ENB = 0;    // Enable LPRC in running modes

----------------------------
API/RTC configuration
----------------------------

RTC_API.RTCC.R = 0x80009001;    // Counter Enabled, API Enabled, LPRC
prescaled by 8 @128KHz used, Trigger enabled
RTC_API.APIVAL.R = 0x00001000;  // API period =32.25msec
```

Caution: For SSWU, ADCSAR must be used in CTU_TRIGGER mode.

11.7.2.3 Activate SSWU sequence

Once all previous registers configuration have been configured, it needs just to perform STANDBY Mode entering in oder to get the desired scenario.

```
//Mode Entry Module (MC_ME): Enter STANDBY Mode
MC_ME.ME_STANDBY0_MC.IRCON.B = 0; /* Switch off IRCOSC during standby */
MC_ME.MCTL.R = 0xD0005AF0;       /*! Write Mode and Key | MC_ME.MCTL */
MC_ME.MCTL.R = 0xD000A50F;  /*! Write Mode and Key inverted | MC_ME.MCTL */
```

12 Security

All SPC58 family devices have a comprehensive set of customer-configurable security features designed to protect code and data from unauthorized access. Some security features are available in all device configurations while other more advanced security features are not available in all device configurations. These more advanced features require more than simply being turned on or off; they require significant involvement for implementation. Consequently, they provide an important opportunity to define the security level of the device.

12.1 Basic security

All SPC58xEx/SPC58xGx devices have the following basic set of security features.

- Device security feature based on the life cycle model with code and data access progressively more restricted as device matures through defined life cycle steps
- Memory security features:
  - NVM censorship support, password protection, one-time-programmable (OTP) flash memory areas, Flash erase counter and tamper detection
  - SRAM and caches initialized to a constant value after reset (power-on reset, when in a test mode, when not booting from internal flash memory); MBIST functionality used in case of power-on and destructive resets to initialize RAMs selected via configuration in UTEST flash; BAF software routines to initialize RAMs when not booting from internal flash memory
- Monitoring of operating conditions
- Unique ID for each device: each SPC58xEx/SPC58xGx device has a unique identification number stored in an OTP flash memory area which can be read by any of the cores on the device
- Secure watchdog timer
- Debugger access control

12.2 Advanced security

Depending on the device configuration, the following features are available:

- Customer programmable Hardware Security Module (HSM): a dedicated security subsystem that includes a processor core, dedicated SRAM, an encryption module, and exclusive access to secure areas of device flash memory; HSM runs code independently from the main device processor cores and can be used to implement advanced security and monitoring functions
- Advanced debugger access control
- Boot modes:
  - Trusted/secure boot support
  - Handshake with BAF supported

12.3 Detailed security information

Details of most of the SPC58xEx/SPC58xGx security features are published in a separate *SPC58xEx/SPC58xGx Microcontroller Security Reference Manual*, which is only available to qualified customers.

13 Calibration and Debug

This chapter discusses the device-specific information for the debug and calibration modules. This device implements a high-speed serial Nexus trace auxiliary port. The Aurora interface allows the Nexus protocol information to be transmitted serially at high speed over four Aurora lanes. The Aurora protocol handles the encoding of the data and striping the information across the four Aurora lanes available on the device. In addition, the LFAST module, a high speed calibration interface, is supported for run control and memory access. The LFAST signals are alternate functions on the device's JTAG pins. The device starts up in JTAG mode and the tool can request that the JTAG interface be converted to the LFAST high-speed interface. All standards used have freely available specifications for tool developers.

The debug and calibration features for this device are provided in two separate dies: the Production die and the Buddy die, or Buddy Device (BD). The Production die is packaged alone in the Production Device (PD) package. The PD is also assembled into an Emulation and Debug Device (ED) package that consists of the Production die along with the Buddy die. The Buddy device supports additional debug capability. The Nexus Aurora interface is only available on the ED. The PD only supports trace to the internal trace/calibration memory.

The Event Out ($\overline{\text{EVTO}}$[0,1]) signals are connected to the external TGOUT[0,1] device pins and the Event In ($\overline{\text{EVTI}}$) signals are connected to the external TGIN[0,1] device pins.

13.1 Core debug support

Internal debug support in the e200z420n3 and e200z425n3 cores allows for software and hardware debug by providing debug functions, such as instruction and data breakpoints and program trace modes. The cores have the capability of exiting reset halted in debug mode, such that no code runs. This allows debuggers to perform operations such as configuration of calibration remap or edit of overlay memory prior to user code execution.

For details, see *Chapter 71: e200z420n3 Core Debug Support* and *Chapter 70: e200z425n3 Core Debug Support*.

13.2 Run control and memory access

Run control is any operation required to control device operation. Run control includes starting and stopping the processor's execution of code, as well as accessing memory while the device is stopped to download code. The Debug and Calibration Interface (DCI) provides the central run control including cross-triggering of breakpoint conditions. This DCI provides support for the IEEE 1149.1 JTAG standard with the JTAG Controller (JTAGC) as well as the IEEE 1149.7 standard with the Compact JTAG (CJTAG) module. The JTAG interface supports both Boundary Scan and Debug modes. It is based on the IEEE 1149.1 and IEEE 1149.7 standards. The JTAG Controller (JTAGC) operates in the standard IEEE 1149.1 5-wire interface. The JTAGC can either be fed directly from the JTAG pins or the CJTAG module, which supports the IEEE 1149.7 standard.

The Sequence Processing Unit (SPU) provides cross-triggering of events from the device. The SPU can access some of the DCU run control features since it can control the $\overline{\text{EVTO}}$ pins and monitor the $\overline{\text{EVTI}}$ pins. Furthermore, the SPU can use system triggers to cause

many different types of actions, including triggering a breakpoint cross-trigger (managed by the DCI).

Table 165. Reference links to related information

Related module	Reference
Production Device Debug and Calibration Interface (DCI)	*Chapter 72: Debug and Calibration Interface (DCI)*
Buddy Device Debug and Calibration Interface (DCI)	*Chapter 102: Buddy Device: Debug and Calibration Interface (DCI)*
JTAG Controller (JTAGC)	*Chapter 73: JTAG Controller (JTAGC)*
IEEE 1149.7 Compact JTAG Test Access Port Controller (CJTAG)	*Chapter 74: IEEE 1149.7 Compact JTAG Test Access Port Controller (CJTAG)*
JTAG Data Communication (JDC)	*Chapter 75: JTAG Data Communication (JDC)*
Sequence Processing Unit (SPU)	*Chapter 76: Sequence Processing Unit (SPU)*

13.2.1 Debug and Calibration Interface (DCI)

The Debug and Calibration Interface (DCI) module provides debug and calibration features. The DCI module includes the device JTAG Controller and the IEEE 1149.7 interface. The DCI provides the following features:

- Debug mode enable control for connected tools or software
- Port-sharing logic to allow the 5-pin debug port to be shared between the JTAG and the LFAST
- IEEE 1149.1 and 1149.7 controllers
- Debug break and cross-triggering control
- Synchronous restart control for all CPUs when exiting debug mode
- Tool hot plug capability
- Security access control
- Simultaneous operation for debug and calibration features

The DCI sends and receives the $\overline{\text{EVTI}}$[1:0] and $\overline{\text{EVTO}}$[1:0] signals to/from the cores and other modules. *Table 166* summarizes the sources and destinations of $\overline{\text{EVTI}}$, $\overline{\text{EVTO}}$, and related signals for the PD and the BD. For example, for the PD, $\overline{\text{EVTI}}$[0] are outputs of the JTAGM and SPU, and inputs to the Cores, HSM, GTM, and NXMC.

Table 166. PD DCI signals

Signal	Cores	HSM	GTM	NXMC	NAR	JTAGM	SPU	DTS
$\overline{\text{EVTI}}$[0]	I	I	I	I	—	O	I/O	—
$\overline{\text{EVTI}}$[1]	—	—	—	—	—	O	I/O	—
$\overline{\text{EVTO}}$[0]	O	O	O	O	—	I	I/O	—
$\overline{\text{EVTO}}$[1]	—	—	—	—	—	I	I/O	O
Core debug event	I	—	—	—	—	—	—	—
DCI cross triggering	O	—	O	—	—	—	—	—

Table 167. BD DCI signals

Signal	RWA	NAR	NAL	JTAGM
$\overline{\text{EVTI}}$[0]	—	I	—	O
$\overline{\text{EVTI}}$[1]	—	—	—	O
$\overline{\text{EVTO}}$[0]	—	—	—	I
$\overline{\text{EVTO}}$[1]	—	—	—	I

The DCI sends the system debug signal, ipg_debug, to the following modules. For details regarding module functionality during system debug, refer to the individual module chapter.

- AMU
- CAN
- LFAST (Interprocessor Communications instance only)
- DSPI
- eDMA
- FCCU
- GTMINT
- I2C
- NXMC
- PIT
- PSI5
- SARADC
- SDADC
- SRX
- SSCM
- STM
- SWT

13.2.1.1 DCI $\overline{\text{EVTx}}$ Pin Multiplexing Control Register (DCI_PINCR)

The DCI $\overline{\text{EVTx}}$ pin multiplexing control register (DCI_PINCR) is described below. The $\overline{\text{EVTI}}$ and $\overline{\text{EVTO}}$ functions share the same set of possible pins. If this register is written to enable both $\overline{\text{EVTI}}$ and $\overline{\text{EVTO}}$ on the same pin, the $\overline{\text{EVTI}}$ function is selected, although this type of programming should be avoided. Programming the $\overline{\text{EVTI}}$ fields to select $\overline{\text{EVTI}}$[0] or $\overline{\text{EVTI}}$[1] inputs to come from more than one pin is considered a programming error and no pin is enabled.

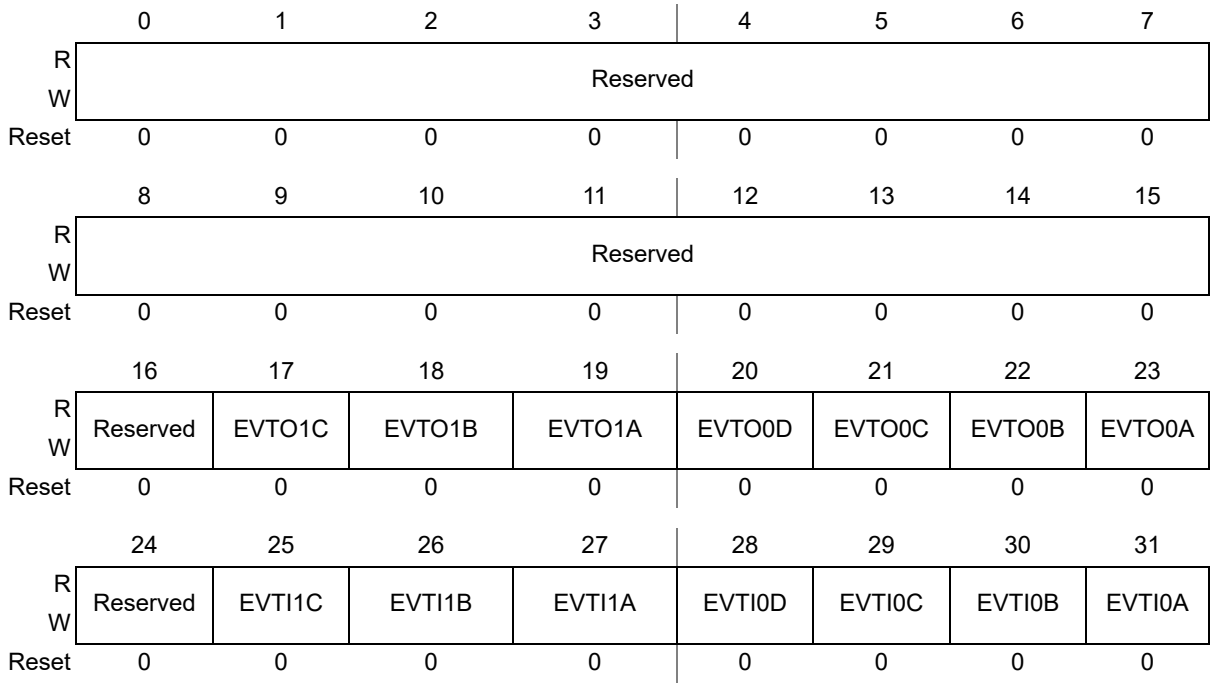


Figure 94. DCI EVTx pin multiplexing control register (DCI_PINCR)

Table 168. DCI_PINCR register field descriptions

Field	Description
17 EVTO1C	Enables $\overline{\text{EVTO}}$[1] function on pin PM[6]
18 EVTO1B	Enables $\overline{\text{EVTO}}$[1] function on pin PK[14]
19 EVTO1A	Enables $\overline{\text{EVTO}}$[1] function on pin PH[11]
20 EVTO0D	Enables $\overline{\text{EVTO}}$[0] function on pin PM[5]
21 EVTO0C	Enables $\overline{\text{EVTO}}$[0] function on pin PM[4]
22 EVTO0B	Enables $\overline{\text{EVTO}}$[0] function on pin PF[15]
23 EVTO0A	Enables $\overline{\text{EVTO}}$[0] function on pin PF[14]
25 EVTI1C	Enables $\overline{\text{EVTI}}$[1] function on pin PM[6]
26 EVTI1B	Enables $\overline{\text{EVTI}}$[1] function on pin PK[14]
27 EVTI1A	Enables $\overline{\text{EVTI}}$[1] function on pin PH[11]

Table 168. DCI_PINCR register field descriptions (continued)

Field	Description
28 EVTI0D	Enables $\overline{\text{EVTI}}$[0] function on pin PM[5]
29 EVTI0C	Enables $\overline{\text{EVTI}}$[0] function on pin PM[4]
30 EVTI0B	Enables $\overline{\text{EVTI}}$[0] function on pin PF[15]
31 EVTI0A	Enables $\overline{\text{EVTI}}$[0] function on pin PF[14]

13.2.1.2 JTAG Controller (JTAGC)

The JTAGC block provides the means to test chip functionality and connectivity while remaining transparent to system logic when not in test mode. Testing is performed via a boundary scan technique, as defined in the IEEE 1149.1-2001 standard. All data input to and output from the JTAGC block is communicated in serial format. All Nexus debug registers are accessed and configured via the JTAG Controller.

13.2.1.3 Compact JTAG (CJTAG)

The CJTAG module implements parts of the IEEE 1149.7 standard for test and debug capabilities.

13.2.2 JTAG Data Communication (JDC)

The JTAG Data Communication (JDC) module allows both JTAG and software access to two registers. These two registers are the challenge/response password authorization registers used by the Hardware Security Module (HSM).

13.2.3 Sequence Processing Unit (SPU)

The Sequence Processing Unit (SPU) provides on-device logic analyzer trigger functions, such as performance monitoring, by using internal device signals as trigger sources. Performance monitor functions are available at both the system and CPU level. Complex trigger and system performance monitor functions are implemented in the SPU module. System-level performance monitor functions are integrated into the SPU complex trigger logic, thus allowing the counting and timing of any debug trigger combinations supported by the SPU.

Various clients generate watchpoints and other triggers when operating in Debug mode. The SPU collects these triggers and uses them as conditions for programmable sequences of states and resultant actions. The SPU provides the capability to create complex debug triggers. By configuring each of the events to trigger specific actions, it achieves complex logic-analyzer-like behavior, providing vital real-time visibility and the ability to debug system activities.

13.2.3.1 SPU Level1 Input Mux Configuration registers

The SPU Level1 Mux inputs for the L1SELn registers (where n = 0 to 7) are described in the following tables.

The L1SEL0 register fields are described in the following table.

Table 169. L1SEL0 register field descriptions

Field	Description
MUX 0	000 CPU0 instruction address compare 1 (watchpoint 0) 001 CPU1 instruction address compare 1 (watchpoint 0) 010 CPU2 instruction address compare 1 (watchpoint 0) 011 Reserved 100 Reserved 101 Reserved 110 CPU0 debug event input 1 (watchpoint 10) 111 CPU0 debug event output 2 (watchpoint 20)
MUX 1	000 CPU0 instruction address compare 2 (watchpoint 1) 001 CPU1 instruction address compare 2 (watchpoint 1) 010 CPU2 instruction address compare 2 (watchpoint 1) 011 Reserved 100 Reserved 101 Reserved 110 CPU1 debug event input 1 (watchpoint 10) 111 CPU1 debug event output 2 (watchpoint 20)
MUX 2	000 CPU0 instruction address compare 3 (watchpoint 2) 001 CPU1 instruction address compare 3 (watchpoint 2) 010 CPU2 instruction address compare 3 (watchpoint 2) 011 Reserved 100 Reserved 101 Reserved 110 CPU2 debug event input 1 (watchpoint 10) 111 CPU2 debug event output 2 (watchpoint 20)
MUX 3	000 CPU0 instruction address compare 4 (watchpoint 3) 001 CPU1 instruction address compare 4 (watchpoint 3) 010 CPU2 instruction address compare 4 (watchpoint 3) 011 Reserved 100 Reserved 101 Reserved 110 Reserved 111 Reserved
MUX 4	000 CPU0 instruction address compare 5 (watchpoint 8) 001 CPU1 instruction address compare 5 (watchpoint 8) 010 CPU2 instruction address compare 5 (watchpoint 8) 011 Reserved 100 Reserved 101 Reserved 110 CPU0 debug event input 2 (watchpoint 11) 111 CPU0 debug event output 3 (watchpoint 21)
MUX 5	000 CPU0 instruction address compare 6 (watchpoint 9) 001 CPU1 instruction address compare 6 (watchpoint 9) 010 CPU2 instruction address compare 6 (watchpoint 9) 011 Reserved 100 Reserved 101 Reserved 110 CPU1 debug event input 2 (watchpoint 11) 111 CPU1 debug event output 3 (watchpoint 21)

Table 169. L1SEL0 register field descriptions (continued)

Field	Description
MUX 6	000 CPU0 instruction address compare 7 (watchpoint 14) 001 CPU1 instruction address compare 7 (watchpoint 14) 010 CPU2 instruction address compare 7 (watchpoint 14) 011 Reserved 100 Reserved 101 Reserved 110 CPU2 debug event input 2 (watchpoint 11) 111 CPU2 debug event output 3 (watchpoint 21)
MUX 7	000 CPU0 instruction address compare 8 (watchpoint 15) 001 CPU1 instruction address compare 8 (watchpoint 15) 010 CPU2 instruction address compare 8 (watchpoint 15) 011 Reserved 100 Reserved 101 Reserved 110 Reserved 111 Reserved

The L1SEL1 register fields are described in the following table.

Table 170. L1SEL1 register field descriptions

Field	Description
MUX 8	000 CPU0 instruction address compare 1 (watchpoint 0) 001 CPU1 instruction address compare 1 (watchpoint 0) 010 CPU2 instruction address compare 1 (watchpoint 0) 011 Reserved 100 Reserved 101 Reserved 110 CPU0 debug event output 0 (watchpoint 12) 111 Reserved
MUX 9	000 CPU0 instruction address compare 2 (watchpoint 1) 001 CPU1 instruction address compare 2 (watchpoint 1) 010 CPU2 instruction address compare 2 (watchpoint 1) 011 Reserved 100 Reserved 101 Reserved 110 CPU1 debug event output 0 (watchpoint 12) 111 Reserved
MUX 10	000 CPU0 instruction address compare 3 (watchpoint 2) 001 CPU1 instruction address compare 3 (watchpoint 2) 010 CPU2 instruction address compare 3 (watchpoint 2) 011 Reserved 100 Reserved 101 Reserved 110 CPU2 debug event output 0 (watchpoint 12) 111 Reserved

Table 170. L1SEL1 register field descriptions (continued)

Field	Description
MUX 11	000 CPU0 instruction address compare 4 (watchpoint 3) 001 CPU1 instruction address compare 4 (watchpoint 3) 010 CPU2 instruction address compare 4 (watchpoint 3) 011 Reserved 100 Reserved 101 Reserved 110 Reserved 111 Reserved
MUX 12	000 CPU0 instruction address compare 5 (watchpoint 8) 001 CPU1 instruction address compare 5 (watchpoint 8) 010 CPU2 instruction address compare 5 (watchpoint 8) 011 Reserved 100 Reserved 101 Reserved 110 CPU0 debug event output 1 (watchpoint 13) 111 Reserved
MUX 13	000 CPU0 instruction address compare 6 (watchpoint 9) 001 CPU1 instruction address compare 6 (watchpoint 9) 010 CPU2 instruction address compare 6 (watchpoint 9) 011 Reserved 100 Reserved 101 Reserved 110 CPU1 debug event output 1 (watchpoint 13) 111 Reserved
MUX 14	000 CPU0 instruction address compare 7 (watchpoint 14) 001 CPU1 instruction address compare 7 (watchpoint 14) 010 CPU2 instruction address compare 7 (watchpoint 14) 011 Reserved 100 Reserved 101 Reserved 110 CPU2 debug event output 1 (watchpoint 13) 111 Reserved
MUX 15	000 CPU0 instruction address compare 8 (watchpoint 15) 001 CPU1 instruction address compare 8 (watchpoint 15) 010 CPU2 instruction address compare 8 (watchpoint 15) 011 Reserved 100 Reserved 101 Reserved 110 Reserved 111 Reserved

The L1SEL2 register fields are described in the following table.

Table 171. L1SEL2 register field descriptions

Field	Description
MUX16	000 CPU0 data address compare 1 (watchpoint 4) 001 CPU1 data address compare 1 (watchpoint 4) 010 CPU2 data address compare 1 (watchpoint 4) 011 Reserved 100 CPU0 write to L1CSR0 101 Reserved 110 Reserved 111 Reserved
MUX 17	000 CPU0 data address compare 2 (watchpoint 5) 001 CPU1 data address compare 2 (watchpoint 5) 010 CPU2 data address compare 2 (watchpoint 5) 011 Reserved 100 CPU0 write to L1CSR1 101 Reserved 110 Reserved 111 Reserved
MUX 18	000 CPU0 data address compare 3 (watchpoint 6) 001 CPU1 data address compare 3 (watchpoint 6) 010 CPU2 data address compare 3 (watchpoint 6) 011 Reserved 100 CPU0 write to E2ECTL0 101 Reserved 110 Reserved 111 Reserved
MUX 19	000 CPU0 data address compare 4 (watchpoint 7) 001 CPU1 data address compare 4 (watchpoint 7) 010 CPU2 data address compare 4 (watchpoint 7) 011 Reserved 100 CPU0 write to DMEMCTL0 101 Reserved 110 Reserved 111 Reserved
MUX 20	000 CPU0 data address compare 1 (watchpoint 4) 001 CPU1 data address compare 1 (watchpoint 4) 010 CPU2 data address compare 1 (watchpoint 4) 011 Reserved 100 CPU0 write to IMEMCTL0 101 Reserved 110 CPU2 write to L1CSR0 111 Reserved
MUX 21	000 CPU0 data address compare 2 (watchpoint 5) 001 CPU1 data address compare 2 (watchpoint 5) 010 CPU2 data address compare 2 (watchpoint 5) 011 Reserved 100 CPU0 write to HID0 101 CPU1 write to MSR 110 CPU2 write to L1CSR1 111 Reserved

Table 171. L1SEL2 register field descriptions (continued)

Field	Description
MUX 22	000 CPU0 data address compare 3 (watchpoint 6) 001 CPU1 data address compare 3 (watchpoint 6) 010 CPU2 data address compare 3 (watchpoint 6) 011 Reserved 100 CPU0 write to HID1 101 CPU1 write to IVPR 110 CPU2 write to E2ECTL0 111 Reserved
MUX 23	000 CPU0 data address compare 4 (watchpoint 7) 001 CPU1 data address compare 4 (watchpoint 7) 010 CPU2 data address compare 4 (watchpoint 7) 011 Reserved 100 CPU0 write to BUCSR 101 CPU1 write to SPEFSCR 110 CPU2 write to DMEMCTL0 111 Reserved

The L1SEL3 register fields are described in the following table.

Table 172. L1SEL3 register field descriptions

Field	Description
MUX 24	000 CPU0 data address compare 1 (watchpoint 4) 001 CPU1 data address compare 1 (watchpoint 4) 010 CPU2 data address compare 1 (watchpoint 4) 011 Reserved 100 CPU0 write to SPEFSCR 101 CPU1 write to BUCSR 110 CPU2 write to IMEMCTL0 111 Reserved
MUX 25	000 CPU0 data address compare 2 (watchpoint 5) 001 CPU1 data address compare 2 (watchpoint 5) 010 CPU2 data address compare 2 (watchpoint 5) 011 Reserved 100 CPU0 write to IVPR 101 CPU1 write to HID1 110 CPU2 write to HID0 111 Reserved
MUX 26	000 CPU0 data address compare 3 (watchpoint 6) 001 CPU1 data address compare 3 (watchpoint 6) 010 CPU2 data address compare 3 (watchpoint 6) 011 Reserved 100 CPU0 write to MSR 101 CPU1 write to HID0 110 CPU2 write to HID1 111 Reserved

Table 172. L1SEL3 register field descriptions (continued)

Field	Description
MUX 27	000 CPU0 data address compare 4 (watchpoint 7) 001 CPU1 data address compare 4 (watchpoint 7) 010 CPU2 data address compare 4 (watchpoint 7) 011 Reserved 100 Reserved 101 CPU1 write to IMEMCTL0 110 CPU2 write to BUCSR 111 Reserved
MUX 28	000 CPU0 data address compare 1 (watchpoint 4) 001 CPU1 data address compare 1 (watchpoint 4) 010 CPU2 data address compare 1 (watchpoint 4) 011 Reserved 100 Reserved 101 CPU1 write to DMEMCTL0 110 CPU2 write to SPEFSCR 111 Reserved
MUX 29	000 CPU0 data address compare 2 (watchpoint 5) 001 CPU1 data address compare 2 (watchpoint 5) 010 CPU2 data address compare 2 (watchpoint 5) 011 Reserved 100 Reserved 101 CPU1 write to E2ECTL0 110 CPU2 write to IVPR 111 Reserved
MUX 30	000 CPU0 data address compare 3 (watchpoint 6) 001 CPU1 data address compare 3 (watchpoint 6) 010 CPU2 data address compare 3 (watchpoint 6) 011 Reserved 100 Reserved 101 CPU1 write to L1CSR1 110 CPU2 write to MSR 111 Reserved
MUX 31	000 CPU0 data address compare 4 (watchpoint 7) 001 CPU1 data address compare 4 (watchpoint 7) 010 CPU2 data address compare 4 (watchpoint 7) 011 Reserved 100 Reserved 101 CPU1 write to L1CSR0 110 Reserved 111 Reserved

The L1SEL4 register fields are described in the following table.

Table 173. L1SEL4 register field descriptions

Field	Description
MUX 32	000 CPU0 performance monitor counter 0 (watchpoint 23) 001 CPU1 performance monitor counter 0 (watchpoint 23) 010 CPU0 memory protection unit (watchpoint 27) 011 Reserved 100 Reserved 101 Reserved 110 Production device NAR full 111 GTM watchpoint 0
MUX 33	000 CPU0 performance monitor counter 1 (watchpoint 24) 001 CPU1 performance monitor counter 1 (watchpoint 24) 010 CPU1 memory protection unit (watchpoint 27) 011 Reserved 100 Reserved 101 Reserved 110 Production device NAR partial full 111 GTM watchpoint 1
MUX 34	000 CPU0 performance monitor counter 2 (watchpoint 25) 001 CPU1 performance monitor counter 2 (watchpoint 25) 010 CPU2 memory protection unit (watchpoint 27) 011 Reserved 100 Reserved 101 Reserved 110 Buddy device NAR partial full 111 GTM watchpoint 2
MUX 35	000 CPU0 performance monitor counter 3 (watchpoint 26) 001 CPU1 performance monitor counter 3 (watchpoint 26) 010 Reserved 011 Reserved 100 Reserved 101 Reserved 110 Buddy device NAR full 111 GTM watchpoint 3
MUX 36	000 CPU2 performance monitor counter 0 (watchpoint 23) 001 Reserved 010 CPU0 performance monitor event input (watchpoint 22) 011 Reserved 100 Reserved 101 Reserved 110 Reserved 111 GTM watchpoint 4
MUX 37	000 CPU2 performance monitor counter 1 (watchpoint 24) 001 Reserved 010 CPU1 performance monitor event input (watchpoint 22) 011 Reserved 100 NXMC1 watchpoint 7 101 Reserved 110 Reserved 111 GTM watchpoint 5

Table 173. L1SEL4 register field descriptions (continued)

Field	Description
MUX 38	000 CPU2 performance monitor counter 2 (watchpoint 25) 001 Reserved 010 CPU2 performance monitor event input (watchpoint 22) 011 Reserved 100 Reserved 101 Reserved 110 Reserved 111 GTM watchpoint 6
MUX 39	000 CPU2 performance monitor counter 3 (watchpoint 26) 001 Reserved 010 Reserved 011 Reserved 100 Reserved 101 Reserved 110 Reserved 111 GTM watchpoint 7

The L1SEL5 register fields are described in the following table.

Table 174. L1SEL5 register field descriptions

Field	Description
MUX 40	000 CPU0 interrupt (watchpoint 16) 001 CPU0 critical interrupt (watchpoint 18) 010 CPU0 data trace control range 1 (watchpoint 29) 011 Reserved 100 Reserved 101 Reserved 110 SPU counter event 0 111 GTM watchpoint 8
MUX 41	000 CPU1 interrupt (watchpoint 16) 001 CPU1 critical interrupt (watchpoint 18) 010 CPU1 data trace control range 1 (watchpoint 29) 011 Reserved 100 Reserved 101 Reserved 110 SPU counter event 1 111 GTM watchpoint 9
MUX 42	000 CPU2 interrupt (watchpoint 16) 001 CPU2 critical interrupt (watchpoint 18) 010 CPU2 data trace control range 1 (watchpoint 29) 011 Reserved 100 Reserved 101 Reserved 110 SPU counter event 2 111 GTM watchpoint 10

Table 174. L1SEL5 register field descriptions (continued)

Field	Description
MUX43	000 Reserved 001 Reserved 010 Reserved 011 Reserved 100 Reserved 101 Reserved 110 SPU counter event 3 111 GTM watchpoint 11
MUX 44	000 CPU0 return (watchpoint 17) 001 CPU0 critical return (watchpoint 19) 010 CPU0 data trace control range 2 (watchpoint 30) 011 Reserved 100 Reserved 101 Reserved 110 SPU counter event 4 111 Reserved
MUX 45	000 CPU1 return (watchpoint 17) 001 CPU1 critical return (watchpoint 19) 010 CPU1 data trace control range 2 (watchpoint 30) 011 Reserved 100 Reserved 101 Reserved 110 SPU counter event 5 111 Reserved
MUX 46	000 CPU2 return (watchpoint 17) 001 CPU2 critical return (watchpoint 19) 010 CPU2 data trace control range 2 (watchpoint 30) 011 Reserved 100 Reserved 101 Reserved 110 SPU counter event 6 111 Reserved
MUX 47	000 Reserved 001 Reserved 010 Reserved 011 Reserved 100 Reserved 101 Reserved 110 SPU counter event 7 111 Reserved

The L1SEL6 register fields are described in the following table.

Table 175. L1SEL6 register field descriptions

Field	Description
MUX 48	000 CPU0 context load (using e_lmvsprw instruction) 001 CPU0 context restore (using e_stmvsprw instruction) 010 CPU0 data trace control range 3 (watchpoint 31) 011 NXMC0 watchpoint 0 100 Reserved 101 Reserved 110 SPU counter event 8 111 Reserved
MUX 49	000 CPU1 context load (using e_lmvsprw instruction) 001 CPU1 context restore (using e_stmvsprw instruction) 010 CPU1 data trace control range 3 (watchpoint 31) 011 NXMC0 watchpoint 1 100 Reserved 101 Reserved 110 SPU counter event 9 111 Reserved
MUX 50	000 CPU2 context load (using e_lmvsprw instruction) 001 CPU2 context restore (using e_stmvsprw instruction) 010 CPU2 data trace control range 3 (watchpoint 31) 011 NXMC0 watchpoint 2 100 Reserved 101 Reserved 110 SPU counter event 10 111 Reserved
MUX 51	000 Reserved 001 Reserved 010 Reserved 011 NXMC0 watchpoint 3 100 Reserved 101 Reserved 110 SPU counter event 11 111 Reserved
MUX 52	000 CPU0 interrupt current priority match to C0PIS 001 CPU0 write to ME bits of MSR 010 Reserved 011 NXMC0 watchpoint 4 100 Reserved 101 Reserved 110 SPU counter event 12 111 Reserved
MUX 53	000 CPU1 interrupt current priority match to C1PIS 001 CPU1 write to ME bits of MSR 010 Reserved 011 NXMC0 watchpoint 5 100 Reserved 101 Reserved 110 SPU counter event 13 111 Reserved

Table 175. L1SEL6 register field descriptions (continued)

Field	Description
MUX 54	000 CPU2 interrupt current priority match to C2PIS 001 CPU2 write to ME bits of MSR 010 Reserved 011 NXMC0 watchpoint 6 100 Reserved 101 Reserved 110 SPU counter event 14 111 Reserved
MUX 55	000 Reserved 001 Reserved 010 Reserved 011 NXMC0 watchpoint 7 100 Reserved 101 Reserved 110 SPU counter event 15 111 Reserved

The L1SEL7 register fields are described in the following table.

Table 176. L1SEL7 register field descriptions

Field	Description
MUX 56	000 CPU0 write to EE bits of MSR 001 CPU0 write to CE bits of MSR 010 NXMC1 watchpoint 0 011 Reserved 100 Reserved 101 Reserved 110 Reserved 111 DCI EVTI0
MUX 57	000 CPU1 write to EE bits of MSR 001 CPU1 write to CE bits of MSR 010 NXMC1 watchpoint 1 011 Reserved 100 Reserved 101 Reserved 110 Reserved 111 DCI EVTI1
MUX 58	000 CPU2 write to EE bits of MSR 001 CPU2 write to CE bits of MSR 010 NXMC1 watchpoint2 011 Reserved 100 Reserved 101 Reserved 110 Reserved 111 DCI EVTO0

Table 176. L1SEL7 register field descriptions (continued)

Field	Description
MUX 59	000 Reserved 001 Reserved 010 NXMC1 watchpoint 3 011 Reserved 100 Reserved 101 Reserved 110 Reserved 111 DCI EVTO1
MUX 60	000 CPU0 processor exception vector match to C0PEVP 001 CPU0 write to DE bits of MSR 010 NXMC1 watchpoint 4 011 Reserved 100 Reserved 101 Reserved 110 Reserved 111 DCI system halt
MUX 61	000 CPU1 processor exception vector match to C1PEVP 001 CPU1 write to DE bits of MSR 010 NXMC1 watchpoint 5 011 Reserved 100 Reserved 101 Reserved 110 Reserved 111 Reserved
MUX 62	000 CPU2 processor exception vector match to C2PEVP 001 CPU2 write to DE bits of MSR 010 NXMC1 watchpoint 6 011 Reserved 100 Reserved 101 Reserved 110 Reserved 111 Reserved
MUX 63	000 Reserved 001 Reserved 010 Reserved 011 Reserved 100 Reserved 101 Reserved 110 Reserved 111 Reserved

13.3 Calibration interface

This device includes an alternate calibration interface for high-speed run control and advanced software control. This calibration interface is based on the LFAST interface. The LFAST based calibration has the ability to perform all of the functions that JTAG supports, only at a faster speed. The MCU implements the slave module, and the pins for the LFAST interface are multiplexed onto the same pins as the JTAG pins. Initially, the device powers up with the JTAG module in control of the pins. Under tool commands, the LFAST interface

can be enabled and the pins become LFAST signals. Once the LFAST interface has been activated, all run control and other JTAG operations are converted in the MCU from LFAST messages into JTAG control operations of the JTAG Controller. The interface is supported entirely in hardware for both LFAST and JTAG modes, with no software or system overhead required to support interface traffic.

Table 177. Reference links to related information

Related module	Reference
JTAG Master (JTAGM)	*Chapter 77: Buddy Device—JTAG Master (JTAGM)*
LFAST Module (LFAST)	*Chapter 80: LVDS Fast Asynchronous Serial Transmission – High Speed Debug*
Development Tool Semaphore (DTS)	*Chapter 81: Development Trigger Semaphore (DTS)*

13.3.1 JTAG Master (JTAGM)

The JTAG Master (JTAGM) acts as JTAG master inside the device. The module has a parallel interface that can exchange data with another serial communication module (such as the LFAST module) or through software.

The data transferred to this module is transformed to produce TCK, TMS, TDI, and TRST outputs and to accept TDO inputs. The module allows these five signals to connect to the JTAGC as if the JTAG data were coming from an external tool. The JTAGM generates all required JTAG scan chains to allow software and high-speed serial communication access to all JTAG-mapped resources.

13.3.2 Debug LFAST

The high-speed calibration interface provides up to a 320 Mbit/s interface between a calibration tool and the device. The calibration interface uses the LFAST as the physical layer between the tool and the JTAGM for access to debug resources. In addition, an LFAST switch allows direct access to read and write memory. The calibration debug LFAST module is one of two instantiations of the LFAST module on the device. The calibration LFAST is configured for slave-only operation and is meant to be used as a calibration interface to provide higher (than JTAG) speed debug and calibration operations.The calibration LFAST interface supports full duplex operation, where the available bandwidths for upstream and downstream traffic are independent.

13.3.3 Development Trigger Semaphore (DTS)

The Development Trigger Semaphore (DTS) module enables software to signal an external tool by driving a persistent (affected only by reset or an external tool) signal on an external device pin. There are a variety of ways that this module can be used, including as a component of an external real-time data acquisition system.

Note: When used as a component of a triggered data-acquisition system, Nexus read/write access is via the JTAG interface of the Nexus debug port and is different than the data acquisition protocol defined in the IEEE-ISTO 5001-2003 or IEEE-ISTO 5001-2011 Nexus standards, which use the Nexus auxiliary port.

13.4 Debug over CAN

As well as supporting debug via JTAG and LFAST interfaces, the device supports the use of a CAN interface for debug. This allows debug of application hardware where access to the dedicated JTAG/LFAST interface signals is not easily available. The debug over CAN mechanism is intended to provide basic debug functionality with some reduction in bandwidth and features compared to use of the dedicated JTAG or LFAST interfaces.

Use of the CAN interface for debug purposes requires the use of some M_CAN, eDMA and JTAGM resources. Some initialization of these resources via software is required, but once this initialization has completed, debug over CAN is possible without any further software overhead.

As the debug over CAN scheme generates internal JTAG messages based on received CAN data, all JTAG clients and included debug resources are accessible. Basic device trace capability is also possible by configuring the device trace hardware to stream to a device overlay/trace RAM, which can be read later using debug over CAN.

The debug over CAN scheme supports the following features:

- Supports operation through CAN_SUB_0_M_CAN_1 or CAN_SUB_0_M_CAN_2 interfaces
  - Makes use of M_CAN debug enhancements
  - Debug message selected by filter configuration SFEC/EFEC=111
- Allows debug of hardware where JTAG access is not available
- Debug traffic carried over application CAN bus
- Debug traffic on CAN coexists with application traffic
- No software overhead after initialization
- flexible selection of CAN identifiers for debug use
- Access to all JTAG debug facilities, includes CPU run control

13.5 Nexus Trace Aurora interface

The Aurora interface allows Nexus protocol information to be transmitted serially at high speed over multiple Aurora lanes to provide advanced trace information from the device. The Aurora protocol handles the encoding of the data and the striping of information across the lanes. The Aurora interface is only available on the Emulation and Debug Device. Trace information is transferred to the Buddy Device for transmission via the Aurora interface.

The Aurora information can be used to reconstruct events or operations that occurred inside the microcontroller. Nexus supports the transmission, through a single Aurora port, of information from multiple trace clients within the MCU.Each Nexus message is tagged with the source client identifier and the type of message. The actual trace information that is available depends on the Nexus client; generally, it can be individually enabled (both the type of message and the client). *Table 178* shows a few examples of Nexus messages. All messages can be enabled to include a timestamp.

Table 178. Nexus message types

Message Type	Description
Program trace	Nexus program trace messages transmit any discontinuities in the program flow, such as branches and interrupts. Multiple types of messages can be generated.
Ownership trace	Ownership trace provides information on process identification changes.
Watchpoint trace	Watchpoint messages are generated any time a watchpoint match occurs in the program or data flow, or as a result of SPU actions. These can be programmed for many types of events within the MCU.
Device identification	The device identification message allows information about the MCU to be transmitted upon startup to allow tools to identify the target system MCU type.
Debug status	Debug status messages provide additional information that may be needed to reconstruct software execution, such as whether the device has entered Debug or Low-power mode.
Data acquisition	Data acquisition messages are an optional feature that allows software control of information to be transmitted.
Error	Error messages are transmitted when an error condition occurs, such as internal buffer overflows.

The Nexus trace information can be output to internal memory, or the parallel information can be serialized for transmission over a serial interface through the Nexus Aurora interface. Nexus trace information can also be routed through the Data Write Processing Unit (DWPU) as described in *Section 13.7: Data trace filtering*.

The Nexus blocks do not have memory-mapped registers. The Nexus registers are accessed by the development tool via the JTAG port using a two step process. First, the specific block is selected by loading the corresponding ACCESS_AUX instruction as described in the JTAGC chapter. Next, after the block is selected, the Nexus registers are enabled by loading the enable instruction to the JTAGC Instruction Register (IR). The enable instructions for the different Nexus clients are shown in the following table. After the enable instruction is received, the development tool has access to the Nexus registers of the selected client.

Table 179. Nexus client JTAG enable instructions

Module	Enable instruction	Opcode
Production Die		
Cores	CORE_ENABLE	0x7C (10 bits)
NXMC	NEXUS_ENABLE	0x0 (4 bits)
GTM		
NAR		
SPU		
Buddy Die		
RWA	CORE_ENABLE	0x7C (10 bits)

Table 179. Nexus client JTAG enable instructions (continued)

Module	Enable instruction	Opcode
NAR	NEXUS_ENABLE	0x0 (4 bits)
NAL		
DWPU		

All of the internal JTAG modules are accessed through a single JTAG connection to the device. In the case of the ED, all JTAG communication is performed through the JTAG interface located in the BD. In the production package configuration, access is performed through the JTAG interface located in the PD.

With the hierarchical JTAG configuration, the BD JTAGC is considered the master JTAG TAP controller. The PD JTAGC TAP controller is selected via an AUX_ACCESS instruction. Other TAP controllers residing on the BD are also selected via AUX_ACCESS instructions (NAR, RWA, BD_DCI). Once control is transferred from the BD JTAGC to the PD JTAGC via the AUX_ACCESS instruction, all other PD TAP controllers are then selected via AUX_ACCESS instructions executed by the PD JTAGC. This is also referenced as TAP-sharing and is shown in the following figure.

Note: On the Emulation device, when the PAUSE-DR/UPDATE-DR state is executed to pass control of the JTAG TAP controller, control is always passed to the top level JTAG TAP controller, in other words, the DCI of the BD. For this reason, it is recommended that all JTAG operations, except when the BD clients need to be accessed, that all PD JTAG access be preceded with ACCESS_PD_JTAGC command. If the device is not an ED, the access is ignored.

Figure 95. JTAG hierarchical view

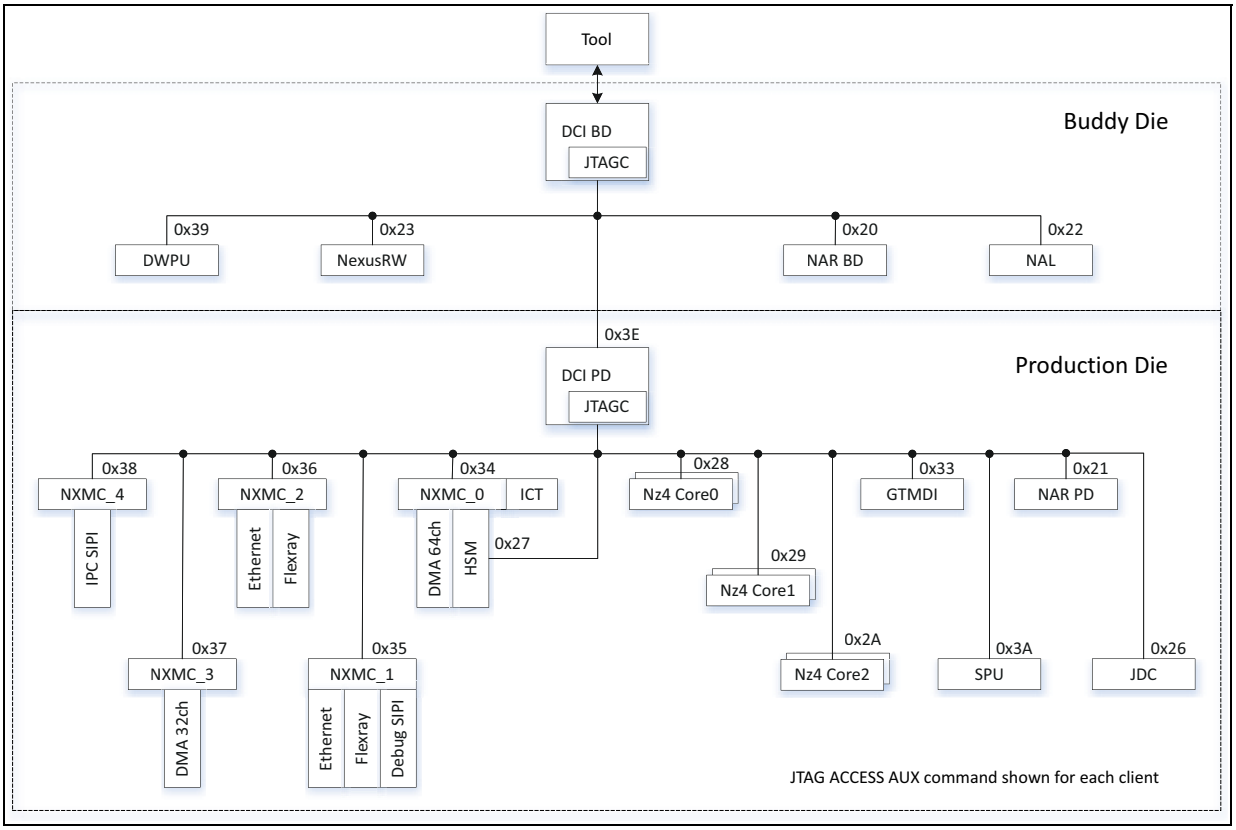


Table 180. Reference links to related information

Related module	Reference
Production Device Nexus Aurora Router (NAR)	*Chapter 82: Nexus Aurora Router (NAR)*
Buddy Device: Nexus Aurora Router (NAR)	*Chapter 105: Buddy Device—Nexus Aurora Router (NAR)*
Buddy Device: Nexus Aurora Link (NAL)	*Chapter 106: Buddy Device—Nexus Aurora Link (NAL)*
Buddy Device: Nexus Aurora PHY (NAP)	*Chapter 107: Buddy Device—Nexus Aurora PHY (NAP)*
Buddy Device: Nexus Read Write Access (RWA)	*Chapter 99: Buddy Device—Nexus Read Write Access (RWA)*

13.5.1 Nexus Aurora overview

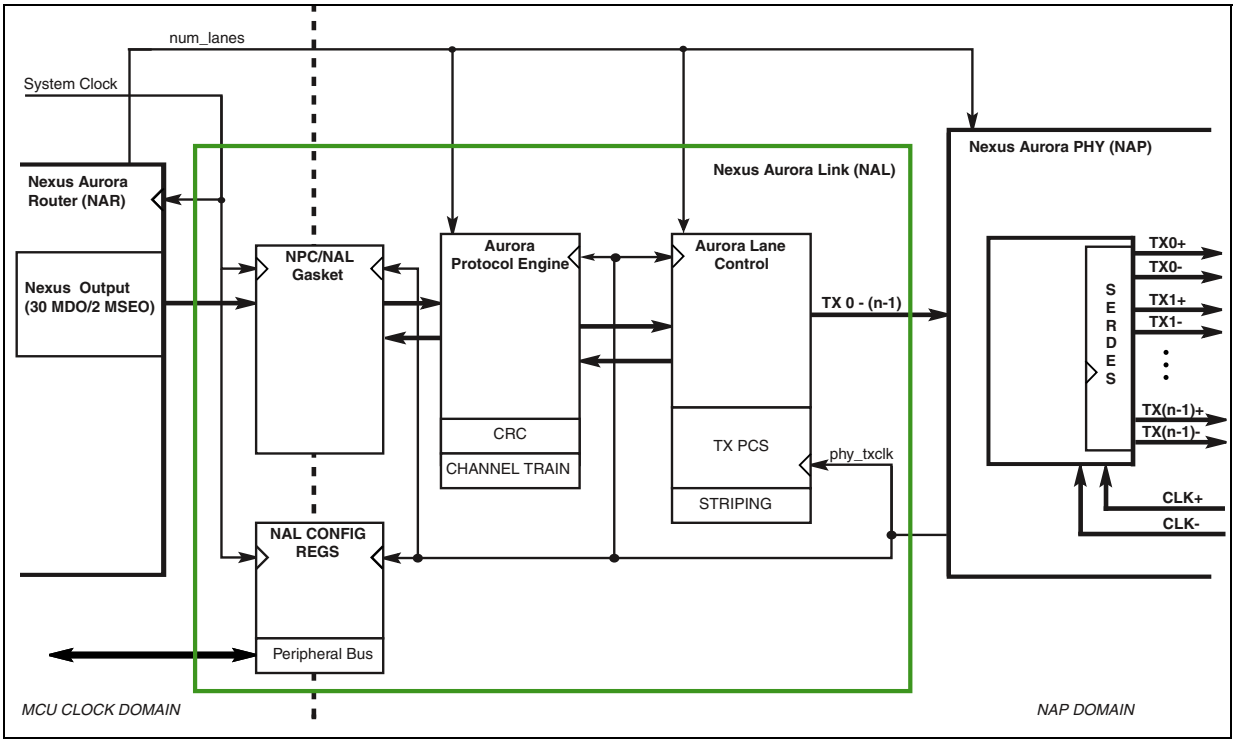
The IEEE-ISTO 5001-2011 standard adds a new type of auxiliary trace port that supports a higher bandwidth. This new interface type uses the Aurora protocol to reformat the parallel Nexus messages into one or more output serial lanes of data, where a lane is defined as a 2-pin Low Voltage Differential Signals (LVDS) interface. The protocol handles striping the data onto the multiple lanes. Each lane uses an 8b/10b encoded LVDS driver. The encoding allows the receiver to recover clocking information.

The Aurora physical interface is based on the XAUI (10 Gigabit Attachment Unit Interface) standard used in communication protocols, which covers the jitter, differential skew, inter-lane skew, and bit error rate requirements.

The Aurora protocol covers lane initialization and channel bonding for both simplex and duplex implementations. The current implementation uses a simplex output-only connection from the MCU to the external tool.

The Aurora interface connects Nexus parallel trace output to an Aurora Protocol Engine (APE), the Aurora Lane Control (ALC), and the Nexus Aurora Physical interface (NAP). *Figure 96* shows the Aurora interface block diagram, where the number of lanes, *n*, is 4.

Figure 96. Aurora interface block diagram

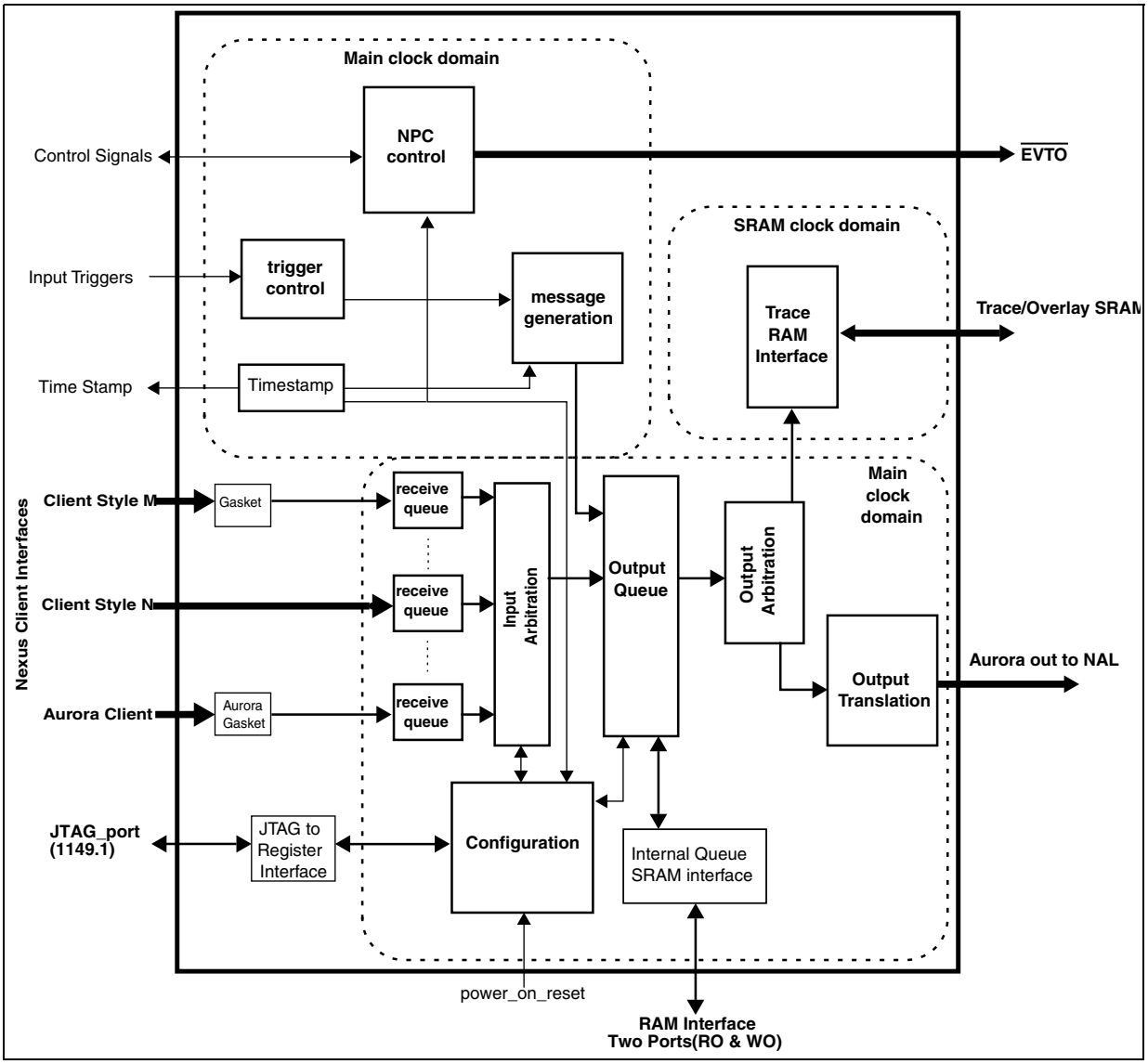


13.5.2 Nexus Aurora Router (NAR)

The Nexus Aurora Router (NAR) manages the Nexus trace information from each of the Nexus clients. The NAR takes the parallel trace information from each of the clients, buffers it to avoid overruns, and routes it to either the internal overlay RAM or the Buddy Device, where it is stored in on-chip SRAM or transmitted out of the device through the Aurora interface.

16 KB of trace SRAM is available on the PD (located in the flash module), and 2 MB of trace SRAM is available in the ED. (Some of this SRAM may be used for calibration overlay content as well as trace content.) Similarly, a smaller NAR output queue is located on the PD (2 KB), and a much larger NAR output queue is located in the ED (8 KB). The physical Aurora pins are only available on the ED. *Figure 97* shows the NAR block diagram.

Figure 97. NAR block diagram



The PD NAR module has ten clients, with client index 1–10. Clients 11–16 are not connected. The client connections and the corresponding NAR client index are shown in *Table 181*. The NAR client index is used in the NAR_STCR, NAR_CSSR, and NAR_CDR registers. The source trace ID is used in the NAR_SFR.

Table 181. PD NAR clients

Client index	Client	Nexus source trace ID (SRC)	Beats per client (Bn)
—	NAR	0b1111 (0xF)	—
1	GTM	0b0111 (0x7)	3
2	HSM	0b0011 (0x3)	1
3	Core 0	0b0000 (0x0)	1

Table 181. PD NAR clients (continued)

Client index	Client	Nexus source trace ID (SRC)	Beats per client (Bn)
4	NXMC Client 0 Concentrator	0b1100 (0xC)	1
5	Core 1	0b0001 (0x1)	1
6	NXMC Client 1 Concentrator	0b1101 (0xD)	1
7	Core 2	0b0010 (0x2)	1
8	NXMC Client 2 Concentrator	0b1000 (0x8)	1
9	NXMC Client 3	0b1001 (0x9)	1
10	NXMC Client 4	0b1010 (0xA)	1
11	—	—	—
12	—	—	—
13	—	—	—
14	—	—	—
15	—	—	—
16	—	—	—

The BD NAR module has one client and the client connections and the corresponding NAR client index are shown in *Table 182*.

Table 182. BD NAR clients

Client index	Client	Nexus source trace ID (SRC)	Beats per client (Bn)
1	PD NAR	0b1110 (0xE)	2

The NAR is limited writing trace information to the overlay RAM. The PD NAR only has access to the 16 KB internal overlay RAM. Therefore, the PD NAR trace memory bus base address registers (NAR_TBALO and NAR_TBAHI) must specify a base address in the range 0x0D000000–0x0D003FFF. The BD NAR also has access to 2 MB of extended overlay RAM. Therefore, the BD NAR trace memory bus base address registers (NAR_TBALO and NAR_TBAHI) must specify a base address in the range 0x0C000000–0x0C1FFFFF.

The NUM_LANE field of the NAR_NAPCR is described in *Table 183*.

Table 183. NAR_NAPCR field descriptions

Field	Description
19:16 NUM_LANE	TX Lane Configuration – Number of enabled TX lanes 0000 Disable NAP and NAL 0001 Reserved 0010 Enable 2 lanes (Lane 0–1) 0011 Reserved 0100 Enable 4 lanes (Lanes 0–3) 0101 Reserved 0110 Reserved 0111 Reserved 1000 Reserved 1001 Reserved ... 1111 Reserved

13.5.3 Nexus Aurora Link (NAL)

The Nexus Aurora Link (NAL) is only included in the Emulation and Debug Device. The Nexus Aurora Link (NAL) consists of all the logic on the MCU needed to implement the connection up to the physical-layer serializer and transmitter (no receiver). It includes Aurora control and data multiplexing, data-encoding and striping, and the Physical Coding Sublayer (PCS) portion of the protocol.

Debug data in the Nexus format from multiple clients is packetized into a 32-bit format by the clients and then collected by the NAR. The NAL encodes the message data (8b/10b), stripes it across the lanes, and inserts Aurora control characters (such as start of message, end of message, idle, clock compensation characters, and so on) around the payload. The NAL PCS drives the Nexus Aurora Physical (NAP) interface.

13.5.4 Nexus Aurora PHY (NAP)

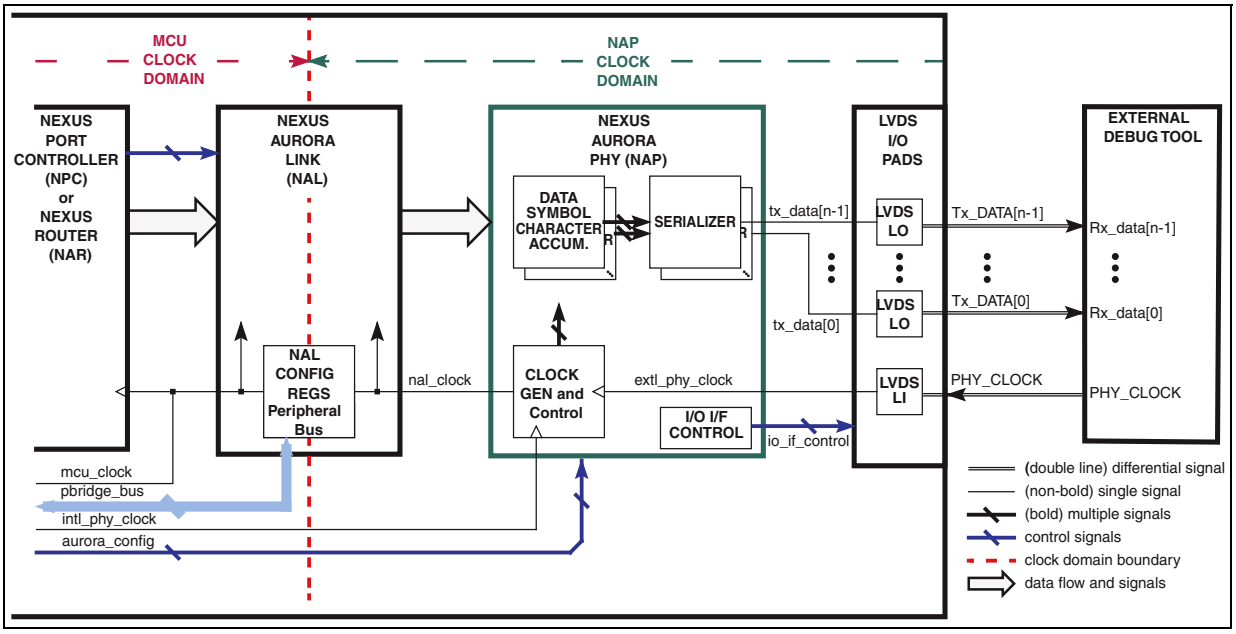
The Nexus Aurora PHY (NAP) is only included in the Emulation and Debug Device. The NAP, in conjunction with the Nexus Aurora Link (NAL), supports one-way simplex high-speed serial communication with an external debugger. The NAP receives 8b/10b encoded data from the ALC block in the NAL and transmits this data onto a given PHY lane's LVDS pads. The NAL performs lane striping with settings in the NAL and NAR specifying lane configuration. Each NAP lane consists of a Data Symbol Character Accumulator that registers the 10-bit character for that lane from the ALC on the rising edge of the NAL clock. A serializer takes the 10-bit character and serializes it into the individual transmit data bits at the PHY clock rate to be transmitted via the LVDS output drivers to the external debugger receive channel.

Note: The NAL performs static link training for an attached debugger PHY. The NAP is not directly aware of the link training operation, since it merely transmits the data presented to it by the NAL.

On the receive data side of the external debugger, it is the responsibility of the external debugger to perform elastic buffering, symbol detection, symbol locking (called “lane alignment” in Aurora terminology), 10b/8b decoding, decode and disparity error tracking, and CRC checking in the received data stream.

Figure 98 shows a block diagram of the Nexus Aurora Physical (NAP) interface and the LVDS output drivers, where the number of lanes, n, is 4.

Figure 98. NAP block diagram



13.6 Nexus clients

Nexus supports the transmission, through a single Aurora port, of information from multiple trace clients within the device.

Table 184. Reference links to related information

Related module	Reference
e200z420n3 Nexus Support	*Chapter 83: e200z420n3 Nexus 3 Module*
e200z425n3 Nexus Support	*Chapter 84: e200z4256 Nexus 3 module*
GTM Development Interface (GTMDI)	*Chapter 49: GTM Development Interface (GTMDI)*
Nexus Crossbar Multi-Master Client (NXMC)	*Chapter 85: Nexus Crossbar Multi-Master Client (NXMC)*

13.6.1 e200z4 Nexus 3

The e200z420n3 and e200z425n3 Nexus 3 modules provide real-time development capabilities for e200z4d processors in compliance with the IEEE-ISTO 5001 standard. This module provides development support capabilities without requiring the use of address and data pins for internal visibility.

The development features supported are Program Trace, Data Trace, Watchpoint Messaging, Ownership Trace, Data Acquisition Messaging, and Read/Write Access via the JTAG interface. The Nexus 3 module also supports two Class 4 features: Watchpoint Triggering and Processor Overrun Control.

13.6.2 GTM Development Interface (GTMDI)

The GTM Development Interface (GTMDI) provides real-time development capabilities for the GTM system including Multi-channel Sequencer (MCS) cores, Advanced Routing Unit (ARU) data trace, Input Channel Filters trace, Output Timer channels, DPLL and Sensor Pattern Evaluation (SPE) module submodules. The GTMDI interfaces to the GTM IP using dedicated signals. The GTMDI operates in conjunction with on-chip debug modules to implement complex debug features. Trace capability is also provided along with a global timestamp that allows real-time trace for several GTM submodule events. The GTMDI implements a Nexus client standard interface and a JTAG standard port controller to provide communication capability via the JTAG port and the on-chip NAR.

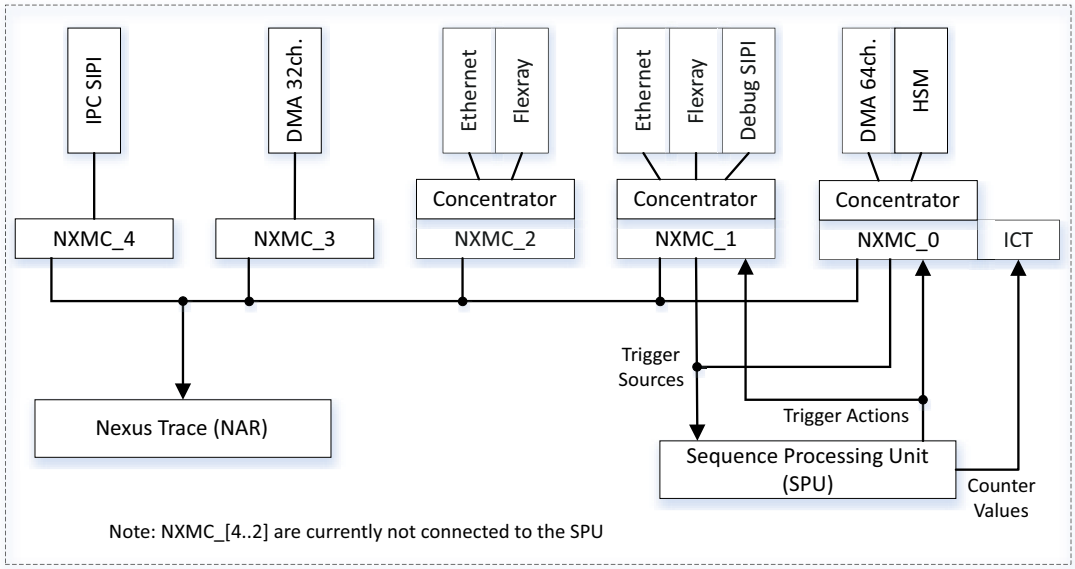
13.6.3 Nexus Crossbar Multi-master Client (NXMC)

The Nexus Crossbar Multi-master Client (NXMC) module provides real-time trace capabilities in compliance with the IEEE-ISTO Nexus 5001-2011 standard. This module provides development support capabilities for SoCs without requiring address and data pins for internal visibility. A portion of the pin interface is also compliant with the IEEE 1149.1 JTAG standard.

To provide Nexus data trace messaging from bus masters on the master ports of the crossbar (XBAR) that do not inherently have a trace client, a Nexus Crossbar Multi-Master Client (NXMC) monitors traffic into the master port of the XBAR. Five NXMCs are implemented, out of which only two interact with the SPU. Trace messages transmitted over the Nexus interface (or stored in trace memory) include which NXMC generated the message as well as an identifier for the pre-concentrator source of the message. In addition to XBAR bus master trace capability, the NXMC also allows for in-circuit trace capability that can be used to trace the Sequence Processing Unit (SPU) counter values. The two external hardware watchpoint triggers of each NXMC are used to generate timestamps (four in total—two from each NXMC) for events from the SPU.

The following figure shows a block diagram illustrating the NXMC integration.

Figure 99. NXMC integration block diagram



The values of the trace master ID or in-circuit client ID, along with the Nexus message source identifier for the different clients are shown in the following table.

Table 185. NXMC source and master client IDs

Client		Nexus trace source ID (SRC)	Trace master ID or In-circuit trace client ID (MASTER)
In-Circuit Trace	SPU	0b1011	0b0000
XBAR Client 0 Concentrator	Hardware Security Module (HSM) data trace	0b1100	0b0000
	eDMA_1 Controller		0b0001
XBAR Client 1 Concentrator	Debug SIPI	0b1101	0b1000
	FlexRay_0		0b1001
	Ethernet_0		0b1010
XBAR Client 2 Concentrator	FlexRay_1	0b1000	0b0100
	Ethernet_1		0b0101
direct master	eDMA_0 controller	0b1001	ob1100
direct master	IPC SIPI	ob1010	ob1110
Data Write Processor Unit (DWPU)	e200zx core	—(1)	0b1111

1. The Nexus trace Source ID is the same as the original core SRC.

13.7 Data trace filtering

The Data Write Processing Unit (DWPU) supports data trace filtering as well as out of range parameter identification.

Table 186. Reference links to related information

Related module	Reference
Data Write Processing Unit (DWPU)	*Chapter 98: Buddy Device—Data Write Processing Unit (DWPU)*

DWPU operates in conjunction with the BD NAR and a small RAM array. DWPU receives its input trace stream from the BD NAR Nexus Out Link. The trace stream is initially decompressed (which includes conversion of relative address and data values to absolute values) to ease subsequent filtering and comparison operations. The decompression process tracks multiple contexts, with individual context tracking required for each supported Nexus client.

The processed stream then passes through an address compare block, that generates tag RAM addresses corresponding to a portion of each data write address within the trace stream. Another portion of the data write addresses is used to select a particular bit from the data word returned by the tag RAM, so that each tag RAM bit corresponds to a unique 64-bit data write address. The value of the selected bit determines whether the data write trace occurrence is passed out of the DWPU back to the BD NAR, or is removed from the trace stream. DWPU returns the data trace filtered and re-formatted stream back to a secondary Nexus client input port on the BD NAR.

14 e200z4d Core Complex Overview

The SPC58xEx/SPC58xGx has five separate cores that perform various computational and control functions. They are organized in three processing channels, two of which with a functional core and a lock step replica.

- The main computational shell consists of 2 processing channels (Main Core_0 and Main Core_1), both of which use an e200z420n3 core. These two cores are used for general computational functions.
- Only Main Core_0 uses an e200z419 core (Checker Core_0) as lock step replica. The e200z419 is a true subset of the e200z420n3 core. When enabled, the checker core operates in lock step mode with the associated main core executing exactly the same instructions. Thus, the checker core insures that the main core is executing correctly.
- The third processing channel is implemented with a lock-step pair of e200z425s and is generally used to control various I/O modules.

This chapter provides overviews of the e200z420n3, the e200z419, and the e200z425n3 cores.

The e200z420n3, e200z419, and e200z425n3 cores are very similar. The register set for all these three cores is identical. The e200z425n3 core differs in hardware from the e200z420n3 core in several ways. The e200z425n3 core and e200z420n3 core have different amount of Instruction Cache. The e200z425n3 has no Data Cache. Also, the e200z425n3 incorporates a Lightweight Signal Processing Extension (LSP) APU to provide support for real time SIMD fixed point embedded numeric operations. The e200z419 core differs in hardware from the e200z420n3 in that no Instruction Cache, Data Cache, I-memory or D-memory are implemented.

14.1 Overview of the e200z420n3, e200z419, and e200z425n3 cores

The e200z processor family is a set of CPU cores that implement low-cost versions of the Power architecture.

The e200z420n3, e200z419, and e200z425n3 are dual-issue, 32-bit Power VLE compliant designs with 32-bit general purpose registers (GPRs).

An Embedded Floating-point (EFPU2) APU is provided to support real-time single-precision embedded numeric operations using the general-purpose registers.

The e200z420n3, e200z419, and e200z425n3 cores implement the VLE (variable-length encoding) ISA, providing improved code density. The VLE ISA is further documented in *PowerISA 2.06*, a separate document. The base *PowerISA 2.06* fixed-length 32-bit instruction set is not directly supported.

The e200z420n3, e200z419, and e200z425n3 processor cores integrate a pair of integer execution units, a branch control unit, instruction fetch unit, load/store unit, and a multi-ported register file capable of sustaining six read and three write operations per clock. Most integer instructions execute in a single clock cycle. Branch target prefetching is performed by the branch unit to allow single-cycle branches in many cases.

A Nexus Class 3+ module is integrated into each e200z420n3, and e200z425n3 processor core.

A Memory Protection Unit is integrated in each e200z420n3, e200z419, and e200z425n3 processor core.

14.2 Features

The following is a list of some of the key features of the e200z420n3, e200z419, and e200z425n3 processor cores:

- Dual issue, 32-bit *PowerISA 2.06*-VLE compliant CPU
- In-order execution and retirement
- Precise exception handling
- Branch processing unit
  - Dedicated branch address calculation adder
  - Branch target prefetching using BTB
  - Return Address Stack
- Load/store unit
  - 2 cycle load latency
  - Fully pipelined
  - Misaligned access support
- 32-bit General Purpose Register (GPR) file
- Dual AHB 2.v6 64-bit System buses
- Local Instruction and Data Memories (IMEM, DMEM) with shared AHB 2.v6 64-bit slave interface
  - e200z420n3 core—16 KB IMEM, 64 KB DMEM
  - e200z425n3 core—16 KB IMEM, 32 KB DMEM
  - e200z419 core does not contain either Instruction or Data MEM
- Memory Protection Unit (MPU) implementing a 24-entry region descriptor table with support for 6 arbitrary-sized instruction memory regions, 12 arbitrary-sized data memory regions, and 6 additional arbitrary-sized instruction or data memory regions
- 2-Way Set Associative Harvard Instruction and Data Cache
  - e200z420n3 core—8 KB ICache, 4 KB DCache
  - e200z425n3 core—8 KB ICache, no DCache
  - e200z419 core does not contain either Instruction or Data Cache
- Embedded Floating-point APU (EFPU2) supporting single-precision floating-point operations
- Lightweight Signal Processing (LSP) APU (e200z425n3 core only) supporting fixed-point multiply/accumulate and integer/fixed point operations using the 32-bit General Purpose Register file.
- Performance Monitor APU supporting execution profiling
- Nexus Class 3-plus Real-time Development Unit
- Power management
  - Low power design with extensive clock gating
  - Power saving modes: doze, nap, sleep, wait
  - Dynamic power management of execution units, caches and Local memories
- Testability
  - Synthesizeable, MuxD scan design
  - ABIST/MBIST for arrays
  - Built-in LBIST unit

14.3 Microarchitecture summary

The e200z420n3, e200z419, and e200z425n3 processor cores utilize a five stage instruction pipeline with two stages for execution. The Instruction Fetch, Instruction Decode/Register File Read/ EA Calc, Execute 0/ Memory Access0, Execute1/Memory Access1, and Register Writeback stages operate in an overlapped fashion allowing single clock instruction execution for most instructions. *Figure 100* shows a block diagram of the e200z architecture.

The integer execution units each consists of a 32-bit Arithmetic Unit (AU), a Logic Unit (LU), a 32-bit Barrel shifter (Shifter), a Mask-Insertion Unit (MIU), a Condition Register manipulation Unit (CRU), a Count-Leading-Zeros unit (CLZ), and result feed-forward hardware. The integer EU1 also supports hardware division and a 32x32 Hardware Multiplier array.

Most arithmetic and logical operations are executed in a single cycle with the exception of multiply, which is implemented with a pipelined hardware array, and the divide instructions. A Count-Leading-Zeros unit operates in a single clock cycle.

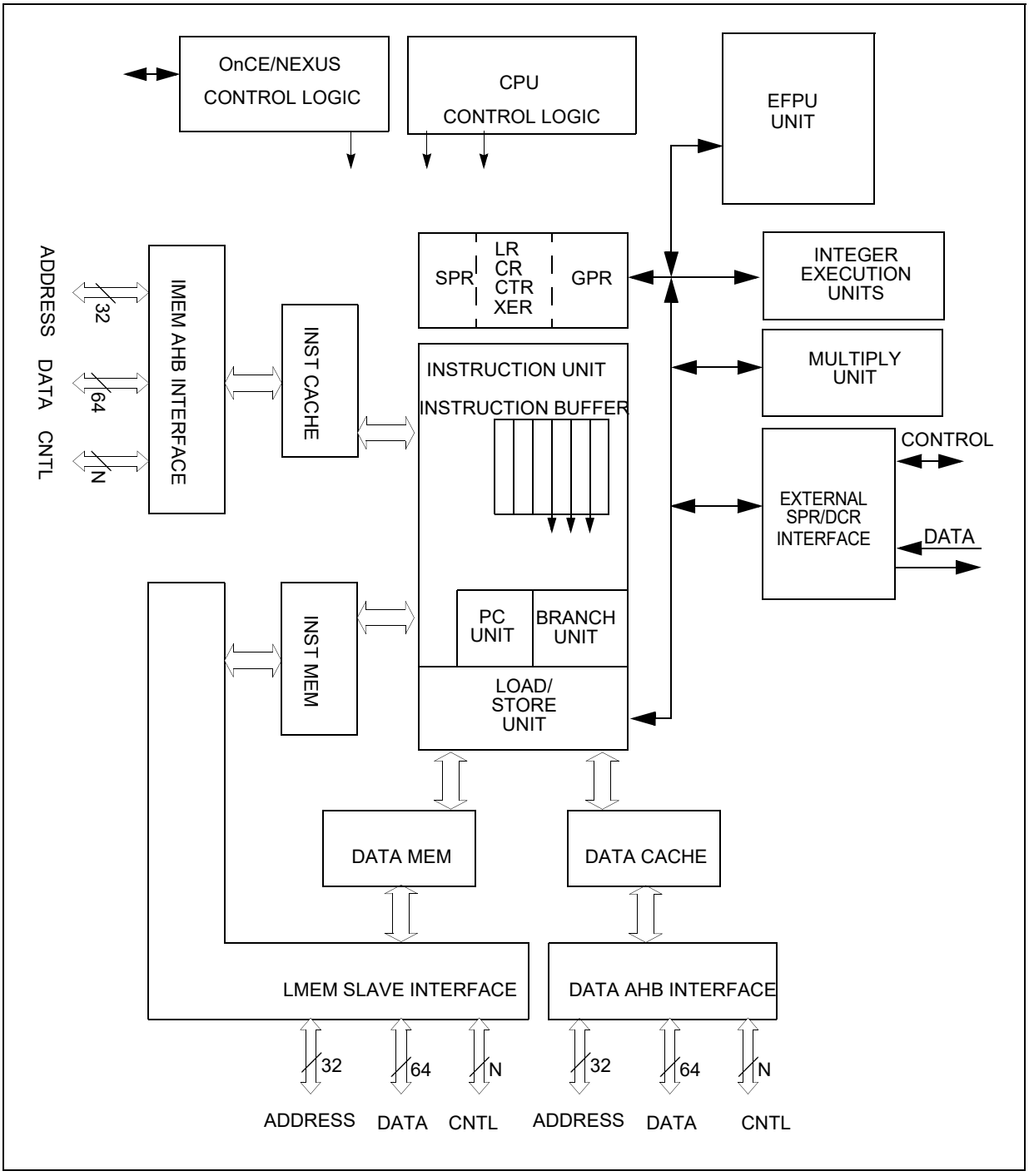
The Instruction Unit contains a PC incrementer and dedicated Branch Address adders to minimize delays during change of flow operations. Sequential prefetching is performed to ensure a supply of instructions into the execution pipeline. Branch target prefetching is performed to accelerate taken branches. Prefetched instructions are placed into an instruction buffer.

Branch target addresses are calculated in parallel with branch instruction decode, resulting in an execution time of two clocks for correctly predicted branches. Conditional branches which are not taken execute in a single clock. Branches with successful BTB target prefetching have an effective execution time of one clock if correctly predicted.

Memory load and store operations are provided for byte, halfword, word (32-bit), and doubleword data with automatic zero or sign extension of byte and halfword load data as well as optional byte reversal of data. These instructions can be pipelined to allow effective single cycle throughput. Load and store multiple word instructions allow low overhead context save and restore operations. The load/store unit contains a dedicated effective address adder to allow effective address generation to be optimized.

The Condition Register unit supports the condition register (CR) and condition register operations defined by the Power architecture. The condition register consists of eight 4-bit fields that reflect the results of certain operations, such as move, integer and floating-point compare, arithmetic, and logical instructions, and provide a mechanism for testing and branching.

Figure 100. e200z block diagram



Vectored and autovectored interrupts are supported by the CPU. Vectored interrupt support is provided to allow multiple interrupt sources to have unique interrupt handlers invoked with no software overhead.

The LSP APU (included only on the e200z425n3 core) supports vector instructions operating on 16-bit and 32-bit fixed-point data types. The EFPU2 APU supports 32-bit IEEE-754 single-precision floating-point formats and operates in a pipelined fashion. The 32-bit

general purpose register file is used for source and destination operands. There is a unified storage model for scalar single-precision floating-point data types of 32-bits and the normal integer type. Low latency floating-point add, subtract, mixed add/subtract, sum, diff, min, max, multiply, multiply-add, multiply-sub, divide, square root, compare, and conversion operations are provided. Most operations can be pipelined.

14.3.1 Instruction unit features

The features of the e200z420n3, e200z419, and e200z425n3 Instruction units are:

- 64-bit path to cache supports fetching of two 32-bit instructions per clock
- Instruction buffer holds up to 8 32-bit instructions
- Dedicated PC incrementer supporting instruction prefetches
- Branch unit with dedicated branch address adder and branch lookahead logic (BTB) supporting single cycle execution of successfully predicted branches

14.3.2 Integer unit features

The e200z420n3, e200z419, and e200z425n3 integer units support single cycle execution of most integer instructions:

- 32-bit AU for arithmetic and comparison operations
- 32-bit LU for logical operations
- 32-bit priority encoder for the count leading zero’s function
- 32-bit single cycle barrel shifter for static shifts and rotates
- 32-bit mask unit for data masking and insertion
- Divider logic for signed and unsigned divide in 4-14 clocks with minimized execution timing (EU1 only)
- Pipelined 32x32 hardware multiplier array supports 32x32->32 multiply with 2 clock latency, 1 clock throughput (EU1 only)

14.3.3 Load/Store unit features

The e200z420n3, e200z419, and e200z425n3 load/store units support load, store, and the load multiple / store multiple instructions:

- 32-bit effective address adder for data memory address calculations
- Pipelined operation supports throughput of one load or store operation per cycle
- Dedicated 64-bit interface to memory supports saving and restoring of up to two registers per cycle for load multiple and store multiple word instructions and context save/restore instructions

14.3.4 MPU features

The features of the MPU are as follows:

- 24-entry fully-associative Range Table
- Arbitrary range size support from 1 byte to 4 GB
- 8-bit Process Identifier
- Bypass capability for individual access types
- Maskable Address and Process Identifier
- Programmable Memory Attributes
- Entry Invalidation Protection
- Write-Once Option
- Alternate Debug Breakpoint/Watchpoint function

14.3.5 Cache features

The features of the Caches are as follows:

- 2-Way Set Associative Harvard Instruction and Data Caches
  - e200z420—8 KB ICache, 4 KB DCache
  - e200z425—8 KB ICache, No DCache
  - e200z419—No ICache, No DCache
- Writethrough Support
- 8-entry Store Buffer
- Linefill Buffer
- 32-bit address bus plus attributes and control
- Separate uni-directional 64-bit read data bus and 64-bit write data bus
- Supports Way locking
- Supports Write allocation policies
- Supports multi-bit EDC with Correction/Auto-invalidation capability for the I and D caches
- Hardware Debug Cache Invalidate Support for the Data Cache
- Software and Hardware Debug access to cache tag, status and data storage via dedicated control registers

14.3.6 Local memory features

The features of the Local Instruction and Data Memories are as follows:

- Local Instruction and Data Memories (IMEM, DMEM) with shared AHB 2.v6 64-bit slave interface
  - e200z420n3—16 KB IMEM, 64 KB DMEM
  - e200z425n3—16 KB IMEM, 32 KB DMEM
  - e200z419—No IMEM, No DMEM
- Instruction and data memories (IMEM, DMEM) are not duplicated on e200z419 core.
- Supports zero wait-state access relative to cache hit timings
- Supports multi-bit (SECDED) ECC with correction/scrubbing capability
- 4-entry ECC R-M-W Store Buffer for DMEM
- 32-bit address bus plus attributes and control
- Separate unidirectional 64-bit read data bus and (for DMEM only) 64-bit write data bus

14.3.7 e200z420n3 and e200z425n3 system bus features

The features of the e200z420n3 and e200z425n3 System Bus interface are as follows:

- Independent instruction and data interfaces
- Advanced microcontroller bus architecture (AMBA) AHB2.v6 protocol
- 32-bit address bus, 64-bit data bus, plus attributes and control
- Data interface provides separate uni-directional 64-bit read and write data buses
- Support for HCLK running at a slower rate than CPU clock

14.3.8 e200z419 system features

Checker Core_0 is implemented using an e200z419 core. When enabled, Checker Core_0s is used in Lock Step with Main_Core_0. The e200z419 core can only be enabled or disabled by the user. For the SPC58xEx/SPC58xGx device, the Checker Core_0s has no system bus and cannot independently execute user code. The core is only used to execute the same instruction stream as Main Core 0.

The e200z419 core is identical to the e200z420n3 with five exceptions:

- IMEM memory array is not present
- ICACHE memory array is not present
- DMEM memory array is not present
- DCACHE memory array is not present
- The e200z419 has a pair of system bus interfaces, which are driven/sampled by the delayed lockstep logic

15 Core (z420n3) Description

15.1 Overview of the Core (z420n3)

Core (z420n3) is a dual-issue 32-bit *PowerISA 2.06* VLE compliant design with 32-bit general purpose registers (GPRs). The Core (z420n3) core implements the VLE (variable-length encoding) ISA, providing improved code density. The VLE ISA is further documented in *PowerISA 2.06*, a separate document.

An Embedded Floating-point (EFPU2) APU is provided to support real-time single-precision floating-point embedded numerics operations using the general-purpose registers.

The Core (z420n3) processor integrates a pair of integer execution units, a branch control unit, instruction fetch unit, load/store unit, and a multi-ported register file capable of sustaining six read and three write operations per clock. Most integer instructions execute in a single clock cycle. Branch target prefetching is performed by the branch unit to allow single-cycle branches in many cases.

Core (z420n3) contains an 8 Kbyte Instruction Cache, and a 4 Kbyte Data Cache, as well as a Nexus Class 3+ real-time debug module with support for program and data trace features as well as extensive trace controls.

A Memory Protection Unit is also included which supports protections of various instruction and data memory areas.

15.2 Register Model

This section describes the registers implemented in the Core (z420n3) core. The *PowerISA 2.06* architecture defines register-to-register operations for all computational instructions.

Core (z420n3) extends usage of the General Purpose Registers to support EFPU APU operations operating on the 32-bit GPR registers.

Figure 101 and *Figure 102* show the complete Core (z420n3) register set, all of which are accessible in supervisor mode. *Figure 103* and *Figure 104* show the subset of registers which are accessible while in user mode. The number to the right of the special-purpose registers (SPRs) is the decimal number used in the instruction syntax to access the register (for example, the integer exception register (XER) is SPR 1).

Figure 101. Core (z420n3) Supervisor Mode Programmer's Model SPRs

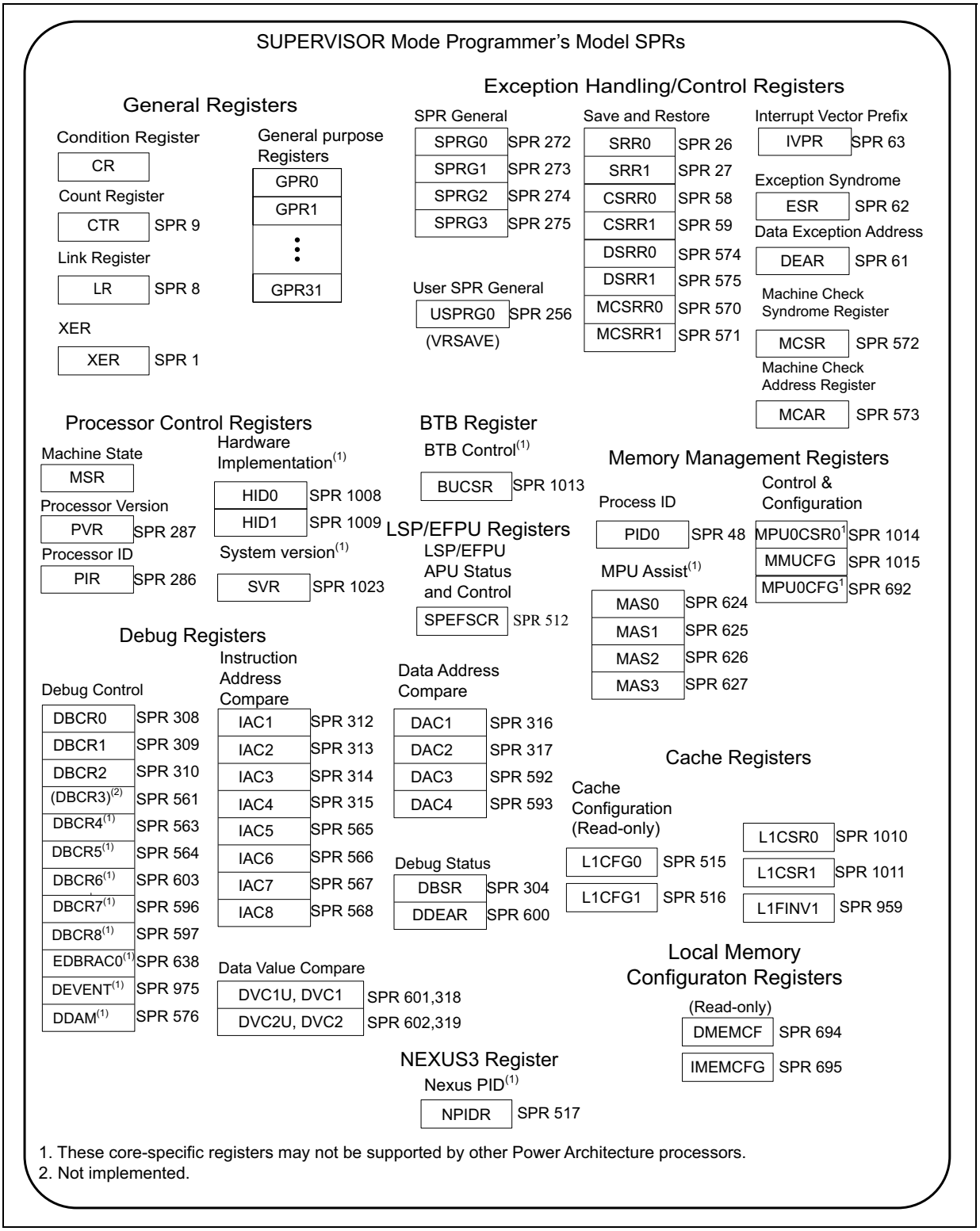


Figure 102. Core (z420n3)Supervisor Mode Programmer's Model DCRs and PMRs

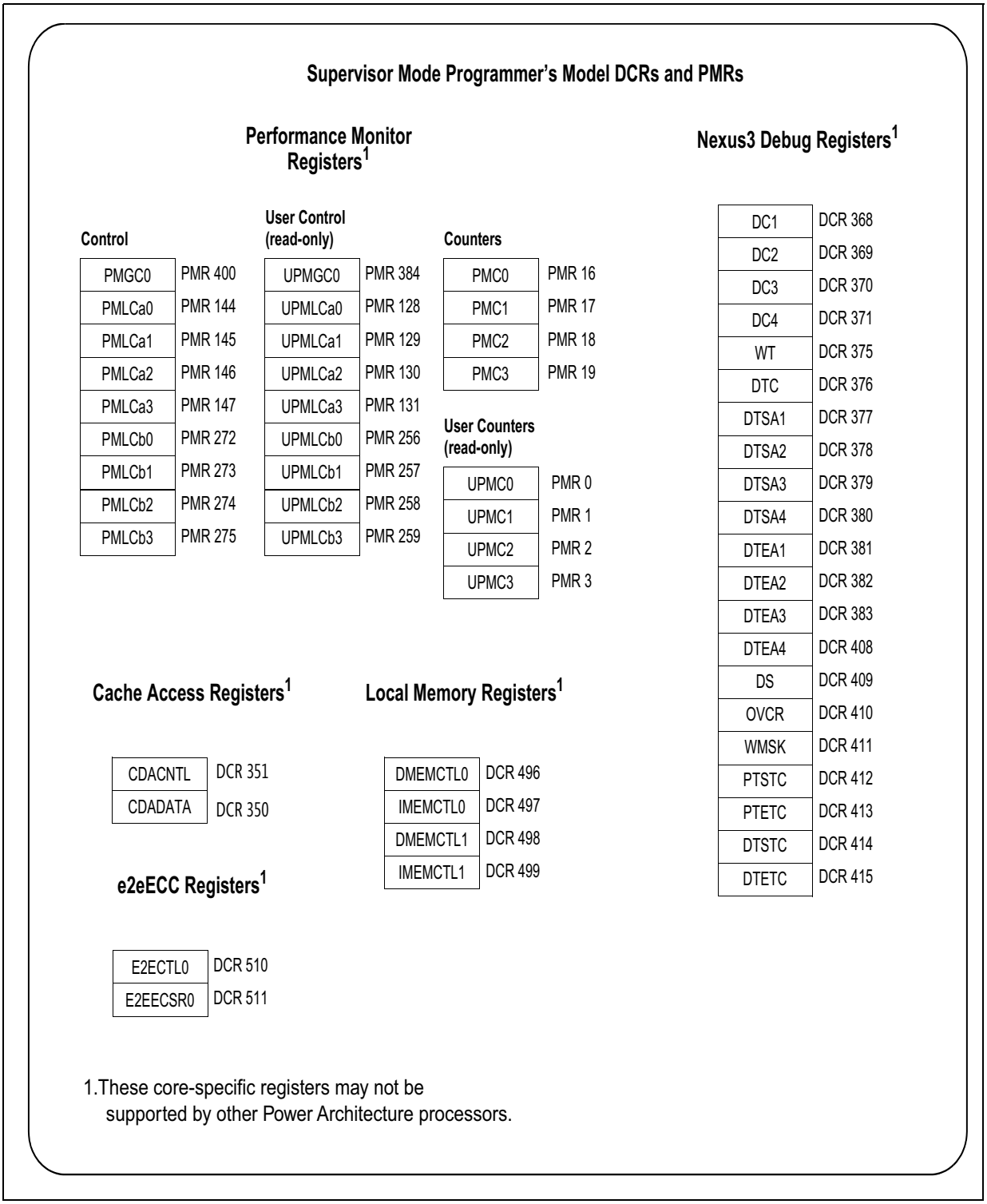


Figure 103. Core (z420n3) User Mode Programmer's Model SPRs

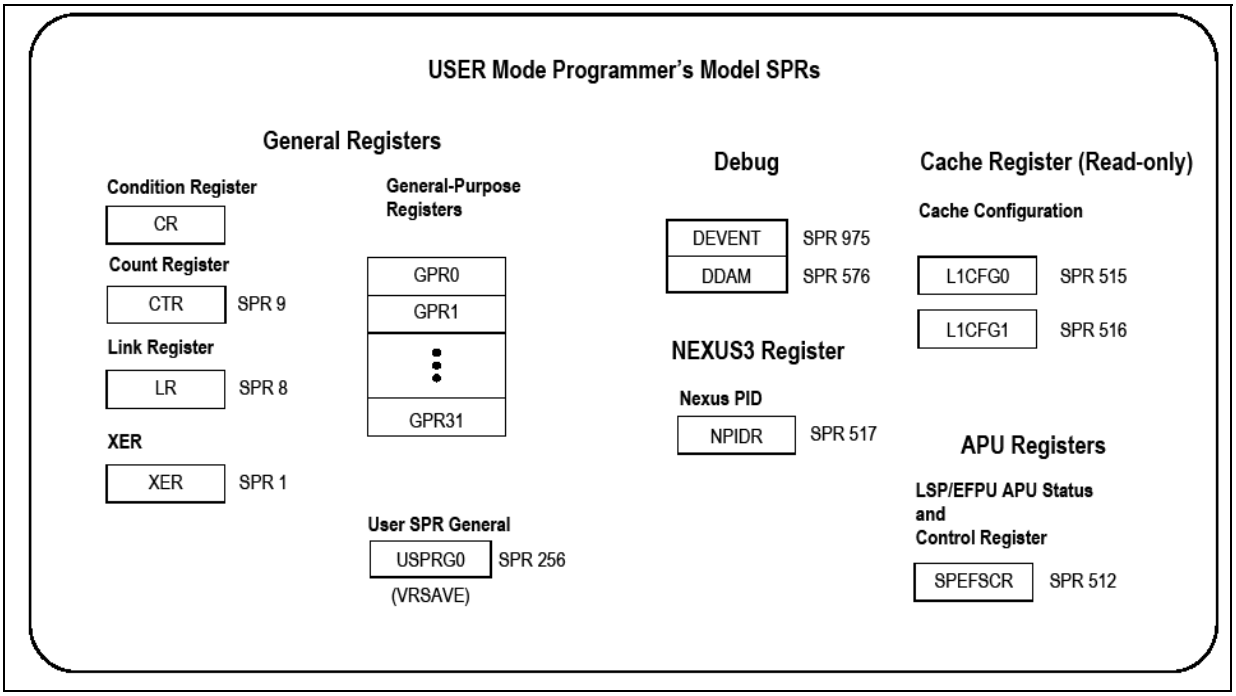
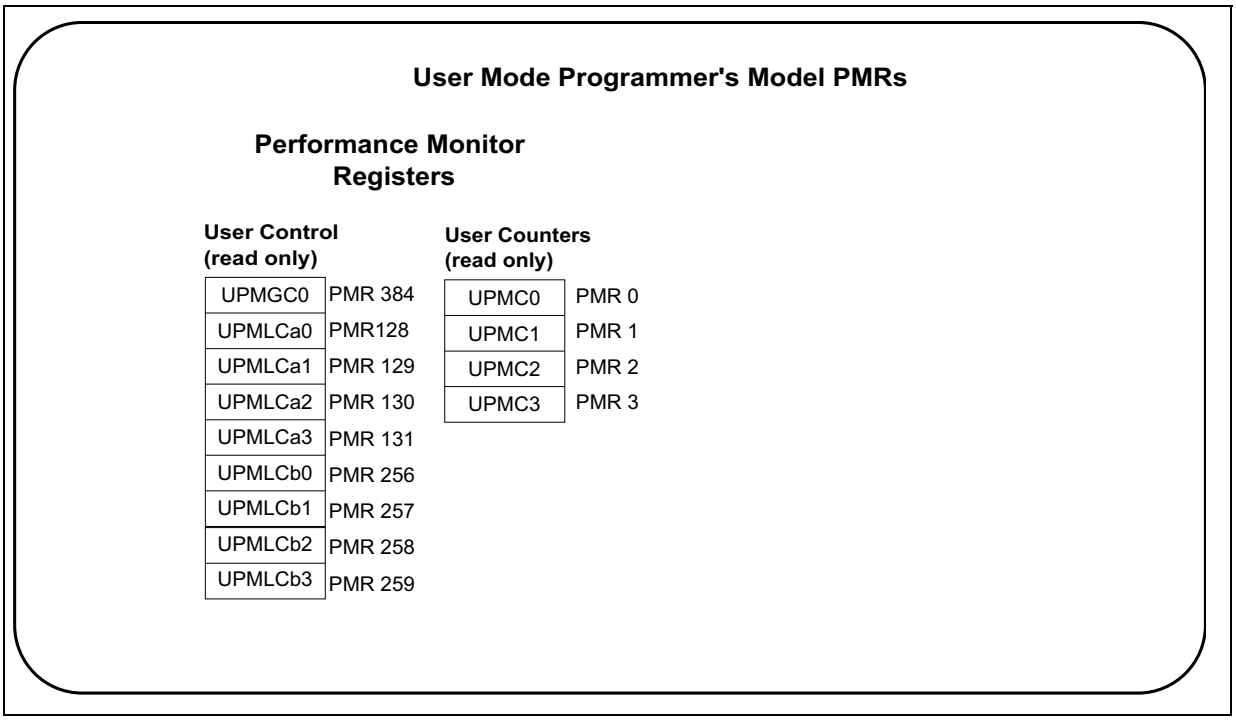


Figure 104. Core (z420n3) User Mode Programmer's Model PMRs



15.3 Dual Issue Operation

The instruction issue unit attempts to issue a pair of instructions each cycle to the execution units. Source operands for each of the instructions are provided from the GPRs or from the operand feed-forward muxes. Data or resource hazards may create stall conditions which cause instruction issue to be stalled for one or more cycles until the hazard is eliminated.

The execution units write the result of a finished instruction onto the proper result bus and into the destination registers. The writeback logic retires an instruction when the instruction has finished execution. Up to three results can be simultaneously written.

Two execution units are provided to allow dual issue of most instructions. Only a single load/store unit is provided. Only a single integer multiply and divide unit is provided, thus a pair of multiply or divide instructions cannot issue simultaneously. In addition, the divide unit is blocking.

Table 187 shows the Core (z420n3) concurrent instruction issue capabilities. Note that data dependencies between instructions will generally preclude dual-issue.

Table 187. Concurrent Instruction Issue Capabilities

Class of Instruction	branch	load/store	scalar integer	scalar float	vector integer non-MAC	vector integer MAC	special
branch	—	yes	yes	yes	yes	yes	—
load/store	yes	—	yes	yes	yes	yes	—
scalar integer	yes	yes	yes(1)	yes	yes	yes(2)	—
scalar float	yes	yes	yes	—	yes	yes	—
vector integer non-MAC	yes	yes	yes	yes	—	—	—
vector integer MAC	yes	yes	yes(2)	yes	—	—	—
special	—	—	—	—	—	—	—

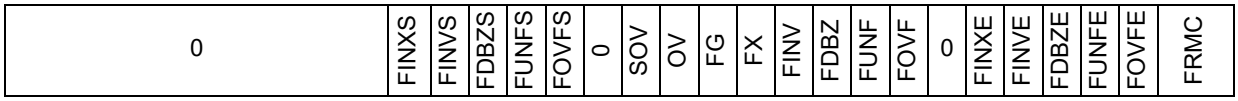
1. excludes multiply or divide class instructions occurring in both issue slots
2. excludes vector MAC/multiply class instructions occurring with scalar multiply, or divide class instructions

15.4 Reservation Instructions and Cache Interactions

The Zen core treats reservation instruction (**lbarx**, **lharx**, **lwarx**, **stbcx**., **sthcx**., and **stwcx**.) accesses as though they were cache inhibited, and forces a cache miss. A reservation access is always issued to the bus.

15.5 Signal Processing Extension / Embedded Floating-point Status and Control Register (SPEFSCR)

Status and control for embedded floating-point uses the SPEFSCR register. The SPEFSCR register is implemented as special purpose register (SPR) number 512 and is read and written by the mfspr and mtspr instructions. The SPEFSCR is shown in *Figure 105*.



SPR - 512; Read/Write; Reset - 0x0

Figure 105. DSP/EFPU Status and Control Register (SPEFSCR)

The SPEFSCR bits are defined in *Table 188*.

Table 188. EFPU Status and Control Register

Bits	Name	Description
0:9	—	Reserved
10	FINXS	Embedded Floating-point Inexact Sticky Flag The FINXS bit is set to 1 whenever the execution of a floating-point instruction delivers an inexact result and no Floating-point Data exception is taken, or if the result of a Floating-point instruction results in overflow (FOVF=1), but Floating-point Overflow exceptions are disabled (FOVFE=0), or if the result of a Floating-point instruction results in underflow (FUNF=1), but Floating-point Underflow exceptions are disabled (FUNFE=0), and no Floating-point Data exception occurs. The FINXS bit remains set until it is cleared by a **mtspr** instruction specifying the SPEFSCR register.
11	FINVS	Embedded Floating-point Invalid Operation Sticky Flag The FINVS bit is set to a 1 when a floating-point instruction sets the FINV bit to 1. The FINVS bit remains set until it is cleared by a **mtspr** instruction specifying the SPEFSCR register.
12	FDBZS	Embedded Floating-point Divide by Zero Sticky Flag The FDBZS bit is set to 1 when a floating-point divide instruction sets the FDBZ bit to 1. The FDBZS bit remains set until it is cleared by a **mtspr** instruction specifying the SPEFSCR register.
13	FUNFS	Embedded Floating-point Underflow Sticky Flag The FUNFS bit is set to 1 when a floating-point instruction sets the FUNF bit to 1. The FUNFS bit remains set until it is cleared by a **mtspr** instruction specifying the SPEFSCR register.
14	FOVFS	Embedded Floating-point Overflow Sticky Flag The FOVFS bit is set to 1 when a floating-point instruction sets the FOVF bit to 1. The FOVFS bit remains set until it is cleared by a **mtspr** instruction specifying the SPEFSCR register.
15	—	Reserved
16	SOV	Summary integer overflow Defined by LSP APU.
17	OV	Integer overflow Defined by LSP APU.
18	FG	Embedded Floating-point Guard bit FG is supplied for use by the Floating-point Round exception handler. FG is zeroed if a Floating-point Data Exception occurs.
19	FX	Embedded Floating-point Sticky bit FX is supplied for use by the Floating-point Round exception handler.FX is zeroed if a Floating-point Data Exception occurs.

Table 188. EFPU Status and Control Register (continued)

Bits	Name	Description
20	FINV	Embedded Floating-point Invalid Operation / Input error The FINV bit is set to 1 if the A or B operand of a floating-point instruction is Infinity, NaN, or Denorm, or if the operation is a divide and the dividend and divisor are both 0.
21	FDBZ	Embedded Floating-point Divide by Zero The FDBZ bit is set to 1 when a floating-point divide instruction executed with divisor of 0, and the l dividend is a finite non-zero number.
22	FUNF	Embedded Floating-point Underflow The FUNF bit is set to 1 when the execution of a floating-point instruction results in an underflow.
23	FOVF	Embedded Floating-point Overflow The FOVF bit is set to 1 when the execution of a floating-point instruction results in an overflow.
24	—	Reserved
25	FINXE	Embedded Floating-point Inexact Exception Enable 0 = Exception disabled 1 = Exception enabled If the exception is enabled, a Floating-point Round exception is taken if the result of a Floating-point instruction does not result in overflow or underflow, and the result is inexact (FG | FX = 1), or if the result of a Floating-point instruction does result in overflow (FOVF=1) but Floating-point Overflow exceptions are disabled (FOVFE=0), or if the result of a Floating-point instruction results in underflow (FUNF=1 or FUNFH=1) but Floating-point Underflow exceptions are disabled (FUNFE=0), and no Floating-point Data exception occurs.
26	FINVE	Embedded Floating-point Invalid Operation / Input Error Exception Enable 0 = Exception disabled 1 = Exception enabled If the exception is enabled, a Floating-point Data exception is taken if the FINV bit is set by a floating-point instruction.
27	FDBZE	Embedded Floating-point Divide by Zero Exception Enable 0 = Exception disabled 1 = Exception enabled If the exception is enabled, a Floating-point Data exception is taken if the FDBZ bit is set by a floating-point instruction.
28	FUNFE	Embedded Floating-point Underflow Exception Enable 0 = Exception disabled 1 = Exception enabled If the exception is enabled, a Floating-point Data exception is taken if the FUNF bit is set by a floating-point instruction.

Table 188. EFPU Status and Control Register (continued)

Bits	Name	Description
29	FOVFE	Embedded Floating-point Overflow Exception Enable 0 = Exception disabled 1 = Exception enabled If the exception is enabled, a Floating-point Data exception is taken if the FOVF bit is set by a floating-point instruction.
30:31	FRMC	Embedded Floating-point Rounding Mode Control 00 = Round to Nearest 01 = Round toward Zero 10 = Round toward +Infinity 11 = Round toward -Infinity

15.6 Cache

This section describes the cache registers, cache control instructions, and various cache operations.

15.6.1 Cache Overview

The Core (z420n3) processor supports a 8 Kbyte 2-way set-associative instruction and 4 Kbyte 2-way set associative data cache with a 32-byte line size. The caches improve system performance by providing low-latency data to the Core (z420n3) instruction and data pipelines, which decouples processor performance from system memory performance.

Instruction and data addresses from the processor are used to index the cache array. If the access address matches a valid cache tag entry, the access hits in the cache.

15.6.2 L1 Cache Control and Status Register 0 (L1CSR0)

The L1 Cache Control and Status Register 0 (L1CSR0) is a 32-bit register used for general control of the data cache as well as providing general control over disabling ways in both caches. The L1CSR0 register is accessed using a **mfspr** or **mtspr** instruction. The SPR number for L1CSR0 is 1010 in decimal. The L1CSR0 register is shown in *Figure 106*.

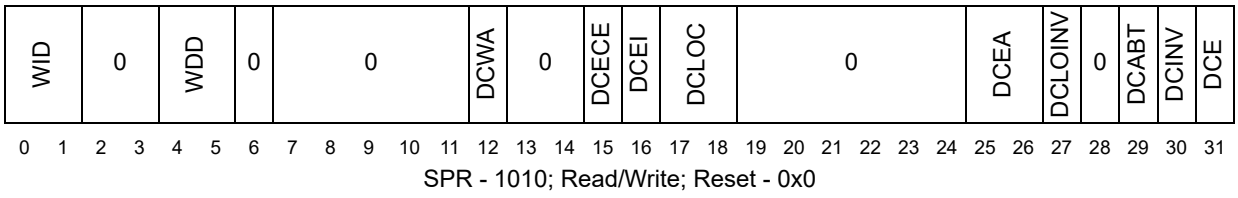


Figure 106. L1 Cache Control and Status Register 0 (L1CSR0)

The L1CSR0 bits are described in *Table 189*.

Table 189. L1CSR0 Field Descriptions

Bits	Name	Description
0:1	WID	Way Instruction Disable. 0 The corresponding way in the instruction cache is available for replacement by instruction miss line fills. 1 The corresponding way instruction cache is not available for replacement by instruction miss line fills. Bit 0 corresponds to way 0. Bit 1 corresponds to way 1. The WID bits may be used for locking ways of the instruction cache, and also affect the replacement policy of the instruction cache.
2:3	—	Reserved(1)
4:5	WDD	Way Data Disable. 0 = The corresponding way in the data cache is available for replacement by data miss line fills. 1 = The corresponding way in the data cache is not available for replacement by data miss line fills. Bit 4 corresponds to way 0. Bit 5 corresponds to way 1. The WDD bits may be used for locking ways of the data cache, and also affect the replacement policy of the data cache.
8:11	—	Reserved(1)
12	DCWA	Data Cache Write Allocation Policy 0 Cache line allocation on a cacheable write miss is disabled 1 Cache line allocation on a cacheable write miss is enabled When enabled is a write misses the cache the cache line is loaded into cache before writing data.
13:14	—	Reserved(1)
15	DCECE	Data Cache Error Checking Enable 0 Error Checking is disabled 1 Error Checking is enabled
16	DCEI	Data Cache Error Injection 0 Cache Error Injection is disabled 1 A double-bit error will be injected on each write into the cache data array by inverting the two uppermost parity check bits. This includes writes due to store hits as well as writes due to cache line refills. DCEI will cause injection of errors regardless of the setting of DCECE, although reporting of errors will be masked while DCECE=0.

Table 189. L1CSR0 Field Descriptions (continued)

Bits	Name	Description
17:18	DCLOC	Data Cache Lockout Control 00 Cache line lockout is disabled (and array LO indicators are ignored). 01 Cache line lockout is enabled. Cache lines with any tag errors or any data errors will have the corresponding lockout indicators set. A Machine check will not be generated for the error if the operation would have normally generated one unless a locked line is locked out. 10 Cache line lockout is enabled. Cache lines with any tag or data errors will have the corresponding lockout indicators set. A Machine check will still be generated for an error if the operation would have normally generated one. 11 Cache line lockout is enabled. Cache lines with uncorrectable tag errors or any data errors will have the corresponding lockout indicators set. A Machine check will not be generated for the error if the operation would have normally generated one unless a locked line is locked out. If a correctable tag error occurs, and is re-detected on recycling the access after a correction cycle for the tag is performed, the line is locked out regardless of detecting a single-bit or multi-bit error. **Note:** For the **dcbi** and **dcbf** instructions, and for specialized load/store instructions, no lockout operation is performed, regardless of error conditions.
19:24	—	Reserved(1)
25:26	DCEA	Data Cache Error Action 00 Error Detection causes Machine Check exception. 01 Error Detection causes Correction/Auto-invalidation. No machine check is generated for most cases. Correction is performed for single-bit tag errors, and lines with multi-bit tag errors are invalidated. Correction is performed for single or multi-bit data errors on cache hits by reloading of the line. 1x Reserved
27	DCLOINV	Data Cache Lockout Indicator Invalidate When DCINV = 0: Reserved, do not set to 1 When DCINV = 1: 0 No cache lockout bit invalidate 1 Cache lockout indicator invalidation operation When written to a ‘1’ in conjunction with writing DCINV to a ‘1’, a cache lockout indicator invalidation operation is initiated by hardware, and the LO bits are cleared, removing all line lockout status. Once complete, this bit is reset to ‘0’. Writing a ‘1’ while an invalidation operation is in progress will result in an undefined operation. Writing a ‘0’ to this bit while an invalidation operation is in progress will be ignored. Cache lockout bit invalidation operations require approximately 66 cycles to complete. Invalidation occurs regardless of the data cache enable (DCE) value.
28	—	Reserved(1)
29	DCABT	Data Cache Operation Aborted Indicates a Cache Invalidate operation was aborted prior to completion. This bit is set by hardware on an aborted condition, and will remain set until cleared by software writing 0 to this bit location.

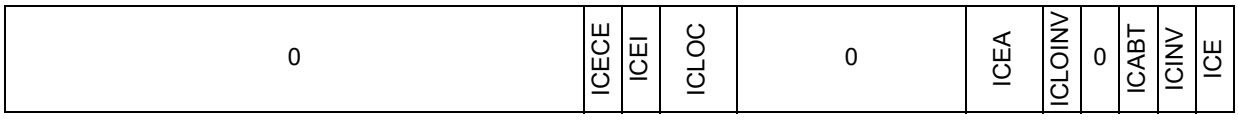
Table 189. L1CSR0 Field Descriptions (continued)

Bits	Name	Description
30	DCINV	Data Cache Invalidate 0 No cache invalidate 1 Cache invalidation operation When written to a '1', a cache invalidation operation is initiated by hardware. Once complete, this bit is reset to '0'. Writing a '1' while an invalidation operation is in progress will result in an undefined operation. Writing a '0' to this bit while an invalidation operation is in progress will be ignored. Cache invalidation operations require approximately 66 cycles to complete. Invalidation occurs regardless of the data cache enable (DCE) value.
31	DCE	Data Cache Enable 0 Cache is disabled 1 Cache is enabled When disabled, cache lookups are not performed for normal load or store accesses. Other L1CSR0 cache control operations are still available. Also, operation of the store buffer is not affected by DCE.

1. These bits are not implemented and should be written with zero for future compatibility.

15.6.3 L1 Cache Control and Status Register 1 (L1CSR1)

The L1 Cache Control and Status Register 1 (L1CSR1) is a 32-bit register used for general control of the instruction cache. The L1CSR1 register is accessed using a **mfspr** or **mtspr** instruction. The SPR number for L1CSR1 is 1011 in decimal. The L1CSR1 register is shown in *Figure 107*.



SPR - 1011; Read/Write; Reset - 0x0

Figure 107. L1 Cache Control and Status Register 1 (L1CSR1)

The L1CSR1 bits are described in *Table 190*.

Table 190. L1CSR1 Field Descriptions

Bits	Name	Description
0:14	—	Reserved(1)
15	ICECE	Instruction Cache Error Checking Enable 0 Error Checking is disabled 1 Error Checking is enabled
16	ICEI	Instruction Cache Error Injection Enable 0 Cache Error Injection is disabled 1 A double-bit error will be injected into each doubleword written into the cache by inverting the two uppermost parity check bits. ICEI will cause injection of errors regardless of the setting of ICECE, although reporting of errors will be masked when ICECE=0.

Table 190. L1CSR1 Field Descriptions (continued)

Bits	Name	Description
17:18	ICLOC	Instruction Cache Lockout Control 00 Cache line lockout is disabled (and array LO indicators are ignored). 01 Cache line lockout is enabled. Cache lines with any tag errors or any data errors will have the corresponding lockout indicators set. A Machine check will not be generated for the error if the operation would have normally generated one unless a locked line is locked out. 10 Cache line lockout is enabled. Cache lines with any tag or data errors will have the corresponding lockout indicators set. A Machine check will still be generated for an error if the operation would have normally generated one. 11 Cache line lockout is enabled. Cache lines with uncorrectable tag errors or any data errors will have the corresponding lockout indicators set. A Machine check will not be generated for the error if the operation would have normally generated one unless a locked line is locked out. If a correctable tag error occurs, and is re-detected on recycling the access after a correction cycle for the tag is performed, the line is locked out regardless of detecting a single-bit or multi-bit error. **Note:** For the **icbi** instruction, no lockout operation is performed, regardless of error conditions.
19:24	—	Reserved(1)
25:26	ICEA	Instruction Cache Error Action 00 Error Detection causes Machine Check exception. 01 Error Detection causes Correction/Auto-invalidation. No machine check is generated for most cases. Correction is performed for single-bit tag errors, and lines with multi-bit tag errors are invalidated. Correction is performed for single or multi-bit data errors on cache hits by reloading of the line. 1x Reserved
27	ICLOINV	Instruction Cache Lockout Indicator Invalidate When ICINV = 0: Reserved, do not set to 1 When ICINV = 1: 0 No cache lockout bit invalidate 1 Cache lockout indicator invalidation operation When written to a ‘1’ in conjunction with writing ICINV to a ‘1’, a cache lockout indicator invalidation operation is initiated by hardware, and the LO bits are cleared, removing all line lockout status. Once complete, this bit is reset to ‘0’. Writing a ‘1’ while an invalidation operation is in progress will result in an undefined operation. Writing a ‘0’ to this bit while an invalidation operation is in progress will be ignored. Cache lockout bit invalidation operations require approximately 66 cycles to complete. Invalidation occurs regardless of the instruction cache enable (ICE) value.
28	—	Reserved(1)
29	ICABT	Instruction Cache Operation Aborted Indicates a Cache Invalidate operation was aborted prior to completion. This bit is set by hardware on an aborted condition, and will remain set until cleared by software writing 0 to this bit location.

Table 190. L1CSR1 Field Descriptions (continued)

Bits	Name	Description
30	ICINV	Instruction Cache Invalidate 0 No cache invalidate 1 Cache invalidation operation When written to a '1', a cache invalidation operation is initiated by hardware. Once complete, this bit is reset to '0'. Writing a '1' while an invalidation operation is in progress will result in an undefined operation. Writing a '0' to this bit while an invalidation operation is in progress will be ignored. Cache invalidation operations require approximately 130 cycles to complete. Invalidation occurs regardless of the enable (ICE) value.
31	ICE	Instruction Cache Enable 0 Cache is disabled 1 Cache is enabled When disabled, cache lookups are not performed for instruction accesses. Other L1CSR1 cache control operations are still available and are not affected by ICE.

1. These bits are not implemented and should be written with zero for future compatibility.

15.6.4 L1 Cache Configuration Register 0 (L1CFG0)

The L1 Cache Configuration Register 0 (L1CFG0) is a 32-bit read-only register. L1CFG0 provides information about the configuration of the Core (z420n3) L1 data cache design. The contents of the L1CFG0 register can be read using a **mfspr** instruction. The SPR number for L1CFG0 is 515 in decimal. The L1CFG0 register is shown in *Figure 108*.

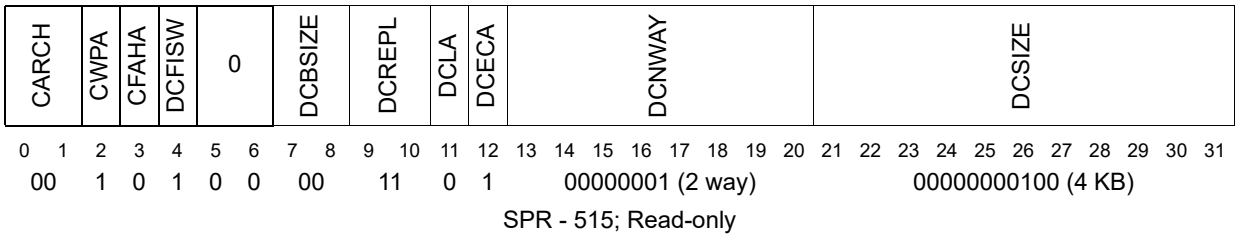


Figure 108. L1 Cache Configuration Register 0 (L1CFG0)

The L1CFG0 bits are described in *Table 191*.

Table 191. L1CFG0 Field Descriptions

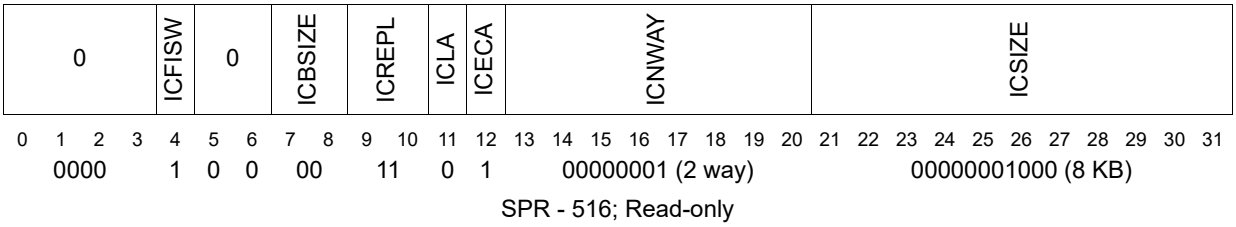
Bits	Name	Description
0:1	CARCH	Cache Architecture 00 - The cache architecture is Harvard
2	CWPA	Cache Way Partitioning Available 1 - The caches support partitioning of way availability for I/D accesses
3	DCFAHA	Data Cache Flush All by Hardware Available 0 - The data cache does not support Flush All in Hardware
4	DCFISWA	Data Cache Flush/Invalidate by Set and Way Available 1 - The data cache supports invalidation by Set and Way via L1FINV0

Table 191. L1CFG0 Field Descriptions (continued)

Bits	Name	Description
5:6	—	Reserved - read as zeros
7:8	DCBSIZE	Data Cache Block Size 00 - The data cache implements a block size of 32 bytes
9:10	DCREPL	Data Cache Replacement Policy 11- The data cache implements a FIFO replacement policy
11	DCLA	Data Cache Locking APU Available 0 - The data cache does not implement the line locking APU
12	DCECA	Data Cache Error Checking Available 1 - The data cache implements error checking
13:20	DCNWAY	Data Cache Number of Ways 0x01 - The data cache is 2-way set-associative
21:31	DCSIZE	Data Cache Size 0x004 - The size of the data cache is 4 Kbyte

15.6.5 L1 Cache Configuration Register 1 (L1CFG1)

The L1 Cache Configuration Register 1 (L1CFG1) is a 32-bit read-only register. L1CFG1 provides information about the configuration of the Core (z420n3) L1 instruction cache design. The contents of the L1CFG1 register can be read using a **mfspr** instruction. The SPR number for L1CFG1 is 516 in decimal. The L1CFG1 register is shown in *Figure 109*.



SPR - 516; Read-only

Figure 109. L1 Cache Configuration Register 1 (L1CFG1)

The L1CFG1 bits are described in *Table 192*.

Table 192. L1CFG1 Field Descriptions

Bits	Name	Description
0:3	—	Reserved - read as zeros
4	ICFISWA	Instruction Cache Flush/Invalidate by Set and Way Available 1 - The instruction cache supports invalidation by Set and Way via L1FINV1
5:6	—	Reserved - read as zeros
7:8	ICBSIZE	Instruction Cache Block Size 00 - The instruction cache implements a block size of 32 bytes
9:10	ICREPL	Instruction Cache Replacement Policy 11 - The instruction cache implements a FIFO replacement policy

Table 192. L1CFG1 Field Descriptions (continued)

Bits	Name	Description
11	ICLA	Instruction Cache Locking APU Available 0 - The instruction cache does not implement the line locking APU
12	ICECA	Instruction Cache Error Checking Available 1 - The instruction cache implements error checking
13:20	ICNWAY	Instruction Cache Number of Ways 0x01 - The instruction cache is 2-way set-associative
21:31	ICSIZE	Instruction Cache Size 0x008 - The size of the instruction cache is 8 Kbyte

15.6.6 Data Cache Software Coherency

Data cache coherency is supported through software operations to invalidate lines using either a **dcbi** or **dcbf** instruction, or by using the L1FINV0 control register.

Data cache misses will force the store buffer to empty prior to performing the access.

15.6.7 Data Cache Hardware Coherency

Data cache coherency is not supported in hardware. Software operations must generally be used to maintain coherency. Debug support is provided however for a Dcache line invalidation operation requested by an external hardware debugger to occur. When requested by an external hardware debugger tool operating through the OnCE port, an individual cache line invalidation operation will be scheduled to occur at the next available DCache access cycle.

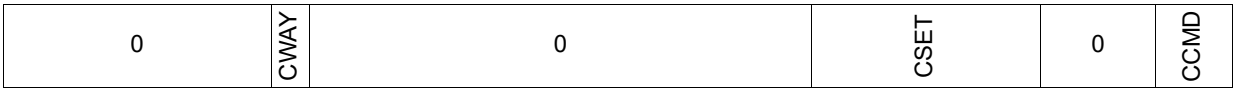
15.6.8 Cache Invalidate by Set and Way

Core (z420n3) supports cache set/way invalidation under software control. The caches may be invalidated by set and way through a **mtspr l1finv{0,1}** instruction.

The L1 Flush and Invalidate Control Registers (L1FINV{0,1}) are 32-bit SPRs used to select a cache set and way to be invalidated. This function is available even when a cache is disabled. L1FINV0 is used for data cache operations, while L1FINV1 is used for instruction cache operations.

15.6.8.1 L1FINV0

The SPR number for L1FINV0 is 1016 in decimal. The L1FINV0 register is shown in *Figure 110*.



SPR - 1016; Read/Write; Reset - 0x0

Figure 110. L1 Flush/Invalidate Register 0 (L1FINV0)

The L1FINV0 bits are described in *Table 193*.

Table 193. L1FINV0 Field Descriptions

Bits	Name	Description
0:6	—	Reserved for way extension
7	CWAY	Cache Way Specifies the data cache way to be selected
8:20	—	Reserved for set extension
21:26	CSET	Cache Set Specifies the cache set to be selected
27:29	—	Reserved for set/command extension
30:31	CCMD	Cache Command 00 = The data contained in this entry is invalidated 01 = Reserved 1x = Reserved

For invalidation operations via L1FINV0, tag ECC errors and data EDC errors are ignored, and the selected line will be invalidated regardless of any error.

15.6.8.2 L1FINV1

The SPR number for L1FINV1 is 959 in decimal. The L1FINV1 register is shown in *Figure 111*.



SPR - 959; Read/Write; Reset - 0x0

Figure 111. L1 Flush/Invalidate Register 1 (L1FINV1)

The L1FINV1 bits are described in *Table 194*.

Table 194. L1FINV1 Field Descriptions

Bits	Name	Description
0:6	—	Reserved for way extension
7	CWAY	Cache Way Specifies the instruction cache way to be selected
8:19	—	Reserved for set extension
20:26	CSET	Cache Set Specifies the instruction cache set to be selected
27:29	—	Reserved for set/command extension
30:31	CCMD	Cache Command 00 = The data contained in this entry is invalidated 01 = Reserved 1x = Reserved

For invalidation operations via L1FINV1, tag ECC errors and data EDC errors are ignored, and the selected line will be invalidated regardless of any error.

15.6.9 Cache EDC/ECC Protection

Cache protection is supported for both the tag and data arrays of each cache. Tag arrays of both instruction and data caches support error detection and correction. Each doubleword in the data arrays of the ICache and for each word in the data arrays of the DCache, have single and double bit error detection.

The error detection codes also cover the cache index address of the cache line, providing additional protection against addressing errors. If any type of error (including a single-bit error in one of the index bits) is encountered in one of the index address bits, it is treated as an uncorrectable tag ECC error, to protect against internal addressing failures.

Tag ECC and data EDC checking is controlled by the $L1CSR0_{DCECE}$ and $L1CSR1_{ICECE}$ control fields. When error checking is enabled, checking is performed on each cache access.

For cache lookups due to instruction fetching, normal loads, or normal stores, if an uncorrectable tag ECC error is detected on any portion of the accessed tags, a tag ECC error is signaled, regardless of whether a cache hit or miss occurs. Otherwise, if a cache hit for an instruction fetch or a normal load occurs and a data EDC error is detected on any portion of the accessed data, an error is also signaled.

Store hits to the Dcache will be placed into one of the RMW buffers to support EDC checkbit operation. If the store is a partial-width store, the cache data word corresponding to the address of the store is read into the next available buffer, the store data is merged, and the new EDC checkbits for the word are calculated. If the store is a full width store, no read of the cache data is performed, since the store will overwrite the full EDC data granularity, and the EDC checkbits can be computed directly. Buffers are managed on a FIFO basis. If no buffer is available to hold the store data, a stall is incurred while a buffer (or two if possible) is written back to the cache to be freed for holding the new store. Buffers are written back to the Dcache during otherwise idle cycles once two buffers are occupied, providing a simple form of store gathering.

Signaling of an ECC error or EDC error may cause a Machine Check exception to occur, and one or more syndrome bits to be set in the Machine Check Syndrome register, or may instead result in a correction/auto-invalidation operation and not result in an exception being signaled, or both may occur, depending on the error action control setting in the appropriate cache control register.

15.6.9.1 Cache Line Lockout

In addition to the ECC/EDC protection employed for the caches, each cache line in the Icache and Dcache has a lockout indicator composed of a redundant set of lockout bits which can be selectively set when certain errors occur in either the tag or data portion of the cache line. Use of the lockout function and control over error conditions which cause a lockout to occur are controlled by $L1CSR\{0,1\}_{[D,I]CLOC}$. When the lockout function is enabled, lines which encounter selected tag ECC or data EDC errors on normal cacheable instruction fetch, load, or store accesses will have their lockout bits set. When the lockout indicator is set, the line will not be replaced and will remain in an invalid state, effectively disabling it. Also, future tag ECC or data EDC errors on the line are ignored. Lockout indicators may be cleared by software performing a cache invalidation operation (setting $L1CSR\{0,1\}_{[D,I]CINV}$) in conjunction with a cache lockout indicator invalidation (setting

L1CSR{0,1}$_{[D,I]CLOINV}$). Three stored redundant lockout bits are provided to allow for single and double-bit parity errors to be detected on the lockout indicators themselves.

When operating in machine check mode, if lockout controls are enabled via L1CSR{0,1}$_{[D,I]CLOC}$, lockout bit parity errors on any line detected on a cacheable cache lookup will generate a machine check exception. If correction/auto-invalidation is instead enabled, on each cache lookup operation, if a single- or double-bit lockout bit parity error is detected in one or more ways and lockout controls are enabled, the lockout error(s) will be corrected by re-writing all lockout bits to the asserted state, and no machine check is generated.

Lockout bit parity errors are ignored for non-cacheable accesses in either error action mode. In addition, to avoid certain exception conditions and for consistency with error reporting, non-cacheable accesses and specialized load/store accesses do not set LO bits; instead, cache contents are ignored for those errors since the cache is bypassed. A future cacheable normal access will perform the lockout function if required.

15.6.10 Cache Error Injection

Cache error injection provides a way to test error recovery by intentionally injecting EDC/ECC errors into the instruction and/or data cache.

Error injection into the instruction cache operates as follows:

- If L1CSR1$_{ICEI}$ is set, any instruction cache line fill to the instruction cache data has the associated two most significant EDC/ECC check bits inverted in the instruction cache data array for each doubleword loaded.

Error injection for the data cache operates as follows:

- If L1CSR0$_{DCEI}$ is set, any cache line fill to the data cache data array has the associated two most significant EDC/ECC check bits inverted in the data array for each word loaded. Additionally, inverted bits are generated for any data stored into the data cache data array on a store hit.

Cache parity error injection is not performed for cache debug write accesses, since parity bit values written can be directly controlled.

In order to clear the EDC/ECC bit errors, a cache invalidation or an invalidation of the lines which have had an injected parity error may be performed. Line invalidation may be performed by an **icbi/dcbi** instruction, or an L1FINV[0,1] invalidation operation.

15.7 Exceptions

As specified by the PowerISA 2.06 architecture, interrupts can be either precise or imprecise, synchronous or asynchronous, and critical or non-critical. Asynchronous exceptions are caused by events external to the processor's instruction execution; synchronous exceptions are directly caused by instructions or an event somehow synchronous to the program flow, such as a context switch. A precise interrupt architecturally guarantees that no instruction beyond the instruction causing the exception has (visibly) executed. Critical interrupts are provided with a separate save/restore register pair (CSRR0/CSRR1) to allow certain critical exceptions to be handled within a non-critical interrupt handler. Machine check interrupts are also provided with a separate save/restore register pair (MCSRR0/MCSRR1) to allow machine check exceptions to be handled within a non-critical or critical interrupt handler. Debug interrupts are also provided with a separate

save/restore register pair (DSRR0/DSRR1) to allow debug exceptions to be handled within a critical, non-critical, or machine check interrupt handler.

Interrupts implemented in Zen and the exception conditions that cause them are listed in *Table 195*.

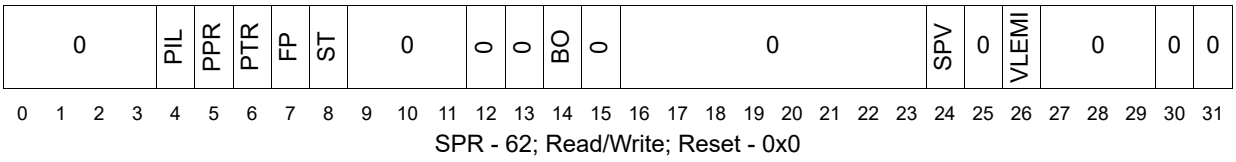
Table 195. Exceptions and Conditions

Interrupt Type	Interrupt Vector Offset Value	Causing Conditions
System reset	none, vector to [p_rstbase[0:29]] || 2'b00	1) Reset
Critical Input	0x00(1)	**p_critint_b** is asserted and MSR_{CE}=1.
Machine check	0x10	1) **p_mcp_b** transitions from negated to asserted 2) ISI or Bus Error on first instruction fetch for an exception handler 3) EDC/ECC Error signaled on cache or Local Memory access 4) External bus error 5) Stack limit check failure
Machine check (NMI)	0x10	1) Non-Maskable Interrupt.
Data Storage	0x20	1) Access control
Instruction Storage	0x30	1) Access control
External Input	0x40(2)	Interrupt Controller interrupt and MSR_{EE}=1
Alignment	0x50	1) **lmw**, **stmw** not word aligned 2) **lwarx** or **stwcx.** not word aligned, **lharx** or **sthcx.** not halfword aligned 3) dcbz
Program	0x60	Illegal, Privileged, Trap
Performance Monitor	0x70	Performance Monitor Enabled Condition or Event w/$PMGC0_{UDI}$=0
System call	0x80	Execution of the System Call (**se_sc**) instruction
Debug	0x90	Trap, Instruction Address Compare, Data Address Compare, Instruction Complete, Branch Taken, Return from Interrupt, Interrupt Taken, Debug Counter, External Debug Event, Unconditional Debug Event, Performance Monitor Enabled Condition or Event w/$PMGC0_{UDI}$=1, MPU
EFPU Data Exception	0xA0	—
EFPU Round Exception	0xB0	—
TBD	0xC0-0xF0	Reserved for future processor use

1. Autovectored Critical Input interrupts use this offset value. Vectored interrupts supply an interrupt vector offset directly.
2. Autovectored External Input interrupts use this offset value. Vectored interrupts supply an interrupt vector offset directly.

15.7.1 Exception Syndrome Register

The Exception Syndrome Register (ESR) provides a *syndrome* to differentiate between exceptions that can generate the same interrupt type.



SPR - 62; Read/Write; Reset - 0x0

Figure 112. Exception Syndrome Register

The ESR bits are defined in *Table 196*.

Table 196. ESR Bit Settings

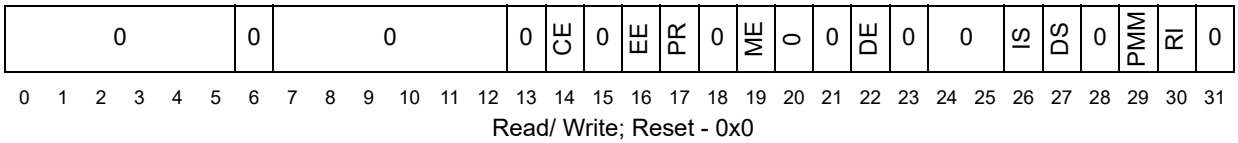
Bit(s)	Name	Description	Associated Interrupt Type
0:3	—	Reserved	—
4	PIL	Illegal Instruction exception (For Core (z420n3), PIL is used for both illegal and unimplemented instructions)	Program
5	PPR	Privileged Instruction exception	Program
6	PTR	Trap exception	Program
7	FP	Floating-point operation	Program
8	ST	Store operation	Alignment Data Storage
9:11	—	Reserved	—
12	--	Reserved	—
13	--	Reserved	—
14	BO	Byte Ordering exception Mismatched Instruction Storage exception	Instruction Storage
15	--	Reserved	—
16:23	—	Reserved	—
24	SPV	EFPU APU Operation	EFPU Floating-point Data Exception EFPU Floating-point Round Exception Alignment Data Storage
25	—	Reserved	—

Table 196. ESR Bit Settings (continued)

Bit(s)	Name	Description	Associated Interrupt Type
26	VLEMI	VLE Mode Instruction	EFPU Floating-point Data Exception EFPU Floating-point Round Exception Data Storage Instruction Storage Alignment Program System Call
27:29	—	Reserved	—
30	—	Reserved	—
31	—	Reserved	—

15.7.2 Machine State Register

The Machine State Register defines the state of the processor. The Zen MSR is shown in *Figure 113*.



Read/ Write; Reset - 0x0

Figure 113. Machine State Register (MSR)

The MSR bits are defined in *Table 197*.

Table 197. MSR Field Descriptions

Bit(s)	Name	Description
0:5	—	Reserved
6	—	Reserved
7:12	—	Reserved
13	—	Reserved
14	CE	Critical Interrupt Enable 0 - Critical Input interrupts are disabled. 1 - Critical Input interrupts are enabled.
15	—	Reserved
16	EE	External Interrupt Enable 0 - External Input interrupts are disabled. 1 - External Input interrupts are enabled.

Table 197. MSR Field Descriptions (continued)

Bit(s)	Name	Description
17	PR	Problem State 0 - The processor is in supervisor mode, can execute any instruction, and can access any resource (e.g. GPRs, SPRs, MSR, etc.). 1 - The processor is in user mode, cannot execute any privileged instruction, and cannot access any privileged resource.
18	—	Reserved
19	ME	Machine Check Enable 0 - Asynchronous Machine Check interrupts are disabled. 1 - Asynchronous Machine Check interrupts are enabled.
20	—	Reserved
21	—	Reserved
22	DE	Debug Interrupt Enable 0 - Debug interrupts are disabled. 1 - Debug interrupts are enabled.
23	—	Reserved
24	—	Reserved
25	—	Reserved
26	—	Reserved
27	—	Reserved
28	—	Reserved
29	PMM	PMM Performance monitor mark bit. System software can set PMM when a marked process is running to enable statistics to be gathered only during the execution of the marked process. MSR_{PR} and MSR_{PMM} together define a state that the processor (supervisor or user) and the process (marked or unmarked) may be in at any time. If this state matches an individual state specified in the Performance Monitor registers PMLCa n, the state for which monitoring is enabled, counting is enabled.
30	RI	Recoverable Interrupt - This bit is provided for software use to detect nested machine check exception conditions. This bit is cleared by hardware when a Machine Check interrupt is taken
31	—	Reserved

15.7.3 Machine Check Syndrome Register (MCSR)

When the processor takes a machine check interrupt, it updates the Machine Check Syndrome register (MCSR) to differentiate between machine check conditions. The MCSR is shown in *Figure 114*.



SPR - 572; Read/Clear; Reset - 0x0

Figure 114. Machine Check Syndrome Register (MCSR)

Table 198 describes MCSR fields. The MCSR indicates the source of a machine check condition.

All bits in the MCSR are implemented as “write ‘1’ to clear”. Software in the machine check handler is expected to clear the MCSR bits it has sampled prior to re-enabling MSR_{ME} to avoid a redundant machine check exception and to prepare for updated status bit information on the next machine check interrupt.

Note that any set bit in the MCSR other than status-type bits will cause a subsequent machine check interrupt once MSR_{ME}=1.

Table 198. Machine Check Syndrome Register (MCSR)

Bit	Name	Description	Exception Type(1)	Recoverable
0	MCP	Machine check input pin	Async Mchk	Maybe
1	IC_DPERR	Instruction Cache data array parity error	Async Mchk	Precise
2	—	Reserved		—
3	DC_DPERR	Data Cache data array parity error	Async Mchk	Maybe
4	EXCP_ERR	ISI or Bus Error on first instruction fetch for an exception handler	Async Mchk	Precise
5	IC_TPERR	Instruction Cache Tag parity error	Async Mchk	Precise
6	DC_TPERR	Data Cache Tag parity error	Async Mchk	Maybe
7	IC_LKERR	Instruction Cache Lock error Indicates a cache control operation or invalidation operation invalidated one or more lines in a locked way of the Icache for certain situations. May also be set on locked line refill error.	Async Mchk	—
8	DC_LKERR	Data Cache Lock error Indicates a cache control operation or invalidation operation invalidated one or more lines in a locked way of the Dcache for certain situations. May also be set on locked line refill error.	Async Mchk	—
10:11	—	Reserved		—
11	NMI	NMI input pin	NMI	—

Table 198. Machine Check Syndrome Register (MCSR) (continued)

Bit	Name	Description	Exception Type(1)	Recoverable
12	MAV	MCAR Address Valid Indicates that the address contained in the MCAR was updated by hardware to correspond to the first detected Async Mchk error condition	Status	—
13	MEA	MCAR holds Effective Address If MAV=1,MEA=1 indicates that the MCAR contains an effective address and MEA=0 indicates that the MCAR contains a physical address	Status	—
14	U	User Indicates the value captured in MCAR was generated in user mode.	Status	—
15	IF	Instruction Fetch Error Report An error occurred during the fetch of an instruction and the instruction attempted to execute. This could be due to an ECC error, or an external bus error. MCSRR0 contains the instruction address.	Error Report	Precise
16	LD	Load type instruction Error Report An error occurred during the attempt to execute the load type instruction located at the address stored in MCSRR0. This could be due to an ECC error, stack limit check error, or an external bus error.	Error Report	Precise
17	ST	Store type instruction Error Report An error occurred during the attempt to execute the store type instruction located at the address stored in MCSRR0. This could be due to an ECC error, stack limit check error, or on certain external bus errors.	Error Report	Precise
18	G	Guarded instruction Error Report An error occurred during the attempt to execute the load or store type instruction located at the address stored in MCSRR0 and the access was guarded and encountered an error on the external bus, or an uncorrectable DMEM error.	Error Report	Precise
19:20	—	Reserved		—
21	STACK_ERR	Stack Access Limit Check Error Indicates a limit check failure on a CPU data access using R1 in the <EA> calculation.	Async Mchk	Precise
22	IMEM_PERR	Instruction Mem (IMEM) Parity Error Indicates an uncorrectable error in the IMEM on a CPU port access.	Async Mchk	Precise
23	DMEM_RDPERR	Data Mem (DMEM) Parity Error Indicates an uncorrectable error in the DMEM on a CPU port read access.	Async Mchk	Precise
24	DMEM_WRPERR	Data Mem (DMEM) Write Parity Error Indicates an uncorrectable error in the DMEM on a CPU port write access.	Async Mchk	Maybe
25:26	—	Reserved		—

Table 198. Machine Check Syndrome Register (MCSR) (continued)

Bit	Name	Description	Exception Type(1)	Recoverable
27	BUS_IRERR	Read bus error on Instruction recovery linefill	Async Mchk	Precise if data used
28	BUS_DRERR	Read bus error on data load or linefill	Async Mchk	Precise if data used
29	BUS_WRERR	Write bus error on store to bus or DMEM imprecise write error	Async Mchk	Unlikely
30	BUS_WRDSI	Write bus error on buffered store to bus with DSI signaled. Set concurrently with BUS_WRERR for this case	Async Mchk	Unlikely
31	—	Reserved		—

1. The Exception Type indicates the exception type associated with a given syndrome bit

   - “Error Report” indicates that this bit is only set for error report exceptions which cause machine check interrupts. These bits are only updated when the machine check interrupt is actually taken. Error report exceptions are not gated by MSR_{ME}. These are synchronous exceptions. These bits will remain set until cleared by software writing a “1” to the bit position(s) to be cleared.

   - “Status” indicates that this bit is provides additional status information regarding the logging of a machine check exception. These bits will remain set until cleared by software writing a “1” to the bit position(s) to be cleared.

   - “NMI” indicates that this bit is only set for the non-maskable interrupt type exception which causes a machine check interrupt. This bit is only updated when the machine check interrupt is actually taken. NMI exceptions are not gated by MSR_{ME}. This is an asynchronous exception. This bit will remain set until cleared by software writing a “1” to the bit position.

   - “Async Mchk” indicates that this bit is set for an asynchronous machine check exception. These bits are set immediately upon detection of the error. Once any “Async Mchk” bit is set in the MCSR, a machine check interrupt will occur if MSR_{ME}=1. If MSR_{ME}=0, the machine check exception will remain pending. These bits will remain set until cleared by software writing a “1” to the bit position(s) to be cleared.

15.7.4 Interrupt Vector Prefix Registers (IVPR)

The Interrupt Vector Prefix Register is used during interrupt processing for determining the starting address of a software handler used to handle an interrupt. The value of the Vector Offset selected for a particular interrupt type is concatenated with the Vector Base value held in the Interrupt Vector Prefix register (IVPR) to form an instruction address from which execution is to begin. The format of IVPR is shown in *Figure 115*.

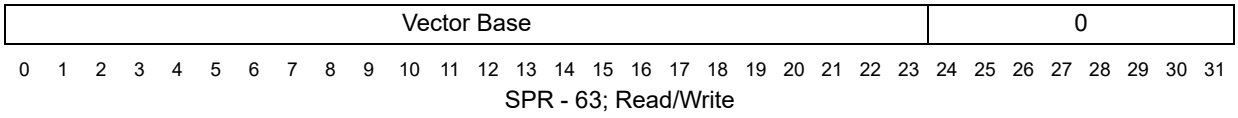


Figure 115. Interrupt Vector Prefix Registers (IVPR)

The IVPR fields are defined in *Table 199*.

Table 199. IVPR Register Fields

Bit(s)	Name	Description
0:23 (32:55)	Vector Base	Vector Base This field is used to define the base location of the vector table, aligned to a 256 byte boundary. This field provides the high-order 24 bits of the location of all interrupt handlers (unless hardware vectoring is used). The vector offset value appropriate for the type of exception being processed is concatenated with the IVPR Vector Base to form the address of the handler in memory. For hardware-vectored interrupts, the value of the supplied interrupt vector is logically OR'ed with the low order bits of the Vec Base Field to determine the interrupt handler address.
24:31 (56:63)	—	Reserved

15.7.5 Interrupt Definitions

15.7.5.1 Critical Input Interrupt (Offset 0x00)

A Critical Input exception is signalled to the processor by the assertion of the critical interrupt pin. If the exception is enabled by MSR_{CE}, the Critical Input interrupt is taken.

A Critical Input interrupt may be delayed by other higher priority exceptions or if MSR_{CE} is cleared when the exception occurs.

Table 200 lists register settings when a Critical Input interrupt is taken.

Table 200. Critical Input Interrupt—Register Settings

Register	Setting Description		
CSRR0	Set to the effective address of the instruction that the processor would have attempted to execute next if no exception conditions were present.		
CSRR1	Set to the contents of the MSR at the time of the interrupt		
MSR	SPV 0 WE 0 CE 0 EE 0 PR 0	FP 0 ME— FE0 0 DE—/0(1)	FE1 0 IS 0 DS 0 PMM 0 RI—
ESR	Unchanged		
MCSR	Unchanged		
DEAR	Unchanged		
Vector	$IVPR_{0:23}$ || 0x00 (autovectored) $IVPR_{0:15}$ || ($IVPR_{16:29}$ | **p_voffset[0:13])** || 2'b00 (non-autovectored)		

1. Clearing of DE is optionally supported by control in HID0.

The MSR_{DE} bit is not automatically cleared by a Critical Input interrupt, but can be configured to be cleared via the HID0 register ($HID0_{CICLRDE}$).

15.7.5.2 Machine Check Interrupt (Offset 0x10)

The Machine Check APU defines a separate set of save/restore registers (MCSRR0/1), a Machine Check Syndrome register (MCSR) to record the source(s) of machine checks, and a Machine Check Address register (MCAR) to hold an address associated with a machine check for certain classes of machine checks. Return from Machine Check instructions (**se_rfmci**) are also provided to support returns using MCSRR0/1.

The MSR_{DE} bit is not automatically cleared by a Machine Check exception, but can be configured to be cleared or left unchanged via the HID0 register ($HID0_{MCCLRDE}$).

When a Machine Check interrupt is taken, registers are updated as shown in *Table 201*.

Table 201. Machine Check Interrupt - Register Settings

Register	Setting Description		
MCSRR0	On a best-effort basis sets this to the address of some instruction that was executing or about to be executing when the machine check condition occurred.		
MCSRR1	Set to the contents of the MSR at the time of the interrupt		
MSR	SPV 0 WE 0 CE 0 EE 0 PR 0	FP 0 ME 0 FE0 0 DE 0/—(1)	FE1 0 IS 0 DS 0 PMM 0 RI 0
ESR	Unchanged		
MCSR	Updated to reflect the source(s) of a machine check. Hardware only sets appropriate bits, no previously set bits are cleared by hardware.		
Vector	$IVPR_{0:23}$ || 0x10		

1. Clearing of DE is optionally supported by control in HID0.

15.7.5.3 Data Storage Interrupt (Offset 0x20)

A Data Storage interrupt (DSI) may occur if no higher priority exception exists and a Read or Write Access Control exception condition exists.

Table 202 lists register settings when a DSI is taken.

Table 202. Data Storage Interrupt—Register Settings

Register	Setting Description		
SRR0	Set to the effective address of the excepting load/store instruction.		
SRR1	Set to the contents of the MSR at the time of the interrupt		
MSR	SPV 0 WE 0 CE— EE 0 PR 0	FP 0 ME— FE0 0 DE—	FE1 0 IS 0 DS 0 PMM 0 RI—
ESR	Access:	[ST], [SPV], VLEMI. All other bits cleared.	
MCSR	Unchanged		

Table 202. Data Storage Interrupt—Register Settings (continued)

Register	Setting Description
DEAR	For Access Control exceptions, set to the effective address of the access which caused the violation.
Vector	$IVPR_{0:23}$ || 0x20

15.7.5.4 Instruction Storage Interrupt (Offset 0x30)

An Instruction Storage interrupt (ISI) occurs when no higher priority exception exists and an Execute Access Control exception occurs.

Table 203 lists register settings when an ISI is taken.

Table 203. Instruction Storage Interrupt—Register Settings

Register	Setting Description		
SRR0	Set to the effective address of the excepting instruction.		
SRR1	Set to the contents of the MSR at the time of the interrupt		
MSR	SPV 0 WE 0 CE— EE 0 PR 0	FP 0 ME— FE0 0 DE—	FE1 0 IS 0 DS 0 PMM 0 RI—
ESR	VLEMI. All other bits cleared.		
MCSR	Unchanged		
DEAR	Unchanged		
Vector	$IVPR_{0:23}$ || 0x30		

15.7.5.5 External Input Interrupt (Offset 0x40)

An External Input exception is signalled to the processor by the assertion of an interrupt from the interrupt controller. The input is a level-sensitive signal expected to remain asserted until core acknowledges the external interrupt. If the input is negated early, recognition of the interrupt request is not guaranteed. When the core detects the exception, if the exception is enabled by MSR_{EE}, it takes the External Input interrupt.

An External Input interrupt may be delayed by other higher priority exceptions or if MSR_{EE} is cleared when the exception occurs.

Table 204 lists register settings when an External Input interrupt is taken.

Table 204. External Input Interrupt—Register Settings

Register	Setting Description
SRR0	Set to the effective address of the instruction that the processor would have attempted to execute next if no exception conditions were present.
SRR1	Set to the contents of the MSR at the time of the interrupt

Table 204. External Input Interrupt—Register Settings (continued)

Register	Setting Description		
MSR	SPV0 WE0 CE— EE0 PR0	FP0 ME— FE00 DE—	FE10 IS0 DS 0 PMM 0 RI—
ESR	Unchanged		
MCSR	Unchanged		
DEAR	Unchanged		
Vector	$IVPR_{0:23}$ || 0x40 (autovectored) $IVPR_{0:15}$ || ($IVPR_{16:29}$ | **p_voffset[0:13])** || 2'b00 (non-autovectored)		

15.7.5.6 Alignment Interrupt (Offset 0x50)

An Alignment exception is generated when any of the following occurs:

- The operand of **lmw** or **stmw** not word aligned.
- The operand of **lwarx** or **stwcx.** not word aligned.
- The operand of **lharx** or **sthcx.** not halfword aligned.
- Execution of a **dcbz** instruction is attempted.

Table 205 lists register settings when an alignment interrupt is taken.

Table 205. Alignment Interrupt—Register Settings

Register	Setting Description		
SRR0	Set to the effective address of the excepting load/store/dcbz instruction.		
SRR1	Set to the contents of the MSR at the time of the interrupt		
MSR	SPV 0 WE 0 CE— EE 0 PR 0	FP 0 ME— FE0 0 DE—	FE1 0 IS 0 DS 0 PMM 0 RI—
ESR	[ST], VLEMI. All other bits cleared.		
MCSR	Unchanged		
DEAR	Set to the effective address of a byte of the load or store access causing the violation.		
Vector	$IVPR_{0:23}$ || 0x50		

15.7.5.7 Program Interrupt (Offset 0x60)

A program interrupt occurs when no higher priority exception exists and one or more of the following exception conditions occur:

- Illegal Instruction exception
- Privileged Instruction exception
- Trap exception

core will invoke an Illegal Instruction program exception on attempted execution of the following instructions:

- Unimplemented instructions
- Instruction from the illegal instruction class
- **mtspr** and **mfspr** instructions with an undefined SPR specified
- **mtdcr** and **mfdcr** instructions with an undefined DCR specified

core will invoke a Privileged Instruction program exception on attempted execution of the following instructions when MSR_{PR}=1 (user mode):

- A privileged instruction
- **mtspr** and **mfspr** instructions which specify a SPRN value with $SPRN_5$=1 (even if the SPR is undefined).

core will invoke an Trap exception on execution of the **tw** instruction if the trap conditions are met and the exception is not also enabled as a Debug interrupt.

core will invoke an Illegal instruction program exception on attempted execution of the instructions **lswi**, **lswx**, **stswi**, **stswx**, **mfapidi**, **mfdcrx**, **mtdcrx**, or on any *PowerISA 2.06* floating-point instruction. All other defined or allocated instructions that are not implemented by core will cause a illegal instruction program exception.

Table 206 lists register settings when a Program interrupt is taken.

Table 206. Program Interrupt—Register Settings

Register	Setting Description		
SRR0	Set to the effective address of the excepting instruction.		
SRR1	Set to the contents of the MSR at the time of the interrupt		
MSR	SPV 0 WE 0 CE— EE 0 PR 0	FP 0 ME— FE0 0 DE—	FE1 0 IS 0 DS 0 PMM 0 RI—
ESR	Illegal, Unimplemented: Privileged: Trap:	PIL, VLEMI. All other bits cleared. PPR, VLEMI. All other bits cleared. PTR, VLEMI. All other bits cleared.	
MCSR	Unchanged		
DEAR	Unchanged		
Vector	$IVPR_{0:23}$ || 0x60		

15.7.5.8 System Call Interrupt (Offset 0x80)

A System Call interrupt occurs when a System Call (**se_sc**) instruction is executed and no higher priority exception exists.

Table 207 lists register settings when a System Call interrupt is taken.

Table 207. System Call Interrupt—Register Settings

Register	Setting Description		
SRR0	Set to the effective address of the instruction *following* the system call instruction.		
SRR1	Set to the contents of the MSR at the time of the interrupt		
MSR	SPV 0 WE 0 CE— EE 0 PR 0	FP 0 ME— FE0 0 DE—	FE1 0 IS 0 DS 0 PMM 0 RI—
ESR	VLEMI. All other bits cleared.		
MCSR	Unchanged		
DEAR	Unchanged		
Vector	$IVPR_{0:23}$ || 0x80		

15.7.5.9 Debug Interrupt (Offset 0x90)

There are multiple sources that can signal a Debug exception. A Debug interrupt occurs when no higher priority exception exists, a Debug exception exists in the Debug Status Register, and Debug interrupts are enabled (both $DBCR0_{IDM}$=1 (internal debug mode) and MSR_{DE}=1).

Table 208 lists register settings when a Debug interrupt is taken.

Table 208. Debug Interrupt—Register Settings

Register	Setting Description		
DSRR0(1)	Set to the effective address of the excepting instruction for IAC, BRT, RET, CRET, and TRAP. Set to the effective address of the next instruction to be executed *following* the excepting instruction for DAC (usually) and ICMP. For a UDE, IRPT, CIRPT, DCNT, or DEVT type exception, set to the effective address of the instruction that the processor would have attempted to execute next if no exception conditions were present.		
DSRR1	Set to the contents of the MSR at the time of the interrupt		
MSR	SPV 0 WE 0 CE—/0(2) EE—/0(2) PR0	FP 0 ME— FE0 0 DE 0	FE1 0 IS 0 DS 0 PMM 0 RI—

Table 208. Debug Interrupt—Register Settings (continued)

Register	Setting Description	
DBSR(3)	Unconditional Debug Event: Instr. Complete Debug Event: Branch Taken Debug Event: Interrupt Taken Debug Event: Critical Interrupt Taken Debug Event: Trap Instruction Debug Event: Instruction Address Compare: Data Address Compare: Debug Notify Interrupt: Return Debug Event: Critical Return Debug Event: External Debug Event: Performance Monitor Debug Event: MPU Debug Event: and optionally, an Imprecise Debug Event flag	 UDE ICMP BRT IRPT CIRPT TRAP {IAC} {DACR, DACW} DNI RET CRET {DEVT1, DEVT2} PMI MPU {IDE}
ESR	Unchanged	
MCSR	Unchanged	
DEAR	Unchanged	
Vector	$IVPR_{0:23}$ || 0x90	

1. Assumes that the Debug interrupt is precise
2. Conditional based on control bits in HID0
3. Note that multiple DBSR bits may be set

15.7.5.10 System Reset Interrupt

The System Reset exception is a non-maskable, asynchronous exception signalled to the processor through the assertion of system-defined signals.

A System reset may be initiated by either a software reset or during power-on reset.

When a reset request occurs, the processor branches to the system reset exception vector without attempting to reach a recoverable state. If reset occurs during normal operation, all operations cease and the machine state is lost.

Table 209 lists register settings when a System Reset interrupt is taken.

Table 209. System Reset Interrupt—Register Settings

Register	Setting Description
CSRR0	Undefined.
CSRR1	Undefined.

Table 209. System Reset Interrupt—Register Settings (continued)

Register	Setting Description		
MSR	SPV 0 WE 0 CE 0 EE 0 PR 0	FP 0 ME 0 FE0 0 DE 0	FE1 0 IS 0 DS 0 PMM 0 RI 0
ESR	Cleared		
DEAR	Undefined		
Vector	[**p_rstbase[0:29]**] || 2'b00		

15.7.5.11 Embedded Floating-point Data Interrupt (Offset 0xA0)

The Embedded Floating-point Data interrupt is taken if no higher priority exception exists and a EFPU Floating-point Data exception is generated. When a Floating-point Data exception occurs, the processor suppresses execution of the instruction causing the exception.

Table 210 lists register settings when a EFPU Floating-point Data interrupt is taken.

Table 210. Embedded Floating-point Data Interrupt—Register Settings

Register	Setting Description		
SRR0	Set to the effective address of the excepting EFPU instruction.		
SRR1	Set to the contents of the MSR at the time of the interrupt		
MSR	SPV 0 WE 0 CE— EE 0 PR 0	FP 0 ME— FE0 0 DE—	FE1 0 IS 0 DS 0 PMM 0 RI—
ESR	SPV, VLEMI. All other bits cleared.		
MCSR	Unchanged		
DEAR	Unchanged		
Vector	$IVPR_{0:23}$ || 0xA0		

15.7.5.12 Embedded Floating-point Round Interrupt (Offset 0xB0)

The Embedded Floating-point Round interrupt is taken when a EFPU floating-point instruction generates an inexact result and inexact exceptions are enabled.

Table 211 lists register settings when a EFPU Floating-point Round interrupt is taken.

Table 211. Embedded Floating-point Round Interrupt—Register Settings

Register	Setting Description		
SRR0	Set to the effective address of the instruction following the excepting EFPU instruction.		
SRR1	Set to the contents of the MSR at the time of the interrupt		
MSR	SPV 0 WE 0 CE— EE 0 PR 0	FP 0 ME— FE0 0 DE—	FE1 0 IS 0 DS 0 PMM 0 RI—
ESR	SPV, VLEMI. All other bits cleared.		
MCSR	Unchanged		
DEAR	Unchanged		
Vector	$IVPR_{0:23}$ || 0xB0		

15.7.5.13 Performance Monitor Interrupt (Offset 0x70)

A performance monitor interrupt that may be generated by an enabled condition or event. An enabled condition or event is as follows:

A PMC*x* register overflow condition occurs with the following settings:

- PMLCax_{CE} = 1; that is, for the given counter the overflow condition is enabled.
- PMCx_{OV} = 1; that is, the given counter indicates an overflow.

For a performance monitor interrupt to be signaled on an enabled condition or event, $PMGC0_{PMIE}$ must be set.

Although an exception condition may occur with MSR_{EE} = 0, the interrupt cannot be taken until MSR_{EE} = 1.

The priority of the performance monitor interrupt is below all other asynchronous interrupts.

Table 212 lists register settings when an performance monitor interrupt is taken.

Table 212. Performance Monitor Interrupt—Register Settings

Register	Setting Description		
SRR0/ DSRR0(1)	Set to the effective address of the next instruction to be executed.		
SRR1/ DSRR1(1)	Set to the contents of the MSR at the time of the interrupt		
MSR	SPV 0 WE 0 CE—/0(2) EE 0/—(3) PR 0	FP 0 ME— FE0 0 DE—/0(4)	FE1 0 IS 0 DS 0 PMM 0 RI—
ESR	Unchanged		
MCSR	Unchanged		

Table 212. Performance Monitor Interrupt—Register Settings (continued)

Register	Setting Description
DEAR	Unchanged
Vector	$IVPR_{0:23}$ || 0x70

1. DSRR0/1 are used if $PMGC0_{UDI}$ = 1'
2. CE is cleared if $PMGC0_{UDI}$ = 1 and $HID0_{DCLRCE}$ = 1
3. EE is not cleared if $PMGC0_{UDI}$ = 1 and $HID0_{DCLREE}$ =0
4. DE is cleared if $PMGC0_{UDI}$ = 1'

15.8 MPU

15.8.1 MPU Overview

The Core (z420n3) Memory Protection Unit (MPU) provides the capability of protecting regions of memory, with the following feature set:

- 24-entry region descriptor table with support for 6 arbitrary-sized instruction memory regions, 12 arbitrary-sized data memory regions, and 6 additional arbitrary-sized regions programmable as instruction or data memory regions
- Ability to set access permissions and memory attributes on a per-region basis
- Process ID aware, with per-bit masking of TID values
- Capability for masking upper address bits in the range comparison
- Capability of bypassing permissions checking for selected access types
- Per-entry write-once logic for entry protection
- Hardware flash invalidation support and per-entry invalidation protection controls
- Ability to optionally utilize region descriptors for generating debug events and watchpoints

Software managed by **mpure** and **mpuwe** instructions

15.8.2 Software Interface and MPU Instructions

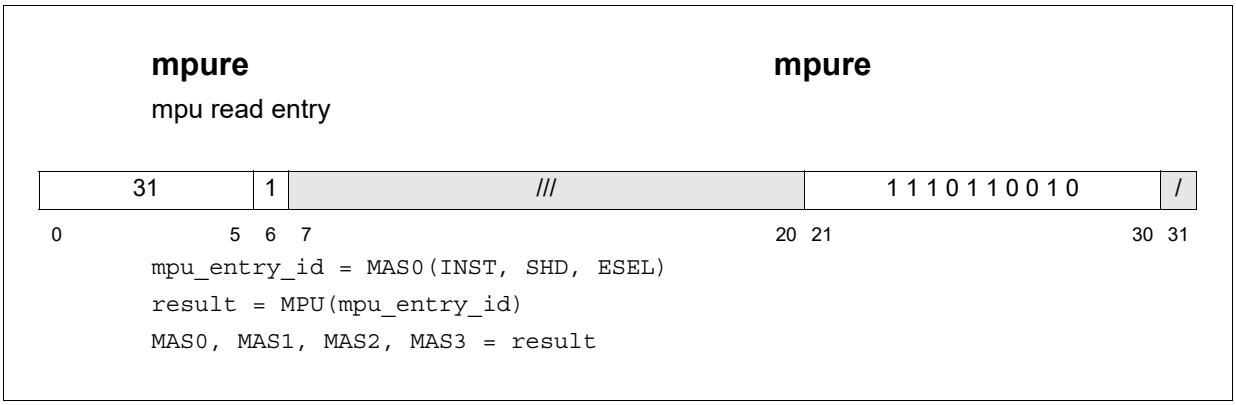
The MPU entries are accessed indirectly through several MPU Assist (MAS) registers. Software can write and read the MPU Assist registers with **mtspr** and **mfspr** instructions. These registers contain information related to reading and writing a given entry within the MPU. Data is read from the MPU into the MAS registers with a **mpure** (MPU read entry) instruction. Data is written to the MPU from the MAS registers with a **mpuwe** (MPU write entry) instruction.

The **mpure**, **mpuwe**, and **mpusync** instructions are privileged.

15.8.3 MPU Read Entry Instruction (mpure)

The MPU read entry instruction causes the content of a single MPU entry to be placed in the MPU assist registers. The entry is specified by the INST, SHD, and ESEL fields of the MAS0 register. The entry contents are placed in the MAS0, MAS1, MAS2, and MAS3 registers.

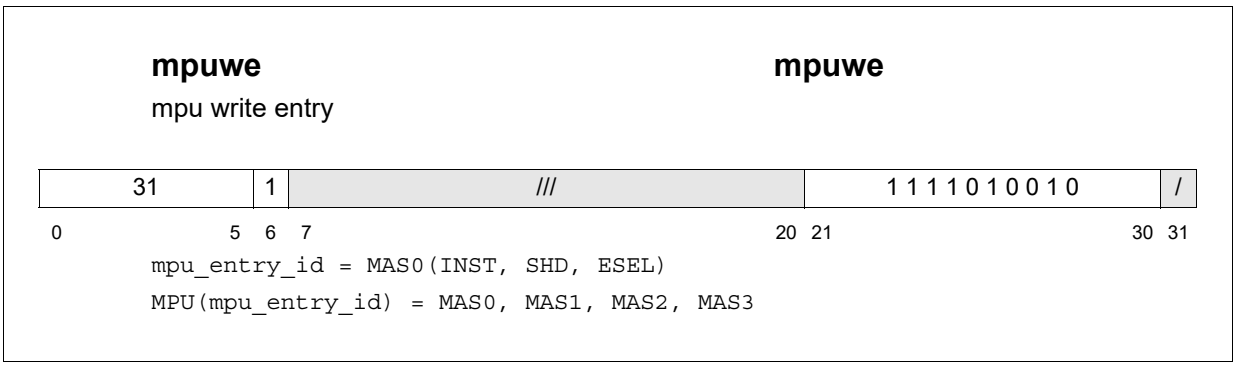
Figure 116. MPU Read Entry Instruction (mpure)



15.8.4 MPU Write Entry Instruction (mpuwe)

The MPU write entry instruction causes the contents of certain fields within the MPU assist registers MAS0, MAS1, MAS2, and MAS3 to be written into a single MPU entry in the MPU. The entry written is specified by the INST, SHD, and ESEL fields of the MAS0 register.

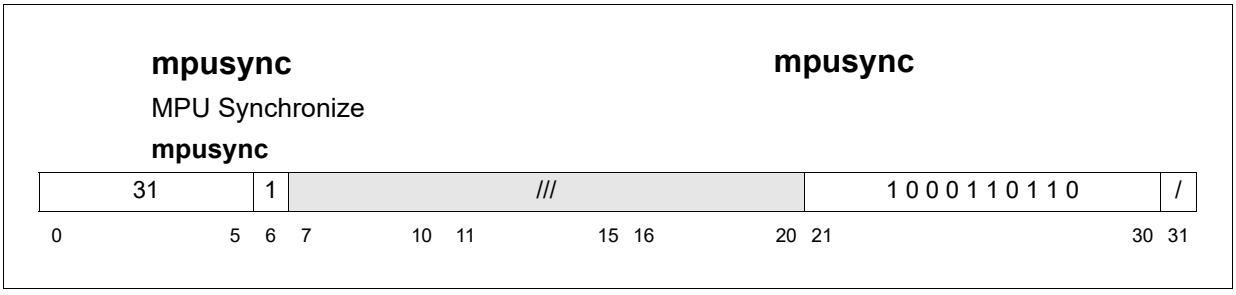
Figure 117. MPU Write Entry Instruction (mpuwe)



15.8.5 MPU Synchronize Instruction (mpusync)

The MPU Synchronize instruction is treated as a privileged no-op by the Core (z420n3).

Figure 118. MPU Synchronize Instruction (mpusync)



15.8.6 MMU/MPU Configuration Register (MMUCFG)

The MMU/MPU Configuration Register (MMUCFG) is a 32-bit read-only register. The SPR number for MMUCFG is 1015 in decimal. MMUCFG provides information about the configuration of the Core (z420n3) MPU design. The MMUCFG register is shown in *Figure 119*.

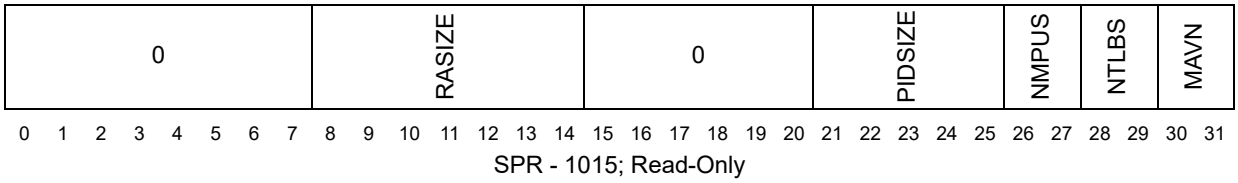


Figure 119. MMU/MPU Configuration Register (MMUCFG)

The MMUCFG bits are described in *Table 213*.

Table 213. MMUCFG Field Descriptions

Bits	Name	Function
0:7	—	Reserved
8:14	RASIZE	Number of Bits of Real Address supported 0100000- This version of the MPU implements 32 real address bits
15:16	—	Reserved
17:20	—	Reserved
21:25	PIDSIZE	PID Register Size 00111 - PID registers contain 8 bits in this version of the MPU
26:27	NMPUS	Number of MPUs 01 - This version of the MMU implements one MPU structure: a fully-associative MPU for MPU0
28:29	NTLBS	Number of TLBs 00 - This version of the MMU implements no TLB structures
30:31	MAVN	MMU Architecture Version Number 11 - This version of the MMU implements Version 3.1 of the MMU Architecture

15.8.7 MPU0 Configuration Register (MPU0CFG)

The MPU0 Configuration Register (MPU0CFG) is a 32-bit read-only register. The SPR number for MPU0CFG is 692 in decimal. MPU0CFG provides information about the configuration of MPU0 in the Core (z420n3) MPU. The MPU0CFG register is shown in *Figure 120*.

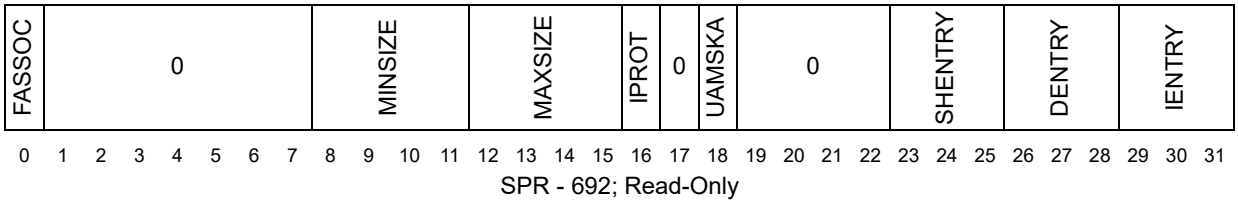


Figure 120. MPU0 Configuration Register (MPU0CFG)

The MPU0CFG bits are described in *Table 214*.

Table 214. MPU0CFG Field Descriptions

Bits	Name	Function
0	FASSOC	Fully Associative 1 - Indicates that MPU0 is fully associative
1:7	—	Reserved for non-fully associative implementation use
8:11	MINSIZE	Minimum Region Size 0x0 - Smallest region size is 1 byte Note: due to the use of access address matching, the effective smallest region size is 8 bytes
12:15	MAXSIZE	Maximum Region Size 0xB - Largest region size is 4 GB
16	IPROT	Invalidate Protect Capability 1 - Invalidate Protect Capability is supported in MPU0
17	—	Reserved
18	UAMSKA	Upper Address Masking Availability 1 - Upper Address Masking is Available
19:22	—	Reserved
23:25	SHENTRY	Number of Shared (configurable for I or D) Entries 0x2 - Indicates that MPU0 contains 6 shared entries
26:28	DENTRY	Number of Data Entries 0x4 - Indicates that MPU0 contains 12 dedicated data entries
29:31	IENTRY	Number of Instruction Entries 0x2 - Indicates that MPU0 contains 6 dedicated Instruction entries

15.8.8 MPU0 Control and Status Register 0 (MPU0CSR0)

The MPU0 Control and Status Register 0 (MPU0CSR0) is a 32-bit register. The SPR number for MPU0CSR0 is 1014 in decimal. MPU0CSR0 controls the operation of the MPU. The MPU0CSR0 register is shown in *Figure 121*.

0	BYPSR	BYPSW	BYPSX	BYPUR	BYPUW	BYPUX	0	DRDEN	DWDEN	IDEN	0	TIDCTL	0	MPUFI	MPUEN
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15	16	17	18	19	20	21	22 23	24	25	26	27	28	29	30	31

SPR - 1014; Read/ Write; Reset - 0x0

Figure 121. MPU0 Control and Status Register 0 (MPU0CSR0)

The MPU0CSR0 bits are described in *Table 215*.

Table 215. MPU0CSR0 - MPU0 Control and Status Register 0

Bits	Name	Description
0:15	—	Reserved
16	BYPSR	Bypass MPU protections for Supervisor Read accesses 0 - No Bypass of MPU protections for Supervisor Read accesses 1 - Bypass MPU protections for Supervisor Read accesses This bit controls whether supervisor-mode read accesses bypass the MPU protection mechanism. When bypass is enabled, supervisor-mode read accesses do not generate a DSI when no MPU match occurs. Supervisor-mode read accesses are still compared to MPU entries, and a hit will provide access attributes (CI, G).
17	BYPSW	Bypass MPU protections for Supervisor Write accesses 0 - No Bypass of MPU protections for Supervisor Write accesses 1 - Bypass MPU protections for Supervisor Write accesses This bit controls whether supervisor-mode write accesses bypass the MPU protection mechanism. When bypass is enabled, supervisor-mode write accesses do not generate a DSI when no MPU match occurs. Supervisor-mode write accesses are still compared to MPU entries, and a hit will provide access attributes (CI, G).
18	BYPSX	Bypass MPU protections for Supervisor Instruction accesses 0 - No Bypass of MPU protections for Supervisor Instruction accesses 1 - Bypass MPU protections for Supervisor Instruction accesses This bit controls whether supervisor-mode instruction accesses bypass the MPU protection mechanism. When bypass is enabled, supervisor-mode instruction accesses do not generate an ISI when no MPU match occurs. Supervisor-mode instruction accesses are still compared to MPU entries, and a hit will provide access attributes (CI).
19	BYPUR	Bypass MPU protections for User Read accesses 0 - No Bypass of MPU protections for User Read accesses 1 - Bypass MPU protections for User Read accesses This bit controls whether user-mode read accesses bypass the MPU protection mechanism. When bypass is enabled, user-mode read accesses do not generate a DSI when no MPU match occurs. User-mode read accesses are still compared to MPU entries, and a hit will provide access attributes (CI, G).
20	BYPUW	Bypass MPU protections for User Write accesses 0 - No Bypass of MPU protections for User Write accesses 1 - Bypass MPU protections for User Write accesses This bit controls whether user-mode write accesses bypass the MPU protection mechanism. When bypass is enabled, user-mode write accesses do not generate a DSI when no MPU match occurs. User-mode write accesses are still compared to MPU entries, and a hit will provide access attributes (CI, G).

Table 215. MPU0CSR0 - MPU0 Control and Status Register 0 (continued)

Bits	Name	Description
21	BYPUX	Bypass MPU protections for User Instruction accesses 0 - No Bypass of MPU protections for User Instruction accesses 1 - Bypass MPU protections for User Instruction accesses This bit controls whether user-mode instruction accesses bypass the MPU protection mechanism. When bypass is enabled, user-mode instruction accesses do not generate an ISI when no MPU match occurs. User-mode instruction accesses are still compared to MPU entries, and a hit will provide access attributes (CI).
22:23	—	Reserved
24	DRDEN	Data Read Debug Enable 0 - MPU debug events are disabled for data read accesses 1 - MPU debug events are enabled for data read accesses This bit controls whether data read accesses are enabled to generate MPU debug events. When disabled, no data read access will generate an MPU debug event, regardless of whether an access match occurs to a valid MPU region descriptor with DEBUG=1. If such a match occurs, no MPU debug event is generated, and no access protection is performed. When enabled, data read accesses are not blocked from generating MPU debug events.
25	DWDEN	Data Write Debug Enable 0 - MPU debug events are disabled for data write accesses 1 - MPU debug events are enabled for data write accesses This bit controls whether data write accesses are enabled to generate MPU debug events. When disabled, no data write access will generate an MPU debug event, regardless of whether an access match occurs to a valid MPU region descriptor with DEBUG=1. If such a match occurs, no MPU debug event is generated, and no access protection is performed. When enabled, data write accesses are not blocked from generating MPU debug events.
26	IDEN	Instruction Debug Enable 0 - MPU debug events are disabled for instruction accesses 1 - MPU debug events are enabled for instruction accesses This bit controls whether instruction accesses are enabled to generate MPU debug events. When disabled, no instruction access will generate an MPU debug event, regardless of whether an access match occurs to a valid MPU region descriptor with DEBUG=1. When enabled, instruction accesses are not blocked from generating MPU debug events.
27	—	Reserved
28	TIDCTL	TID usage Control 0 - TID comparisons performed normally 1 - no TID comparisons are performed for matching while in Supervisor mode When TIDCTL is set to ‘1’, the TID match is disabled (forced match) in Supervisor mode.
29	—	Reserved

Table 215. MPU0CSR0 - MPU0 Control and Status Register 0 (continued)

Bits	Name	Description
30	MPUFI	MPU flash invalidate 0 - No flash invalidate 1 - MPU1 invalidation operation When written to a '1', a MPU invalidation operation is initiated by hardware. Once complete, this bit is reset to '0'. Writing a '1' while an invalidation operation is in progress will result in an undefined operation. Writing a '0' to this bit while an invalidation operation is in progress will be ignored. MPU invalidation operations require 3 cycles to complete. Note that entries which have the IPROT bit set will not be invalidated.
31	MPUEN	MPU Enable 0 - MPU is disabled 1 - MPU is enabled This bit enables operation of the MPU. When enabled, access addresses are compared to each entry in the MPU for a match condition. If no match condition occurs, and the access type is not enabled to bypass the MPU protections, then an ISI or DSI condition is signaled for the access. Note that this bit is ignored for matching entries with DEBUG=1 if in EDM and hardware owns MPU Debug entries.

15.8.9 MPU Assist Registers (MAS)

The Core (z420n3) uses four special purpose registers (MAS0, MAS1, MAS2, and MAS3) to facilitate reading and writing MPU entries. The MAS registers can be read or written using the **mfspr** and **mtspr** instructions.

The MAS0 register is shown in *Figure 122*. Fields are defined in *Table 216*.

VALID	IPROT	SEL	0	RO	DEBUG	INST	SHD	0	ESEL	0	UAMSK	UW	SW	UX/UR	SX/SR	IOVR	GOVR (1)	1	1	I	0	G (1)	0
0	1	2 3	4	5	6	7	8	9 10 11	12 13 14 15	16	17 18 19	20	21	22	23	24	25	26	27	28	29	30	31

SPR - 624; Read/ Write; Reset - Unaffected

Figure 122. MPU Assist Registers 0 (MAS0)

1. This bit is not implemented in dedicated instruction entries, and is ignored in shared entries with INST=1

Table 216. MAS0 —MPU Read/Write and Replacement Control

Bit	Name	Comments, or Function when Set
0	VALID	MPU Entry Valid 0 - This MPU entry is invalid 1 - This MPU entry is valid
1	IPROT	Invalidation Protect 0 - Entry is not protected from invalidation 1 - Entry is protected from invalidation

Table 216. MAS0 —MPU Read/Write and Replacement Control (continued)

Bit	Name	Comments, or Function when Set
2:3	SEL	Selects MPU for access: 00 - Reserved, no access 01 - Reserved, no access 10 - Select MPU 11 - Reserved, no access
4	—	Reserved
5	RO	Read-Only 0 - This MPU entry is writable 1 - This MPU entry is Read-only When set to ‘1’, the entry is no longer writable by software. Once set, this bit will remain set until a processor reset occurs.
6	DEBUG	Debug Control for Entry 0 - This MPU entry is used for access protections and allows accesses when matched based on protection attributes 1 - This MPU entry is used for generating a debug event when a match occurs and the access protections would allow access This bit is used to assign an entry for debug event use instead of normal access protection use. When used for debug purposes, an MPU debug event is generated when a hit to the entry occurs and the access permissions allow the access.
7	INST	Instruction Entry 0 - This MPU entry is used for matching data accesses only 1 - This MPU entry is used for matching instruction accesses only When SHD=0, this bit is used to select the INST entry portion of the region descriptor table for **mpure** and **mpuwe** instruction operations. When SHD=1, this bit is used to indicate whether the entry in the Shared portion of the region descriptor table is assigned as an entry for either instruction access or data access matching. Only one of these access types is monitored for matching by a given shared region descriptor entry.
8	SHD	Shared Entry Select 0 - The shared portion of the region descriptor table is not accessed on a **mpure** or **mpuwe** operation. Either the instruction portion or the data portion is accessed based on the setting of INST. The entry within the selected portion is based on ESEL. 1 - The shared portion of the region descriptor table is accessed on a **mpure** or **mpuwe** operation. The instruction and data portions are not accessed. The INST bit is used to indicate whether the entry in the Shared portion of the region descriptor table is assigned as an entry for instruction access or data access matching. Only one of these access types is monitored for matching by a given shared region descriptor entry. ESEL defines which shared entry is selected. This bit is used to control selection of the Shared portion of the region descriptor table.
9:11	—	Reserved
12:15	ESEL	Entry select for MPU This field is used to select an entry for reading or writing in conjunction with the settings of SHD and INST. Only valid entry numbers should be used in this field, otherwise the result of an operation is boundedly undefined.
16	—	Reserved

Table 216. MAS0 —MPU Read/Write and Replacement Control (continued)

Bit	Name	Comments, or Function when Set
17:19	UAMSK	Upper Address Mask Control 000 - No upper address bits are masked, address matching uses all 32 address bits for accesses 001 - The most significant access address bit is forced to zero before matching 010 - The two most significant access address bits are forced to zero before matching 101 - The upper 5 access address bits are forced to zero before matching 11x - Reserved, do not use This field is used to support masking of upper address bits before performing range comparisons. Masked bits will be forced to 0 for the purposes of the range compare. Range upper and lower bounds should be set accordingly.
20	UW	User Mode Write Permission 0 - No User mode Write permission 1 - User mode has Write permission Determines User mode Write (W) permission when INST=0. Ignored when INST=1.
21	SW	Supervisor Mode Write / Read Permission 0 - No Supervisor mode Write permission 1 - Supervisor mode has Write permission Determines Supervisor mode Write (W) permission when INST=0. Ignored when INST=1.
22	UX/UR	User Mode Execute / Read Permission 0 - No User mode Execute/ Read permission 1 - User mode has Execute/ Read permission Determines User mode Execute (X) permission when INST=1, or User mode Read (R) permission when INST=0.
23	SX/SR	Supervisor Mode Execute / Read Permission 0 - No Supervisor mode Execute/ Read permission 1 - Supervisor mode has Execute/ Read permission Determines Supervisor mode Execute (X) permission when INST=1, or Supervisor mode Read (R) permission when INST=0.
24	IOVR	Cache-Inhibit attribute Override 0 - No override of I attribute by entry 1 - Entry I bit overrides I attribute Determines whether this matching entry overrides the I bit settings of other matching entries and the Cache-Inhibited region configuration signals
25	GOVR/—	G attribute Override 0 - No override of G attribute by entry 1 - Entry G bit overrides G attribute Determines whether this entry overrides the G bit settings of other matching entries and the Guarded region configuration signals This bit is not implemented in dedicated instruction entries, and is ignore in shared entries with INST=1.
26	—	Reserved
27	—	Reserved

Table 216. MAS0 —MPU Read/Write and Replacement Control (continued)

Bit	Name	Comments, or Function when Set
28	I	Cache Inhibited 0 - This region is considered cacheable 1 - This region is considered cache-inhibited This attribute may be overridden by the cache-inhibited region configuration input settings.
29	—	Reserved
30	G/—	Guarded 0 - Accesses to this region are not guarded, and can be performed before it is known if they are required by the sequential execution model 1 - All loads and stores to this region are performed without speculation (i.e. they are known to be required) This attribute may be overridden by the guarded region configuration input settings. This bit is not implemented in dedicated instruction entries, and is ignore in shared entries with INST=1.
31	—	Reserved

The MAS1 register is shown in *Figure 123*. Fields are defined in *Table 217*.

0	TID	0	TIDMSK

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

SPR - 625; Read/ Write; Reset - Unaffected

Figure 123. MPU Assist Register 1 (MAS1)

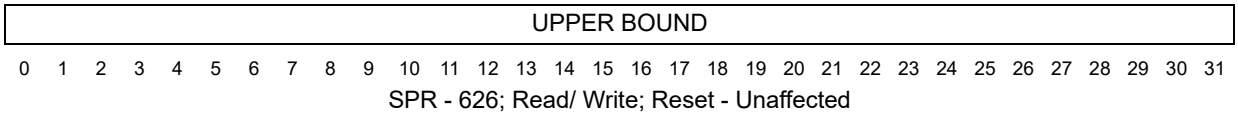
Table 217. MAS1 —Descriptor Context and Configuration Control

Bit	Name	Comments, or Function when Set
0:7	—	Reserved
8:15	TID	Region ID bits This field is compared with the current process ID of the access address. A TID value of 0 defines an entry as global and matches with all process IDs.

Table 217. MAS1 —Descriptor Context and Configuration Control

Bit	Name	Comments, or Function when Set
16:23	—	Reserved
24:31	TIDMSK	Region ID mask 0xxxxxxx - TID[0] comparison to PID0[0] not masked 1xxxxxxx - TID[0] comparison to PID0[0] is masked x0xxxxxx - TID[1] comparison to PID0[1] not masked x1xxxxxx - TID[1] comparison to PID0[1] is masked xx0xxxxx - TID[2] comparison to PID0[2] not masked xx1xxxxx - TID[2] comparison to PID0[2] is masked xxx0xxxx - TID[3] comparison to PID0[3] not masked xxx1xxxx - TID[3] comparison to PID0[3] is masked xxxx0xxx - TID[4] comparison to PID0[4] not masked xxxx1xxx - TID[4] comparison to PID0[4] is masked xxxxx0xx - TID[5] comparison to PID0[5] not masked xxxxx1xx - TID[5] comparison to PID0[5] is masked xxxxxx0x - TID[6] comparison to PID0[6] not masked xxxxxx1x - TID[6] comparison to PID0[6] is masked xxxxxxx0 - TID[7] comparison to PID0[7] not masked xxxxxxx1 - TID[7] comparison to PID0[7] is masked This field controls whether bits within the TID to PID[0] comparison are masked for determining a range match. When a bit is masked, the value of that TID and PID0 bit is ignored for matching purposes.

The MAS2 register is shown in *Figure 124*. Fields are defined in *Table 218*.



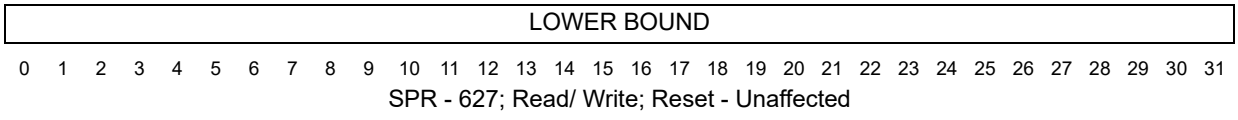
SPR - 626; Read/ Write; Reset - Unaffected

Figure 124. MPU Assist Register 2 (MAS2)

Table 218. MAS2 - Upper Bound Control

Bit	Name	Comments, or Function when Set
0:31	UPPER BOUND	Upper bound of address range covered by entry. Entry match requires LOWER_BOUND <= EA <= UPPER_BOUND

The MAS3 register is shown in *Figure 125*. Fields are defined in *Table 219*.



SPR - 627; Read/ Write; Reset - Unaffected

Figure 125. MPU Assist Register 3 (MAS3)

Table 219. MAS3 - Lower Bound Control

Bit	Name	Comments, or Function when Set
0:31	LOWER BOUND	Lower bound of address range covered by entry. Entry match requires LOWER_BOUND <= EA <= UPPER_BOUND

15.8.10 MAS Registers Summary

The MAS registers are summarized in *Figure 126*.

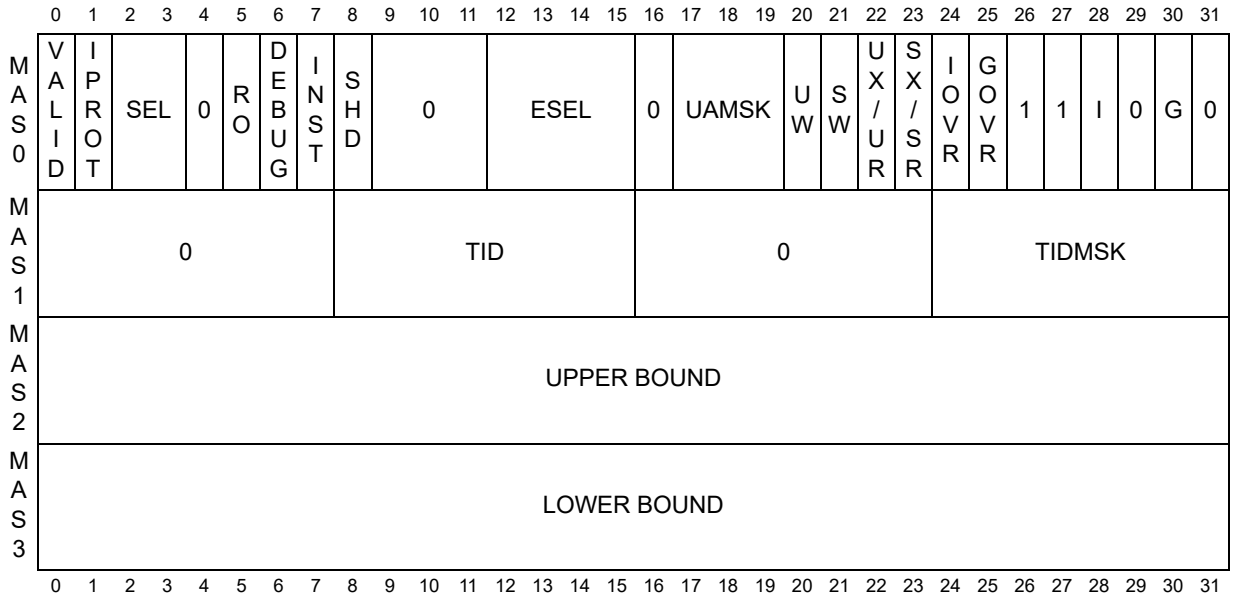


Figure 126. MPU Assist Registers Summary

15.9 Local memories

15.9.1 Local instruction and data memory overview

In order to provide for low latency memory access for critical instruction routines and data operands, local instruction (IMEM) and data (DMEM) memory capabilities have been added to the next generation processor cores. These memories provide for low latency access (zero wait-state) similar to an instruction or data cache, but do not suffer from the overhead of cache miss or cache write through operations. Instead, they provide “tightly coupled” storage.

15.9.2 Local memory control and configuration

DMEM and IMEM local memory configuration is provided by a set of privileged read-only SPRs which are accessed using **mfspr** instructions. DMEM and IMEM local memory is controlled by a set of privileged device control registers (DCRs) which are accessed using the **mfdcr** and **mtdcr** instructions. These registers and a description of the operation of various features are described in the following subsections.

15.9.2.1 DMEM Configuration Register 0 (DMEMCFG0)

The DMEM Configuration Register 0 (DMEMCFG0) is a 32-bit read-only register. DMEMCFG0 provides information about the configuration of the Core (z425n3) local data memory design. The contents of the DMEMCFG0 register can be read using a **mfspr** instruction. The SPR number for DMEMCFG0 is 694 in decimal. The DMEMCFG0 register is shown in *Figure 127*.

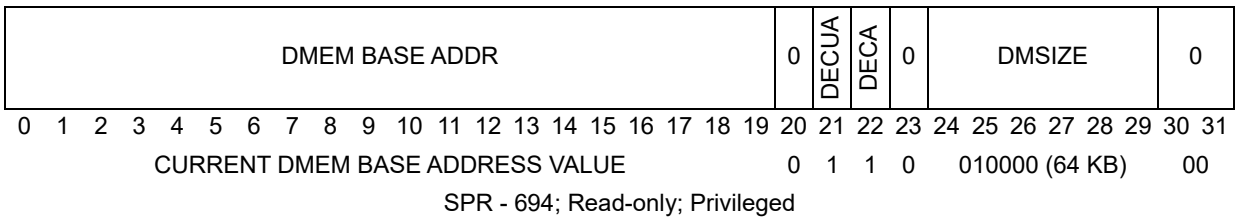


Figure 127. DMEM Configuration Register 0 (DMEMCFG0)

The DMEMCFG0 fields are described in *Table 220*.

Table 220. DMEMCFG0 Field Descriptions

Bits	Name	Description
0:19	DMEM BASE ADDR	DMEM BASE ADDRESS (CPU Port) This field defines the current Base Address value being used for the CPU port to the DMEM. This field reflects the value of the external DMEM base address port inputs when the external base address port inputs are enabled via the DBAPD control bit, or the value of the BASE ADDRESS field of DMEMCTL0 when the external base address port inputs are disabled via the DBAPD control bit. **Note:** low order bits of this field are driven to 0's when the DMEM size is > 4 Kbyte
20	-	Reserved
21	DECUA	DECUE - DMEM Error Correction Update Available 0 - Error Correction Update is not available 1 - Error Correction Update is available for correction of a correctable single-bit error detected on a read access DECUA indicates an error scrubbing function for the DMEM is available.
22	DECA	DECUE - DMEM Error Correction Available 0 - Error Correction is not available 1 - Error Correction is available DECA indicates an error correction function for the DMEM is available.
23	-	Reserved
24:29	DMSIZE	DMSIZE - DMEM Size 010000 - The size of the DMEM is 64 Kbyte.
30:31	-	Reserved

15.9.2.2 DMEM Control Register 0 (DMEMCTL0)

The DMEM Control Register 0 (DMEMCTL0) controls operation of certain functions of the DMEM logic.



DCR - 496; Read/Write; Reset(1) - 19'b0 || p_dmem_rstcfg[8,6:7,0] || 4'b0 || p_dmem_rstcfg[1:5]; Privileged

Figure 128. DMEM Control Register 0 (DMEMCTL0)

1. reset by **m_por** and **slv_reset_b**. Unaffected by **p_reset_b**.

Table 221. DMEMCTL0 Field Descriptions

Bits	Name	Description
0:16	DMEM BASE ADDR	DMEM BASE ADDRESS Field (CPU Port) This field defines the Base Address used for the CPU port to the DMEM when the external base address port inputs are disabled via the DBAPD control bit.
17:18	-	Reserved
19	DBYPCB	DBYPCB - DMEM Bypass Cache Bypass CPU accesses 0 - Non-decorated Cache Bypass loads and Store with Writethrough CPU accesses do not bypass the DMEM CPU port 1 - Non-decorated Cache Bypass loads and Store with Writethrough CPU accesses bypass the DMEM CPU port and are routed out through the BIU to the DAHB external interface. DBYPCB can be used to force Non-decorated Cache Bypass loads and Store with Writethrough (**lbcbx**, **lhcbx**, **lwcbx**, and **stwwtx**, **sthwtx**, **stbwtx**) accesses which target the DMEM address space to be presented to the external bus. Note that the protection settings in DMEMCTL1 are still applied to these accesses. **Note:** this field is updated on **m_por** and **slv_reset_b** based on the value of the **p_dmem_rstcfg[8]** input.
20	DBYPAT	DBYPAT - DMEM Bypass Atomic CPU accesses 0 - Atomic CPU accesses do not bypass the DMEM CPU port 1 - Atomic CPU accesses bypass the DMEM CPU port and are routed out through the BIU to the DAHB external interface. DBYPAT can be used to force atomic (load and reserve, store conditional) accesses to the DMEM address space to be presented to external reservation hardware in systems which support reservation or decoration functionality (or both), since no internal reservation hardware is present in the CPU for the DMEM. In general, software must not rely on normal CPU write accesses to clear a reservation, since they are not monitored by reservation hardware; instead only store conditional accesses should be used for this purpose. Note that the protection settings in DMEMCTL1 are still applied to these accesses. **Note:** this field is updated on **m_por** and **slv_reset_b** based on the value of the **p_dmem_rstcfg[6]** input.

Table 221. DMEMCTL0 Field Descriptions (continued)

Bits	Name	Description
21	DBYPDEC	DBYPDEC - DMEM Bypass Decorated CPU accesses 0 - Decorated Load and Store CPU accesses do not bypass the DMEM CPU port 1 - Decorated Load and Store CPU accesses bypass the DMEM CPU port and are routed out through the BIU to the DAHB external interface. DBYPDEC can be used to force decorated accesses (**lbdx, lhdx, lwdx, lbdcbx, lhdcbx, lwdcbx, stbdx, sthdx, stwdx, dsn, dsncb**) to the DMEM address space to be presented to external decoration hardware for systems which support this functionality, since no internal decoration hardware is present in the CPU. Note that the protection settings in DMEMCTL1 are still applied to these accesses. **Note:** this field is updated on **m_por** and **slv_reset_b** based on the value of the **p_dmem_rstcfg[7]** input.
22	DECUE	DECUE - DMEM Error Correction Update Enable 0 - Error Correction Update is disabled 1 - Error Correction Update is enabled. A correctable single-bit error detected on a read access from either the CPU or the slave port will cause the DMEM to be re-written with the corrected data. (Write accesses perform this correction automatically during partial-width writes.) DECUE can be used to provide an error scrubbing function for the DMEM. **Note:** this field is updated on **m_por** and **slv_reset_b** based on the value of the **p_dmem_rstcfg[0]** input.
23	-	Reserved
24	DBAPD	DBAPD - DMEM Base Address Port Disable 0 - The external DMEM base address port (**p_dmem_baseaddr[0:n]**) is used to define the base address of the DMEM for CPU accesses. 1 - The external DMEM base address port (**p_dmem_baseaddr[0:n]**) is disabled for use, and instead the BASE ADDR field value is used to define the base address of the DMEM for CPU accesses. This bit controls usage of the external Base Address port (**p_dmem_baseaddr[0:n]**) to define the base address of the DMEM for CPU port accesses. It has no effect on DMEM slave port accesses.
25	DPEIE	DPEIE - DMEM Processor Error Injection Enable 0 - Error Injection is disabled 1 - A double-bit error will be injected on each CPU port write into the DMEM data array by inverting the two uppermost parity check bits (p_dchk[0:1]). DPEIE will cause injection of errors regardless of the setting of DCPECE, although reporting of errors to the CPU will be masked while DCPECE=0.
26	DICWE	DICWE - DMEM Imprecise CPU Write Enable 0 - Imprecise writes by the CPU are disabled. All partial-width write ECC errors are reported precisely (error report machine check). 1 - Imprecise writes by the CPU are enabled. In certain cases, partial-width write errors may be reported imprecisely (async machine check only). DICWE may allow for increased partial-width write performance when set.

Table 221. DMEMCTL0 Field Descriptions (continued)

Bits	Name	Description
27:28	DSWCE	DSWCE - DMEM Slave port Write Check/Correct Enable 00 - Slave write data is not checked or corrected for errors. 01 - Slave write data is checked but not corrected for errors. Detected errors generate a slave port ERROR response and no Write is performed to the DMEM. 10 - Slave write data is checked and corrected for errors on all writes. Single-bit correctable errors do not automatically generate an ERROR response, but are instead corrected. 11 - Slave write data is checked and corrected for errors on partial-width (1, 2, 3, 5, 6, or 7 byte, or misaligned 4-byte writes) writes. Single-bit correctable errors on partial-width writes do not automatically generate an ERROR response, but are instead corrected. Aligned word or double word writes are not checked or corrected for errors. **Note:** this field is updated on **m_por** and **slv_reset_b** based on the values of the **p_dmem_rstcfg[1:2]** inputs.
29	DDAUEC	DDAUEC - DMEM Disable Address Use in Error Check 0 - Use of access address is enabled in checkbit and syndrome generation. 1 - Use of access address is disabled in checkbit and syndrome generation. This bit controls usage of the address portion of the ECC H matrix. When use is disabled, the address portions are not used for checkbit/syndrome generation for DMEM CPU or slave port accesses. **Note:** this field is updated on **m_por** and **slv_reset_b** based on the value of the **p_dmem_rstcfg[3]** input.
30	DCECE	DCPECE - DMEM CPU Port ECC Enable (CPU port) 0 - End-to-End ECC is disabled for the CPU interface. No checking of read data is performed by the CPU. Writes will still generate the proper check bit values to be written. 1- End-to-End ECC is enabled for performing ECC on the CPU interface. Error checking of read data is performed with single-bit error correction. Detection of uncorrectable single- or multi-bit errors on a CPU port access cause a machine check to be generated. **Note:** this field is updated on **m_por** and **slv_reset_b** based on the value of the **p_dmem_rstcfg[4]** input.
31	DSECE	DSECE - DMEM Slave port Error Checking Enable (Slave port) 0 - End-to-End ECC checking logic is disabled for the Slave interface. No checking of read data is performed during read-modify-write operations for Slave port partial-width writes. Writes will still generate the proper check bit values to be written. 1 - End-to-End ECC is enabled for performing ECC on the Slave Port interface. Error checking of read data is performed with single-bit error correction during read-modify-write operations for partial-width writes. Detection of uncorrectable single- or multi-bit errors on a Slave port partial-width write access causes a bus error ERROR response to be generated. **Note:** this field is updated on **m_por** and **slv_reset_b** based on the value of the **p_dmem_rstcfg[5]** input.

15.9.2.3 DMEM Control Register 1 (DMEMCTL1)

The DMEM Control Register 1 (DMEMCTL1) provides protection functions for the DMEM. Separate Supervisor-mode and User-mode read and write access controls allow accesses to be granted, denied, or conditionally granted independently for four equally-sized blocks of DMEM. Conditional granting of access permissions causes the MPU memory protection functions to be employed, all other settings override MPU settings. The DMEMCTL1 settings are applied to all accesses which target DMEM address range, regardless of

whether the access may bypass the DMEM based on settings in DMEMCTL0$_{DBYPCB,}$ DBYPATF, DBYPDEC·



DCR - 498; Read/Write; Reset(1) - 32‘b0; Privileged

Figure 129. DMEM Control Register 1 (DMEMCTL1)

1. reset by **m_por** and **p_reset_b**.

Table 222. DMEMCTL1 Field Descriptions

Bits	Name	Description
0:1	DSWAC3	DSWAC3 - DMEM Supervisor Write Access Control for Quadrant 3 00 - Supervisor-mode CPU write accesses are allowed to quadrant 3 of DMEM 01 - Supervisor-mode CPU write accesses are not allowed to quadrant 3 of DMEM 10 - Supervisor-mode CPU write accesses are conditionally allowed to quadrant 3 of DMEM based on memory protection logic 11 - Reserved **Note:** Individual control fields are provided for four equal sized sections of DMEM regardless of DMEM size. A single control field is used for each access based on the upper two address bits used to index into the DMEM. DSWAC3 is used when these two bits = 2‘b11
2:3	DSWAC2	DSWAC2 - DMEM Supervisor Write Access Control for Quadrant 2 00 - Supervisor-mode CPU write accesses are allowed to quadrant 2 of DMEM 01 - Supervisor-mode CPU write accesses are not allowed to quadrant 2 of DMEM 10 - Supervisor-mode CPU write accesses are conditionally allowed to quadrant 2 of DMEM based on memory protection logic 11 - Reserved **Note:** Individual control fields are provided for four equal sized sections of DMEM regardless of DMEM size. A single control field is used for each access based on the upper two address bits used to index into the DMEM. DSWAC2 is used when these two bits = 2‘b10
4:5	DSWAC1	DSWAC1 - DMEM Supervisor Write Access Control for Quadrant 1 00 - Supervisor-mode CPU write accesses are allowed to quadrant 1 of DMEM 01 - Supervisor-mode CPU write accesses are not allowed to quadrant 1 of DMEM 10 - Supervisor-mode CPU write accesses are conditionally allowed to quadrant 1 of DMEM based on memory protection logic 11 - Reserved **Note:** Individual control fields are provided for four equal sized sections of DMEM regardless of DMEM size. A single control field is used for each access based on the upper two address bits used to index into the DMEM. DSWAC1 is used when these two bits = 2‘b01

Table 222. DMEMCTL1 Field Descriptions (continued)

Bits	Name	Description
6:7	DSWAC0	DSWAC0 - DMEM Supervisor Write Access Control for Quadrant 0 00 - Supervisor-mode CPU write accesses are allowed to quadrant 0 of DMEM 01 - Supervisor-mode CPU write accesses are not allowed to quadrant 0 of DMEM 10 - Supervisor-mode CPU write accesses are conditionally allowed to quadrant 0 of DMEM based on memory protection logic 11 - Reserved **Note:** Individual control fields are provided for four equal sized sections of DMEM regardless of DMEM size. A single control field is used for each access based on the upper two address bits used to index into the DMEM. DSWAC0 is used when these two bits = 2‘b00
8:9	DSRAC3	DSRAC3 - DMEM Supervisor Read Access Control for Quadrant 3 00 - Supervisor-mode CPU read accesses are allowed to quadrant 3 of DMEM 01 - Supervisor-mode CPU read accesses are not allowed to quadrant 3 of DMEM 10 - Supervisor-mode CPU read accesses are conditionally allowed to quadrant 3 of DMEM based on memory protection logic 11 - Reserved **Note:** Individual control fields are provided for four equal sized sections of DMEM regardless of DMEM size. A single control field is used for each access based on the upper two address bits used to index into the DMEM. DSRAC3 is used when these two bits = 2‘b11
10:11	DSRAC2	DSRAC2 - DMEM Supervisor Read Access Control for Quadrant 2 00 - Supervisor-mode CPU read accesses are allowed to quadrant 2 of DMEM 01 - Supervisor-mode CPU read accesses are not allowed to quadrant 2 of DMEM 10 - Supervisor-mode CPU read accesses are conditionally allowed to quadrant 2 of DMEM based on memory protection logic 11 - Reserved **Note:** Individual control fields are provided for four equal sized sections of DMEM regardless of DMEM size. A single control field is used for each access based on the upper two address bits used to index into the DMEM. DSRAC2 is used when these two bits = 2‘b10
12:13	DSRAC1	DSRAC1 - DMEM Supervisor Read Access Control for Quadrant 1 00 - Supervisor-mode CPU read accesses are allowed to quadrant 1 of DMEM 01 - Supervisor-mode CPU read accesses are not allowed to quadrant 1 of DMEM 10 - Supervisor-mode CPU read accesses are conditionally allowed to quadrant 1 of DMEM based on memory protection logic 11 - Reserved **Note:** Individual control fields are provided for four equal sized sections of DMEM regardless of DMEM size. A single control field is used for each access based on the upper two address bits used to index into the DMEM. DSRAC1 is used when these two bits = 2‘b01
14:15	DSRAC0	DSRAC0 - DMEM Supervisor Read Access Control for Quadrant 0 00 - Supervisor-mode CPU read accesses are allowed to quadrant 0 of DMEM 01 - Supervisor-mode CPU read accesses are not allowed to quadrant 0 of DMEM 10 - Supervisor-mode CPU read accesses are conditionally allowed to quadrant 0 of DMEM based on memory protection logic 11 - Reserved **Note:** Individual control fields are provided for four equal sized sections of DMEM regardless of DMEM size. A single control field is used for each access based on the upper two address bits used to index into the DMEM. DSRAC0 is used when these two bits = 2‘b00

Table 222. DMEMCTL1 Field Descriptions (continued)

Bits	Name	Description
16:17	DUWAC3	DUWAC3 - DMEM User Write Access Control for Quadrant 3 00 - User-mode CPU write accesses are allowed to quadrant 3 of DMEM 01 - User-mode CPU write accesses are not allowed to quadrant 3 of DMEM 10 - User-mode CPU write accesses are conditionally allowed to quadrant 3 of DMEM based on memory protection logic 11 - Reserved **Note:** Individual control fields are provided for four equal sized sections of DMEM regardless of DMEM size. A single control field is used for each access based on the upper two address bits used to index into the DMEM. DUWAC3 is used when these two bits = 2‘b11
18:19	DUWAC2	DUWAC2 - DMEM User Write Access Control for Quadrant 2 00 - User-mode CPU write accesses are allowed to quadrant 2 of DMEM 01 - User-mode CPU write accesses are not allowed to quadrant 2 of DMEM 10 - User-mode CPU write accesses are conditionally allowed to quadrant 2 of DMEM based on memory protection logic 11 - Reserved **Note:** Individual control fields are provided for four equal sized sections of DMEM regardless of DMEM size. A single control field is used for each access based on the upper two address bits used to index into the DMEM. DUWAC2 is used when these two bits = 2‘b10
20:21	DUWAC1	DUWAC1 - DMEM User Write Access Control for Quadrant 1 00 - User-mode CPU write accesses are allowed to quadrant 1 of DMEM 01 - User-mode CPU write accesses are not allowed to quadrant 1 of DMEM 10 - User-mode CPU write accesses are conditionally allowed to quadrant 1 of DMEM based on memory protection logic 11 - Reserved **Note:** Individual control fields are provided for four equal sized sections of DMEM regardless of DMEM size. A single control field is used for each access based on the upper two address bits used to index into the DMEM. DUWAC1 is used when these two bits = 2‘b01
22:23	DUWAC0	DUWAC0 - DMEM User Write Access Control for Quadrant 0 00 - User-mode CPU write accesses are allowed to quadrant 0 of DMEM 01 - User-mode CPU write accesses are not allowed to quadrant 0 of DMEM 10 - User-mode CPU write accesses are conditionally allowed to quadrant 0 of DMEM based on memory protection logic 11 - Reserved **Note:** Individual control fields are provided for four equal sized sections of DMEM regardless of DMEM size. A single control field is used for each access based on the upper two address bits used to index into the DMEM. DUWAC0 is used when these two bits = 2‘b00
24:25	DURAC3	DURAC3 - DMEM User Read Access Control for Quadrant 3 00 - User-mode CPU read accesses are allowed to quadrant 3 of DMEM 01 - User-mode CPU read accesses are not allowed to quadrant 3 of DMEM 10 - User-mode CPU read accesses are conditionally allowed to quadrant 3 of DMEM based on memory protection logic 11 - Reserved **Note:** Individual control fields are provided for four equal sized sections of DMEM regardless of DMEM size. A single control field is used for each access based on the upper two address bits used to index into the DMEM. DURAC3 is used when these two bits = 2‘b11

Table 222. DMEMCTL1 Field Descriptions (continued)

Bits	Name	Description
26:27	DURAC2	DURAC2 - DMEM User Read Access Control for Quadrant 2 00 - User-mode CPU read accesses are allowed to quadrant 2 of DMEM 01 - User-mode CPU read accesses are not allowed to quadrant 2 of DMEM 10 - User-mode CPU read accesses are conditionally allowed to quadrant 2 of DMEM based on memory protection logic 11 - Reserved **Note:** Individual control fields are provided for four equal sized sections of DMEM regardless of DMEM size. A single control field is used for each access based on the upper two address bits used to index into the DMEM. DURAC2 is used when these two bits = 2'b10
28:29	DURAC1	DURAC1 - DMEM User Read Access Control for Quadrant 1 00 - User-mode CPU read accesses are allowed to quadrant 1 of DMEM 01 - User-mode CPU read accesses are not allowed to quadrant 1 of DMEM 10 - User-mode CPU read accesses are conditionally allowed to quadrant 1 of DMEM based on memory protection logic 11 - Reserved **Note:** Individual control fields are provided for four equal sized sections of DMEM regardless of DMEM size. A single control field is used for each access based on the upper two address bits used to index into the DMEM. DURAC1 is used when these two bits = 2'b01
30:31	DURAC0	DURAC0 - DMEM User Read Access Control for Quadrant 0 00 - User-mode CPU read accesses are allowed to quadrant 0 of DMEM 01 - User-mode CPU read accesses are not allowed to quadrant 0 of DMEM 10 - User-mode CPU read accesses are conditionally allowed to quadrant 0 of DMEM based on memory protection logic 11 - Reserved **Note:** Individual control fields are provided for four equal sized sections of DMEM regardless of DMEM size. A single control field is used for each access based on the upper two address bits used to index into the DMEM. DURAC0 is used when these two bits = 2'b00

15.9.2.4 IMEM Configuration Register 0 (IMEMCFG0)

The IMEM Configuration Register 0 (IMEMCFG0) is a 32-bit read-only register. IMEMCFG0 provides information about the configuration of the e200z420n3 local instruction memory design. The contents of the IMEMCFG0 register can be read using a **mfspr** instruction. The SPR number for IMEMCFG0 is 695 in decimal. The IMEMCFG0 register is shown in *Figure 130*.

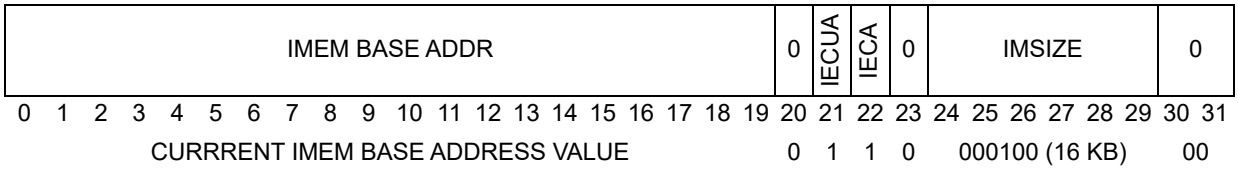


Figure 130. IMEM Configuration Register 0 (IMEMCFG0)

Table 223. IMEMCFG0 Field Descriptions

Bits	Name	Description
0:19	IMEM BASE ADDR	IMEM BASE ADDRESS (CPU Port) This field defines the current Base Address value being used for the CPU port to the IMEM. This field reflects the value of the external IMEM base address port inputs when the external base address port inputs are enabled via the DBAPD control bit, or the value of the BASE ADDRESS field of IMEMCFG0 when the external base address port inputs are disabled via the DBAPD control bit. **Note:** Low order bits of this field are driven to 0's when the IMEM size is > 4 KBs.
20	-	Reserved
21	IECUA	IECUE - IMEM Error Correction Update Available 0 - Error Correction Update is not available 1 - Error Correction Update is available for correction of a correctable single-bit error detected on a read access IECUA indicates an error scrubbing function for the IMEM is available.
22	IECA	IECUE - IMEM Error Correction Available 0 - Error Correction is not available 1 - Error Correction is available IECA indicates an error correction function for the IMEM is available.
23	-	Reserved
24:29	IMSIZE	IMSIZE - IMEM Size 000100 - The size of the IMEM is 16 KBs
30:31	-	Reserved

15.9.2.5 IMEM Control Register 0 (IMEMCTL0)

The IMEM Control Register 0 (IMEMCTL0) controls operation of certain functions of the IMEM logic.



DCR - 497; Read/Write; Reset(1) - 22'b0 ||| p_imem_rstcfg[0] || 4'b0 || p_imem_rstcfg[1:5]; Privileged

Figure 131. IMEM Control Register 0 (IMEMCTL0)

1. reset by **m_por** and **slv_reset_b**. Unaffected by **p_reset_b**.

Table 224. IMEMCTL0 Field Descriptions

Bits	Name	Description
0:17	IMEM BASE ADDR	IMEM BASE ADDRESS Field (CPU Port) This field defines the Base Address used for the CPU port to the IMEM when the external base address port inputs are disabled via the BAPD control bit. **Note:** Changes to this value and to the BAPD control bit must be performed carefully by software to ensure coherency is maintained. Specifically, software must ensure that the BTB is invalidated, and that no cached entries are present in the ICache for the memory space to be occupied by the IMEM.
18:19	-	Reserved for Base Address extension
20:21	-	Reserved
22	IECUE	IECUE - IMEM Error Correction Update Enable 0 - Error Correction Update is disabled 1 - Error Correction Update is enabled. A correctable single-bit error detected on CPU instruction fetches or slave port read or partial-width write cycles will cause the IMEM to be re-written with the corrected data. IECUE can be used to provide an error scrubbing function. **Note:** This field is updated on **m_por** and **slv_reset_b** based on the value of the **p_imem_rstcfg[0]** input.
23	-	Reserved
24	IBAPD	IBAPD - IMEM Base Address Port Disable 0 - The external IMEM base address port (**p_imem_baseaddr[0:n]**) is used to define the base address of the IMEM for CPU accesses. 1 - The external IMEM base address port (**p_imem_baseaddr[0:n]**) is disabled for use, and instead the BASE ADDR field value is used to define the base address of the IMEM for CPU accesses. This bit controls usage of the external Base Address port (**p_imem_baseaddr[0:n]**) to define the base address of the IMEM for CPU port accesses. It has no effect on IMEM slave port accesses.
25:26	-	Reserved
27:28	ISWCE	ISWCE - IMEM Slave port Write Check/Correct Enable 00 - Slave write data is not checked or corrected for errors. 01 - Slave write data is checked but not corrected for errors. Detected errors generate a slave port ERROR response and no Write is performed to the IMEM. 10 - Slave write data is checked and corrected for errors on all writes. Single-bit correctable errors do not automatically generate an ERROR response, but are instead corrected. 11 - Slave write data is checked and corrected for errors on partial-width (1, 2, 3, 4, 5, 6, or 7 byte) writes. Single-bit correctable errors on partial-width writes do not automatically generate an ERROR response, but are instead corrected. **Note:** This field is updated on **m_por** and **slv_reset_b** based on the values of the **p_imem_rstcfg[1:2]** inputs.

Table 224. IMEMCTL0 Field Descriptions (continued)

Bits	Name	Description
29	IDAUEC	IDAUEC - IMEM Disable Address Use in Error Check 0 - Use of access address is enabled in checkbit and syndrome generation. 1 - Use of access address is disabled in checkbit and syndrome generation. This bit controls usage of the address portion of the ECC H matrix. When use is disabled, the address portions are not used. for checkbit/syndrome generation for IMEM CPU or slave port accesses. **Note:** This field is updated on **m_por** and **slv_reset_b** based on the value of the **p_imem_rstcfg[3]** input.
30	ICECE	ICPECE - IMEM CPU Port ECC Enable (CPU port) 0 - End-to-End ECC is disabled for the CPU interface. No checking of read data is performed by the CPU. Writes will still generate the proper check bit values to be written. 1- End-to-End ECC is enabled for performing ECC on the CPU interface. Error checking of read data is performed with single-bit error correction. Detection of uncorrectable single- or multi-bit errors on a CPU port access cause a machine check to be generated. **Note:** This field is updated on **m_por** and **slv_reset_b** based on the value of the **p_imem_rstcfg[4]** input.
31	ISECE	ISECE - IMEM Slave port Error Checking Enable (Slave port) 0 - End-to-End ECC checking logic is disabled for the Slave interface. No checking of read data is performed during read-modify-write operations for partial-width writes. Writes will still generate the proper check bit values to be written. 1 - End-to-End ECC is enabled for performing ECC on the Slave Port interface. Error checking of read data is performed during read-modify-write operations for partial-width writes with single-bit error correction. Detection of uncorrectable single- or multi-bit errors on a Slave port access causes a bus error ERROR response to be generated. **Note:** This field is updated on **m_por** and **slv_reset_b** based on the value of the **p_imem_rstcfg[5]** input.

15.9.2.6 IMEM Control Register 1 (IMEMCTL1)

The IMEM Control Register 1 (IMEMCTL1) provides protection functions for the IMEM. Separate Supervisor-mode and User-mode instruction fetch access controls allow accesses to be granted, denied, or conditionally granted independently for four equally-sized blocks of IMEM. Conditional granting of access permissions causes the MPU memory protection functions to be employed, all other settings override MPU settings.

0	ISXAC3	ISXAC2	ISXAC1	ISXAC0	0	IUXAC3	IUXAC2	IUXAC1	IUXAC0
0 1 2 3 4 5 6 7	8 9	10 11	12 13	14 15	16 17 18 19 20 21 22 23	24 25	26 27	28 29	30 31

DCR - 499; Read/Write; Reset(1) - 32'b0; Privileged

Figure 132. IMEM Control Register 1 (IMEMCTL1)

1. reset by **m_por** and **p_reset_b**.

Table 225. IMEMCTL1 Field Descriptions

Bits	Name	Description
0:7	-	Reserved
8:9	ISXAC3	ISXAC3 - IMEM Supervisor Instruction Fetch Access Control for Quadrant 3 00 - Supervisor-mode CPU instruction fetch accesses are allowed to quadrant 3 of IMEM 01 - Supervisor-mode CPU instruction fetch accesses are not allowed to quadrant 3 of IMEM 10 - Supervisor-mode CPU instruction fetch accesses are conditionally allowed to quadrant 3 of IMEM based on memory protection logic 11 - Reserved **Note:** Individual control fields are provided for four equal sized sections of IMEM regardless of IMEM size. A single control field is used for each access based on the upper two address bits used to index into the IMEM. ISXAC3 is used when these two bits = 2‘b11
10:11	ISXAC2	ISXAC2 - IMEM Supervisor Instruction Fetch Access Control for Quadrant 2 00 - Supervisor-mode CPU instruction fetch accesses are allowed to quadrant 2 of IMEM 01 - Supervisor-mode CPU instruction fetch accesses are not allowed to quadrant 2 of IMEM 10 - Supervisor-mode CPU instruction fetch accesses are conditionally allowed to quadrant 2 of IMEM based on memory protection logic 11 - Reserved **Note:** Individual control fields are provided for four equal sized sections of IMEM regardless of IMEM size. A single control field is used for each access based on the upper two address bits used to index into the IMEM. ISXAC2 is used when these two bits = 2‘b10
12:13	ISXAC1	ISXAC1 - IMEM Supervisor Instruction Fetch Access Control for Quadrant 1 00 - Supervisor-mode CPU instruction fetch accesses are allowed to quadrant 1 of IMEM 01 - Supervisor-mode CPU instruction fetch accesses are not allowed to quadrant 1 of IMEM 10 - Supervisor-mode CPU instruction fetch accesses are conditionally allowed to quadrant 1 of IMEM based on memory protection logic 11 - Reserved **Note:** Individual control fields are provided for four equal sized sections of IMEM regardless of IMEM size. A single control field is used for each access based on the upper two address bits used to index into the IMEM. ISXAC1 is used when these two bits = 2‘b01
14:15	ISXAC0	ISXAC0 - IMEM Supervisor Instruction Fetch Access Control for Quadrant 0 00 - Supervisor-mode CPU instruction fetch accesses are allowed to quadrant 0 of IMEM 01 - Supervisor-mode CPU instruction fetch accesses are not allowed to quadrant 0 of IMEM 10 - Supervisor-mode CPU instruction fetch accesses are conditionally allowed to quadrant 0 of IMEM based on memory protection logic 11 - Reserved **Note:** Individual control fields are provided for four equal sized sections of IMEM regardless of IMEM size. A single control field is used for each access based on the upper two address bits used to index into the IMEM. ISXAC0 is used when these two bits = 2‘b00
16:23	-	Reserved

Table 225. IMEMCTL1 Field Descriptions (continued)

Bits	Name	Description
24:25	IUXAC3	IUXAC3 - IMEM User Instruction Fetch Access Control for Quadrant 3 00 - User-mode CPU instruction fetch accesses are allowed to quadrant 3 of IMEM 01 - User-mode CPU instruction fetch accesses are not allowed to quadrant 3 of IMEM 10 - User-mode CPU instruction fetch accesses are conditionally allowed to quadrant 3 of IMEM based on memory protection logic 11 - Reserved **Note:** Individual control fields are provided for four equal sized sections of IMEM regardless of IMEM size. A single control field is used for each access based on the upper two address bits used to index into the IMEM. IUXAC3 is used when these two bits = 2‘b11
26:27	IUXAC2	IUXAC2 - IMEM User Instruction Fetch Access Control for Quadrant 2 00 - User-mode CPU instruction fetch accesses are allowed to quadrant 2 of IMEM 01 - User-mode CPU instruction fetch accesses are not allowed to quadrant 2 of IMEM 10 - User-mode CPU instruction fetch accesses are conditionally allowed to quadrant 2 of IMEM based on memory protection logic 11 - Reserved **Note:** Individual control fields are provided for four equal sized sections of IMEM regardless of IMEM size. A single control field is used for each access based on the upper two address bits used to index into the IMEM. IUXAC2 is used when these two bits = 2‘b10
28:29	IUXAC1	IUXAC1 - IMEM User Instruction Fetch Access Control for Quadrant 1 00 - User-mode CPU instruction fetch accesses are allowed to quadrant 1 of IMEM 01 - User-mode CPU instruction fetch accesses are not allowed to quadrant 1 of IMEM 10 - User-mode CPU instruction fetch accesses are conditionally allowed to quadrant 1 of IMEM based on memory protection logic 11 - Reserved **Note:** Individual control fields are provided for four equal sized sections of IMEM regardless of IMEM size. A single control field is used for each access based on the upper two address bits used to index into the IMEM. IUXAC1 is used when these two bits = 2‘b01
30:31	IUXAC0	IUXAC0 - IMEM User Instruction Fetch Access Control for Quadrant 0 00 - User-mode CPU instruction fetch accesses are allowed to quadrant 0 of IMEM 01 - User-mode CPU instruction fetch accesses are not allowed to quadrant 0 of IMEM 10 - User-mode CPU instruction fetch accesses are conditionally allowed to quadrant 0 of IMEM based on memory protection logic 11 - Reserved **Note:** Individual control fields are provided for four equal sized sections of IMEM regardless of IMEM size. A single control field is used for each access based on the upper two address bits used to index into the IMEM. IUXAC0 is used when these two bits = 2‘b00

15.10 End-to-End ECC support

This section describes End-to-End Error detection and correction capability (e2eECC) enhancements the Core (z425n3) to address additional diagnostic capabilities for the next generation of embedded designs.

15.10.1 e2eECC control and configuration

e2eECC is controlled by a set of privileged device control registers (DCRs) which are accessed using the **mfdcr** and **mtdcr** instructions. These registers and a description of the operation of various features are described in the following subsections.

15.10.1.1 End-to-End ECC Control Register 0 (E2ECTL0)

The End-to-End ECC Control Register 0 (E2ECTL0) controls operation of the e2eECC logic.

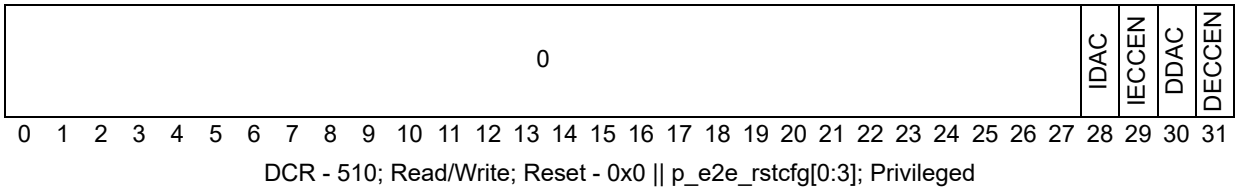


Figure 133. e2eECC Control Register 0 (E2ECTL0)

Table 226. E2ECTL0 Field Descriptions

Bits	Name	Description
0:27	-	These bits are reserved
28	IDAC	IDAC - Instruction Disable Address Checking 0 - Use of access address is enabled in checkbit and syndrome generation. 1 - Use of access address is disabled in checkbit and syndrome generation. This bit controls usage of the address portion of the ECC H matrix. When use is disabled, the address portion are not used for checkbit/syndrome generation for Instruction AHB accesses. **Note:** this field is updated on **p_reset_b** based on the value of the **p_e2e_rstcfg[0]** input.
29	IECCEN	Instruction ECC Enable Field 0 - End-to-End ECC is disabled for the Instruction interface. No checking of instruction fetch data is performed. 1 - End-to-End ECC is enabled for performing error detection and correction on the Instruction interface. Checking of instruction fetch data is performed with correction of single-bit data errors. Detection of any address errors or uncorrectable multi-bit data errors cause a machine check to be generated if the instruction fetch information is attempted to be used for instruction decoding and execution. **Note:** this field is updated on **p_reset_b** based on the value of the **p_e2e_rstcfg[1]** input.

Table 226. E2ECTL0 Field Descriptions (continued)

Bits	Name	Description
30	DDAC	DDAC - Data Disable Address Checking 0 - Use of access address is enabled in checkbit and syndrome generation. 1 - Use of access address is disabled in checkbit and syndrome generation. This bit controls usage of the address portion of the ECC H matrix. When use is disabled, the address portion are not used for checkbit/syndrome generation for Data AHB accesses. **Note:** this field is updated on **p_reset_b** based on the value of the **p_e2e_rstcfg[2]** input.
31	DECCEN	Data ECC Enable Field 0 - End-to-End ECC is disabled for the Data interface. No checking of read data is performed. Writes will still generate the proper check bit values to be supplied to external storage devices. 1 - End-to-End ECC is enabled for performing error detection and correction on the Data interface. Checking of read data is performed with correction of single-bit data errors. Detection of any address errors, or uncorrectable multi-bit data errors cause a machine check to be generated. Writes generate the proper check bit values to be supplied to external storage devices. **Note:** this field is updated on **p_reset_b** based on the value of the **p_e2e_rstcfg[3]** input.

Enabling of e2eECC operation is performed by setting the [I,D]ECCEN bit of the E2ECTL0 DCR to a one. On reset, e2eECC operation may be disabled from detecting or correcting errors based on reset configuration inputs, and no exceptions will be signaled for non-zero calculated syndrome values. This allows for the proper initialization of stored checkbits in any volatile storage by the CPU without causing error conditions due to uninitialized storage. Following the initialization, software may enable e2eECC operation by writing the appropriate values to the $E2ECTL0_{[I,D]ECCEN}$ control bits.

The $E2ECTL0_{[I,D]DAC}$ control bits can be used to remove the address checking component of the ECC logic for the Instruction AHB and Data AHB when address inclusion is not desired.

15.10.1.2 End-to-end ECC error generation by software

Error generation provides a way to test error recovery and portions of the error detection/correction logic by intentionally injecting parity errors into the driven checkbit information on the external bus during write operations. No provision is made to generate internal errors on read data due to timing issues; instead, software may intentionally generate errors in the checkbit values to emulate data, address or checkbit error on external bus write cycles, and then may read back the written data to cause an error to be detected.

The End-to-End ECC Error Control/Status Register (E2EECSR0) controls operation of the e2eECC error injection logic. It allows for generation of a single error injection operation on the next CPU data write to the external bus only, or for a series of error injection write operations to the external bus, using the WRC and INVC control fields. The WRCHKBIT/CHKINVT field controls which of the **p_hwchkbit[7:0]** outputs are affected.

Control is provided to allow for either inverting the value(s) of one or more of the checkbit outputs **p_hwchkbit[7:0]** in hardware on one or more external write accesses, or for software to directly supply checkbit values to be used for the external write access(es).

The End-to-End ECC Error Control/Status Register (E2EECSR0) also supports access to raw checkbit values via the RCHKBIT status field. This field is updated with checkbit values received on each CPU data read operation to either the DMEM or to the external bus.

Note that Nexus3 accesses do not cause error injection or capturing of read checkbits, regardless of the settings of the E2EECSR0 register.

Figure 134 shows the layout of E2EECSR0.

0	RCHKBIT	0	WRC	INVC	0	WCHKBIT/CHKINVT
0 1 2 3	4 5 6 7 8 9 10 11	12 13 14 15	16 17	18 19	20 21 22 23	24 25 26 27 28 29 30 31

DCR - 511; Read/Write; Reset - 0x0; Privileged

Figure 134. e2eECC Error Control/Status Register 0 (E2EECSR0)

Table 227. E2EECSR0 Field Descriptions

Bits	Name	Description
0:3	-	These bits are reserved
4:11	RCHKBIT	Read Checkbits This field provides the raw checkbits received on the last CPU data access to the DMEM or to external memory via the Data BIU. This field corresponds bit-wise to **p_hrchkbit[7:0]**. for external reads and to **p_dchk[0:7]** for DMEM read accesses. Software may use this information to determine the stored checkbits of external memories or the DMEM by performing CPU load operations and then accessing this field prior to the next load operation (assuming interrupts are masked).
12:15	-	These bits are reserved
16:17	WRC	Write Control 00 - No write checkbit substitution is performed by this control bit 01 - Checkbit substitution is performed for the next CPU external write access (only) by driving **p_hwchkbit[7:0]** with the values in the WCHKBIT control field. Software must clear this field before re-writing to 01 in order to generate a subsequent checkbit substitution operation. 10 - Checkbit substitution is performed for each CPU external write access while this field remains set to 10 by driving **p_hwchkbit[7:0]** with the values in the WCHKBIT control field. This field must be cleared by software to discontinue further checkbit substitution. 11- Reserved **Note:** checkbit substitution has priority over Error injection
18:19	INVC	Invert Control 00 - No error injection is performed by this control bit 01 - Error injection is performed for the next CPU external write access (only) by modifying **p_hwchkbit[7:0]** based on CHKINVT. Software must clear this field before re-writing to 01 in order to generate a subsequent error injection operation. 10 - Error injection is performed for each CPU external write access while this field remains set to 10 by modifying **p_hwchkbit[7:0]** based on CHKINVT. This field must be cleared by software to discontinue further error generation. 11 - Reserved **Note:** checkbit substitution has priority over Error injection

Table 227. E2EECSR0 Field Descriptions (continued)

Bits	Name	Description
20:23	-	These bits are reserved
24:31	WCHKBIT/ CHKINVT	Write Checkbits /Checkbit Invert Mask This field provides the checkbit substitution values when performing checkbit substitution via the WRC control field, and controls which checkbits are inverted when performing error injection via the INVC control field. This field corresponds bit-wise to **p_hwchkbit[7:0]**. For Checkbit inversion operations via error injection: 0 - Checkbit is driven normally 1 - Checkbit is inverted before being driven out on **p_hwchkbit[7:0]** when error injection occurs.

16 Core (z425n3) description

16.1 Overview of the Core (z425n3)

Core (z425n3) is a dual-issue 32-bit *PowerISA 2.06* VLE compliant design with 32-bit general purpose registers (GPRs). The Core (z425n3) core implements the VLE (variable-length encoding) ISA, providing improved code density. The VLE ISA is further documented in *PowerISA 2.06*, a separate document.

An Embedded Floating-point (EFPU2) APU is provided to support real-time single-precision floating-point embedded numerics operations using the general-purpose registers.

A Lightweight Signal Processing Extension (LSP) APU is provided to support real-time SIMD fixed point embedded numerics operations using the general-purpose registers. All arithmetic instructions that execute in the core operate on data in the general purpose registers (GPRs). The GPRs support register pairing in order to support vector instructions defined by the LSP APU. These instructions operate on a vector pair of 16-bit or 32-bit data types, and deliver vector and scalar results.

The Core (z425n3) processor integrates a pair of integer execution units, a branch control unit, instruction fetch unit, load/store unit, and a multi-ported register file capable of sustaining six read and three write operations per clock. Most integer instructions execute in a single clock cycle. Branch target prefetching is performed by the branch unit to allow single-cycle branches in many cases.

Core (z425n3) contains an 8 Kbyte Instruction Cache, as well as a Nexus Class 3+ real-time debug module with support for program and data trace features as well as extensive trace controls.

A Memory Protection Unit is also included which supports protections of various instruction and data memory areas.

16.2 Register model

This section describes the registers implemented in the Core (z425n3) core. The *PowerISA 2.06* architecture defines register-to-register operations for all computational instructions.

Core (z425n3) extends usage of the General Purpose Registers to support LSP and EFPU and LSP APU operations operating on the 32-bit GPR registers.

Figure 135 and *Figure 136* show the complete Core (z425n3) register set, all of which are accessible in supervisor mode. *Figure 137* and *Figure 138* show the subset of registers which are accessible while in user mode. The number to the right of the special-purpose registers (SPRs) is the decimal number used in the instruction syntax to access the register (for example, the integer exception register (XER) is SPR 1).

Figure 135. Core (z425n3) Supervisor Mode Programmer's Model SPRs

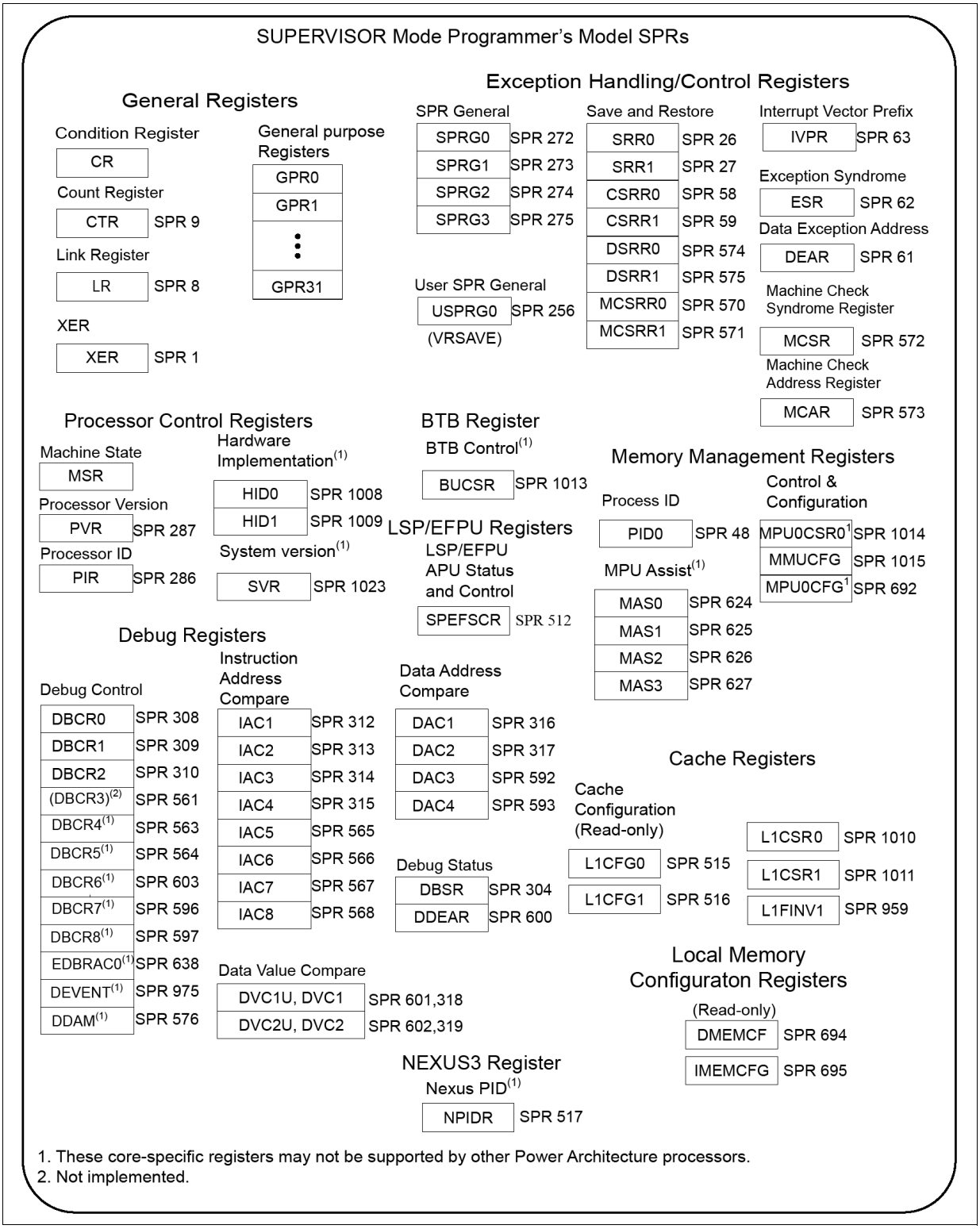
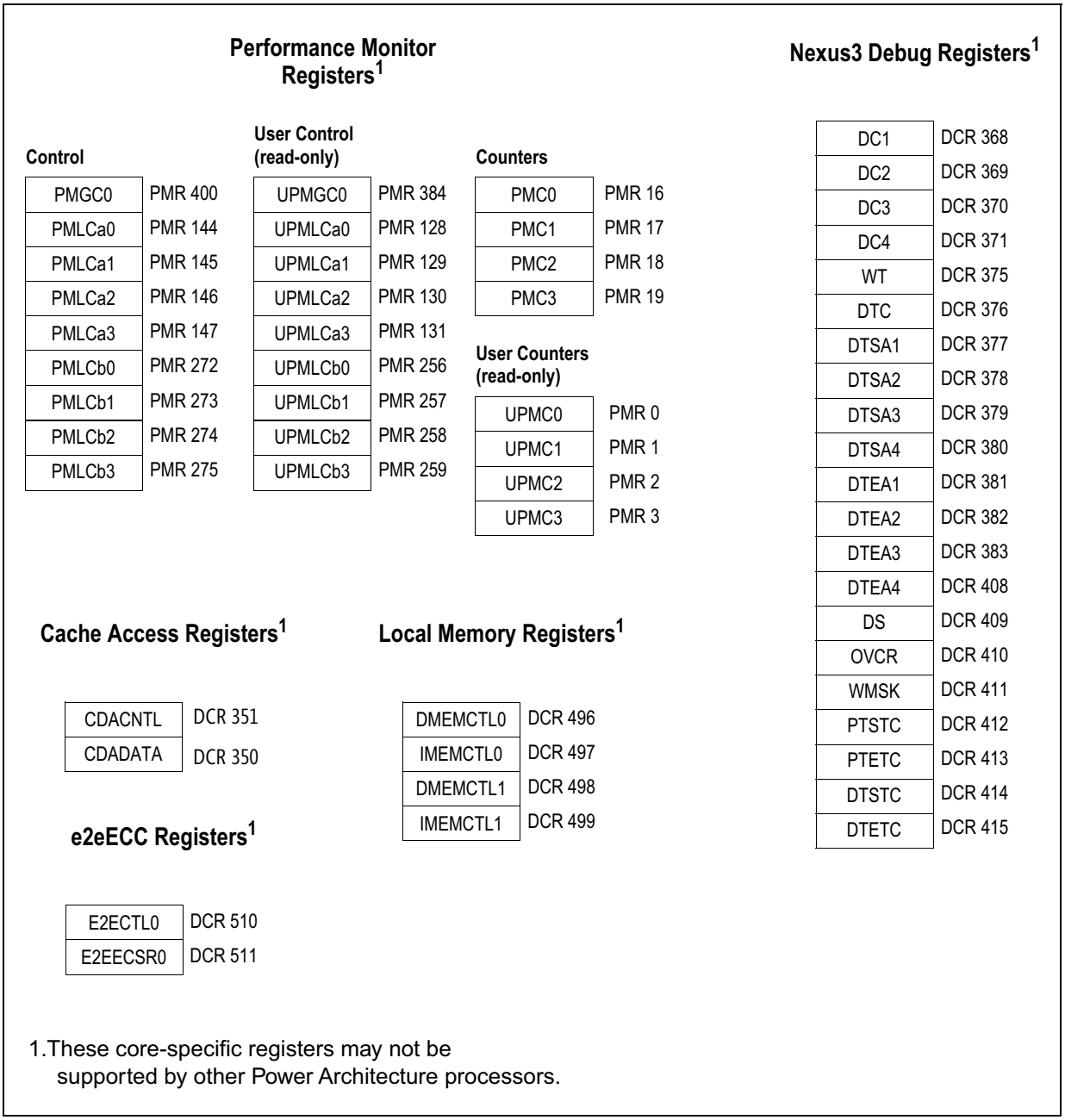


Figure 136. Core (z425n3) Supervisor Mode Programmer's Model DCRs and PMRs



1.These core-specific registers may not be supported by other Power Architecture processors.

Figure 137. Core (z425n3) User Mode Programmer's Model SPRs

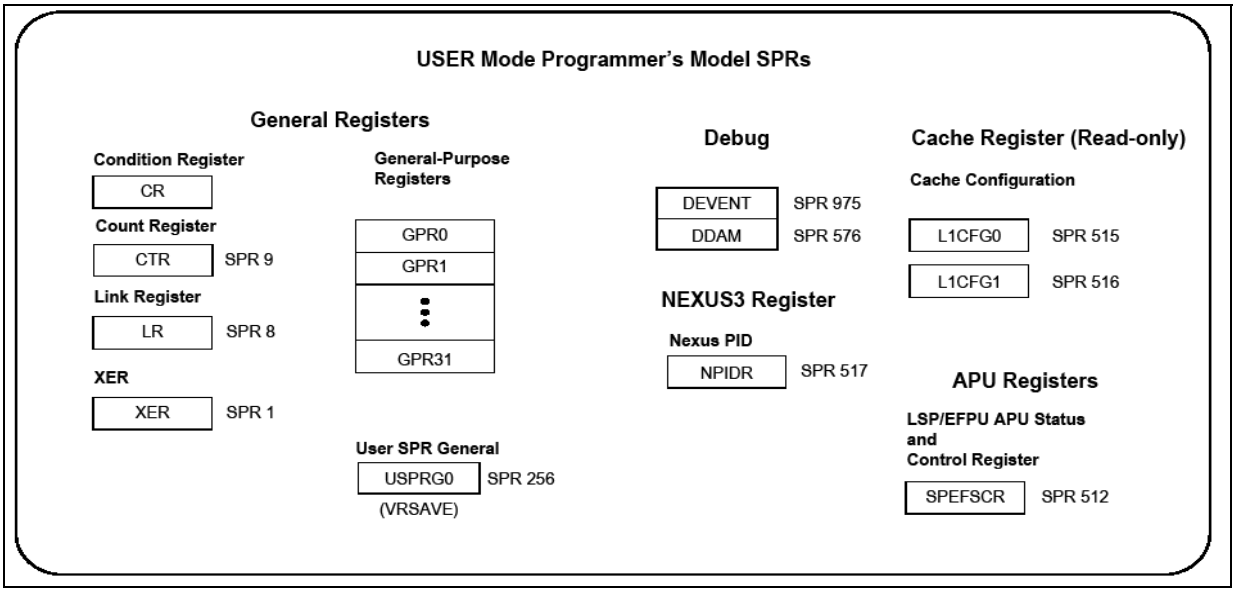
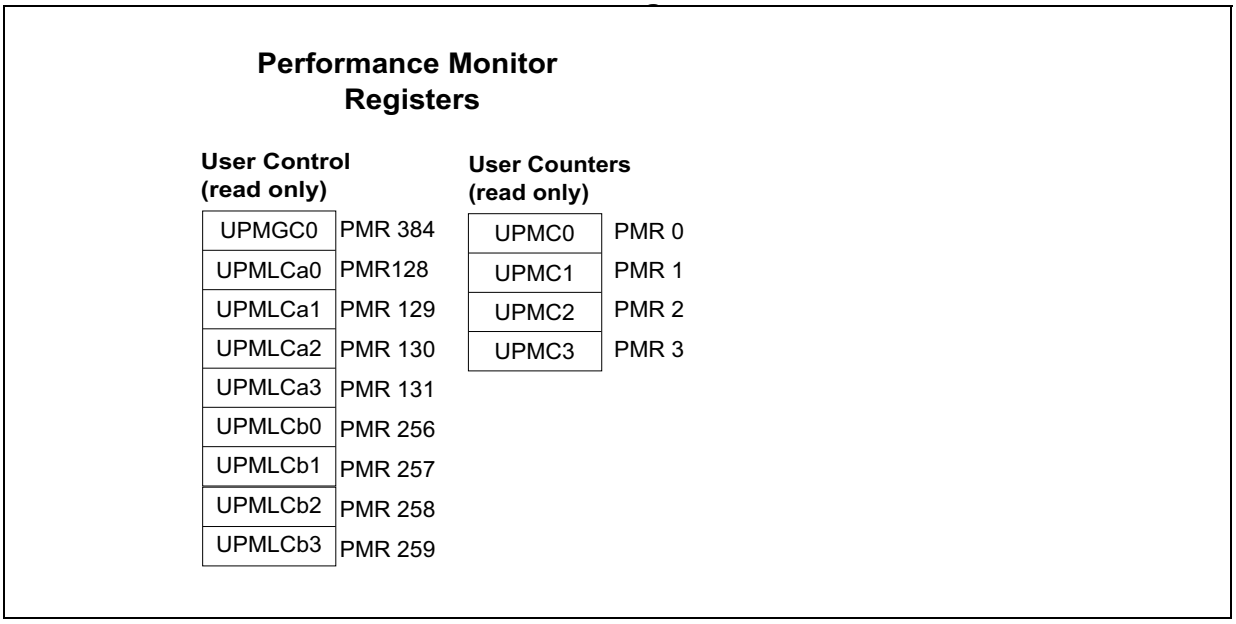


Figure 138. Core (z425n3) User Mode Programmer's Model PMRs



16.3 Dual issue operation

The instruction issue unit attempts to issue a pair of instructions each cycle to the execution units. Source operands for each of the instructions are provided from the GPRs or from the operand feed-forward multiplexers. Data or resource hazards may create stall conditions which cause instruction issue to be stalled for one or more cycles until the hazard is eliminated.

The execution units write the result of a finished instruction onto the proper result bus and into the destination registers. The writeback logic retires an instruction when the instruction has finished execution. Up to three results can be simultaneously written.

Two execution units are provided to allow dual issue of most instructions. Only a single load/store unit is provided. Only a single integer multiply and divide unit is provided, thus a pair of multiply or divide instructions cannot issue simultaneously. In addition, the divide unit is blocking.

Table 228 shows the Core (z425n3) concurrent instruction issue capabilities. Note that data dependencies between instructions will generally preclude dual-issue.

Table 228. Concurrent Instruction Issue capabilities

Class of Instruction	branch	load/store	scalar integer	scalar float	vector integer non-MAC	vector integer MAC	special
branch	—	yes	yes	yes	yes	yes	—
load/store	yes	—	yes	yes	yes	yes	—
scalar integer	yes	yes	yes(1)	yes	yes	yes(2)	—
scalar float	yes	yes	yes	—	yes	yes	—
vector integer non-MAC	yes	yes	yes	yes	—	—	—
vector integer MAC	yes	yes	yes(2)	yes	—	—	—
special	—	—	—	—	—	—	—

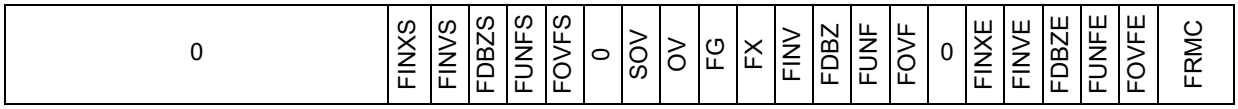
1. excludes multiply or divide class instructions occurring in both issue slots
2. excludes vector MAC/multiply class instructions occurring with scalar multiply, or divide class instructions

16.4 Reservation instructions and cache interactions

The Zen core treats reservation instruction (**lbarx**, **lharx**, **lwarx**, **stbcx.**, **sthcx.** and **stwcx.**) accesses as though they were cache inhibited. A reservation access is always issued to the bus.

16.5 Signal Processing Extension / Embedded Floating-point Status and Control Register (SPEFSCR)

Status and control for embedded floating-point uses the SPEFSCR register. This register is also used by the LSP APU. The SPEFSCR register is implemented as special purpose register (SPR) number 512 and is read and written by the **mfspr** and **mtspr** instructions. The SPEFSCR is shown in *Figure 139*.



SPR - 512; Read/Write; Reset - 0x0

Figure 139. LSP/EFPU Status and Control Register (SPEFSCR)

The SPEFSCR bits are defined in *Table 229*.

Table 229. LSP/EFPU Status and Control Register

Bits	Name	Description
0:9	—	Reserved
10	FINXS	Embedded Floating-point Inexact Sticky Flag The FINXS bit is set to 1 whenever the execution of a floating-point instruction delivers an inexact result and no Floating-point Data exception is taken, or if the result of a Floating-point instruction results in overflow (FOVF=1), but Floating-point Overflow exceptions are disabled (FOVFE=0), or if the result of a Floating-point instruction results in underflow (FUNF=1), but Floating-point Underflow exceptions are disabled (FUNFE=0), and no Floating-point Data exception occurs. The FINXS bit remains set until it is cleared by a **mtspr** instruction specifying the SPEFSCR register.
11	FINVS	Embedded Floating-point Invalid Operation Sticky Flag The FINVS bit is set to a 1 when a floating-point instruction sets the FINV bit to 1. The FINVS bit remains set until it is cleared by a **mtspr** instruction specifying the SPEFSCR register.
12	FDBZS	Embedded Floating-point Divide by Zero Sticky Flag The FDBZS bit is set to 1 when a floating-point divide instruction sets the FDBZ bit to 1. The FDBZS bit remains set until it is cleared by a **mtspr** instruction specifying the SPEFSCR register.
13	FUNFS	Embedded Floating-point Underflow Sticky Flag The FUNFS bit is set to 1 when a floating-point instruction sets the FUNF bit to 1. The FUNFS bit remains set until it is cleared by a **mtspr** instruction specifying the SPEFSCR register.
14	FOVFS	Embedded Floating-point Overflow Sticky Flag The FOVFS bit is set to 1 when a floating-point instruction sets the FOVF bit to 1. The FOVFS bit remains set until it is cleared by a **mtspr** instruction specifying the SPEFSCR register.
15	—	Reserved
16	SOV	Summary integer overflow Defined by LSP APU.
17	OV	Integer overflow Defined by LSP APU.
18	FG	Embedded Floating-point Guard bit FG is supplied for use by the Floating-point Round exception handler. FG is zeroed if a Floating-point Data Exception occurs.

Table 229. LSP/EFPU Status and Control Register (continued)

Bits	Name	Description
19	FX	Embedded Floating-point Sticky bit FX is supplied for use by the Floating-point Round exception handler.FX is zeroed if a Floating-point Data Exception occurs.
20	FINV	Embedded Floating-point Invalid Operation / Input error The FINV bit is set to 1 if the A or B operand of a floating-point instruction is Infinity, NaN, or Denorm, or if the operation is a divide and the dividend and divisor are both 0.
21	FDBZ	Embedded Floating-point Divide by Zero The FDBZ bit is set to 1 when a floating-point divide instruction executed with divisor of 0, and the l dividend is a finite non-zero number.
22	FUNF	Embedded Floating-point Underflow The FUNF bit is set to 1 when the execution of a floating-point instruction results in an underflow.
23	FOVF	Embedded Floating-point Overflow The FOVF bit is set to 1 when the execution of a floating-point instruction results in an overflow.
24	—	Reserved
25	FINXE	Embedded Floating-point Inexact Exception Enable 0 = Exception disabled 1 = Exception enabled If the exception is enabled, a Floating-point Round exception is taken if the result of a Floating-point instruction does not result in overflow or underflow, and the result is inexact (FG | FX = 1), or if the result of a Floating-point instruction does result in overflow (FOVF=1) but Floating-point Overflow exceptions are disabled (FOVFE=0), or if the result of a Floating-point instruction results in underflow (FUNF=1 or FUNFH=1) but Floating-point Underflow exceptions are disabled (FUNFE=0), and no Floating-point Data exception occurs.
26	FINVE	Embedded Floating-point Invalid Operation / Input Error Exception Enable 0 = Exception disabled 1 = Exception enabled If the exception is enabled, a Floating-point Data exception is taken if the FINV bit is set by a floating-point instruction.
27	FDBZE	Embedded Floating-point Divide by Zero Exception Enable 0 = Exception disabled 1 = Exception enabled If the exception is enabled, a Floating-point Data exception is taken if the FDBZ bit is set by a floating-point instruction.
28	FUNFE	Embedded Floating-point Underflow Exception Enable 0 = Exception disabled 1 = Exception enabled If the exception is enabled, a Floating-point Data exception is taken if the FUNF bit is set by a floating-point instruction.

Table 229. LSP/EFPU Status and Control Register (continued)

Bits	Name	Description
29	FOVFE	Embedded Floating-point Overflow Exception Enable 0 = Exception disabled 1 = Exception enabled If the exception is enabled, a Floating-point Data exception is taken if the FOVF bit is set by a floating-point instruction.
30:31	FRMC	Embedded Floating-point Rounding Mode Control 00 = Round to Nearest 01 = Round toward Zero 10 = Round toward +Infinity 11 = Round toward -Infinity

16.6 Cache

This chapter describes the cache registers, cache control instructions, and various cache operations.

16.6.1 Cache overview

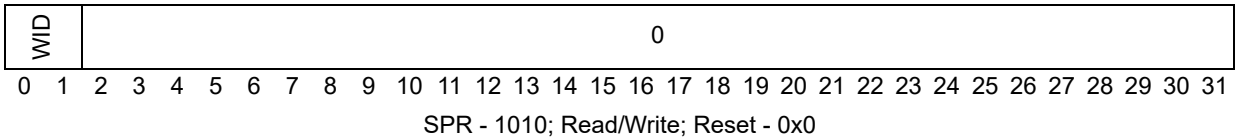
The Core (z425n3) processor supports an 8 Kbyte 2-way set-associative instruction cache with a 32-byte line size. The cache improves system performance by providing low-latency data to the Core (z425n3) instruction pipeline, which decouples processor performance from system memory performance.

The Core (z425n3) processor also contains an 8-entry store buffer to decouple store completion to the processor from store completion on the data interface to further improve system performance.

Instruction addresses from the processor are used to index the cache array. If the access address matches a valid cache tag entry, the access hits in the cache.

16.6.2 L1 Cache Control and Status Register 0 (L1CSR0)

The L1 Cache Control and Status Register 0 (L1CSR0) is a 32-bit register used for general control of a data cache as well as providing general control over disabling ways in both caches. The L1CSR0 register implements the WID control field for instruction cache way disabling. The other fields are reserved. The L1CSR0 register is accessed using a **mfspr** or **mtspr** instruction. The SPR number for L1CSR0 is 1010 in decimal. The L1CSR0 register is shown in *Figure 140*.



SPR - 1010; Read/Write; Reset - 0x0

Figure 140. L1 Cache Control and Status Register 0 (L1CSR0)

The L1CSR0 bits are described in *Table 230*.

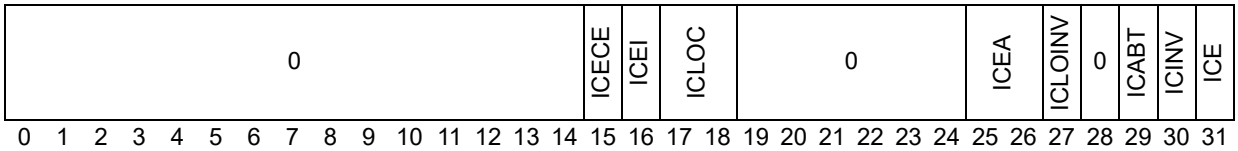
Table 230. L1CSR0 Field Descriptions

Bits	Name	Description
0:1	WID	Way Instruction Disable. 0 = The corresponding way in the instruction cache is available for replacement by instruction miss line fills. 1 = The corresponding way instruction cache is not available for replacement by instruction miss line fills. Bit 0 corresponds to way 0. Bit 1 corresponds to way 1. The WID bits may be used for locking ways of the instruction cache, and also affect the replacement policy of the instruction cache.
2:31	—	Reserved(1)

1. These bits are not implemented and should be written with zero for future compatibility.

16.6.3 L1 Cache Control and Status Register 1 (L1CSR1)

The L1 Cache Control and Status Register 1 (L1CSR1) is a 32-bit register used for general control of the instruction cache. The L1CSR1 register is accessed using a **mfspr** or **mtspr** instruction. The SPR number for L1CSR1 is 1011 in decimal. The L1CSR1 register is shown in *Figure 141*.



SPR - 1011; Read/Write; Reset - 0x0

Figure 141. L1 Cache Control and Status Register 1 (L1CSR1)

The L1CSR1 bits are described in *Table 231*.

Table 231. L1CSR1 Field Descriptions

Bits	Name	Description
0:14	—	Reserved(1)
15	ICECE	Instruction Cache Error Checking Enable 0 - Error Checking is disabled 1 - Error Checking is enabled
16	ICEI	Instruction Cache Error Injection Enable 0 - Cache Error Injection is disabled 1 - A double-bit error will be injected into each double word written into the cache by inverting the two uppermost parity check bits. ICEI will cause injection of errors regardless of the setting of ICECE, although reporting of errors will be masked when ICECE=0.

Table 231. L1CSR1 Field Descriptions (continued)

Bits	Name	Description
17:18	ICLOC	Instruction Cache Lockout Control 00 - Cache line lockout is disabled (and array LO indicators are ignored). 01 - Cache line lockout is enabled. Cache lines with any tag errors or any data errors will have the corresponding lockout indicators set. A Machine check will not be generated for the error if the operation would have normally generated one unless a locked line is locked out. 10 - Cache line lockout is enabled. Cache lines with any tag or data errors will have the corresponding lockout indicators set. A Machine check will still be generated for an error if the operation would have normally generated one. 11 - Cache line lockout is enabled. Cache lines with uncorrectable tag errors or any data errors will have the corresponding lockout indicators set. A Machine check will not be generated for the error if the operation would have normally generated one unless a locked line is locked out. If a correctable tag error occurs, and is re-detected on recycling the access after a correction cycle for the tag is performed, the line is locked out regardless of detecting a single-bit or multi-bit error. **Note:** for the **icbi** instruction, no lockout operation is performed, regardless of error conditions.
19:24	—	Reserved(1)
25:26	ICEA	Instruction Cache Error Action 00 - Error Detection causes Machine Check exception. 01 - Error Detection causes Correction/Auto-invalidation. No machine check is generated for most cases. Correction is performed for single-bit tag errors, and lines with multi-bit tag errors are invalidated. Correction is performed for single or multi-bit data errors on cache hits by reloading of the line. 1x - Reserved
27	ICLOINV	Instruction Cache Lockout Indicator Invalidate When ICINV = 0: Reserved, do not set to 1 When ICINV = 1: 0 - No cache lockout bit invalidate 1 - Cache lockout indicator invalidation operation When written to a ‘1’ in conjunction with writing ICINV to a ‘1’, a cache lockout indicator invalidation operation is initiated by hardware, and the LO bits are cleared, removing all line lockout status. Once complete, this bit is reset to ‘0’. Writing a ‘1’ while an invalidation operation is in progress will result in an undefined operation. Writing a ‘0’ to this bit while an invalidation operation is in progress will be ignored. Cache lockout bit invalidation operations require approximately 66 cycles to complete. Invalidation occurs regardless of the instruction cache enable (ICE) value.
28	—	Reserved(1)
29	ICABT	Instruction Cache Operation Aborted Indicates a Cache Invalidate operation was aborted prior to completion. This bit is set by hardware on an aborted condition, and will remain set until cleared by software writing 0 to this bit location.

Table 231. L1CSR1 Field Descriptions (continued)

Bits	Name	Description
30	ICINV	Instruction Cache Invalidate 0 - No cache invalidate 1 - Cache invalidation operation When written to a ‘1’, a cache invalidation operation is initiated by hardware. Once complete, this bit is reset to ‘0’. Writing a ‘1’ while an invalidation operation is in progress will result in an undefined operation. Writing a ‘0’ to this bit while an invalidation operation is in progress will be ignored. Cache invalidation operations require approximately 130 cycles to complete. Invalidation occurs regardless of the enable (ICE) value.
31	ICE	Instruction Cache Enable 0 - Cache is disabled 1 - Cache is enabled When disabled, cache lookups are not performed for instruction accesses. Other L1CSR1 cache control operations are still available and are not affected by ICE.

1. These bits are not implemented and should be written with zero for future compatibility.

16.6.4 L1 Cache Configuration Register 0 (L1CFG0)

The L1 Cache Configuration Register 0 (L1CFG0) is a 32-bit read-only register. L1CFG0 provides information about the configuration of the Core (z425n3) L1 data cache design. The contents of the L1CFG0 register can be read using a **mfspr** instruction. The SPR number for L1CFG0 is 515 in decimal. The L1CFG0 register is shown in *Figure 142*.

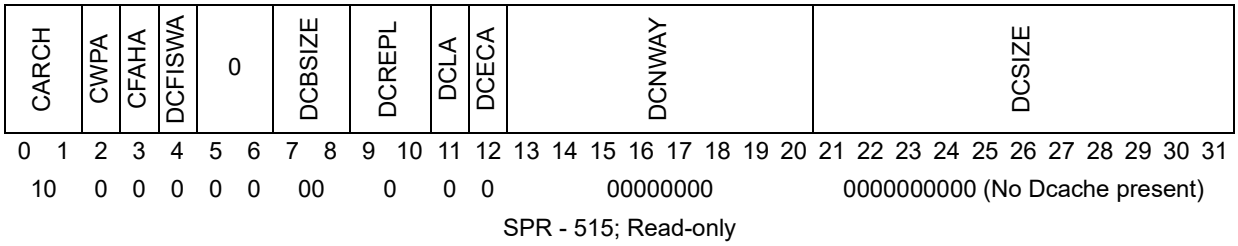


Figure 142. L1 Cache Configuration Register 0 (L1CFG0)

The L1CFG0 register indicates a cache architecture value (CARCH) of 2‘b10, indicating that an ICache but no DCache is present on Core (z425n3).

The L1CFG0 bits are described in *Table 232*.

Table 232. L1CFG0 Field Descriptions

Bits	Name	Description
0:1	CARCH	Cache Architecture 00 - The cache architecture is Harvard
2	CWPA	Cache Way Partitioning Available 1 - The caches support partitioning of way availability for I/D accesses

Table 232. L1CFG0 Field Descriptions (continued)

Bits	Name	Description
3	DCFAHA	Data Cache Flush All by Hardware Available 0 - The data cache does not support Flush All in Hardware
4	DCFISWA	Data Cache Flush/Invalidate by Set and Way Available 1 - The data cache supports invalidation by Set and Way via L1FINV0
5:6	—	Reserved - read as zeros
7:8	DCBSIZE	Data Cache Block Size 00 - The data cache implements a block size of 32 bytes
9:10	DCREPL	Data Cache Replacement Policy 11- The data cache implements a FIFO replacement policy
11	DCLA	Data Cache Locking APU Available 0 - The data cache does not implement the line locking APU
12	DCECA	Data Cache Error Checking Available 1 - The data cache implements error checking
13:20	DCNWAY	Data Cache Number of Ways 0x01 - The data cache is 2-way set-associative
21:31	DCSIZE	Data Cache Size 0x004 - The size of the data cache is 4 Kbyte

16.6.5 L1 Cache Configuration Register 1 (L1CFG1)

The L1 Cache Configuration Register 1 (L1CFG1) is a 32-bit read-only register. L1CFG1 provides information about the configuration of the Core (z425n3) L1 instruction cache design. The contents of the L1CFG1 register can be read using a **mfspr** instruction. The SPR number for L1CFG1 is 516 in decimal. The L1CFG1 register is shown in *Figure 143*.

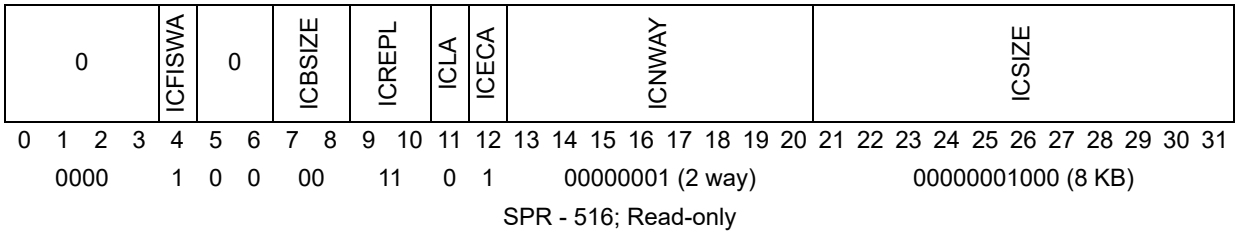


Figure 143. L1 Cache Configuration Register 1 (L1CFG1)

The L1CFG1 bits are described in *Table 233*.

Table 233. L1CFG1 Field Descriptions

Bits	Name	Description
0:3	—	Reserved - read as zeros
4	ICFISWA	Instruction Cache Flush/Invalidate by Set and Way Available 1 - The instruction cache supports invalidation by Set and Way via L1FINV1
5:6	—	Reserved - read as zeros

Table 233. L1CFG1 Field Descriptions (continued)

Bits	Name	Description
7:8	ICBSIZE	Instruction Cache Block Size 00 - The instruction cache implements a block size of 32 bytes
9:10	ICREPL	Instruction Cache Replacement Policy 11 - The instruction cache implements a FIFO replacement policy
11	ICLA	Instruction Cache Locking APU Available 0 - The instruction cache does not implement the line locking APU
12	ICECA	Instruction Cache Error Checking Available 1 - The instruction cache implements error checking
13:20	ICNWAY	Instruction Cache Number of Ways 0x01 - The instruction cache is 2-way set-associative
21:31	ICSIZE	Instruction Cache Size 0x008 - The size of the instruction cache is 8 Kbyte

16.6.6 Cache Invalidate by Set and Way

Core (z425n3) supports cache set/way invalidation under software control. The instruction cache may be invalidated by set and way through a **mtspr l1finv1** instruction.

The L1 Flush and Invalidate Control Registers (L1FINV1) are 32-bit SPRs used to select a cache set and way to be invalidated. This function is available even when the instruction cache is disabled.

16.6.6.1 L1FINV1

The SPR number for L1FINV1 is 959 in decimal. The L1FINV1 register is shown in *Figure 144*.



SPR - 959; Read/Write; Reset - 0x0

Figure 144. L1 Flush/Invalidate Register 1 (L1FINV1)

The L1FINV1 bits are described in *Table 234*.

Table 234. L1FINV1 Field Descriptions

Bits	Name	Description
0:6	—	Reserved for way extension
7	CWAY	Cache Way Specifies the instruction cache way to be selected
8:19	—	Reserved for set extension
20:26	CSET	Cache Set Specifies the instruction cache set to be selected

Table 234. L1FINV1 Field Descriptions (continued)

Bits	Name	Description
27:29	—	Reserved for set/command extension
30:31	CCMD	Cache Command 00 = The data contained in this entry is invalidated 01 = Reserved 1x = Reserved

For invalidation operations via L1FINV1, tag ECC errors and data EDC errors are ignored, and the selected line will be invalidated regardless of any error.

16.6.7 Cache EDC/ECC protection

Cache protection is supported for both the tag and data arrays of the instruction cache. Tag array support error detection and correction. Each double word in the data arrays of the ICache have single and double bit error detection.

The error detection codes also cover the cache index address of the cache line, providing additional protection against addressing errors. If any type of error (including a single-bit error in one of the index bits) is encountered in one of the index address bits, it is treated as an uncorrectable tag ECC error, to protect against internal addressing failures.

Tag ECC and data EDC checking is controlled by the $L1CSR1_{ICECE}$ control field. When error checking is enabled, checking is performed on each cache access.

For cache lookups, if an uncorrectable tag ECC error is detected on any portion of the accessed tags, a tag ECC error is signaled, regardless of whether a cache hit or miss occurs. Otherwise, if a cache hit for an instruction fetch occurs and a data EDC error is detected on any portion of the accessed data, an error is also signaled.

Signaling of an ECC error or EDC error may cause a Machine Check exception to occur, and one or more syndrome bits to be set in the Machine Check Syndrome register, or may instead result in a correction/auto-invalidation operation and not result in an exception being signaled, or both may occur, depending on the error action control setting in the appropriate cache control register.

16.6.7.1 Cache line lockout

In addition to the ECC/EDC protection employed for the cache, each cache line in the Icache has a lockout indicator composed of a redundant set of lockout bits which can be selectively set when certain errors occur in either the tag or data portion of the cache line. Use of the lockout function and control over error conditions which cause a lockout to occur are controlled by $L1CSR1_{ICLOC}$. When the lockout function is enabled, lines which encounter selected tag ECC or data EDC errors on normal cacheable instruction fetch, load, or store accesses will have their lockout bits set. When the lockout indicator is set, the line will not be replaced and will remain in an invalid state, effectively disabling it. Also, future tag ECC or data EDC errors on the line are ignored. Lockout indicators may be cleared by software performing a cache invalidation operation (setting $L1CSR1_{ICINV}$) in conjunction with a cache lockout indicator invalidation (setting $L1CSR1_{ICLOINV}$). Three stored redundant lockout bits are provided to allow for single and double-bit parity errors to be detected on the lockout indicators themselves.

When operating in machine check mode, if lockout controls are enabled via $L1CSR1_{ICLOC}$, lockout bit parity errors on any line detected on a cacheable cache lookup will generate a machine check exception. If correction/auto-invalidation is instead enabled, on each cache lookup operation, if a single- or double-bit lockout bit parity error is detected in one or more ways and lockout controls are enabled, the lockout error(s) will be corrected by re-writing all lockout bits to the asserted state, and no machine check is generated.

Lockout bit parity errors are ignored for non-cacheable accesses in either error action mode. In addition, to avoid certain exception conditions and for consistency with error reporting, non-cacheable accesses and specialized load/store accesses, do not set LO bits; instead, cache contents are ignored for those errors since the cache is bypassed. A future cacheable normal access will perform the lockout function if required.

16.6.8 Cache error injection

Cache error injection provides a way to test error recovery by intentionally injecting EDC/ECC errors into the instruction or data cache, or both.

Error injection into the instruction cache operates as follows:

- If $L1CSR1_{ICEI}$ is set, any instruction cache line fill to the instruction cache data has the associated two most significant EDC/ECC check bits inverted in the instruction cache data array for each double word loaded.

Cache parity error injection is not performed for cache debug write accesses, since parity bit values written can be directly controlled.

In order to clear the EDC/ECC bit errors, a cache invalidation or an invalidation of the lines which have had an injected parity error may be performed. Line invalidation may be performed by an **icbi** instruction, or an L1FINV1 invalidation operation.

16.7 Exceptions

Interrupts implemented in Zen and the exception conditions that cause them are listed in *Table 235*.

Table 235. Exceptions and conditions

Interrupt Type	Interrupt Vector Offset Value	Causing Conditions
System reset	none, vector to [p_rstbase[0:29]] || 2'b00	1) Reset
Critical Input	0x00(1)	**p_critint_b** is asserted and MSR_{CE}=1
Machine check	0x10	1) **p_mcp_b** transitions from negated to asserted 2) ISI or Bus Error on first instruction fetch for an exception handler 3) EDC/ECC Error signaled on cache or Local Memory access 4) External bus error 5) Stack limit check failure
Machine check (NMI)	0x10	1) Non-Maskable Interrupt

Table 235. Exceptions and conditions (continued)

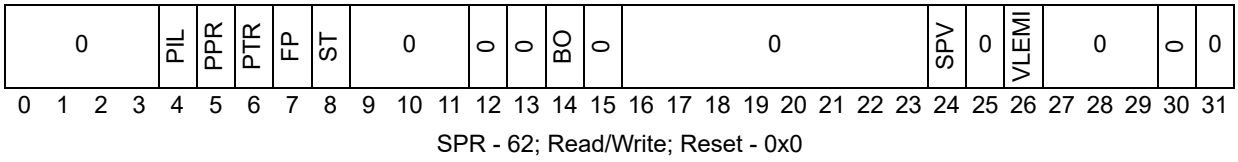
Interrupt Type	Interrupt Vector Offset Value	Causing Conditions
Data Storage	0x20	1) Access control
Instruction Storage	0x30	1) Access control
External Input	0x40(2)	Interrupt Controller interrupt and MSR_{EE}=1
Alignment	0x50	1) **lmw**, **stmw** not word aligned 2) **lwarx** or **stwcx.** not word aligned, **lharx** or **sthcx.** not halfword aligned 3) dcbz 4) LSP ld and st instructions not properly aligned
Program	0x60	Illegal, Privileged, Trap
Performance Monitor	0x70	Performance Monitor Enabled Condition or Event w/$PMGC0_{UDI}$=0
System call	0x80	Execution of the System Call (**se_sc**) instruction
Debug	0x90	Trap, Instruction Address Compare, Data Address Compare, Instruction Complete, Branch Taken, Return from Interrupt, Interrupt Taken, Debug Counter, External Debug Event, Unconditional Debug Event, Performance Monitor Enabled Condition or Event w/$PMGC0_{UDI}$=1, MPU
EFPU Data Exception	0xA0	
EFPU Round Exception	0xB0	

1. Autovectored Critical Input interrupts use this offset value. Vectored interrupts supply an interrupt vector offset directly.
2. Autovectored External Input interrupts use this offset value. Vectored interrupts supply an interrupt vector offset directly.

16.7.1 Exception syndrome register

The Exception Syndrome Register (ESR) provides a *syndrome* to differentiate between exceptions that can generate the same interrupt type.

The ESR bits are defined in *Table 236*.



SPR - 62; Read/Write; Reset - 0x0

Figure 145. Exception Syndrome Register (ESR)

Table 236. ESR Bit Settings

Bit(s)	Name	Description	Associated Interrupt Type
0:3	—	Reserved	—
4	PIL	Illegal Instruction exception (For Core (z425n3), PIL is used for both illegal and unimplemented instructions)	Program
5	PPR	Privileged Instruction exception	Program
6	PTR	Trap exception	Program
7	FP	Floating-point operation	Program
8	ST	Store operation	Alignment Data Storage
9:11	—	Reserved	—
12	--	Reserved	—
13	--	Reserved	—
14	BO	Byte Ordering exception Mismatched Instruction Storage exception	Instruction Storage
15	--	Reserved	—
16:23	—	Reserved	—
24	SPV	LSP/EFPU APU Operation	EFPU Floating-point Data Exception EFPU Floating-point Round Exception Alignment Data Storage
25	—	Reserved	—
26	VLEMI	VLE Mode Instruction	EFPU Floating-point Data Exception EFPU Floating-point Round Exception Data Storage Instruction Storage Alignment Program System Call
27:29	—	Reserved	—
30	—	Reserved	—
31	—	Reserved	—

16.7.2 Machine State Register

The Machine State Register defines the state of the processor. The Zen MSR is shown in *Figure 146*.

0:5	6	7:12	13	14	15	16	17	18	19	20	21	22	23	24:25	26	27	28	29	30	31
0	0	0	0	CE	0	EE	PR	0	ME	0	0	DE	0	0	IS	DS	0	PMM	RI	0

Read/ Write; Reset - 0x0

Figure 146. Machine State Register (MSR)

The MSR bits are defined in *Table 237*.

Table 237. MSR Field Descriptions

Bit(s)	Name	Description
0:5	—	Reserved
6	—	Reserved
7:12	—	Reserved
13	—	Reserved
14	CE	Critical Interrupt Enable 0 - Critical Input interrupts are disabled. 1 - Critical Input interrupts are enabled.
15	—	Reserved
16	EE	External Interrupt Enable 0 - External Input interrupts are disabled. 1 - External Input interrupts are enabled.
17	PR	Problem State 0 - The processor is in supervisor mode, can execute any instruction, and can access any resource (such as GPRs, SPRs and MSR) 1 - The processor is in user mode, cannot execute any privileged instruction, and cannot access any privileged resource.
18	—	Reserved
19	ME	Machine Check Enable 0 - Asynchronous Machine Check interrupts are disabled. 1 - Asynchronous Machine Check interrupts are enabled.
20	—	Reserved
21	—	Reserved
22	DE	Debug Interrupt Enable 0 - Debug interrupts are disabled. 1 - Debug interrupts are enabled.
23	—	Reserved
24	—	Reserved
25	—	Reserved
26	—	Reserved
27	—	Reserved
28	—	Reserved

Table 237. MSR Field Descriptions (continued)

Bit(s)	Name	Description
29	PMM	PMM Performance monitor mark bit. System software can set PMM when a marked process is running to enable statistics to be gathered only during the execution of the marked process. MSR_{PR} and MSR_{PMM} together define a state that the processor (supervisor or user) and the process (marked or unmarked) may be in at any time. If this state matches an individual state specified in the Performance Monitor registers PMLCa n, the state for which monitoring is enabled, counting is enabled.
30	RI	Recoverable Interrupt - This bit is provided for software use to detect nested machine check exception conditions. This bit is cleared by hardware when a Machine Check interrupt is taken
31	—	Reserved

16.7.3 Machine Check Syndrome Register (MCSR)

When the processor takes a machine check interrupt, it updates the Machine Check Syndrome register (MCSR) to differentiate between machine check conditions. The MCSR is shown in *Figure 147*.

MCP	IC_DPERR	0	DC_DPERR	EXCP_ERR	IC_TPERR	DC_TPERR	IC_LKERR	DC_LKERR	0	NMI	MAV	MEA	U	IF	LD	ST	G	0	STACK_ERR	IMEM_PERR	DMEM_RDPERR	DMEM_WRPERR	0	BUS_IRERR	BUS_DRERR	BUS_WRERR	BUS_WRDSI	0
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31

SPR - 572; Read/Clear; Reset - 0x0

Figure 147. Machine Check Syndrome Register (MCSR)

Table 238 describes MCSR fields. The MCSR indicates the source of a machine check condition.

All bits in the MCSR are implemented as “write ‘1’ to clear”. Software in the machine check handler is expected to clear the MCSR bits it has sampled prior to re-enabling MSR_{ME} to avoid a redundant machine check exception and to prepare for updated status bit information on the next machine check interrupt.

Note that any set bit in the MCSR other than status-type bits will cause a subsequent machine check interrupt once MSR_{ME}=1.

Table 238. Machine Check Syndrome Register (MCSR)

Bit	Name	Description	Exception Type(1)	Recoverable
0	MCP	Machine check input pin	Async Mchk	Maybe
1	IC_DPERR	Instruction Cache data array parity error	Async Mchk	Precise
2	—	Reserved		—

Table 238. Machine Check Syndrome Register (MCSR) (continued)

Bit	Name	Description	Exception Type(1)	Recoverable
3	DC_DPERR(2)	Data Cache data array parity error	Async Mchk	Maybe
4	EXCP_ERR	ISI or Bus Error on first instruction fetch for an exception handler	Async Mchk	Precise
5	IC_TPERR	Instruction Cache Tag parity error	Async Mchk	Precise
6	DC_TPERR(2)	Data Cache Tag parity error	Async Mchk	Maybe
7	IC_LKERR	Instruction Cache Lock error Indicates a cache control operation or invalidation operation invalidated one or more lines in a locked way of the Icache for certain situations. May also be set on locked line refill error.	Async Mchk	—
8	DC_LKERR(2)	Data Cache Lock error Indicates a cache control operation or invalidation operation invalidated one or more lines in a locked way of the Dcache for certain situations. May also be set on locked line refill error.	Async Mchk	—
10:11	—	Reserved		—
11	NMI	NMI input pin	NMI	—
12	MAV	MCAR Address Valid Indicates that the address contained in the MCAR was updated by hardware to correspond to the first detected Async Mchk error condition	Status	—
13	MEA	MCAR holds Effective Address If MAV=1,MEA=1 indicates that the MCAR contains an effective address and MEA=0 indicates that the MCAR contains a physical address	Status	—
14	U	User Indicates the value captured in MCAR was generated in user mode.	Status	—
15	IF	Instruction Fetch Error Report An error occurred during the fetch of an instruction and the instruction attempted to execute. This could be due to an ECC error, or an external bus error. MCSRR0 contains the instruction address.	Error Report	Precise
16	LD	Load type instruction Error Report An error occurred during the attempt to execute the load type instruction located at the address stored in MCSRR0. This could be due to an ECC error, stack limit check error, or an external bus error.	Error Report	Precise

Table 238. Machine Check Syndrome Register (MCSR) (continued)

Bit	Name	Description	Exception Type(1)	Recoverable
17	ST	Store type instruction Error Report An error occurred during the attempt to execute the store type instruction located at the address stored in MCSRR0. This could be due to an ECC error, stack limit check error, or on certain external bus errors.	Error Report	Precise
18	G	Guarded instruction Error Report An error occurred during the attempt to execute the load or store type instruction located at the address stored in MCSRR0 and the access was guarded and encountered an error on the external bus, or an uncorrectable DMEM error.	Error Report	Precise
19:20	—	Reserved		—
21	STACK_ERR	Stack Access Limit Check Error Indicates a limit check failure on a CPU data access using R1 in the <EA> calculation.	Async Mchk	Precise
22	IMEM_PERR	Instruction Mem (IMEM) Parity Error Indicates an uncorrectable error in the IMEM on a CPU port access.	Async Mchk	Precise
23	DMEM_RDPERR	Data Mem (DMEM) Parity Error Indicates an uncorrectable error in the DMEM on a CPU port read access.	Async Mchk	Precise
24	DMEM_WRPERR	Data Mem (DMEM) Write Parity Error Indicates an uncorrectable error in the DMEM on a CPU port write access.	Async Mchk	Maybe
25:26	—	Reserved		—
27	BUS_IRERR	Read bus error on Instruction recovery linefill	Async Mchk	Precise if data used
28	BUS_DRERR	Read bus error on data load	Async Mchk	Precise if data used
29	BUS_WRERR	Write bus error on store to bus or DMEM imprecise write error	Async Mchk	Unlikely
30	BUS_WRDSI	Write bus error on buffered store to bus with DSI signaled. Set concurrently with BUS_WRERR for this case	Async Mchk	Unlikely
31	—	Reserved		—

1. The Exception Type indicates the exception type associated with a given syndrome bit

   - “Error Report” indicates that this bit is only set for error report exceptions which cause machine check interrupts. These bits are only updated when the machine check interrupt is actually taken. Error report exceptions are not gated by MSR_{ME}. These are synchronous exceptions. These bits will remain set until cleared by software writing a “1” to the bit position(s) to be cleared.

   - “Status” indicates that this bit is provides additional status information regarding the logging of a machine check exception. These bits will remain set until cleared by software writing a “1” to the bit position(s) to be cleared.

   - “NMI” indicates that this bit is only set for the non-maskable interrupt type exception which causes a machine check interrupt. This bit is only updated when the machine check interrupt is actually taken. NMI exceptions are not gated by MSR_{ME}. This is an asynchronous exception. This bit will remain set until cleared by software writing a “1” to the bit position.

   - “Async Mchk” indicates that this bit is set for an asynchronous machine check exception. These bits are set immediately upon detection of the error. Once any “Async Mchk” bit is set in the MCSR, a machine check interrupt will occur if MSR_{ME}=1. If MSR_{ME}=0, the machine check exception will remain pending. These bits will remain set until cleared by software writing a “1” to the bit position(s) to be cleared.

2. Will not be set by Core (z425n3) since no data cache is present.

16.7.4 Interrupt Vector Prefix Registers (IVPR)

The Interrupt Vector Prefix Register is used during interrupt processing for determining the starting address of a software handler used to handle an interrupt. The value of the Vector Offset selected for a particular interrupt type is concatenated with the Vector Base value held in the Interrupt Vector Prefix register (IVPR) to form an instruction address from which execution is to begin. The format of IVPR is shown in *Figure 148*.

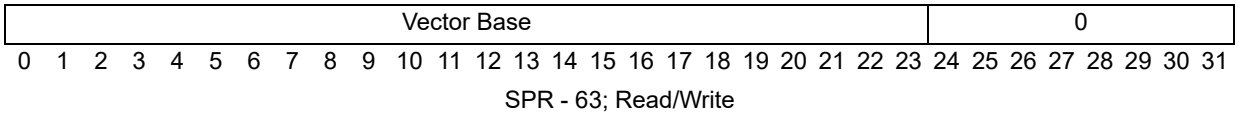


Figure 148. Zen Interrupt Vector Prefix Register (IVPR)

The IVPR fields are defined in *Table 239*.

Table 239. IVPR Register Fields

Bit(s)	Name	Description
0:23 (32:55)	Vector Base	Vector Base This field is used to define the base location of the vector table, aligned to a 256 byte boundary. This field provides the high-order 24 bits of the location of all interrupt handlers (unless hardware vectoring is used). The vector offset value appropriate for the type of exception being processed is concatenated with the IVPR Vector Base to form the address of the handler in memory. For hardware-vectored interrupts, the value of the supplied interrupt vector is logically OR’ed with the low order bits of the Vec Base Field to determine the interrupt handler address.
24:31 (56:63)	—	Reserved

16.7.5 Interrupt definitions

16.7.5.1 Critical input interrupt (Offset 0x00)

A Critical Input exception is signaled to the processor by the assertion of the critical interrupt pin. If the exception is enabled by MSR_{CE}, the Critical Input interrupt is taken.

A Critical Input interrupt may be delayed by other higher priority exceptions or if MSR_{CE} is cleared when the exception occurs.

Table 240 lists register settings when a Critical Input interrupt is taken.

Table 240. Critical Input Interrupt—Register Settings

Register	Setting Description		
CSRR0	Set to the effective address of the instruction that the processor would have attempted to execute next if no exception conditions were present.		
CSRR1	Set to the contents of the MSR at the time of the interrupt		
MSR	SPV 0 WE 0 CE 0 EE 0 PR 0	FP 0 ME— FE0 0 DE—/0(1)	FE1 0 IS 0 DS 0 PMM 0 RI—
ESR	Unchanged		
MCSR	Unchanged		
DEAR	Unchanged		
Vector	$IVPR_{0:23}$ || 0x00 (autovectored) $IVPR_{0:15}$ || ($IVPR_{16:29}$ | **p_voffset[0:13])** || 2‘b00 (non-autovectored)		

1. Clearing of DE is optionally supported by control in HID0.

The MSR_{DE} bit is not automatically cleared by a Critical Input interrupt, but can be configured to be cleared via the HID0 register ($HID0_{CICLRDE}$).

16.7.5.2 Machine check interrupt (Offset 0x10)

The Machine Check APU defines a separate set of save/restore registers (MCSRR0/1), a Machine Check Syndrome register (MCSR) to record the source(s) of machine checks, and a Machine Check Address register (MCAR) to hold an address associated with a machine check for certain classes of machine checks. Return from Machine Check instructions (**se_rfmci**) are also provided to support returns using MCSRR0/1.

The MSR_{DE} bit is not automatically cleared by a Machine Check exception, but can be configured to be cleared or left unchanged via the HID0 register ($HID0_{MCCLRDE}$).

When a Machine Check interrupt is taken, registers are updated as shown in *Table 241*.

Table 241. Machine Check Interrupt - Register Settings

Register	Setting Description
MCSRR0	On a best-effort basis sets this to the address of some instruction that was executing or about to be executing when the machine check condition occurred.
MCSRR1	Set to the contents of the MSR at the time of the interrupt

Table 241. Machine Check Interrupt - Register Settings (continued)

Register	Setting Description		
MSR	SPV 0 WE 0 CE 0 EE 0 PR 0	FP 0 ME 0 FE0 0 DE 0/—(1)	FE1 0 IS 0 DS 0 PMM 0 RI 0
ESR	Unchanged		
MCSR	Updated to reflect the source(s) of a machine check. Hardware only sets appropriate bits, no previously set bits are cleared by hardware.		
Vector	$IVPR_{0:23}$ || 0x10		

1. Clearing of DE is optionally supported by control in HID0.

16.7.5.3 Data storage interrupt (Offset 0x20)

A Data Storage interrupt (DSI) may occur if no higher priority exception exists and a Read or Write Access Control exception condition exists.

Table 242 lists register settings when a DSI is taken.

Table 242. Data Storage Interrupt—Register Settings

Register	Setting Description		
SRR0	Set to the effective address of the excepting load/store instruction.		
SRR1	Set to the contents of the MSR at the time of the interrupt		
MSR	SPV 0 WE 0 CE— EE 0 PR 0	FP 0 ME— FE0 0 DE—	FE1 0 IS 0 DS 0 PMM 0 RI—
ESR	Access:	[ST], [SPV], VLEMI. All other bits cleared.	
MCSR	Unchanged		
DEAR	For Access Control exceptions, set to the effective address of the access which caused the violation.		
Vector	$IVPR_{0:23}$ || 0x20		

16.7.5.4 Instruction storage interrupt (Offset 0x30)

An Instruction Storage interrupt (ISI) occurs when no higher priority exception exists and an Execute Access Control exception occurs.

Table 243 lists register settings when an ISI is taken.

Table 243. Instruction Storage Interrupt—Register Settings

Register	Setting Description		
SRR0	Set to the effective address of the excepting instruction.		
SRR1	Set to the contents of the MSR at the time of the interrupt		
MSR	SPV 0 WE 0 CE— EE 0 PR 0	FP 0 ME— FE0 0 DE—	FE1 0 IS 0 DS 0 PMM 0 RI—
ESR	VLEMI. All other bits cleared.		
MCSR	Unchanged		
DEAR	Unchanged		
Vector	$IVPR_{0:23}$ || 0x30		

16.7.5.5 External input interrupt (Offset 0x40)

An External Input exception is signaled to the processor by the assertion of an interrupt from the interrupt controller. The input is a level-sensitive signal expected to remain asserted until core acknowledges the external interrupt. If the input is negated early, recognition of the interrupt request is not guaranteed. When the core detects the exception, if the exception is enabled by MSR_{EE}, it takes the External Input interrupt.

An External Input interrupt may be delayed by other higher priority exceptions or if MSR_{EE} is cleared when the exception occurs.

Table 244 lists register settings when an External Input interrupt is taken.

Table 244. External Input Interrupt—Register Settings

Register	Setting Description		
SRR0	Set to the effective address of the instruction that the processor would have attempted to execute next if no exception conditions were present.		
SRR1	Set to the contents of the MSR at the time of the interrupt		
MSR	SPV 0 WE 0 CE— EE 0 PR 0	FP 0 ME— FE0 0 DE—	FE1 0 IS 0 DS 0 PMM 0 RI—
ESR	Unchanged		
MCSR	Unchanged		
DEAR	Unchanged		
Vector	$IVPR_{0:23}$ || 0x40 (autovectored) $IVPR_{0:15}$ || ($IVPR_{16:29}$ | **p_voffset[0:13])** || 2'b00 (non-autovectored)		

16.7.5.6 Alignment interrupt (Offset 0x50)

An Alignment exception is generated when any of the following occurs:

- The operand of **lmw** or **stmw** not word aligned.
- The operand of **lwarx** or **stwcx.** not word aligned.
- The operand of **lharx** or **sthcx.** not halfword aligned.
- Execution of a **dcbz** instruction is attempted.
- Execution of a LSP APU load or store instruction which is not properly aligned.

Table 245 lists register settings when an alignment interrupt is taken.

Table 245. Alignment Interrupt—Register Settings

Register	Setting Description		
SRR0	Set to the effective address of the excepting load/store/dcbz instruction.		
SRR1	Set to the contents of the MSR at the time of the interrupt		
MSR	SPV 0 WE 0 CE — EE 0 PR 0	FP 0 ME— FE0 0 DE—	FE1 0 IS 0 DS 0 PMM 0 RI—
ESR	[ST], [SPV], VLEMI. All other bits cleared.		
MCSR	Unchanged		
DEAR	Set to the effective address of a byte of the load or store access causing the violation.		
Vector	$IVPR_{0:23}$ || 0x50		

16.7.5.7 Program interrupt (Offset 0x60)

A program interrupt occurs when no higher priority exception exists and one or more of the following exception conditions occur:

- Illegal Instruction exception
- Privileged Instruction exception
- Trap exception

core will invoke an Illegal Instruction program exception on attempted execution of the following instructions:

- Unimplemented instructions
- Instruction from the illegal instruction class
- **mtspr** and **mfspr** instructions with an undefined SPR specified
- **mtdcr** and **mfdcr** instructions with an undefined DCR specified

core will invoke a Privileged Instruction program exception on attempted execution of the following instructions when MSR_{PR}=1 (user mode):

- A privileged instruction
- **mtspr** and **mfspr** instructions which specify a SPRN value with $SPRN_5$=1 (even if the SPR is undefined).

core will invoke an Trap exception on execution of the **tw** instruction if the trap conditions are met and the exception is not also enabled as a Debug interrupt.

core will invoke an Illegal instruction program exception on attempted execution of the instructions **lswi**, **lswx**, **stswi**, **stswx**, **mfapidi**, **mfdcrx**, **mtdcrx**, or on any *PowerISA 2.06* floating-point instruction. All other defined or allocated instructions that are not implemented by core will cause a illegal instruction program exception.

Table 246 lists register settings when a Program interrupt is taken.

Table 246. Program Interrupt—Register Settings

Register	Setting Description		
SRR0	Set to the effective address of the excepting instruction.		
SRR1	Set to the contents of the MSR at the time of the interrupt		
MSR	SPV 0 WE 0 CE— EE 0 PR 0	FP 0 ME— FE0 0 DE—	FE1 0 IS 0 DS 0 PMM 0 RI—
ESR	Illegal, Unimplemented: Privileged: Trap:	PIL, VLEMI. All other bits cleared. PPR, VLEMI. All other bits cleared. PTR, VLEMI. All other bits cleared.	
MCSR	Unchanged		
DEAR	Unchanged		
Vector	$IVPR_{0:23}$ || 0x60		

16.7.5.8 System call interrupt (Offset 0x80)

A System Call interrupt occurs when a System Call (**se_sc**) instruction is executed and no higher priority exception exists.

Table 247 lists register settings when a System Call interrupt is taken.

Table 247. System Call Interrupt—Register Settings

Register	Setting Description		
SRR0	Set to the effective address of the instruction *following* the system call instruction.		
SRR1	Set to the contents of the MSR at the time of the interrupt		
MSR	SPV 0 WE 0 CE— EE 0 PR 0	FP 0 ME— FE0 0 DE —	FE1 0 IS 0 DS 0 PMM 0 RI—
ESR	VLEMI. All other bits cleared.		
MCSR	Unchanged		

Table 247. System Call Interrupt—Register Settings (continued)

Register	Setting Description
DEAR	Unchanged
Vector	$IVPR_{0:23}$ || 0x80

16.7.5.9 Debug interrupt (Offset 0x90)

There are multiple sources that can signal a Debug exception. A Debug interrupt occurs when no higher priority exception exists, a Debug exception exists in the Debug Status Register, and Debug interrupts are enabled (both $DBCR0_{IDM}$=1 (internal debug mode) and MSR_{DE}=1).

Table 248 lists register settings when a Debug interrupt is taken.

Table 248. Debug Interrupt—Register Settings

Register	Setting Description		
DSRR0(1)	Set to the effective address of the excepting instruction for IAC, BRT, RET, CRET, and TRAP. Set to the effective address of the next instruction to be executed *following* the excepting instruction for DAC (usually) and ICMP. For a UDE, IRPT, CIRPT, DCNT, or DEVT type exception, set to the effective address of the instruction that the processor would have attempted to execute next if no exception conditions were present.		
DSRR1	Set to the contents of the MSR at the time of the interrupt		
MSR	SPV 0 WE 0 CE—/0(2) EE—/0(2) PR 0	FP 0 ME— FE0 0 DE 0	FE1 0 IS 0 DS 0 PMM 0 RI—

Table 248. Debug Interrupt—Register Settings (continued)

Register	Setting Description	
DBSR(3)	Unconditional Debug Event: Instr. Complete Debug Event: Branch Taken Debug Event: Interrupt Taken Debug Event: Critical Interrupt Taken Debug Event: Trap Instruction Debug Event: Instruction Address Compare: Data Address Compare: Debug Notify Interrupt: Return Debug Event: Critical Return Debug Event: External Debug Event: Performance Monitor Debug Event: MPU Debug Event: and optionally, an Imprecise Debug Event flag	UDE ICMP BRT IRPT CIRPT TRAP {IAC} {DACR, DACW} DNI RET CRET {DEVT1, DEVT2} PMI MPU {IDE}
ESR	Unchanged	
MCSR	Unchanged	
DEAR	Unchanged	
Vector	$IVPR_{0:23}$ || 0x90	

1. assumes that the Debug interrupt is precise.
2. conditional based on control bits in HID0.
3. Note that multiple DBSR bits may be set.

16.7.5.10 System reset interrupt

The System Reset exception is a non-maskable, asynchronous exception signaled to the processor through the assertion of system-defined signals.

A System reset may be initiated by either a software reset or during power-on reset.

When a reset request occurs, the processor branches to the system reset exception vector without attempting to reach a recoverable state. If reset occurs during normal operation, all operations cease and the machine state is lost.

Table 249 lists register settings when a System Reset interrupt is taken.

Table 249. System Reset Interrupt—Register Settings

Register	Setting Description		
CSRR0	Undefined.		
CSRR1	Undefined.		
MSR	SPV 0 WE 0 CE 0 EE 0 PR 0	FP 0 ME 0 FE0 0 DE 0	FE1 0 IS 0 DS 0 PMM 0 RI 0
ESR	Cleared		
DEAR	Undefined		
Vector	[**p_rstbase[0:29]**] || 2'b00		

16.7.5.11 Embedded floating-point data interrupt (Offset 0xA0)

The Embedded Floating-point Data interrupt is taken if no higher priority exception exists and a EFPU Floating-point Data exception is generated. When a Floating-point Data exception occurs, the processor suppresses execution of the instruction causing the exception.

Table 250 lists register settings when a EFPU Floating-point Data interrupt is taken.

Table 250. Embedded Floating-point Data Interrupt—Register Settings

Register	Setting Description		
SRR0	Set to the effective address of the excepting EFPU instruction.		
SRR1	Set to the contents of the MSR at the time of the interrupt		
MSR	SPV 0 WE 0 CE— EE 0 PR 0	FP 0 ME— FE0 0 DE—	FE1 0 IS 0 DS 0 PMM 0 RI—
ESR	SPV, VLEMI. All other bits cleared.		
MCSR	Unchanged		
DEAR	Unchanged		
Vector	$IVPR_{0:23}$ || 0xA0		

16.7.5.12 Embedded floating-point round interrupt (Offset 0xB0)

The Embedded Floating-point Round interrupt is taken when a EFPU floating-point instruction generates an inexact result and inexact exceptions are enabled.

Table 251 lists register settings when a EFPU Floating-point Round interrupt is taken.

Table 251. Embedded Floating-point Round Interrupt—Register Settings

Register	Setting Description		
SRR0	Set to the effective address of the instruction following the excepting EFPU instruction.		
SRR1	Set to the contents of the MSR at the time of the interrupt		
MSR	SPV 0 WE 0 CE— EE 0 PR 0	FP 0 ME— FE0 0 DE—	FE1 0 IS 0 DS 0 PMM 0 RI—
ESR	SPV, VLEMI. All other bits cleared.		
MCSR	Unchanged		
DEAR	Unchanged		
Vector	$IVPR_{0:23}$ || 0xB0		

16.7.5.13 Performance monitor interrupt (Offset 0x70)

A performance monitor interrupt that may be generated by an enabled condition or event. An enabled condition or event is as follows:

A PMC*x* register overflow condition occurs with the following settings:

- PMLCax_{CE} = 1; that is, for the given counter the overflow condition is enabled.
- PMCx_{OV} = 1; that is, the given counter indicates an overflow.

For a performance monitor interrupt to be signaled on an enabled condition or event, $PMGC0_{PMIE}$ must be set.

Although an exception condition may occur with MSR_{EE} = 0, the interrupt cannot be taken until MSR_{EE} = 1.

The priority of the performance monitor interrupt is below all other asynchronous interrupts.

Table 252 lists register settings when an performance monitor interrupt is taken.

Table 252. Performance Monitor Interrupt—Register Settings

Register	Setting Description		
SRR0/ DSRR0(1)	Set to the effective address of the next instruction to be executed.		
SRR1/ DSRR1(1)	Set to the contents of the MSR at the time of the interrupt		
MSR	SPV 0 WE 0 CE—/0(2) EE 0/—(3) PR 0	FP 0 ME— FE0 0 DE—/0(4)	FE1 0 IS 0 DS 0 PMM 0 RI—
ESR	Unchanged		
MCSR	Unchanged		

Table 252. Performance Monitor Interrupt—Register Settings (continued)

Register	Setting Description
DEAR	Unchanged
Vector	$IVPR_{0:23}$ || 0x70

1. DSRR0/1 are used if $PMGC0_{UDI}$ = 1'.
2. CE is cleared if $PMGC0_{UDI}$ = 1 and $HID0_{DCLRCE}$ = 1.
3. EE is not cleared if $PMGC0_{UDI}$ = 1 and $HID0_{DCLREE}$ =0.
4. DE is cleared if $PMGC0_{UDI}$ = 1'.

16.8 MPU

16.8.1 MPU overview

The Core (z425n3) Memory Protection Unit (MPU) provides the capability of protecting regions of memory, with the following feature set:

- 24-entry region descriptor table with support for 6 arbitrary-sized instruction memory regions, 12 arbitrary-sized data memory regions, and 6 additional arbitrary-sized regions programmable as instruction or data memory regions
- Ability to set access permissions and memory attributes on a per-region basis
- Process ID aware, with per-bit masking of TID values
- Capability for masking upper address bits in the range comparison
- Capability of bypassing permissions checking for selected access types
- Per-entry write-once logic for entry protection
- Hardware flash invalidation support and per-entry invalidation protection controls
- Ability to optionally utilize region descriptors for generating debug events and watchpoints

Software managed by **mpure** and **mpuwe** instructions

16.8.2 Software interface and MPU instructions

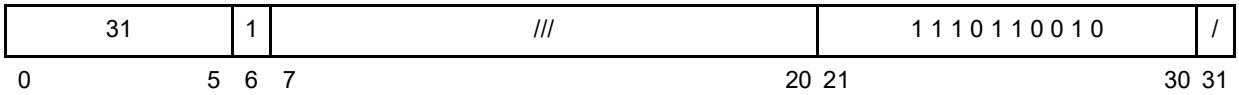
The MPU entries are accessed indirectly through several MPU Assist (MAS) registers. Software can write and read the MPU Assist registers with **mtspr** and **mfspr** instructions. These registers contain information related to reading and writing a given entry within the MPU. Data is read from the MPU into the MAS registers with a **mpure** (MPU read entry) instruction. Data is written to the MPU from the MAS registers with a **mpuwe** (MPU write entry) instruction.

The **mpure**, **mpuwe**, and **mpusync** instructions are privileged.

16.8.3 MPU read entry instruction (mpure)

The MPU read entry instruction causes the content of a single MPU entry to be placed in the MPU assist registers. The entry is specified by the INST, SHD, and ESEL fields of the MAS0 register. The entry contents are placed in the MAS0, MAS1, MAS2, and MAS3 registers.

mpure **mpure**
mpu readentry

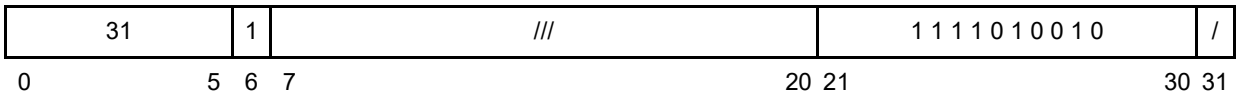



```
mpu_entry_id = MAS0(INST, SHD, ESEL)
result = MPU(mpu_entry_id)
MAS0, MAS1, MAS2, MAS3 = result
```

16.8.4 MPU Write Entry Instruction (mpuwe)

The MPU write entry instruction causes the contents of certain fields within the MPU assist registers MAS0, MAS1, MAS2, and MAS3 to be written into a single MPU entry in the MPU. The entry written is specified by the INST, SHD, and ESEL fields of the MAS0 register.

mpuwe **mpuwe**
mpu write entry



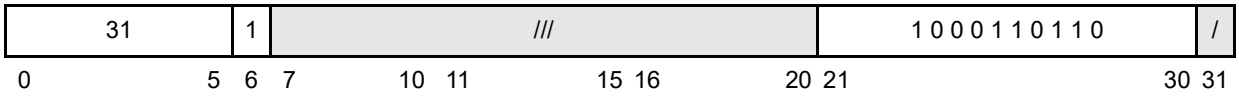

```
mpu_entry_id = MAS0(INST, SHD, ESEL)
MPU(mpu_entry_id) = MAS0, MAS1, MAS2, MAS3
```

16.8.5 MPU Synchronize Instruction (mpusync)

The MPU Synchronize instruction is treated as a privileged no-op by the Core (z425n3).

mpusync **mpusync**
MPU Synchronize

mpusync



16.8.6 MMU/MPU Configuration Register (MMUCFG)

The MMU/MPU Configuration Register (MMUCFG) is a 32-bit read-only register. The SPR number for MMUCFG is 1015 in decimal. MMUCFG provides information about the configuration of the Core (z425n3) MPU design. The MMUCFG register is shown in *Figure 149*.

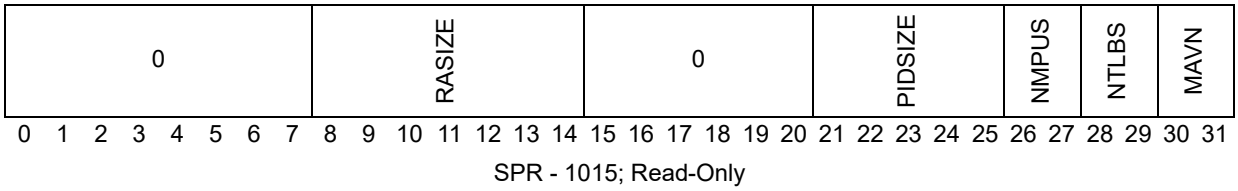


Figure 149. MMU/MPU Configuration Register (MMUCFG)

The MMUCFG bits are described in *Table 253*.

Table 253. MMUCFG Field Descriptions

Bits	Name	Function
0:7	—	Reserved
8:14	RASIZE	Number of Bits of Real Address supported 0100000- This version of the MPU implements 32 real address bits
15:16	—	Reserved
17:20	—	Reserved
21:25	PIDSIZE	PID Register Size 00111 - PID registers contain 8 bits in this version of the MPU
26:27	NMPUS	Number of MPUs 01 - This version of the MMU implements one MPU structure: a fully-associative MPU for MPU0
28:29	NTLBS	Number of TLBs 00 - This version of the MMU implements no TLB structures
30:31	MAVN	MMU Architecture Version Number 11 - This version of the MMU implements Version 3.1 of the MMU Architecture

16.8.7 MPU0 Configuration Register (MPU0CFG)

The MPU0 Configuration Register (MPU0CFG) is a 32-bit read-only register. The SPR number for MPU0CFG is 692 in decimal. MPU0CFG provides information about the configuration of MPU0 in the Core (z425n3) MPU. The MPU0CFG register is shown in *Figure 150*.

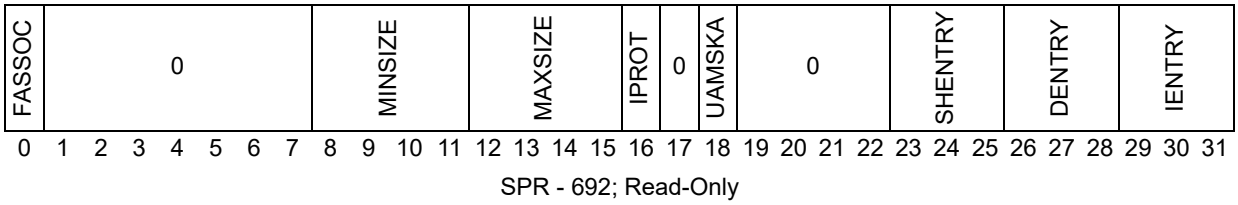


Figure 150. MPU0 Configuration Register (MPU0CFG)

The MPU0CFG bits are described in *Table 254*.

Table 254. MPU0CFG Field Descriptions

Bits	Name	Function
0	FASSOC	Fully Associative 1 - Indicates that MPU0 is fully associative
1:7	—	Reserved for non-fully associative implementation use
8:11	MINSIZE	Minimum Region Size 0x0 - Smallest region size is 1 byte **Note:** due to the use of access address matching, the effective smallest region size is 8 bytes
12:15	MAXSIZE	Maximum Region Size 0xB - Largest region size is 4 GB
16	IPROT	Invalidate Protect Capability 1 - Invalidate Protect Capability is supported in MPU0
17	—	Reserved
18	UAMSKA	Upper Address Masking Availability 1 - Upper Address Masking is Available
19:22	—	Reserved
23:25	SHENTRY	Number of Shared (configurable for I or D) Entries 0x2 - Indicates that MPU0 contains 6 shared entries
26:28	DENTRY	Number of Data Entries 0x4 - Indicates that MPU0 contains 12 dedicated data entries
29:31	IENTRY	Number of Instruction Entries 0x2 - Indicates that MPU0 contains 6 dedicated Instruction entries

16.8.8 MPU0 Control and Status Register 0 (MPU0CSR0)

The MPU0 Control and Status Register 0 (MPU0CSR0) is a 32-bit register. The SPR number for MPU0CSR0 is 1014 in decimal. MPU0CSR0 controls the operation of the MPU. The MPU0CSR0 register is shown in *Figure 151*.



SPR - 1014; Read/ Write; Reset - 0x0

Figure 151. MPU0 Control and Status Register 0 (MPU0CSR0)

The MPU0CSR0 bits are described in *Table 255*.

Table 255. MPU0CSR0 - MPU0 Control and Status Register 0

Bits	Name	Description
0:15	—	Reserved
16	BYPSR	Bypass MPU protections for Supervisor Read accesses 0 - No Bypass of MPU protections for Supervisor Read accesses 1 - Bypass MPU protections for Supervisor Read accesses This bit controls whether supervisor-mode read accesses bypass the MPU protection mechanism. When bypass is enabled, supervisor-mode read accesses do not generate a DSI when no MPU match occurs. Supervisor-mode read accesses are still compared to MPU entries, and a hit will provide access attributes (CI, G).
17	BYPSW	Bypass MPU protections for Supervisor Write accesses 0 - No Bypass of MPU protections for Supervisor Write accesses 1 - Bypass MPU protections for Supervisor Write accesses This bit controls whether supervisor-mode write accesses bypass the MPU protection mechanism. When bypass is enabled, supervisor-mode write accesses do not generate a DSI when no MPU match occurs. Supervisor-mode write accesses are still compared to MPU entries, and a hit will provide access attributes (CI, G).
18	BYPSX	Bypass MPU protections for Supervisor Instruction accesses 0 - No Bypass of MPU protections for Supervisor Instruction accesses 1 - Bypass MPU protections for Supervisor Instruction accesses This bit controls whether supervisor-mode instruction accesses bypass the MPU protection mechanism. When bypass is enabled, supervisor-mode instruction accesses do not generate an ISI when no MPU match occurs. Supervisor-mode instruction accesses are still compared to MPU entries, and a hit will provide access attributes (CI).
19	BYPUR	Bypass MPU protections for User Read accesses 0 - No Bypass of MPU protections for User Read accesses 1 - Bypass MPU protections for User Read accesses This bit controls whether user-mode read accesses bypass the MPU protection mechanism. When bypass is enabled, user-mode read accesses do not generate a DSI when no MPU match occurs. User-mode read accesses are still compared to MPU entries, and a hit will provide access attributes (CI, G).
20	BYPUW	Bypass MPU protections for User Write accesses 0 - No Bypass of MPU protections for User Write accesses 1 - Bypass MPU protections for User Write accesses This bit controls whether user-mode write accesses bypass the MPU protection mechanism. When bypass is enabled, user-mode write accesses do not generate a DSI when no MPU match occurs. User-mode write accesses are still compared to MPU entries, and a hit will provide access attributes (CI, G).
21	BYPUX	Bypass MPU protections for User Instruction accesses 0 - No Bypass of MPU protections for User Instruction accesses 1 - Bypass MPU protections for User Instruction accesses This bit controls whether user-mode instruction accesses bypass the MPU protection mechanism. When bypass is enabled, user-mode instruction accesses do not generate an ISI when no MPU match occurs. User-mode instruction accesses are still compared to MPU entries, and a hit will provide access attributes (CI).
22:23	—	Reserved

Table 255. MPU0CSR0 - MPU0 Control and Status Register 0 (continued)

Bits	Name	Description
24	DRDEN	Data Read Debug Enable 0 - MPU debug events are disabled for data read accesses 1 - MPU debug events are enabled for data read accesses This bit controls whether data read accesses are enabled to generate MPU debug events. When disabled, no data read access will generate an MPU debug event, regardless of whether an access match occurs to a valid MPU region descriptor with DEBUG=1. If such a match occurs, no MPU debug event is generated, and no access protection is performed. When enabled, data read accesses are not blocked from generating MPU debug events.
25	DWDEN	Data Write Debug Enable 0 - MPU debug events are disabled for data write accesses 1 - MPU debug events are enabled for data write accesses This bit controls whether data write accesses are enabled to generate MPU debug events. When disabled, no data write access will generate an MPU debug event, regardless of whether an access match occurs to a valid MPU region descriptor with DEBUG=1. If such a match occurs, no MPU debug event is generated, and no access protection is performed. When enabled, data write accesses are not blocked from generating MPU debug events.
26	IDEN	Instruction Debug Enable 0 - MPU debug events are disabled for instruction accesses 1 - MPU debug events are enabled for instruction accesses This bit controls whether instruction accesses are enabled to generate MPU debug events. When disabled, no instruction access will generate an MPU debug event, regardless of whether an access match occurs to a valid MPU region descriptor with DEBUG=1. When enabled, instruction accesses are not blocked from generating MPU debug events.
27	—	Reserved
28	TIDCTL	TID usage Control 0 - TID comparisons performed normally 1 - no TID comparisons are performed for matching while in Supervisor mode When TIDCTL is set to ‘1’, the TID match is disabled (forced match) in Supervisor mode.
29	—	Reserved

Table 255. MPU0CSR0 - MPU0 Control and Status Register 0 (continued)

Bits	Name	Description
30	MPUFI	MPU flash invalidate 0 - No flash invalidate 1 - MPU1 invalidation operation When written to a '1', a MPU invalidation operation is initiated by hardware. Once complete, this bit is reset to '0'. Writing a '1' while an invalidation operation is in progress will result in an undefined operation. Writing a '0' to this bit while an invalidation operation is in progress will be ignored. MPU invalidation operations require 3 cycles to complete. Note that entries which have the IPROT bit set will not be invalidated.
31	MPUEN	MPU Enable 0 - MPU is disabled 1 - MPU is enabled This bit enables operation of the MPU. When enabled, access addresses are compared to each entry in the MPU for a match condition. If no match condition occurs, and the access type is not enabled to bypass the MPU protections, then an ISI or DSI condition is signaled for the access. Note that this bit is ignored for matching entries with DEBUG=1 if in EDM and hardware owns MPU Debug entries.

16.8.9 MPU Assist registers (MAS)

The Core (z425n3) uses four special purpose registers (MAS0, MAS1, MAS2, and MAS3) to facilitate reading and writing MPU entries. The MAS registers can be read or written using the **mfspr** and **mtspr** instructions.

The MAS0 register is shown in *Figure 152*. Fields are defined in *Table 256*.

VALID	IPROT	SEL	0	RO	DEBUG	INST	SHD	0	ESEL	0	UAMSK	UW	SW	UX/UR	SX/SR	IOVR	GOVR(1)	1	1	I	0	G(1)	0
0	1	2 3	4	5	6	7	8	9 10 11	12 13 14 15	16	17 18 19	20	21	22	23	24	25	26	27	28	29	30	31

SPR - 624; Read/ Write; Reset - Unaffected

Figure 152. MPU Assist Register 0 (MAS0)

1. This bit is not implemented in dedicated instruction entries, and is ignore in shared entries with INST=1

Table 256. MAS0 —MPU Read/Write and Replacement Control

Bit	Name	Comments, or Function when Set
0	VALID	MPU Entry Valid 0 - This MPU entry is invalid 1 - This MPU entry is valid
1	IPROT	Invalidation Protect 0 - Entry is not protected from invalidation 1 - Entry is protected from invalidation

Table 256. MAS0 —MPU Read/Write and Replacement Control (continued)

Bit	Name	Comments, or Function when Set
2:3	SEL	Selects MPU for access: 00 - Reserved, no access 01 - Reserved, no access 10 - Select MPU 11 - Reserved, no access
4	—	Reserved
5	RO	Read-Only 0 - This MPU entry is writable 1 - This MPU entry is Read-only When set to ‘1’, the entry is no longer writable by software. Once set, this bit will remain set until a processor reset occurs.
6	DEBUG	Debug Control for Entry 0 - This MPU entry is used for access protections and allows accesses when matched based on protection attributes 1 - This MPU entry is used for generating a debug event when a match occurs and the access protections would allow access This bit is used to assign an entry for debug event use instead of normal access protection use. When used for debug purposes, an MPU debug event is generated when a hit to the entry occurs and the access permissions allow the access.
7	INST	Instruction Entry 0 - This MPU entry is used for matching data accesses only 1 - This MPU entry is used for matching instruction accesses only When SHD=0, this bit is used to select the INST entry portion of the region descriptor table for **mpure** and **mpuwe** instruction operations. When SHD=1, this bit is used to indicate whether the entry in the Shared portion of the region descriptor table is assigned as an entry for either instruction access or data access matching. Only one of these access types is monitored for matching by a given shared region descriptor entry.
8	SHD	Shared Entry Select 0 - The shared portion of the region descriptor table is not accessed on a **mpure** or **mpuwe** operation. Either the instruction portion or the data portion is accessed based on the setting of INST. The entry within the selected portion is based on ESEL. 1 - The shared portion of the region descriptor table is accessed on a **mpure** or **mpuwe** operation. The instruction and data portions are not accessed. The INST bit is used to indicate whether the entry in the Shared portion of the region descriptor table is assigned as an entry for instruction access or data access matching. Only one of these access types is monitored for matching by a given shared region descriptor entry. ESEL defines which shared entry is selected. This bit is used to control selection of the Shared portion of the region descriptor table.
9:11	—	Reserved
12:15	ESEL	Entry select for MPU. This field is used to select an entry for reading or writing in conjunction with the settings of SHD and INST. Only valid entry numbers should be used in this field, otherwise the result of an operation is boundedly undefined.

Table 256. MAS0 —MPU Read/Write and Replacement Control (continued)

Bit	Name	Comments, or Function when Set
16	—	Reserved
17:19	UAMSK	Upper Address Mask Control 000 - No upper address bits are masked, address matching uses all 32 address bits for accesses 001 - The most significant access address bit is forced to zero before matching 010 - The two most significant access address bits are forced to zero before matching. 101 - The upper 5 access address bits are forced to zero before matching 11x - Reserved, do not use This field is used to support masking of upper address bits before performing range comparisons. Masked bits will be forced to 0 for the purposes of the range compare. Range upper and lower bounds should be set accordingly.
20	UW	User Mode Write Permission 0 - No User mode Write permission 1 - User mode has Write permission Determines User mode Write (W) permission when INST=0. Ignored when INST=1.
21	SW	Supervisor Mode Write / Read Permission 0 - No Supervisor mode Write permission 1 - Supervisor mode has Write permission Determines Supervisor mode Write (W) permission when INST=0. Ignored when INST=1.
22	UX/UR	User Mode Execute / Read Permission 0 - No User mode Execute/ Read permission 1 - User mode has Execute/ Read permission Determines User mode Execute (X) permission when INST=1, or User mode Read (R) permission when INST=0.
23	SX/SR	Supervisor Mode Execute / Read Permission 0 - No Supervisor mode Execute/ Read permission 1 - Supervisor mode has Execute/ Read permission Determines Supervisor mode Execute (X) permission when INST=1, or Supervisor mode Read (R) permission when INST=0.
24	IOVR	Cache-Inhibit attribute Override 0 - No override of I attribute by entry 1 - Entry I bit overrides I attribute Determines whether this matching entry overrides the I bit settings of other matching entries and the Cache-Inhibited region configuration signals
25	GOVR/—	G attribute Override 0 - No override of G attribute by entry 1 - Entry G bit overrides G attribute Determines whether this entry overrides the G bit settings of other matching entries and the Guarded region configuration signals This bit is not implemented in dedicated instruction entries, and is ignore in shared entries with INST=1.
26	—	Reserved
27	—	Reserved

Table 256. MAS0 —MPU Read/Write and Replacement Control (continued)

Bit	Name	Comments, or Function when Set
28	I	Cache Inhibited 0 - This region is considered cacheable 1 - This region is considered cache-inhibited This attribute may be overridden by the cache-inhibited region configuration input settings.
29	—	Reserved
30	G/—	Guarded 0 - Accesses to this region are not guarded, and can be performed before it is known if they are required by the sequential execution model 1 - All loads and stores to this region are performed without speculation (that is they are known to be required) This attribute may be overridden by the guarded region configuration input settings. This bit is not implemented in dedicated instruction entries, and is ignore in shared entries with INST=1.
31	—	Reserved

The MAS1 register is shown in *Figure 153*. Fields are defined in *Table 257*.

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

SPR - 625; Read/ Write; Reset - Unaffected

Figure 153. MPU Assist Register 1 (MAS1)

Table 257. MAS1 —Descriptor Context and Configuration Control

Bit	Name	Comments, or Function when Set
0:7	—	Reserved
8:15	TID	Region ID bits This field is compared with the current process ID of the access address. A TID value of 0 defines an entry as global and matches with all process IDs.

Table 257. MAS1 —Descriptor Context and Configuration Control (continued)

Bit	Name	Comments, or Function when Set
16:23	—	Reserved
24:31	TIDMSK	Region ID mask 0xxxxxxx - TID[0] comparison to PID0[0] not masked 1xxxxxxx - TID[0] comparison to PID0[0] is masked x0xxxxxx - TID[1] comparison to PID0[1] not masked x1xxxxxx - TID[1] comparison to PID0[1] is masked xx0xxxxx - TID[2] comparison to PID0[2] not masked xx1xxxxx - TID[2] comparison to PID0[2] is masked xxx0xxxx - TID[3] comparison to PID0[3] not masked xxx1xxxx - TID[3] comparison to PID0[3] is masked xxxx0xxx - TID[4] comparison to PID0[4] not masked xxxx1xxx - TID[4] comparison to PID0[4] is masked xxxxx0xx - TID[5] comparison to PID0[5] not masked xxxxx1xx - TID[5] comparison to PID0[5] is masked xxxxxx0x - TID[6] comparison to PID0[6] not masked xxxxxx1x - TID[6] comparison to PID0[6] is masked xxxxxxx0 - TID[7] comparison to PID0[7] not masked xxxxxxx1 - TID[7] comparison to PID0[7] is masked This field controls whether bits within the TID to PID[0] comparison are masked for determining a range match. When a bit is masked, the value of that TID and PID0 bit is ignored for matching purposes.

The MAS2 register is shown in *Figure 154*. Fields are defined in *Table 258*.

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

SPR - 626; Read/ Write; Reset - Unaffected

Figure 154. MPU Assist Register 2 (MAS2)

Table 258. MAS2 - Upper Bound Control

Bit	Name	Comments, or Function when Set
0:31	UPPER BOUND	Upper bound of address range covered by entry. Entry match requires LOWER_BOUND <= EA <= UPPER_BOUND

The MAS3 register is shown in *Figure 155*. Fields are defined in *Table 259*.

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

SPR - 627; Read/ Write; Reset - Unaffected

Figure 155. MPU Assist Register 3 (MAS3)

Table 259. MAS3 - Lower Bound Control

Bit	Name	Comments, or Function when Set
0:31	LOWER BOUND	Lower bound of address range covered by entry. Entry match requires LOWER_BOUND <= EA <= UPPER_BOUND

16.8.10 MAS registers summary

The MAS registers are summarized in *Figure 156*.

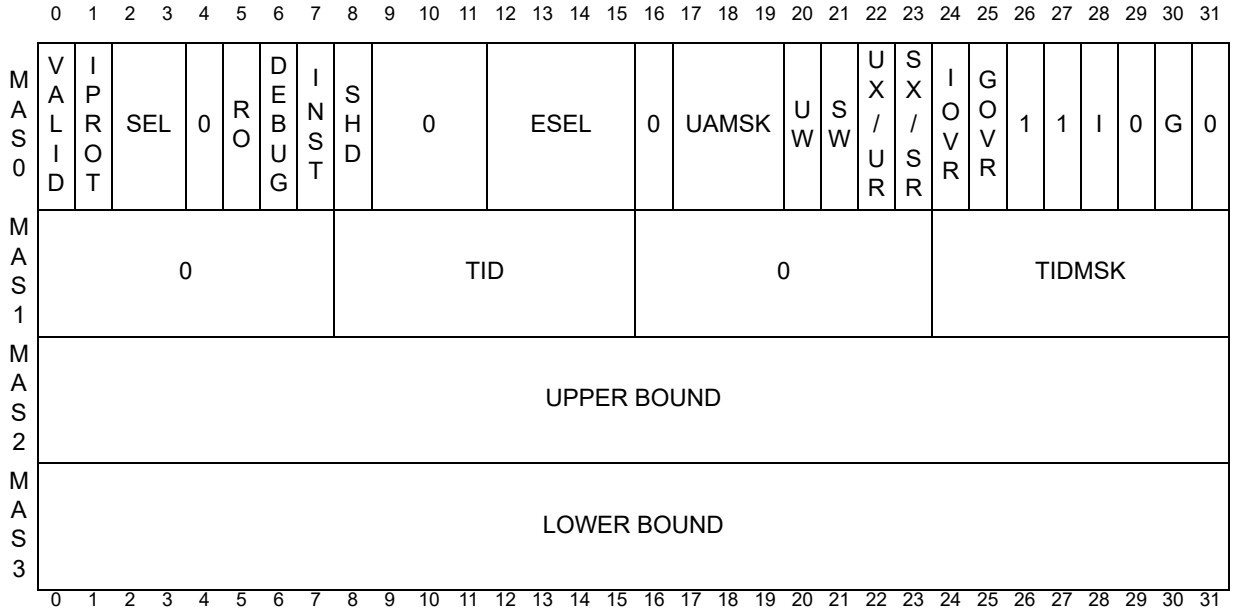


Figure 156. MPU Assist Registers Summary

16.9 Local memories

16.9.1 Local instruction and data memory overview

In order to provide for low latency memory access for critical instruction routines and data operands, local instruction (IMEM) and data (DMEM) memory capabilities have been added to the next generation processor cores. These memories provide for low latency access (zero wait-state) similar to an instruction or data cache, but do not suffer from the overhead of cache miss or cache Writethrough operations. Instead, they provide “tightly coupled” storage.

16.9.2 Local memory control and configuration

DMEM and IMEM local memory configuration is provided by a set of privileged read-only SPRs which are accessed using **mfspr** instructions. DMEM and IMEM local memory is controlled by a set of privileged device control registers (DCRs) which are accessed using the **mfdcr** and **mtdcr** instructions. These registers and a description of the operation of various features are described in the following subsections.

16.9.2.1 DMEM Configuration Register 0 (DMEMCFG0)

The DMEM Configuration Register 0 (DMEMCFG0) is a 32-bit read-only register. DMEMCFG0 provides information about the configuration of the Core (z425n3) local data memory design. The contents of the DMEMCFG0 register can be read using a **mfspr** instruction. The SPR number for DMEMCFG0 is 694 in decimal. The DMEMCFG0 register is shown in *Figure 157*.

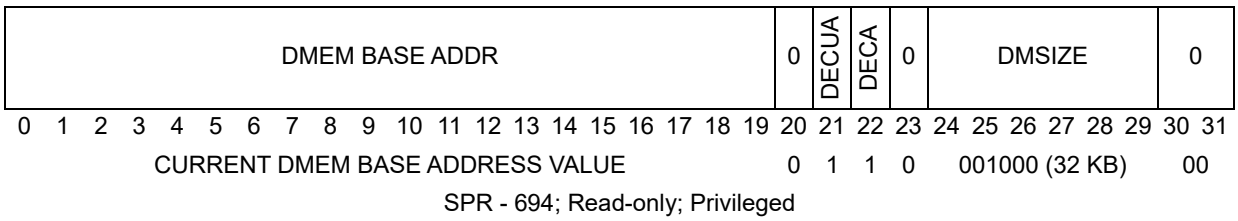


Figure 157. DMEM Configuration Register 0 (DMEMCFG0)

The DMEMCFG0 fields are described in *Table 260*.

Table 260. DMEMCFG0 Field Descriptions

Bits	Name	Description
0:19	DMEM BASE ADDR	DMEM BASE ADDRESS (CPU Port) This field defines the current Base Address value being used for the CPU port to the DMEM. This field reflects the value of the external DMEM base address port inputs when the external base address port inputs are enabled via the DBAPD control bit, or the value of the BASE ADDRESS field of DMEMCFG0 when the external base address port inputs are disabled via the DBAPD control bit. **Note:** low order bits of this field are driven to 0's when the DMEM size is > 4 Kbyte
20	-	Reserved
21	DECUA	DECUE - DMEM Error Correction Update Available 0 - Error Correction Update is not available 1 - Error Correction Update is available for correction of a correctable single-bit error detected on a read access DECUA indicates an error scrubbing function for the DMEM is available.
22	DECA	DECUE - DMEM Error Correction Available 0 - Error Correction is not available 1 - Error Correction is available DECA indicates an error correction function for the DMEM is available.
23	-	Reserved
24:29	DMSIZE	DMSIZE - DMEM Size 001000 - The size of the DMEM is 32 Kbyte
30:31	-	Reserved

16.9.2.2 DMEM Control Register 0 (DMEMCTL0)

The DMEM Control Register 0 (DMEMCTL0) controls operation of certain functions of the DMEM logic.

DMEM BASE ADDR	0	DBYPCB	DBYPAT	DBYPDEC	DECUE	0	DBAPD	DPEIE	DICWE	DSWCE	DDAUEC	DCECE	DSECE
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16	17 18	19	20	21	22	23	24	25	26	27 28	29	30	31

DCR - 496; Read/Write; Reset(1) - 19‘b0 || p_dmem_rstcfg[8,6:7,0] || 4‘b0 || p_dmem_rstcfg[1:5]; Privileged

Figure 158. DMEM Control Register 0 (DMEMCTL0)

1. reset by **m_por** and **slv_reset_b**. Unaffected by **p_reset_b**.

Table 261. DMEMCTL0 Field Descriptions

Bits	Name	Description
0:16	DMEM BASE ADDR	DMEM BASE ADDRESS Field (CPU Port) This field defines the Base Address used for the CPU port to the DMEM when the external base address port inputs are disabled via the DBAPD control bit.
17:18	-	Reserved
19	DBYPCB	DBYPCB - DMEM Bypass Cache Bypass CPU accesses 0 - Non-decorated Cache Bypass loads and Store with Writethrough CPU accesses do not bypass the DMEM CPU port 1 - Non-decorated Cache Bypass loads and Store with Writethrough CPU accesses bypass the DMEM CPU port and are routed out through the BIU to the DAHB external interface. DBYPCB can be used to force Non-decorated Cache Bypass loads and Store with Writethrough (**lbcbx**, **lhcbx**, **lwcbx**, and **stwwtx**, **sthwtx**, **stbwtx**) accesses which target the DMEM address space to be presented to the external bus. Note that the protection settings in DMEMCTL1 are still applied to these accesses. **Note:** this field is updated on **m_por** and **slv_reset_b** based on the value of the **p_dmem_rstcfg[8]** input.
20	DBYPAT	DBYPAT - DMEM Bypass Atomic CPU accesses 0 - Atomic CPU accesses do not bypass the DMEM CPU port 1 - Atomic CPU accesses bypass the DMEM CPU port and are routed out through the BIU to the DAHB external interface. DBYPAT can be used to force atomic (load and reserve, store conditional) accesses to the DMEM address space to be presented to external reservation hardware in systems which support reservation or decoration functionality (or both), since no internal reservation hardware is present in the CPU for the DMEM. In general, software must not rely on normal CPU write accesses to clear a reservation, since they are not monitored by reservation hardware; instead only store conditional accesses should be used for this purpose. Note that the protection settings in DMEMCTL1 are still applied to these accesses. **Note:** this field is updated on **m_por** and **slv_reset_b** based on the value of the **p_dmem_rstcfg[6]** input.

Table 261. DMEMCTL0 Field Descriptions (continued)

Bits	Name	Description
21	DBYPDEC	DBYPDEC - DMEM Bypass Decorated CPU accesses 0 - Decorated Load and Store CPU accesses do not bypass the DMEM CPU port 1 - Decorated Load and Store CPU accesses bypass the DMEM CPU port and are routed out through the BIU to the DAHB external interface. DBYPDEC can be used to force decorated accesses (**lbdx, lhdx, lwdx, lbdcbx, lhdcbx, lwdcbx, stbdx, sthdx, stwdx, dsn, dsncb**) to the DMEM address space to be presented to external decoration hardware for systems which support this functionality, since no internal decoration hardware is present in the CPU. Note that the protection settings in DMEMCTL1 are still applied to these accesses. **Note:** this field is updated on **m_por** and **slv_reset_b** based on the value of the **p_dmem_rstcfg[7]** input.
22	DECUE	DECUE - DMEM Error Correction Update Enable 0 - Error Correction Update is disabled 1 - Error Correction Update is enabled. A correctable single-bit error detected on a read access from either the CPU or the slave port will cause the DMEM to be re-written with the corrected data. (Write accesses perform this correction automatically during partial-width writes.) DECUE can be used to provide an error scrubbing function for the DMEM. **Note:** this field is updated on **m_por** and **slv_reset_b** based on the value of the **p_dmem_rstcfg[0]** input.
23	-	Reserved
24	DBAPD	DBAPD - DMEM Base Address Port Disable 0 - The external DMEM base address port (**p_dmem_baseaddr[0:n]**) is used to define the base address of the DMEM for CPU accesses. 1 - The external DMEM base address port (**p_dmem_baseaddr[0:n]**) is disabled for use, and instead the BASE ADDR field value is used to define the base address of the DMEM for CPU accesses. This bit controls usage of the external Base Address port (**p_dmem_baseaddr[0:n]**) to define the base address of the DMEM for CPU port accesses. It has no effect on DMEM slave port accesses.
25	DPEIE	DPEIE - DMEM Processor Error Injection Enable 0 - Error Injection is disabled 1 - A double-bit error will be injected on each CPU port write into the DMEM data array by inverting the two uppermost parity check bits (p_dchk[0:1]). DPEIE will cause injection of errors regardless of the setting of DCPECE, although reporting of errors to the CPU will be masked while DCPECE=0.
26	DICWE	DICWE - DMEM Imprecise CPU Write Enable 0 - Imprecise writes by the CPU are disabled. All partial-width write ECC errors are reported precisely (error report machine check). 1 - Imprecise writes by the CPU are enabled. In certain cases, partial-width write errors may be reported imprecisely (async machine check only). DICWE may allow for increased partial-width write performance when set.

Table 261. DMEMCTL0 Field Descriptions (continued)

Bits	Name	Description
27:28	DSWCE	DSWCE - DMEM Slave port Write Check/Correct Enable 00 - Slave write data is not checked or corrected for errors. 01 - Slave write data is checked but not corrected for errors. Detected errors generate a slave port ERROR response and no Write is performed to the DMEM. 10 - Slave write data is checked and corrected for errors on all writes. Single-bit correctable errors do not automatically generate an ERROR response, but are instead corrected. 11 - Slave write data is checked and corrected for errors on partial-width (1, 2, 3, 5, 6, or 7 byte, or misaligned 4-byte writes) writes. Single-bit correctable errors on partial-width writes do not automatically generate an ERROR response, but are instead corrected. Aligned word or double word writes are not checked or corrected for errors. **Note:** this field is updated on **m_por** and **slv_reset_b** based on the values of the **p_dmem_rstcfg[1:2]** inputs.
29	DDAUEC	DDAUEC - DMEM Disable Address Use in Error Check 0 - Use of access address is enabled in checkbit and syndrome generation. 1 - Use of access address is disabled in checkbit and syndrome generation. This bit controls usage of the address portion of the ECC H matrix. When use is disabled, the address portions are not used for checkbit/syndrome generation for DMEM CPU or slave port accesses. **Note:** this field is updated on **m_por** and **slv_reset_b** based on the value of the **p_dmem_rstcfg[3]** input.
30	DCECE	DCPECE - DMEM CPU Port ECC Enable (CPU port) 0 - End-to-End ECC is disabled for the CPU interface. No checking of read data is performed by the CPU. Writes will still generate the proper check bit values to be written. 1- End-to-End ECC is enabled for performing ECC on the CPU interface. Error checking of read data is performed with single-bit error correction. Detection of uncorrectable single- or multi-bit errors on a CPU port access cause a machine check to be generated. **Note:** this field is updated on **m_por** and **slv_reset_b** based on the value of the **p_dmem_rstcfg[4]** input.
31	DSECE	DSECE - DMEM Slave port Error Checking Enable (Slave port) 0 - End-to-End ECC checking logic is disabled for the Slave interface. No checking of read data is performed during read-modify-write operations for Slave port partial-width writes. Writes will still generate the proper check bit values to be written. 1 - End-to-End ECC is enabled for performing ECC on the Slave Port interface. Error checking of read data is performed with single-bit error correction during read-modify-write operations for partial-width writes. Detection of uncorrectable single- or multi-bit errors on a Slave port partial-width write access causes a bus error ERROR response to be generated. **Note:** this field is updated on **m_por** and **slv_reset_b** based on the value of the **p_dmem_rstcfg[5]** input.

16.9.2.3 DMEM Control Register 1 (DMEMCTL1)

The DMEM Control Register 1 (DMEMCTL1) provides protection functions for the DMEM. Separate Supervisor-mode and User-mode read and write access controls allow accesses to be granted, denied, or conditionally granted independently for four equally-sized blocks of DMEM. Conditional granting of access permissions causes the MPU memory protection functions to be employed, all other settings override MPU settings. The DMEMCTL1 settings are applied to all accesses which target DMEM address range, regardless of

whether the access may bypass the DMEM based on settings in DMEMCTL0$_{DBYPCB,}$ DBYPATF, DBYPDEC·



DCR - 498; Read/Write; Reset(1) - 32‘b0; Privileged

Figure 159. DMEM Control Register 1 (DMEMCTL1)

1. reset by **m_por** and **p_reset_b**.

Table 262. DMEMCTL1 Field Descriptions

Bits	Name	Description
0:1	DSWAC3	DSWAC3 - DMEM Supervisor Write Access Control for Quadrant 3 00 - Supervisor-mode CPU write accesses are allowed to quadrant 3 of DMEM 01 - Supervisor-mode CPU write accesses are not allowed to quadrant 3 of DMEM 10 - Supervisor-mode CPU write accesses are conditionally allowed to quadrant 3 of DMEM based on memory protection logic 11 - Reserved **Note:** individual control fields are provided for four equal sized sections of DMEM regardless of DMEM size. A single control field is used for each access based on the upper two address bits used to index into the DMEM. DSWAC3 is used when these two bits = 2‘b11
2:3	DSWAC2	DSWAC2 - DMEM Supervisor Write Access Control for Quadrant 2 00 - Supervisor-mode CPU write accesses are allowed to quadrant 2 of DMEM 01 - Supervisor-mode CPU write accesses are not allowed to quadrant 2 of DMEM 10 - Supervisor-mode CPU write accesses are conditionally allowed to quadrant 2 of DMEM based on memory protection logic 11 - Reserved **Note:** individual control fields are provided for four equal sized sections of DMEM regardless of DMEM size. A single control field is used for each access based on the upper two address bits used to index into the DMEM. DSWAC2 is used when these two bits = 2‘b10
4:5	DSWAC1	DSWAC1 - DMEM Supervisor Write Access Control for Quadrant 1 00 - Supervisor-mode CPU write accesses are allowed to quadrant 1 of DMEM 01 - Supervisor-mode CPU write accesses are not allowed to quadrant 1 of DMEM 10 - Supervisor-mode CPU write accesses are conditionally allowed to quadrant 1 of DMEM based on memory protection logic 11 - Reserved **Note:** individual control fields are provided for four equal sized sections of DMEM regardless of DMEM size. A single control field is used for each access based on the upper two address bits used to index into the DMEM. DSWAC1 is used when these two bits = 2‘b01

Table 262. DMEMCTL1 Field Descriptions (continued)

Bits	Name	Description
6:7	DSWAC0	DSWAC0 - DMEM Supervisor Write Access Control for Quadrant 0 00 - Supervisor-mode CPU write accesses are allowed to quadrant 0 of DMEM 01 - Supervisor-mode CPU write accesses are not allowed to quadrant 0 of DMEM 10 - Supervisor-mode CPU write accesses are conditionally allowed to quadrant 0 of DMEM based on memory protection logic 11 - Reserved **Note:** individual control fields are provided for four equal sized sections of DMEM regardless of DMEM size. A single control field is used for each access based on the upper two address bits used to index into the DMEM. DSWAC0 is used when these two bits = 2‘b00
8:9	DSRAC3	DSRAC3 - DMEM Supervisor Read Access Control for Quadrant 3 00 - Supervisor-mode CPU read accesses are allowed to quadrant 3 of DMEM 01 - Supervisor-mode CPU read accesses are not allowed to quadrant 3 of DMEM 10 - Supervisor-mode CPU read accesses are conditionally allowed to quadrant 3 of DMEM based on memory protection logic 11 - Reserved **Note:** individual control fields are provided for four equal sized sections of DMEM regardless of DMEM size. A single control field is used for each access based on the upper two address bits used to index into the DMEM. DSRAC3 is used when these two bits = 2‘b11
10:11	DSRAC2	DSRAC2 - DMEM Supervisor Read Access Control for Quadrant 2 00 - Supervisor-mode CPU read accesses are allowed to quadrant 2 of DMEM 01 - Supervisor-mode CPU read accesses are not allowed to quadrant 2 of DMEM 10 - Supervisor-mode CPU read accesses are conditionally allowed to quadrant 2 of DMEM based on memory protection logic 11 - Reserved **Note:** individual control fields are provided for four equal sized sections of DMEM regardless of DMEM size. A single control field is used for each access based on the upper two address bits used to index into the DMEM. DSRAC2 is used when these two bits = 2‘b10
12:13	DSRAC1	DSRAC1 - DMEM Supervisor Read Access Control for Quadrant 1 00 - Supervisor-mode CPU read accesses are allowed to quadrant 1 of DMEM 01 - Supervisor-mode CPU read accesses are not allowed to quadrant 1 of DMEM 10 - Supervisor-mode CPU read accesses are conditionally allowed to quadrant 1 of DMEM based on memory protection logic 11 - Reserved **Note:** individual control fields are provided for four equal sized sections of DMEM regardless of DMEM size. A single control field is used for each access based on the upper two address bits used to index into the DMEM. DSRAC1 is used when these two bits = 2‘b01
14:15	DSRAC0	DSRAC0 - DMEM Supervisor Read Access Control for Quadrant 0 00 - Supervisor-mode CPU read accesses are allowed to quadrant 0 of DMEM 01 - Supervisor-mode CPU read accesses are not allowed to quadrant 0 of DMEM 10 - Supervisor-mode CPU read accesses are conditionally allowed to quadrant 0 of DMEM based on memory protection logic 11 - Reserved **Note:** individual control fields are provided for four equal sized sections of DMEM regardless of DMEM size. A single control field is used for each access based on the upper two address bits used to index into the DMEM. DSRAC0 is used when these two bits = 2‘b00

Table 262. DMEMCTL1 Field Descriptions (continued)

Bits	Name	Description
16:17	DUWAC3	DUWAC3 - DMEM User Write Access Control for Quadrant 3 00 - User-mode CPU write accesses are allowed to quadrant 3 of DMEM 01 - User-mode CPU write accesses are not allowed to quadrant 3 of DMEM 10 - User-mode CPU write accesses are conditionally allowed to quadrant 3 of DMEM based on memory protection logic 11 - Reserved **Note:** individual control fields are provided for four equal sized sections of DMEM regardless of DMEM size. A single control field is used for each access based on the upper two address bits used to index into the DMEM. DUWAC3 is used when these two bits = 2‘b11
18:19	DUWAC2	DUWAC2 - DMEM User Write Access Control for Quadrant 2 00 - User-mode CPU write accesses are allowed to quadrant 2 of DMEM 01 - User-mode CPU write accesses are not allowed to quadrant 2 of DMEM 10 - User-mode CPU write accesses are conditionally allowed to quadrant 2 of DMEM based on memory protection logic 11 - Reserved **Note:** individual control fields are provided for four equal sized sections of DMEM regardless of DMEM size. A single control field is used for each access based on the upper two address bits used to index into the DMEM. DUWAC2 is used when these two bits = 2‘b10
20:21	DUWAC1	DUWAC1 - DMEM User Write Access Control for Quadrant 1 00 - User-mode CPU write accesses are allowed to quadrant 1 of DMEM 01 - User-mode CPU write accesses are not allowed to quadrant 1 of DMEM 10 - User-mode CPU write accesses are conditionally allowed to quadrant 1 of DMEM based on memory protection logic 11 - Reserved **Note:** individual control fields are provided for four equal sized sections of DMEM regardless of DMEM size. A single control field is used for each access based on the upper two address bits used to index into the DMEM. DUWAC1 is used when these two bits = 2‘b01
22:23	DUWAC0	DUWAC0 - DMEM User Write Access Control for Quadrant 0 00 - User-mode CPU write accesses are allowed to quadrant 0 of DMEM 01 - User-mode CPU write accesses are not allowed to quadrant 0 of DMEM 10 - User-mode CPU write accesses are conditionally allowed to quadrant 0 of DMEM based on memory protection logic 11 - Reserved **Note:** individual control fields are provided for four equal sized sections of DMEM regardless of DMEM size. A single control field is used for each access based on the upper two address bits used to index into the DMEM. DUWAC0 is used when these two bits = 2‘b00

Table 262. DMEMCTL1 Field Descriptions (continued)

Bits	Name	Description
24:25	DURAC3	DURAC3 - DMEM User Read Access Control for Quadrant 3 00 - User-mode CPU read accesses are allowed to quadrant 3 of DMEM 01 - User-mode CPU read accesses are not allowed to quadrant 3 of DMEM 10 - User-mode CPU read accesses are conditionally allowed to quadrant 3 of DMEM based on memory protection logic 11 - Reserved **Note:** individual control fields are provided for four equal sized sections of DMEM regardless of DMEM size. A single control field is used for each access based on the upper two address bits used to index into the DMEM. DURAC3 is used when these two bits = 2‘b11
26:27	DURAC2	DURAC2 - DMEM User Read Access Control for Quadrant 2 00 - User-mode CPU read accesses are allowed to quadrant 2 of DMEM 01 - User-mode CPU read accesses are not allowed to quadrant 2 of DMEM 10 - User-mode CPU read accesses are conditionally allowed to quadrant 2 of DMEM based on memory protection logic 11 - Reserved **Note:** individual control fields are provided for four equal sized sections of DMEM regardless of DMEM size. A single control field is used for each access based on the upper two address bits used to index into the DMEM. DURAC2 is used when these two bits = 2‘b10
28:29	DURAC1	DURAC1 - DMEM User Read Access Control for Quadrant 1 00 - User-mode CPU read accesses are allowed to quadrant 1 of DMEM 01 - User-mode CPU read accesses are not allowed to quadrant 1 of DMEM 10 - User-mode CPU read accesses are conditionally allowed to quadrant 1 of DMEM based on memory protection logic 11 - Reserved **Note:** individual control fields are provided for four equal sized sections of DMEM regardless of DMEM size. A single control field is used for each access based on the upper two address bits used to index into the DMEM. DURAC1 is used when these two bits = 2‘b01
30:31	DURAC0	DURAC0 - DMEM User Read Access Control for Quadrant 0 00 - User-mode CPU read accesses are allowed to quadrant 0 of DMEM 01 - User-mode CPU read accesses are not allowed to quadrant 0 of DMEM 10 - User-mode CPU read accesses are conditionally allowed to quadrant 0 of DMEM based on memory protection logic 11 - Reserved **Note:** individual control fields are provided for four equal sized sections of DMEM regardless of DMEM size. A single control field is used for each access based on the upper two address bits used to index into the DMEM. DURAC0 is used when these two bits = 2‘b00

16.9.2.4 IMEM Configuration Register 0 (IMEMCFG0)

The IMEM Configuration Register 0 (IMEMCFG0) is a 32-bit read-only register. IMEMCFG0 provides information about the configuration of the Core (z425n3) local instruction memory design. The contents of the IMEMCFG0 register can be read using a **mfspr** instruction. The SPR number for IMEMCFG0 is 695 in decimal. The IMEMCFG0 register is shown in *Figure 160*.

IMEM BASE ADDR	0	IECUA	IECA	0	IMSIZE	0
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19	20	21	22	23	24 25 26 27 28 29	30 31
CURRRENT IMEM BASE ADDRESS VALUE	0	1	1	0	000100 (16 KB)	00

SPR - 695; Read-only; Privileged

Figure 160. IMEM Configuration Register 0 (IMEMCFG0)

The IMEMCFG0 fields are described in *Table 263*.

Table 263. IMEMCFG0 Field Descriptions

Bits	Name	Description
0:19	IMEM BASE ADDR	IMEM BASE ADDRESS (CPU Port) This field defines the current Base Address value being used for the CPU port to the IMEM. This field reflects the value of the external IMEM base address port inputs when the external base address port inputs are enabled via the DBAPD control bit, or the value of the BASE ADDRESS field of IMEMCFG0 when the external base address port inputs are disabled via the DBAPD control bit. **Note:** low order bits of this field are driven to 0's when the IMEM size is > 4 Kbyte
20	-	Reserved
21	IECUA	IECUE - IMEM Error Correction Update Available 0 - Error Correction Update is not available 1 - Error Correction Update is available for correction of a correctable single-bit error detected on a read access IECUA indicates an error scrubbing function for the IMEM is available.
22	IECA	IECUE - IMEM Error Correction Available 0 - Error Correction is not available 1 - Error Correction is available IECA indicates an error correction function for the IMEM is available.
23	-	Reserved
24:29	IMSIZE	IMSIZE - IMEM Size 000100 - The size of the IMEM is 16 Kbyte
30:31	-	Reserved

16.9.2.5 IMEM Control Register 0 (IMEMCTL0)

The IMEM Control Register 0 (IMEMCTL0) controls operation of certain functions of the IMEM logic.

IMEM BASE ADDR	0	0	IECUE	0	IBAPD	0	ISWCE	IDAUEC	ICECE	ISECE
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17	18 19	20 21	22	23	24	25 26	27 28	29	30	31

DCR - 497; Read/Write; Reset(1) - 22'b0 ||| p_imem_rstcfg[0] || 4'b0 || p_imem_rstcfg[1:5]; Privileged

Figure 161. IMEM Control Register 0 (IMEMCTL0)

1. reset by **m_por** and **slv_reset_b**. Unaffected by **p_reset_b**.

Table 264. IMEMCTL0 Field Descriptions

Bits	Name	Description
0:17	IMEM BASE ADDR	IMEM BASE ADDRESS Field (CPU Port) This field defines the Base Address used for the CPU port to the IMEM when the external base address port inputs are disabled via the BAPD control bit. **Note:** changes to this value and to the BAPD control bit must be performed carefully by software to ensure coherency is maintained. Specifically, software must ensure that the BTB is invalidated, and that no cached entries are present in the ICache for the memory space to be occupied by the IMEM.
18:19	-	Reserved for Base Address extension
20:21	-	Reserved
22	IECUE	IECUE - IMEM Error Correction Update Enable 0 - Error Correction Update is disabled 1 - Error Correction Update is enabled. A correctable single-bit error detected on CPU instruction fetches or slave port read or partial-width write cycles will cause the IMEM to be re-written with the corrected data. IECUE can be used to provide an error scrubbing function. **Note:** this field is updated on **m_por** and **slv_reset_b** based on the value of the **p_imem_rstcfg[0]** input.
23	-	Reserved
24	IBAPD	IBAPD - IMEM Base Address Port Disable 0 - The external IMEM base address port (**p_imem_baseaddr[0:n]**) is used to define the base address of the IMEM for CPU accesses. 1 - The external IMEM base address port (**p_imem_baseaddr[0:n]**) is disabled for use, and instead the BASE ADDR field value is used to define the base address of the IMEM for CPU accesses. This bit controls usage of the external Base Address port (**p_imem_baseaddr[0:n]**) to define the base address of the IMEM for CPU port accesses. It has no effect on IMEM slave port accesses.
25:26	-	Reserved
27:28	ISWCE	ISWCE - IMEM Slave port Write Check/Correct Enable 00 - Slave write data is not checked or corrected for errors. 01 - Slave write data is checked but not corrected for errors. Detected errors generate a slave port ERROR response and no Write is performed to the IMEM. 10 - Slave write data is checked and corrected for errors on all writes. Single-bit correctable errors do not automatically generate an ERROR response, but are instead corrected. 11 - Slave write data is checked and corrected for errors on partial-width (1, 2, 3, 4, 5, 6, or 7 byte) writes. Single-bit correctable errors on partial-width writes do not automatically generate an ERROR response, but are instead corrected. **Note:** this field is updated on **m_por** and **slv_reset_b** based on the values of the **p_imem_rstcfg[1:2]** inputs.

Table 264. IMEMCTL0 Field Descriptions (continued)

Bits	Name	Description
29	IDAUEC	IDAUEC - IMEM Disable Address Use in Error Check 0 - Use of access address is enabled in checkbit and syndrome generation. 1 - Use of access address is disabled in checkbit and syndrome generation. This bit controls usage of the address portion of the ECC H matrix. When use is disabled, the address portions are not used. for checkbit/syndrome generation for IMEM CPU or slave port accesses. **Note:** this field is updated on **m_por** and **slv_reset_b** based on the value of the **p_imem_rstcfg[3]** input.
30	ICECE	ICPECE - IMEM CPU Port ECC Enable (CPU port) 0 - End-to-End ECC is disabled for the CPU interface. No checking of read data is performed by the CPU. Writes will still generate the proper check bit values to be written. 1- End-to-End ECC is enabled for performing ECC on the CPU interface. Error checking of read data is performed with single-bit error correction. Detection of uncorrectable single- or multi-bit errors on a CPU port access cause a machine check to be generated. **Note:** this field is updated on **m_por** and **slv_reset_b** based on the value of the **p_imem_rstcfg[4]** input.
31	ISECE	ISECE - IMEM Slave port Error Checking Enable (Slave port) 0 - End-to-End ECC checking logic is disabled for the Slave interface. No checking of read data is performed during read-modify-write operations for partial-width writes. Writes will still generate the proper check bit values to be written. 1 - End-to-End ECC is enabled for performing ECC on the Slave Port interface. Error checking of read data is performed during read-modify-write operations for partial-width writes with single-bit error correction. Detection of uncorrectable single- or multi-bit errors on a Slave port access causes a bus error ERROR response to be generated. **Note:** this field is updated on **m_por** and **slv_reset_b** based on the value of the **p_imem_rstcfg[5]** input.

16.9.2.6 IMEM Control Register 1 (IMEMCTL1)

The IMEM Control Register 1 (IMEMCTL1) provides protection functions for the IMEM. Separate Supervisor-mode and User-mode instruction fetch access controls allow accesses to be granted, denied, or conditionally granted independently for four equally-sized blocks of IMEM. Conditional granting of access permissions causes the MPU memory protection functions to be employed, all other settings override MPU settings.

0	ISXAC3	ISXAC2	ISXAC1	ISXAC0	0	IUXAC3	IUXAC2	IUXAC1	IUXAC0
0 1 2 3 4 5 6 7	8 9	10 11	12 13	14 15	16 17 18 19 20 21 22 23	24 25	26 27	28 29	30 31

DCR - 499; Read/Write; Reset(1) - 32'b0; Privileged

Figure 162. IMEM Control Register 1 (IMEMCTL1)

1. reset by **m_por** and **p_reset_b**.

Table 265. IMEMCTL1 Field Descriptions

Bits	Name	Description
0:7	-	Reserved
8:9	ISXAC3	ISXAC3 - IMEM Supervisor Instruction Fetch Access Control for Quadrant 3 00 - Supervisor-mode CPU instruction fetch accesses are allowed to quadrant 3 of IMEM 01 - Supervisor-mode CPU instruction fetch accesses are not allowed to quadrant 3 of IMEM 10 - Supervisor-mode CPU instruction fetch accesses are conditionally allowed to quadrant 3 of IMEM based on memory protection logic 11 - Reserved **Note:** individual control fields are provided for four equal sized sections of IMEM regardless of IMEM size. A single control field is used for each access based on the upper two address bits used to index into the IMEM. ISXAC3 is used when these two bits = 2‘b11
10:11	ISXAC2	ISXAC2 - IMEM Supervisor Instruction Fetch Access Control for Quadrant 2 00 - Supervisor-mode CPU instruction fetch accesses are allowed to quadrant 2 of IMEM 01 - Supervisor-mode CPU instruction fetch accesses are not allowed to quadrant 2 of IMEM 10 - Supervisor-mode CPU instruction fetch accesses are conditionally allowed to quadrant 2 of IMEM based on memory protection logic 11 - Reserved **Note:** individual control fields are provided for four equal sized sections of IMEM regardless of IMEM size. A single control field is used for each access based on the upper two address bits used to index into the IMEM. ISXAC2 is used when these two bits = 2‘b10
12:13	ISXAC1	ISXAC1 - IMEM Supervisor Instruction Fetch Access Control for Quadrant 1 00 - Supervisor-mode CPU instruction fetch accesses are allowed to quadrant 1 of IMEM 01 - Supervisor-mode CPU instruction fetch accesses are not allowed to quadrant 1 of IMEM 10 - Supervisor-mode CPU instruction fetch accesses are conditionally allowed to quadrant 1 of IMEM based on memory protection logic 11 - Reserved **Note:** individual control fields are provided for four equal sized sections of IMEM regardless of IMEM size. A single control field is used for each access based on the upper two address bits used to index into the IMEM. ISXAC1 is used when these two bits = 2‘b01
14:15	ISXAC0	ISXAC0 - IMEM Supervisor Instruction Fetch Access Control for Quadrant 0 00 - Supervisor-mode CPU instruction fetch accesses are allowed to quadrant 0 of IMEM 01 - Supervisor-mode CPU instruction fetch accesses are not allowed to quadrant 0 of IMEM 10 - Supervisor-mode CPU instruction fetch accesses are conditionally allowed to quadrant 0 of IMEM based on memory protection logic 11 - Reserved **Note:** individual control fields are provided for four equal sized sections of IMEM regardless of IMEM size. A single control field is used for each access based on the upper two address bits used to index into the IMEM. ISXAC0 is used when these two bits = 2‘b00
16:23	-	Reserved

Table 265. IMEMCTL1 Field Descriptions (continued)

Bits	Name	Description
24:25	IUXAC3	IUXAC3 - IMEM User Instruction Fetch Access Control for Quadrant 3 00 - User-mode CPU instruction fetch accesses are allowed to quadrant 3 of IMEM 01 - User-mode CPU instruction fetch accesses are not allowed to quadrant 3 of IMEM 10 - User-mode CPU instruction fetch accesses are conditionally allowed to quadrant 3 of IMEM based on memory protection logic 11 - Reserved **Note:** individual control fields are provided for four equal sized sections of IMEM regardless of IMEM size. A single control field is used for each access based on the upper two address bits used to index into the IMEM. IUXAC3 is used when these two bits = 2‘b11
26:27	IUXAC2	IUXAC2 - IMEM User Instruction Fetch Access Control for Quadrant 2 00 - User-mode CPU instruction fetch accesses are allowed to quadrant 2 of IMEM 01 - User-mode CPU instruction fetch accesses are not allowed to quadrant 2 of IMEM 10 - User-mode CPU instruction fetch accesses are conditionally allowed to quadrant 2 of IMEM based on memory protection logic 11 - Reserved **Note:** individual control fields are provided for four equal sized sections of IMEM regardless of IMEM size. A single control field is used for each access based on the upper two address bits used to index into the IMEM. IUXAC2 is used when these two bits = 2‘b10
28:29	IUXAC1	IUXAC1 - IMEM User Instruction Fetch Access Control for Quadrant 1 00 - User-mode CPU instruction fetch accesses are allowed to quadrant 1 of IMEM 01 - User-mode CPU instruction fetch accesses are not allowed to quadrant 1 of IMEM 10 - User-mode CPU instruction fetch accesses are conditionally allowed to quadrant 1 of IMEM based on memory protection logic 11 - Reserved **Note:** individual control fields are provided for four equal sized sections of IMEM regardless of IMEM size. A single control field is used for each access based on the upper two address bits used to index into the IMEM. IUXAC1 is used when these two bits = 2‘b01
30:31	IUXAC0	IUXAC0 - IMEM User Instruction Fetch Access Control for Quadrant 0 00 - User-mode CPU instruction fetch accesses are allowed to quadrant 0 of IMEM 01 - User-mode CPU instruction fetch accesses are not allowed to quadrant 0 of IMEM 10 - User-mode CPU instruction fetch accesses are conditionally allowed to quadrant 0 of IMEM based on memory protection logic 11 - Reserved **Note:** individual control fields are provided for four equal sized sections of IMEM regardless of IMEM size. A single control field is used for each access based on the upper two address bits used to index into the IMEM. IUXAC0 is used when these two bits = 2‘b00

16.10 End-to-End ECC support

This chapter describes End-to-End Error detection and correction capability (e2eECC) enhancements the Core (z425n3) to address additional diagnostic capabilities for the next generation of embedded designs.

16.10.1 e2eECC control and configuration

e2eECC is controlled by a set of privileged device control registers (DCRs) which are accessed using the **mfdcr** and **mtdcr** instructions. These registers and a description of the operation of various features are described in the following subsections.

16.10.1.1 End-to-End ECC Control Register 0 (E2ECTL0)

The End-to-End ECC Control Register 0 (E2ECTL0) controls operation of the e2eECC logic.

Note: Even if e2eECC is disabled, storage are protected by ECC as reported in Section 6.4.4: ECC.



DCR - 510; Read/Write; Reset - 0x0 || p_e2e_rstcfg[0:3]; Privileged

Figure 163. e2eECC Control Register 0 (E2ECTL0)

Table 266. E2ECTL0 Field Descriptions

Bits	Name	Description
0:27	-	These bits are reserved
28	IDAC	IDAC - Instruction Disable Address Checking 0 - Use of access address is enabled in checkbit and syndrome generation. 1 - Use of access address is disabled in checkbit and syndrome generation. This bit controls usage of the address portion of the ECC H matrix. When use is disabled, the address portion are not used for checkbit/syndrome generation for Instruction AHB accesses. **Note:** this field is updated on **p_reset_b** based on the value of the **p_e2e_rstcfg[0]** input.
29	IECCEN	Instruction ECC Enable Field 0 - End-to-End ECC is disabled for the Instruction interface. No checking of instruction fetch data is performed. 1 - End-to-End ECC is enabled for performing error detection and correction on the Instruction interface. Checking of instruction fetch data is performed with correction of single-bit data errors. Detection of any address errors or uncorrectable multi-bit data errors cause a machine check to be generated if the instruction fetch information is attempted to be used for instruction decoding and execution. **Note:** this field is updated on **p_reset_b** based on the value of the **p_e2e_rstcfg[1]** input.

Table 266. E2ECTL0 Field Descriptions (continued)

Bits	Name	Description
30	DDAC	DDAC - Data Disable Address Checking 0 - Use of access address is enabled in checkbit and syndrome generation. 1 - Use of access address is disabled in checkbit and syndrome generation. This bit controls usage of the address portion of the ECC H matrix. When use is disabled, the address portion are not used for checkbit/syndrome generation for Data AHB accesses. **Note:** this field is updated on **p_reset_b** based on the value of the **p_e2e_rstcfg[2]** input.
31	DECCEN	Data ECC Enable Field 0 - End-to-End ECC is disabled for the Data interface. No checking of read data is performed. Writes will still generate the proper check bit values to be supplied to external storage devices. 1 - End-to-End ECC is enabled for performing error detection and correction on the Data interface. Checking of read data is performed with correction of single-bit data errors. Detection of any address errors, or uncorrectable multi-bit data errors cause a machine check to be generated. Writes generate the proper check bit values to be supplied to external storage devices. **Note:** this field is updated on **p_reset_b** based on the value of the **p_e2e_rstcfg[3]** input.

Enabling of e2eECC operation is performed by setting the [I,D]ECCEN bit of the E2ECTL0 DCR to a one. On reset, e2eECC operation may be disabled from detecting or correcting errors based on reset configuration inputs, and no exceptions will be signaled for non-zero calculated syndrome values. This allows for the proper initialization of stored checkbits in any volatile storage by the CPU without causing error conditions due to uninitialized storage. Following the initialization, software may enable e2eECC operation by writing the appropriate values to the $E2ECTL0_{[I,D]ECCEN}$ control bits.

The $E2ECTL0_{[I,D]DAC}$ control bits can be used to remove the address checking component of the ECC logic for the Instruction AHB and Data AHB when address inclusion is not desired.

16.10.1.2 End-to-end ECC error generation by software

Error generation provides a way to test error recovery and portions of the error detection/correction logic by intentionally injecting parity errors into the driven checkbit information on the external bus during write operations. No provision is made to generate internal errors on read data due to timing issues; instead, software may intentionally generate errors in the checkbit values to emulate data, address or checkbit error on external bus write cycles, and then may read back the written data to cause an error to be detected.

The End-to-End ECC Error Control/Status Register (E2EECSR0) controls operation of the e2eECC error injection logic. It allows for generation of a single error injection operation on the next CPU data write to the external bus only, or for a series of error injection write operations to the external bus, using the WRC and INVC control fields. The WRCHKBIT/CHKINVT field controls which of the **p_hwchkbit[7:0]** outputs are affected.

Control is provided to allow for either inverting the value(s) of one or more of the checkbit outputs **p_hwchkbit[7:0]** in hardware on one or more external write accesses, or for software to directly supply checkbit values to be used for the external write access(es).

The End-to-End ECC Error Control/Status Register (E2EECSR0) also supports access to raw checkbit values via the RCHKBIT status field. This field is updated with checkbit values received on each CPU data read operation to either the DMEM or to the external bus.

Note that Nexus3 accesses do not cause error injection or capturing of read checkbits, regardless of the settings of the E2EECSR0 register.

Figure 164 shows the layout of E2EECSR0.

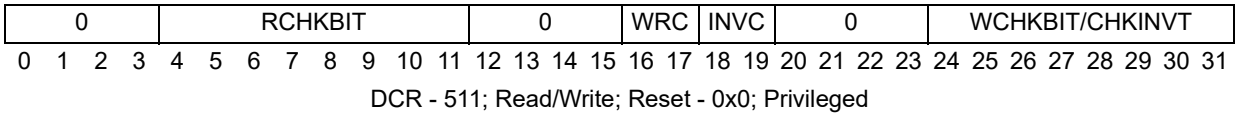


Figure 164. e2eECC Error Control/Status Register 0 (E2EECSR0)

Table 267. E2EECSR0 Field Descriptions

Bits	Name	Description
0:3	-	These bits are reserved
4:11	RCHKBIT	Read Checkbits This field provides the raw checkbits received on the last CPU data access to the DMEM or to external memory via the Data BIU. This field corresponds bit-wise to **p_hrchkbit[7:0]**. for external reads and to **p_dchk[0:7]** for DMEM read accesses. Software may use this information to determine the stored checkbits of external memories or the DMEM by performing CPU load operations and then accessing this field prior to the next load operation (assuming interrupts are masked).
12:15	-	These bits are reserved
16:17	WRC	Write Control 00 - No write checkbit substitution is performed by this control bit 01 - Checkbit substitution is performed for the next CPU external write access (only) by driving **p_hwchkbit[7:0]** with the values in the WCHKBIT control field. Software must clear this field before re-writing to 01 in order to generate a subsequent checkbit substitution operation. 10 - Checkbit substitution is performed for each CPU external write access while this field remains set to 10 by driving **p_hwchkbit[7:0]** with the values in the WCHKBIT control field. This field must be cleared by software to discontinue further checkbit substitution. 11- Reserved **Note:** checkbit substitution has priority over Error injection
18:19	INVC	Invert Control 00 - No error injection is performed by this control bit 01 - Error injection is performed for the next CPU external write access (only) by modifying **p_hwchkbit[7:0]** based on CHKINVT. Software must clear this field before re-writing to 01 in order to generate a subsequent error injection operation. 10 - Error injection is performed for each CPU external write access while this field remains set to 10 by modifying **p_hwchkbit[7:0]** based on CHKINVT. This field must be cleared by software to discontinue further error generation. 11 - Reserved **Note:** checkbit substitution has priority over Error injection

Table 267. E2EECSR0 Field Descriptions (continued)

Bits	Name	Description
20:23	-	These bits are reserved
24:31	WCHKBIT/ CHKINVT	Write Checkbits /Checkbit Invert Mask This field provides the checkbit substitution values when performing checkbit substitution via the WRC control field, and controls which checkbits are inverted when performing error injection via the INVC control field. This field corresponds bit-wise to **p_hwchkbit[7:0]**. For Checkbit inversion operations via error injection: 0 - Checkbit is driven normally 1 - Checkbit is inverted before being driven out on **p_hwchkbit[7:0]** when error injection occurs.

17 System Integration Unit Lite2 (SIUL2)

17.1 Introduction

17.1.1 Overview

The System Integration Unit Lite2 (SIUL2) provides control over all the I/O ports on this device and supports 13 ports with up to 16 bits of bidirectional, general-purpose input and output signals. It supports 16 external interrupts with trigger event configuration. *Figure 165* is the block diagram of SIUL2 and its interfaces to other system components for this device.

The SIUL2 provides the GPIO ports on the device.

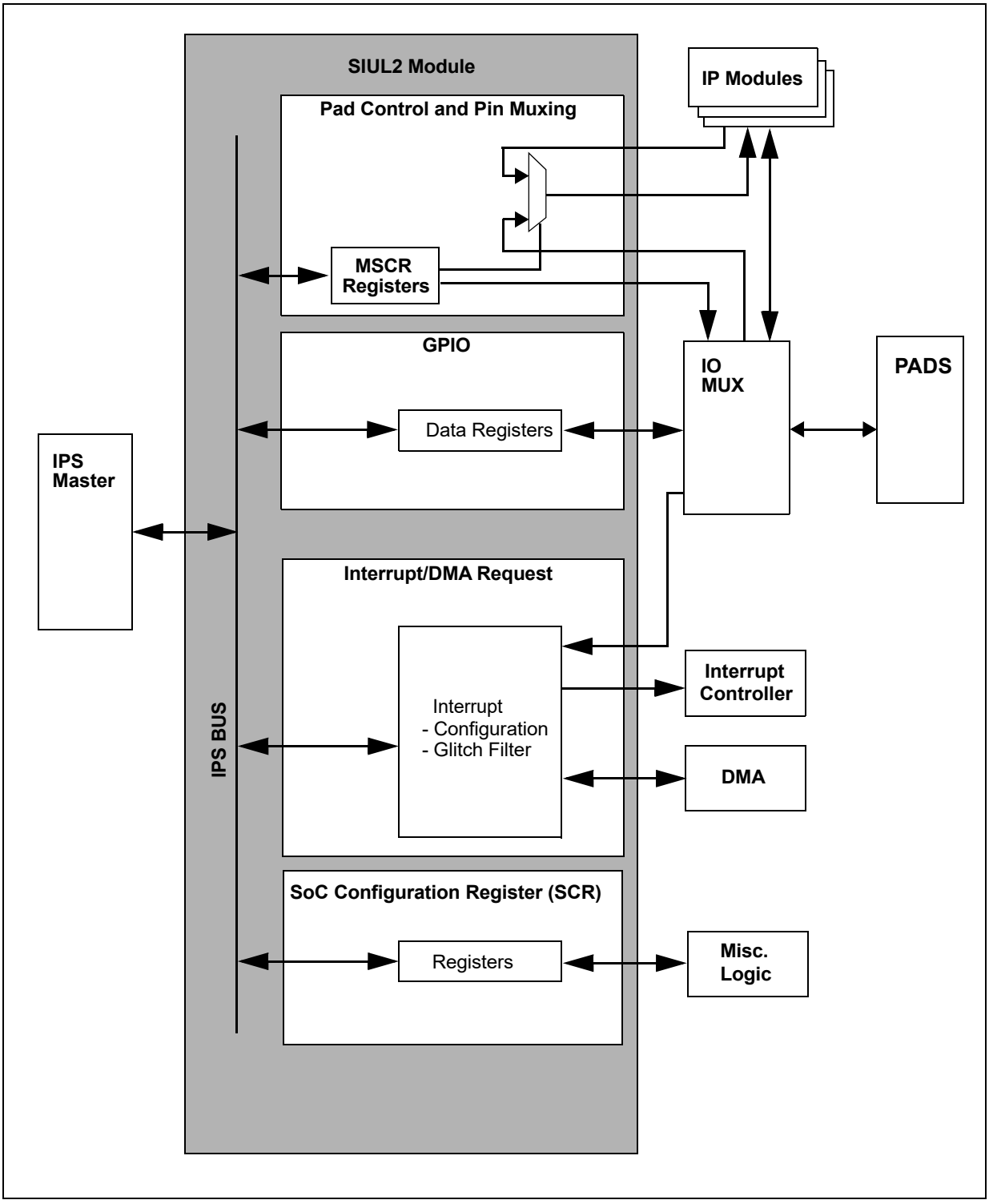
- When configured as output, you can write to the internal GPDO register to control the state driven on the associated output pad.
- When configured as input, you can detect the state of the associated pad by reading the value from the internal GPDI register.
- When configured as input and output, the pad value can be read back, which can be used as a method of checking if the driven value on the associated I/O pad.

In order to assist in software development, there are different access mechanisms to the GPIO data registers. These differing mechanisms allow support for port access or for bit manipulation without the need to use read-modify-write operations or access with Power Architecture reservation:

- Access to two 16-bit ports in one access
- Read/Write access to a single bit
- A 16-bit port write with a bit mask, using single 32-bit access

There are 16 external interrupts and 16 DMA inputs to the SIUL2 that map to EIRQ pins on the device. The interrupt/DMA sources can be configured to have a digital filter to reject short glitches on the inputs.

Figure 165. SIUL2 block diagram[9]



9. For pinout details, refer to the SPC58xEx/SPC58xGx pinout Microsoft Excel file attached to the IO_Definition document.

17.1.2 Features

The SIUL2 supports these distinctive features:

- 13 GPIO ports with data control[10]
  - Drive data to up to 16 independent I/O channels
  - Sample data from up to 16 independent I/O channels
- Two 16-bit registers can be read/written with one access to a 32-bit port if needed.
- External interrupt/DMA request support with:
  - 2 system interrupt vectors for 16 interrupt sources (16 connected to EIRQ pins for this device) with independent interrupt mask
  - 16 programmable digital glitch filters (one for each EIRQ)
  - Independent DMA channels for each EIRQ pin
  - Edge detection

Details about the electrical parameters and setting of functional pads can be found in the SPC58xEx/SPC58xGx pinout Microsoft Excel file attached to the IO_Definition document. These are used to configure the following pad features:

- Drive strength
- Output impedance control
- Open drain/source output enable
- Slew rate control of JTAG and FlexRay pins
- Internal weak pull control
- Pin function assignment
- Safe mode behavior configuration
- TTL, CMOS and Automotive input levels

17.1.3 Register protection

The System Integration Unit Lite2 uses the Register Protection Scheme to protect the individual registers from accidental writes. The following registers can be protected:

- SIUL2 DMA/Interrupt Request Enable Register 0(SIUL2_DIRER0)
- SIUL2 DMA/Interrupt Request Select Register 0(SIUL2_DIRSR0)
- SIUL2 Interrupt Rising-Edge Event Enable Register 0(SIUL2_IREER0)
- SIUL2 Interrupt Falling-Edge Event Enable Register 0(SIUL2_IFEER0)
- SIUL2 Interrupt Filter Enable Register 0(SIUL2_IFER0)
- SIUL2 Interrupt Filter Maximum Counter Register (SIUL2_IFMCR0–SIUL2_IFMCR31)
- SIUL2 Interrupt Filter Clock Prescaler Register (SIUL2_IFCPR)
- SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR0–SIUL2_MSCR1023)

Details about the protected registers at specific addresses can be found in *Chapter 91: Register Protection (REG_PROT)* and *Chapter 7: Device configuration*.

10. For pinout details, refer to the SPC58xEx/SPC58xGx pinout Microsoft Excel file attached to the IO_Definition document.

17.2 Memory map and register description

This section provides a detailed description of all registers accessible in this module.

17.2.1 Memory map

Table 268 gives an overview of the SIUL2 registers implemented.

Note: Reserved registers will be read as 0, write will have no effect. If supported and enabled in this MCU, a transfer error will be issued when trying to access completely reserved register space. A 8-bit or 16-bit access to an "only 32-bit supported access register" gives a transfer error. Unimplemented register accesses give transfer error, that is due to register not being present.

For SIUL2_MSCR_MUX_512-SIUL2_MSCR_MUX1023, only 32/16-bit accesses are supported (8-bit writes do not occur and read zeros).

Table 268. SIUL2 memory map

Address offset	Register name	Location
0x0004	MCU ID Register #1 (SIUL2_MIDR1)	*Section 17.2.2.1*
0x0008	MCU ID Register #2 (SIUL2_MIDR2)	*Section 17.2.2.2*
0x0010	DMA/Interrupt Status Flag Register 0 (SIUL2_DISR0)	*Section 17.2.2.3*
0x0018	DMA/Interrupt Request Enable Register 0 (SIUL2_DIRER0)	*Section 17.2.2.4*
0x0020	DMA/Interrupt Request Select Register 0 (SIUL2_DIRSR0)	*Section 17.2.2.5*
0x0028	Interrupt Rising-Edge Event Enable Register 0 (SIUL2_IREER0)	*Section 17.2.2.6*
0x0030	Interrupt Falling-Edge Event Enable Register 0 (SIUL2_IFEER0)	*Section 17.2.2.7*
0x0038	Interrupt Filter Enable Register 0 (SIUL2_IFER0)	*Section 17.2.2.8*
0x0040–0x00BC	Interrupt Filter Maximum Counter Register 0–Interrupt Filter Maximum Counter Register 31 (SIUL2_IFMCR0–SIUL2_IFMCR31)	*Section 17.2.2.9*
0x00C0	Interrupt Filter Clock Prescaler (SIUL2_IFCPR)	*Section 17.2.2.10*
0x0100	SoC Configuration Register 0 (SIUL2_SCR0)	*Section 17.2.2.11*
0x240-0x0A3C	I/O Pin Multiplexed Signal Configuration Register 0–I/O Pin Multiplexed Signal Configuration Register 511 (SIUL2_MSCR_IO_0–SIUL2_MSCR_IO_511)	*Section 17.2.2.12*
0x0A40-0x123C	I/O Pin Multiplexed Signal Configuration Register 512–I/O Pin Multiplexed Signal Configuration Register 1023 (SIUL2_MSCR0_MUX_512–SIUL2_MSCR_MUX_1023)	*Section 17.2.2.13*
0x1300–0x14FF	GPIO Pad Data Out Register 0–GPIO Pad Data Out Register 511 (SIUL2_GPDO0–SIUL2_GPDO511)	*Section 17.2.2.14*
0x1500–0x16FF	GPIO Pad Data In Register 0–GPIO Pad Data In Register 511 (SIUL2_GPDI0–SIUL2_GPDI511)	*Section 17.2.2.15*
0x1700–0x173E	Parallel GPIO Pad Data Out Register 0–Parallel GPIO Pad Data Out Register 12 (SIUL2_PGPDO0–SIUL2_PGPDO12)	*Section 17.2.2.16*

Table 268. SIUL2 memory map (continued)

Address offset	Register name	Location
0x1740–0x177E	Parallel GPIO Pad Data In Register 0–Parallel GPIO Pad Data In Register 12 (SIUL2_PGPDI0–SIUL2_PGPDI12)	*Section 17.2.2.17*
0x1780–0x17FC	Masked Parallel GPIO Pad Data Out Register 0–Masked Parallel GPIO Pad Data Out Register 12 (SIUL2_MPGPDO0–SIUL2_MPGPDO12)	*Section 17.2.2.18*

17.2.2 Register descriptions

This section describes in address order all the SIUL2 registers.

17.2.2.1 MCU ID Register #1 (SIUL2_MIDR1)

This register contains identification information about the device.

Offset: 0x0004 Access: Read-only

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	PARTNUM															
W																
Reset	0	1	0	1	1	0	0	0	1	0	0	0	1	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	ED	PKG					0	0	MAJOR_MASK				MINOR_MASK			
W																
Reset	0	0(1)	0(1)	0(1)	0(1)	0(1)	0	0	X(2)	X(2)	X(2)	X(2)	X(2)	X(2)	X(2)	X(2)

1. Only possible to change value by modifying the side band signals (in the Test Flash). Values are set at factory and cannot be modified.
2. Value is set at factory and will vary each revision of device

Figure 166. SIUL2 MCU ID Register #1 (SIUL2_MIDR1)

Table 269. SIUL2_MIDR1 field descriptions

Field	Description
0:15 PARTNUM	MCU Part Number These digits identify the part number of the chip. Default is: – SPC58xEx: 0x5888 – SPC58xGx: 0x5846
16 ED	Read back as ‘1’ if an ED device. Reads back as ‘0’ in PD.

Table 269. SIUL2_MIDR1 field descriptions (continued)

Field	Description
17:21 PKG	Package Settings Can by read by software to determine the package type that is used for the particular device: 01000 BGA292 01001 QFP100 01101 QFP144 10001 QFP176 11111 KGD All other combinations are reserved for future use.
24:27 MAJOR_MASK	Major Mask revision
28:31 MINOR_MASK	Minor Mask revision

17.2.2.2 MCU ID Register #2 (SIUL2_MIDR2)

This register contains identification information about the device.

Offset: 0x0008 Access: Read-only

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	SF	FLASH_SIZE_1			FLASH_SIZE_2				0	0	0	0	0	0	0	0
W																
Reset	1	1	0	0	0	0	1	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	FAMILYNUM								0	0	0	0	0	0	0	0
W																
Reset	0	1	0	0	0	1	0	1	0	0	0	0	0	0	0	0

Figure 167. SIUL2 MCU ID Register #2 (SIUL2_MIDR2)

Table 270. SIUL2_MIDR2 field description

Field	Description
0 SF	Manufacturer: 0 Reserved 1 STMicroelectronics
1:4 FLASH_SIZE_1	Coarse granularity for Flash memory size Needs to be combined with FLASH_SIZE_2 to calculate the actual memory size: 0000 16 KB 0001 32 KB 0010 64 KB 0011 128 KB 0100 256 KB 0101 512 KB 0110 1 MB 0111 2 MB 1000 4 MB 1001 8 MB 1010 16 MB 1011 32 MB 1100 64 MB 1101 128 MB 1110 256 MB 1111 512 MB
5:8 FLASH_SIZE_2	Fine granularity for Flash memory size Needs to be combined with FLASH_SIZE_1 to calculate the actual memory size: 0000 0 KB 0001 (FLASH_SIZE_1) / 8 0010 2 * (FLASH_SIZE_1) / 8 0011 3 * (FLASH_SIZE_1) / 8 0100 4 * (FLASH_SIZE_1) / 8 0101 5 * (FLASH_SIZE_1) / 8 0110 6 * (FLASH_SIZE_1) / 8 0111 7 * (FLASH_SIZE_1) / 8 1000 8 * (FLASH_SIZE_1) / 8 1001 9 * (FLASH_SIZE_1) / 8 1010 10 * (FLASH_SIZE_1) / 8 1011 11 * (FLASH_SIZE_1) / 8 1100 12 * (FLASH_SIZE_1) / 8 1101 13 * (FLASH_SIZE_1) / 8 1110 14 * (FLASH_SIZE_1) / 8 1111 15 * (FLASH_SIZE_1) / 8
16:23 FAMILYNUM	ASCII character in MCU part number: 0x45 “E”, 0x47 “G“ All other values reserved for future use.

17.2.2.3 DMA/Interrupt Status Flag Register 0 (SIUL2_DISR0)

The DMA/Interrupt Status Register contains flag bits that record an event on the external IRQ pins. When an event as defined in IRQ Rising-Edge Event Enable Register (SIUL2_IREER0) and IRQ Falling-Edge Event Enable Register (SIUL2_IFEER0) occurs, the corresponding flag bit is set. The IRQ Flag bit is set irrespective of the corresponding DMA/Interrupt Request Enable bit in DMA/Interrupt Request Enable Register (SIUL2_DIRER0) is enabled. The EIF bit remains set until cleared by software or through

the servicing of a DMA request. The EIF bits are cleared by writing a '1' to the bits. A write of '0' has no effect.

This register contains the interrupt flags.

Offset: 0x0010 Access: User read/write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	EIF15	EIF14	EIF13	EIF12	EIF11	EIF10	EIF9	EIF8	EIF7	EIF6	EIF5	EIF4	EIF3	EIF2	EIF1	EIF0
W	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 168. SIUL2 DMA/Interrupt Status Flag Register (SIUL2_DISR0)

Table 271. SIUL2_DISR0 field descriptions

Field	Description
16:31 EIF[15:0]	External Interrupt Status Flag x This flag can be cleared only by writing 1. Writing 0 has no effect. If enabled (SIUL2_DIRER[x]), EIF[x] causes an interrupt or DMA request. 0 No interrupt or DMA event has occurred on the pad 1 An interrupt or DMA event as defined by SIUL2_IREER[x] and SIUL2_IFEER[x] has occurred

17.2.2.4 DMA/Interrupt Request Enable Register 0 (SIUL2_DIRER0)

The DMA/Interrupt Request Enable Register enables the assertion of DMA or interrupt request if the corresponding External IRQ Flag bit is set in SIUL2 DMA/Interrupt Status Flag Register 0 (SIUL2_DISR0). The type of request enabled is determined by the corresponding DMA/Interrupt Request Select bit in SIUL2 DMA/Interrupt Request Select Register 0 (SIUL2_DIRSR0). This register enables the interrupt messaging to the interrupt controller.

Offset: 0x0018 Access: User read/write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	EIRE15	EIRE14	EIRE13	EIRE12	EIRE11	EIRE10	EIRE9	EIRE8	EIRE7	EIRE6	EIRE5	EIRE4	EIRE3	EIRE2	EIRE1	EIRE0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 169. SIUL2 DMA/Interrupt Request Enable Register 0 (SIUL2_DIRER0)

Table 272. SIUL2_DIRER0 field descriptions

Field	Description
16:31 EIRE[15:0]	External Interrupt or DMA Request Enable x 0 Interrupt or DMA requests from the corresponding EIF[x] bit are disabled 1 Set EIF[x] bit causes either a DMA or an interrupt request depending on SIUL2_DIRSR

17.2.2.5 DMA/Interrupt Request Select Register 0 (SIUL2_DIRSR0)

The DIRSR selects between the DMA or interrupt request. If the corresponding bits are set in SIUL2 DMA/Interrupt Status Flag Register 0 (SIUL2_DISR0) and SIUL2 DMA/Interrupt Request Enable Register 0 (SIUL2_DIRER0), then the DMA/Interrupt Request Select bit determines whether DMA or an interrupt request is asserted. EIRQ are the external interrupt package pins on the device.

Offset: 0x0020 Access: User read/write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	DIRS15	DIRS14	DIRS13	DIRS12	DIRS11	DIRS10	DIRS9	DIRS8	DIRS7	DIRS6	DIRS5	DIRS4	DIRS3	DIRS2	DIRS1	DIRS0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 170. SIUL2 DMA/Interrupt Request Select Register 0 (SIUL2_DIRSR0)

Table 273. SIUL2_DIRSR0 field descriptions

Field	Description
16:31 DIRS[15:0]	DMA/Interrupt Request Select Register Selects between DMA request or external interrupt request when an edge-triggered event occurs on the corresponding pin. 0 Interrupt request is selected 1 DMA request is selected

17.2.2.6 Interrupt Rising-Edge Event Enable Register 0 (SIUL2_IREER0)

This register is used to enable the rising-edge triggered events on the corresponding interrupt pads.

Offset: 0x0028 Access: User read/write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	IREE15	IREE14	IREE13	IREE12	IREE11	IREE10	IREE9	IREE8	IREE7	IREE6	IREE5	IREE4	IREE3	IREE2	IREE1	IREE0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 171. SIUL2 Interrupt Rising-Edge Event Enable Register 0 (SIUL2_IREER0)

Table 274. SIUL2_IREER0 field descriptions

Field	Description
16:31 IREE[15:0]	Enable rising-edge events to cause the EIF[x] bit to be set. 0 Rising-edge event is disabled 1 Rising-edge event is enabled

17.2.2.7 Interrupt Falling-Edge Event Enable Register 0 (SIUL2_IFEER0)

This register is used to enable falling-edge triggered events on the corresponding interrupt pads.

Note: If both IREE bit and IFEE bit are cleared for the same interrupt source, the interrupt status flag for the corresponding external interrupt will never be set.

Offset: 0x0030 Access: User read/write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	IFEE15	IFEE14	IFEE13	IFEE12	IFEE11	IFEE10	IFEE9	IFEE8	IFEE7	IFEE6	IFEE5	IFEE4	IFEE3	IFEE2	IFEE1	IFEE0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 172. SIUL2 Interrupt Falling-Edge Event Enable Register 0 (SIUL2_IFEER0)

Table 275. SIUL2_IFEER0 field descriptions

Field	Description
16:31 IFEE[15:0]	Enable falling-edge events to cause the EIF[x] bit to be set. 0 Falling-edge event is disabled 1 Falling-edge event is enabled

17.2.2.8 Interrupt Filter Enable Register 0 (SIUL2_IFER0)

This register is used to enable a digital filter counter on the corresponding interrupt pads to filter out glitches on the inputs.

Offset: 0x0038 Access: User read/write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	IFE15	IFE14	IFE13	IFE12	IFE11	IFE10	IFE9	IFE8	IFE7	IFE6	IFE5	IFE4	IFE3	IFE2	IFE1	IFE0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 173. SIUL2 Interrupt Filter Enable Register 0 (SIUL2_IFER0)

Table 276. SIUL2_IFER0 field descriptions

Field	Description
16:31 IFE[15:0]	Enable digital glitch filter on the interrupt pad input. 0 Filter is disabled 1 Filter is enabled

17.2.2.9 Interrupt Filter Maximum Counter Register (SIUL2_IFMCR*n*)

These registers are used to configure the filter counter associated with each digital glitch filter.

Offset: 0x0040 + n * 0x4 (n = 0 to 31) Access: User read/write

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

Figure 174. SIUL2 Interrupt Filter Maximum Counter Register (SIUL2_IFMCR*n*)

Table 277. SIUL2_IFMCR*n* field descriptions

Field	Description
28:31 MAXCNT	Maximum Interrupt Filter Counter setting (MAXCNTx) Filter Period = TCK*MAXCNTx + n*TCK where (n = -1 to 3) MAXCNT can be 0 to 15 (for MAXCNT<3 filter will behave as all PASS filter) TCK is the Prescaled filter clock period, which is the IRC clock prescaled to IFCP value, specified in SIUL2_IFCPR T(IRC): Basic filter clock period 62.5 ns (f = 16 MHz) Note that filter delay is 2 system clock cycles more than filter period. In general, Filter Period and Filter delay are not integer multiple of TCK because input to Filter is asynchronous whereas output is synchronous with respect to the TCK. T(DELAY) = (1 to 5) * TCK + MAXCOUNT * TCK. The SW is to expect the worst filter delay value

17.2.2.10 Interrupt Filter Clock Prescaler Register (SIUL2_IFCPR)

This register configures a clock prescaler that selects the clock for all digital filters. The prescaler is applied to the input clock to the SIUL2, which is the system PBRIDGE clock counter in the SIUL2. Refer to *Chapter 27: Clocking* for information on clocking.

Offset: 0x00C0 Access: User read/write

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

Figure 175. SIUL2 Interrupt Filter Clock Prescaler Register (SIUL2_IFCPR)

Table 278. SIUL2_IFCPR field descriptions

Field	Description
28:31 IFCP	Interrupt Filter Clock Prescaler setting Prescaled Filter Clock Period = T(IRC) x (IFCP + 1) T(IRC) is the internal oscillator period. IFCP can be 0 to 15.

17.2.2.11 Soc Configuration Register 0 (SIUL2_SCR0)

The ETHERNET_0_MODE bit is used to select between the Media Independent Interface (MII) and the Reduced Media Independent Interface (RMII) mode for the Ethernet.

MASK_OPCn is used to prevent the SSWU OPC channel n to drive the pad during standby.

PADn_MUX_SEL is used to maintain the pad value at standby exit (PADn_MUX_SEL = 0)

Note: PADn_MUX_SEL must be set to 1 to be able to change the pads value and direction after stand-by while system is in the RUN mode.

Offset: 0x0100 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	ETHERNET_0_MODE	ETHERNET_1_MODE	LVDS0_OPT2	LVDS1_OPT2	0	0	0	0	ESR0_ASSERT	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	PAD83_OPC_MASK	PAD84_OPC_MASK	PAD111_OPC_MASK	PAD95_OPC_MASK	PAD92_OPC_MASK	PAD89_OPC_MASK	PAD27_OPC_MASK	PAD58_OPC_MASK	PAD83_MUX_SEL	PAD84_MUX_SEL	PAD111_MUX_SEL	PAD95_MUX_SEL	PAD92_MUX_SEL	PAD89_MUX_SEL	PAD27_MUX_SEL	PAD58_MUX_SEL
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 176. Soc Configuration Register 0 (SIUL2_SCR0)

Table 279. SIUL2_SCR0 field descriptions

Field	Description
0 ETHERNET_0_MODE	Ethernet Controller mode setting 0 RMII is the default state. 1 MII is selected. **Note:** MII/RMII pad configuration is done independently in the SIUL2 MSCR registers.
1 ETHERNET_1_MODE	Ethernet Controller mode setting 0 RMII is the default state. 1 MII is selected. **Note:** MII/RMII pad configuration is done independently in the SIUL2 MSCR registers.
2 LVDS0_OPT2	LVDS 0 Termination register is enabled for LVDS RX when pad 119/120 ILS[1:0] = 10 is set. 1 Termination register is disabled for LVDS RX when pad 119/120 ILS[1:0] = 10 is set.
3 LVDS1_OPT2	LVDS 0 Termination register is enabled for LVDS RX when pad 142/143 ILS[1:0] = 10 is set. 1 Termination register is disabled for LVDS RX when pad 142/143 ILS[1:0] = 10 is set.

Table 279. SIUL2_SCR0 field descriptions (continued)

Field	Description
8 ESR0_ASSERT	Active high signal enabling to force ESR0 as a strong pull-down 0 ESR0 status is controlled by RGM 1 ESR0 status is forced low if ESR0 is configured as output only (associated DCF record DCF.UTEST_MISCELLANEOUS[ESR0_Gate] is low)
16 PAD83_OPC_MASK	PAD[83] OPC functionality 0 channel OPC drives pad during stand-by mode. 1 channel OPC does not drive pad during stand-by mode.
17 PAD84_OPC_MASK	PAD[84] OPC functionality 0 channel OPC drives pad during stand-by mode. 1 channel OPC does not drive pad during stand-by mode.
18 PAD111_OPC_MASK	PAD[111] OPC functionality 0 channel OPC drives pad during stand-by mode. 1 channel OPC does not drive pad during stand-by mode.
19 PAD95_OPC_MASK	PAD[95] OPC functionality 0 channel OPC drives pad during stand-by mode. 1 channel OPC does not drive pad during stand-by mode.
20 PAD92_OPC_MASK	PAD[92] OPC functionality 0 channel OPC drives pad during stand-by mode. 1 channel OPC does not drive pad during stand-by mode.
21 PAD89_OPC_MASK	PAD[89] OPC functionality 0 channel OPC drives pad during stand-by mode. 1 channel OPC does not drive pad during stand-by mode.
22 PAD27_OPC_MASK	PAD[27] OPC functionality 0 channel OPC drives pad during stand-by mode. 1 channel OPC does not drive pad during stand-by mode.
23 PAD58_OPC_MASK	PAD[58] OPC functionality 0 channel OPC drives pad during stand-by mode. 1 channel OPC does not drive pad during stand-by mode.
24 PAD83_MUX_SEL	PAD[83] multiplexer select 0 on leaving standby, this pin is set to the value that was latched on this pin when entering standby mode 1 on leaving standby, this pin is set to a new value
25 PAD84_MUX_SEL	PAD[84] multiplexer select 0 on leaving standby, this pin is set to the value that was latched on this pin when entering standby mode 1 on leaving standby, this pin is set to a new value
26 PAD111_MUX_SEL	PAD[111] multiplexer select 0 on leaving standby, this pin is set to the value that was latched on this pin when entering standby mode 1 on leaving standby, this pin is set to a new value
27 PAD95_MUX_SEL	PAD[95] multiplexer select 0 on leaving standby, this pin is set to the value that was latched on this pin when entering standby mode 1 on leaving standby, this pin is set to a new value

Table 279. SIUL2_SCR0 field descriptions (continued)

Field	Description
28 PAD92_MUX_SEL	PAD[92] multiplexer select 0 on leaving standby, this pin is set to the value that was latched on this pin when entering standby mode 1 on leaving standby, this pin is set to a new value
29 PAD89_MUX_SEL	PAD[89] multiplexer select 0 on leaving standby, this pin is set to the value that was latched on this pin when entering standby mode 1 on leaving standby, this pin is set to a new value
30 PAD27_MUX_SEL	PAD[27] multiplexer select 0 on leaving standby, this pin is set to the value that was latched on this pin when entering standby mode 1 on leaving standby, this pin is set to a new value
31 PAD58_MUX_SEL	PAD[58] multiplexer select 0 on leaving standby, this pin is set to the value that was latched on this pin when entering standby mode 1 on leaving standby, this pin is set to a new value

17.2.2.12 I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO*n*)

Refer to *Section 17.2.3: Multiplexed Signal Configuration Registers* for an introduction to the MSCRs.

This section defines the I/O pin MSCRs. These registers select the output function on the associated I/O pin. The I/O MSCRs control the drive strength, output drive circuit, pull up/down, input buffer enable, input hysteresis, input level selection, and safe-mode operation of the associated I/O pin.

Note: Not all pads have all bits, such as only some pads have hysteresis. Refer to the I/O Signal Description and Input Multiplexing Tables (Microsoft Excel file) attached to the IO_Definition document.

Note that when an I/O pin is configured as a single-ended input, the input buffer must be enabled for the pin using the MSCR[IBE] bit. This enables the input to all input destinations connected to the pin, including the GPIO input. In this way, the GPIO can be read at any time, regardless of the pin function. For IP blocks that have multiple input sources, the input source is selected in the multiplexed input selection MSCRs, and the associated I/O pin MSCR for the selected input must have the MSCR[IBE] bit enabled.

All bits and fields in the I/O MSCRs are applicable for all GPIO ports on the device. This includes all ports on the device with digital output function. For ADC input ports with digital input only (no output), the OERC, ODC, SMC, INV and SSS bits/fields do not apply. The SSS is always 8b0 for input only ports, enabling the general purpose input. The analog input path is enabled in the ADC logic when the channel is selected for conversion.

The MSCR reset state is detailed in the I/O Signal Table of the SPC58xEx/SPC58xGx pinout Microsoft Excel file attached to the IO_Definition document.

Offset: 0x0240 + n * 0x4 (n = 0 to 511) Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	OERC		0	ODC			SMC	0	ILS		IBE	0	WPDE	WPUE
W																
Reset	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	INV	0	0	0	0	0	0	0	SSS							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 177. I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IO*n*)

Table 280. SIUL2_MSCR_IO*n* field description

Field	Description
2:3 OERC	Output Edge Rate Control Specifies the output impedance, drive strength and slew rate of the associated pin. Refer to output Impedance columns in the I/O Signal Description Table for the applicable edge rates for each I/O pin on the device. The I/O Signal Description and Input Multiplexing Tables (Microsoft Excel file) are attached to the IO_Definition document. Refer to the device Data Sheet for the electrical characteristics of the weak, medium, strong, and very strong pad types **Note:** OERC[2] programming is required for forward compatibility. 00 Weak drive 01 Medium drive 10 Strong drive 11 Very Strong drive
5:7 ODC	Output Drive Control Specifies the type of output drive control for the associated pin. 000 Output buffer disabled 001 Open-drain 010 Push-pull 011 Open-source 100 Microsecond Channel LVDS 101 LFAST LVDS 110 Reserved 111 Reserved **Note:** In order to configure the LVDS-Input functionality of the pins supporting it, set ODC=000 (Output buffer disabled)
8 SMC	Safe Mode Control Specifies whether the chip disables the pin's output buffer when the chip enters Safe mode. Resets to 0 except for PB[11] ERROR0 pin which resets to 1. 0 Disable (the output buffer returns to its previous state when the chip leaves Safe mode) 1 Does not disable

Table 280. SIUL2_MSCR_IO*n* field description (continued)

Field	Description
10:11 ILS	Input Level Selection Specifies the logic family for the associated pin, which determines its logic switching levels. Refer to the device Data Sheet for the electrical characteristics of the I/O pad input buffer types. – in case of pin with MSCR[10] bit configurable, following logic levels are valid: 00 Automotive for SPC58xEx, Reserved for SPC58xGx 01 TTL 10 LVDS 11 CMOS **Note:** In order to configure the LVDS-Input functionality of the pins supporting it, set – in case of pin with MSCR[10] bit not configurable, following logic levels are valid: MSCR[10] – not implemented, don't care MSCR[11] = 0: CMOS for SPC58xGx, Auto for SPC58xEx MSCR[11] = 1: TTL for SPC58xGx/SPC58xEx Refer to the IO_Definition excel file to check if MSCR[10] bit of a specific pin is configurable or not
12 IBE	Input Buffer Enable Enables the associated pin's input buffer. 0 Disabled 1 Enabled
14 WPDE	Weak Pulldown Enable Used only when the associated destination is a chip pin. Enables the associated pin's weak pulldown resistor. If pin has analog input-only functionality: when WPDE and WPUE are both enabled that pin is configured with Strong pulldown. On all other pins (including analog bidirectional) this configuration means neither a pullup nor a pulldown. 0 Disabled 1 Enabled
15 WPUE	Weak Pullup Enable Used only when the associated destination is a chip pin. Enables the associated pin's weak pullup resistor. If pin has analog input-only functionality: when WPDE and WPUE are both enabled that pin is configured with Strong pulldown. On all other pins (including analog bidirectional) this configuration means neither a pullup nor a pulldown. 0 Disabled 1 Enabled

Table 280. SIUL2_MSCR_IO*n* field description (continued)

Field	Description
16 INV	Invert The output selected by the corresponding MSCR SSS field can be inverted before it is driven on the I/O pin with the INV bit. **Note:** Use of the INV bit is not supported when the ODC is configured for open-drain or open-source modes, and the output data will not be as expected for an inverted open-drain or open-source connection. 0 The output selected by the SSS field is not inverted before being driven to the pin 1 The output selected by the SSS field is inverted before being driven to the pin
24:31 SSS	Source Signal Select Selects which source signal is connected to the associated destination (chip pin or module port). For a chip pin, the source signals are outputs from module ports. For a module port, the source signals are either outputs from module ports or inputs from chip pins. The meaning of each value depends on the destination. Refer to *Section 17.2.3.1* and *Section 17.2.3.2*.

Table 281. I/O MSCR Reset State Exceptions

Port	Critical Function	SMC	WPDE	WPUE
PA[4]	—	0	0	1
PA[5]	JCOMP	1	1	0
PA[6]	TCK	1	1	0
PA[7]	TMS	1	0	1
PA[8]	TDI	0	0	1
PA[9]	TDO	0	0	1(1)
PA[14]	—	0	0	1
PB[11]	ERROR	1	0	0
PC[10:15]	—	0	0	1
PD[6:7]	—	0	0	1
PE[12]	—	0	0	1
PF[13]	—	0	0	1
PI[14:15]	—	0	0	1
PK[12:15]	—	0	0	1
PM[0:8]	—	0	0	1

1. When JTAG is enabled (JCOMP asserted) during reset, the TDO pin is pulled up during reset, but Hi-Z after. If JTAG is disabled, the TDO pin defaults to input/pull-up during and after reset.

17.2.2.13 Multiplexed Signal Configuration Registers for Multiplexed Input Selection (SIUL2_MSCR_MUX*n*)

Refer to *Section 17.2.3: Multiplexed Signal Configuration Registers* for an introduction to the MSCRs.

This section defines the multiplexed input selection MSCRs. These registers select the input source for IP blocks on the device that have inputs from multiple sources, both other IP blocks and I/O pins. It is possible to configure the pad to be in input without having to connect it externally. This is possible, either by enabling internal pull-up/pull-down or disabling the input buffer (but in this case data is seen as 0 internally).

Offset: 0x240 + *n* * 0x4 (*n* = 512 to 1023) Access: User read/write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	INV	0	0	0	0	0	0	0	SSS							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 178. Multiplexed Signal Configuration Registers for Multiplexed Input Selection (SIUL2_MSCR_MUX*n*)

Table 282. SIUL2_MSCR_MUX*n* field descriptions

Field	Description
16 INV	Invert The input source signal selected by the corresponding SSS field for this MSCR can be inverted before it connects to the destination input with the INV bit. 0 SSS selected input signal is connected directly to the destination input 1 SSS selected input signal is inverted before it is connected to the destination input
24:31 SSS	Source Signal Select Selects which source signal is connected to the associated destination (chip pin or module port). For a chip pin, the source signals are outputs from module ports. For a module port, the source signals are either outputs from module ports or inputs from chip pins. The meaning of each value depends on the destination. Refer to *Section 17.2.3.1* and *Section 17.2.3.2*.

17.2.2.14 GPIO Pad Data Output Registers (SIUL2_GPDO*n*)

These registers can be used to set or clear a single GPIO pad with a byte access. For mapping of ports to GPDO registers, refer to the I/O Signal Description and Input Multiplexing Tables (Microsoft Excel file) attached to the IO_Definition document. Note that the GPDO registers do not apply for ports PA[5], PA[6] and PA[7]. The PA ports are JTAG pins with no GPIO. Writes to the GPDO register corresponding to these ports have no effect.

Note: For the exact number of registers, refer to the SPC58xEx/SPC58xGx pinout Microsoft Excel file attached to the IO_Definition document.

Offset: 0x1300 + n * 0x1 (n = 0 to 511) Access: User read/write

	0	1	2	3	4	5	6	7
R	0	0	0	0	0	0	0	PDO
W								
Reset	0	0	0	0	0	0	0	0

Figure 179. SIUL2 GPIO Pad Data Output register (SIUL2_GPDO*n*)

Table 283. SIUL2_GPDO*n* field descriptions

Field	Description
7 PDO	Pad Data Out This bit stores the data to be driven out on the external GPIO pad controlled by this register. 0 Logic low value is driven on the corresponding GPIO pad when the pad is configured as an output 1 Logic high value is driven on the corresponding GPIO pad when the pad is configured as an output

17.2.2.15 GPIO Pad Data Input Registers (SIUL2_GPDI*n*)

These registers can be used to read the GPIO pad data with a byte access. For mapping of ports to GPDI registers, refer to the I/O Signal Description and Input Multiplexing Tables (Microsoft Excel file) attached to the IO_Definition document. Note that the GPDI registers do not apply for ports PA[5], PA[6] and PA[7]. These three PA ports are JTAG pins with no GPIO. The GPDI values corresponding to these ports always read a value of zero.

Note: For the exact number of registers, refer to the SPC58xEx/SPC58xGx pinout Microsoft Excel file attached to the IO_Definition document.

Offset: 0x1500 + n * 0x1 (n = 0 to 511) Access: User read/write

	0	1	2	3	4	5	6	7
R	0	0	0	0	0	0	0	PDI
W								
Reset	0	0	0	0	0	0	0	0

Figure 180. SIUL2 GPIO Pad Data Input register (SIUL2_GPDI*n*)

Table 284. SIUL2_GPDI*n* field descriptions

Field	Description
7 PDI	Pad Data In This bit stores the value of the external GPIO pad associated with this register. 0 The value of the data in signal for the corresponding GPIO pad is logic low 1 The value of the data in signal for the corresponding GPIO pad is logic high

17.2.2.16 Parallel GPIO Pad Data Out Register (SIUL2_PGPDO*n*)

These registers are used to set or clear the respective pads of the device. Parallel port registers for input (PGPDI) and output (PGPDO) are provided to allow a complete port to be

written or read in one operation, dependent on the individual pad configuration. The difference between PGPDO and GPDO is that PGPDO registers can be used to set the values of all output pins assigned to a device port with a single 16-bit register write vs. the GPDO registers, which are used to set the value on a specific pin with a byte write. Note that the PGPDO registers do not apply for ports PA[5], PA[6] and PA[7]. These PA ports are JTAG pins with no GPIO. Writes to the PGPDO register corresponding to these ports have no effect.

Note: The SIUL2_PGPDO registers access the same physical resources as the SIUL2_PDO and SUIL2_MPGPDO address locations. Some examples of the mapping:

- PPDO[0][0] = PDO[0]
- PPDO[2][0] = PDO[32]
- PPDO[31][15] = PDO[511]

Offset: 0x1700 + n*0x2 (n=0 to 12) Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	PPDO															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 181. SIUL2 Parallel GPIO Pad Data Out Register (SIUL2_PGPDO)

Table 285. SIUL2_PGPDO*n* field descriptions

Field	Description
0:15 PPDO	Parallel Pad Data Out Write or read the data register that stores the value to be driven on the pad in output mode. Access to this register location is coherent with access to the bit-wise GPIO Pad Data Output Registers (SIUL2_GPDO). The x and y bit indexes define which PPDO register bit is equivalent to which PDO register bit according to the following equation: PPDO[x][y] = PDO[(x*16)+y]

17.2.2.17 Parallel GPIO Pad Data In Register (SIUL2_PGPDI*n*)

These registers hold the synchronized input value from the pads. Parallel port registers for input (PGPDI) and output (PGPDO) are provided to allow a complete port to be written or read in one operation, dependent on the individual pad configuration. The difference between PGPDI and GPDI registers is that PGPDI registers can be used to read the values of all input pins assigned to a device port with a single 16-bit register read vs. the GPDI registers, which are used to read the value on a specific pin with a byte read. Note that the GPDI registers do not apply for ports PA[5], PA[6] and PA[7]. These ports are JTAG pins with no GPIO. The GPDI values corresponding to these ports always read a value of zero.

Offset: 0x1740 + n*0x2 (n=0 to 12) Access: User read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	PPDI															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 182. SIUL2 Parallel GPIO Pad Data In Register (SIUL2_PGPDI)

Table 286. SIUL2_PGPDI*n* field descriptions

Field	Description
0:15 PPDI	Parallel Pad Data In Reads the current pad value. Access to this register location is coherent with access to the bit-wise GPIO Pad Data Input Registers (SIUL2_GPDI). The x and y bit indexes define which PPDI register bit is equivalent to which PDI register bit according to the following equation: PPDI[x][y] = PDI[(x*16)+y]

17.2.2.18 Masked Parallel GPIO Pad Data Out Register (SIUL2_MPGPDO*n*)

This register can be used to selectively modify the pad values associated to PPDO[x][0:15]. The MPGPDO[x] register may only be accessed with 32-bit writes. 8-bit or 16-bit writes will not modify any bits in the register and cause a transfer error response by the module. Read access will return 0. Note that the MPGPDO registers do not apply for ports PA[5], PA[6] and PA[7]. These are JTAG pins with no GPIO. Writes to the MPGPDO register corresponding to these ports have no effect.

Offset: 0x1780 + n*0x4 (n=0 to 12) Access: User read/write

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

Figure 183. SIUL2 Masked Parallel GPIO Pad Data Out Register (SIUL2_MPGPDO)

Table 287. SIUL2_MPGPDO*n* field descriptions

Field	Description
0:15 MASK	Mask Field Each bit corresponds to one data bit in the MPPDO[x] register at the same bit location. 0 The associated bit value in the MPPDO[x] field is ignored 1 The associated bit value in the MPPDO[x] field is written
16:31 MPPDO	Masked Parallel Pad Data Out Write the data register that stores the value to be driven on the pad in output mode. Access to this register location is coherent with access to the bit-wise GPIO Pad Data Output Registers (PDO). The x and y bit indexes define which MPPDO register bit is equivalent to which PDO register bit according to the following equation: MPPDO[x][y] = PDO[(x * 16) + y]

17.2.3 Multiplexed Signal Configuration Registers

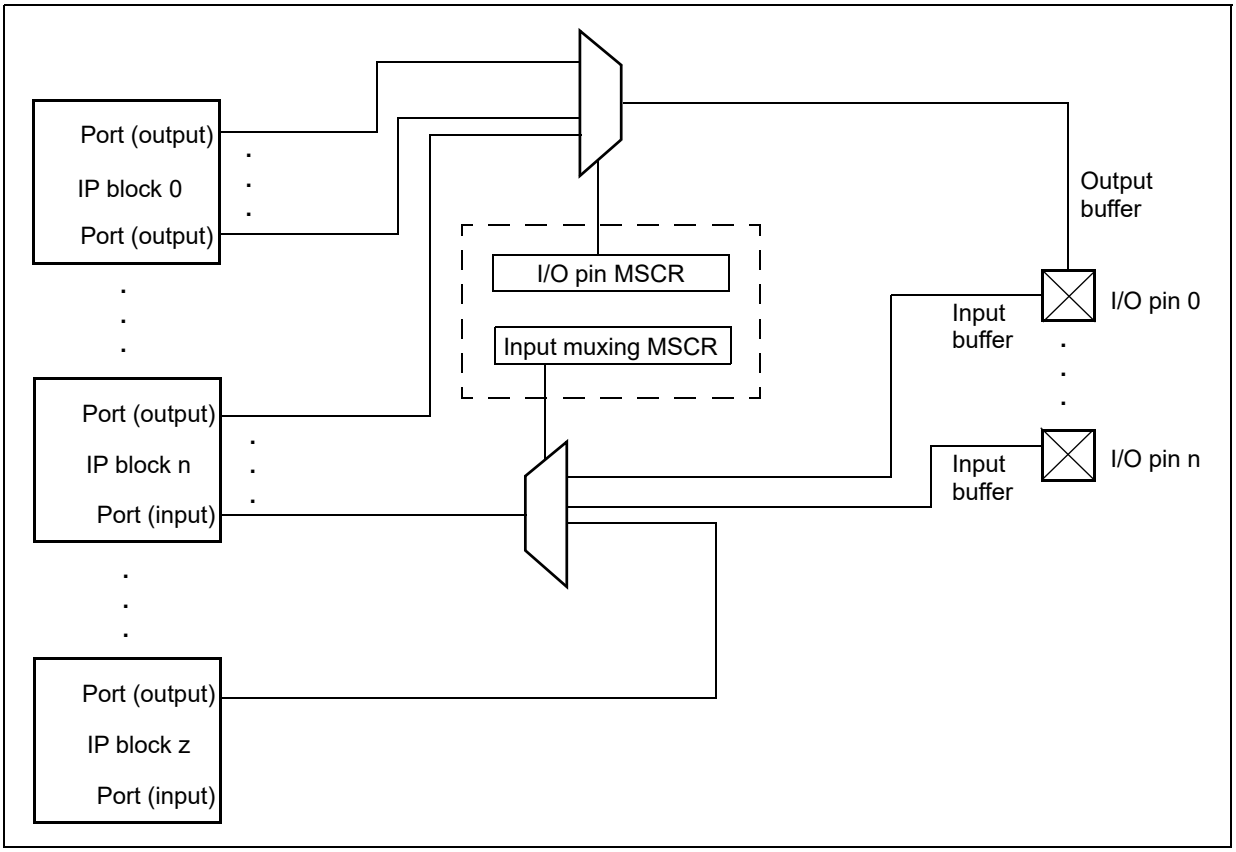
The MSCRs control the I/O pin function and electrical properties for each pin on the device, and are used to select the input signal for IP blocks on the device that have multiple input sources. There is a dedicated MSCR register for each I/O pin on the device. Each MSCR register is independent of the others, except in the case of an LVDS configuration, where two MSCR registers must be configured together in this mode. When LVDS is configured, CMOS pad is disabled (ibe and obe = 0) and when LVDS is not configured, (either of two MSCRs not configured for lvds) then the lvds pad is disabled (lvds_ibe and lvds_obe = 0).

The MSCRs are divided into two sets:

- I/O pin control registers, refer to *Section 17.2.2.12: I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IOn)*
- Multiplexed input selection registers, refer to *Section 17.2.2.13: Multiplexed Signal Configuration Registers for Multiplexed Input Selection (SIUL2_MSCR_MUXn)*

The signal data flow from IP blocks on the device to and from the I/O pins, and from IP block to IP block is given in *Figure 184*.

Figure 184. MSCR I/O pin and IP block port connectivity



17.2.3.1 Chip-pin MSCR assignments and related information

For device pin MSCR register definitions, refer to the SPC58xEx/SPC58xGx I/O Signal Description and Input Multiplexing Tables (Microsoft Excel file) attached to the IO_Definition document.

17.2.3.2 Module-port MSCR assignments and SSS values

For each internal module port that is or can be configured as an input and has more than one possible source. Refer to the SPC58xEx/SPC58xGx I/O Signal Description and Input Multiplexing Tables (Microsoft Excel file) attached to the IO_Definition document. The Input Multiplexing Table shows the assigned MSCR, and the SSS values and their corresponding source signals.

17.3 Functional description

17.3.1 General

This section provides a complete functional description of the SIUL2.

17.3.2 Pad control

The SIUL2 controls the electrical characteristics of the pads on the device through the MSCR registers. It provides a consistent interface for all pads, both on a by-port and a by-bit basis. The following describes the pad characteristics that can be supported by the SIUL2.

The required controls are:

- Slew rate control
  - This is needed to offer improved EMC performance.
  - Support for a fast and slow slew rate.
- Output Impedance Control
  - 4 output impedances, drive strength and slew rates supported.
- Internal weak pull capability
  - This is needed to offer flexibility in the device configuration and the elimination of external hardware in some cases.
  - The configuration of the pull on any pad should be independently controlled to be either pull-up, pull-down or no-pull enabled.
- Safe Mode behavior configuration
- Open drain/source enable
  - Needed to support different pads muxed (for example mux IIC, with non open drain pads).
- Pin Function Assignment
  - This setting defines which chip function has control over the output of the pad.

The setting of each pad out of reset is fixed per MCU, but can be configured individually. In this way it is possible to select special pull settings or peripheral pad ownership per design.

It is possible for the user software to configure each pad independently of all other pads on the device or other pads grouped within a single port. This allows different pad types to be grouped together in ports and allows the necessary flexibility needed for the pads individual operation. This is achieved by grouping all of the above functions into a single register for each pad on the device, and allows each pad to be configured with a single write to one register and allowing simplified duplication of software for each pad with indexed changes for each pad.

17.3.3 General purpose input or output pads (GPIO)

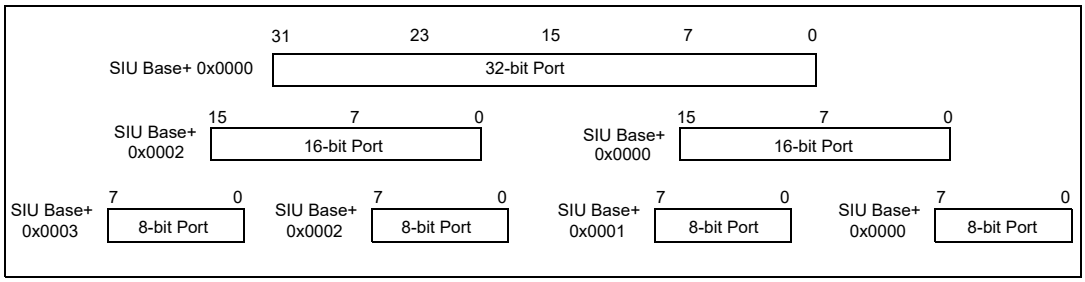
The SIUL2 allows each pad to be configured as either a General Purpose Input Output pad (GPIO), and as one or more alternate functions (input or output), the function of which is determined by the peripheral that will use the pad.

GPIO pads can also optionally be implemented without any alternate function.

The SIUL2 is designed to manage 202 GPIO pads (205 pads less 3 reserved for JTAG) organized as ports that can be accessed for data reads and writes as 32-bit, 16-bit or 8-bit.

As shown in *Figure 185*, all port accesses are identical with each read or write being performed only at a different location to access a different port width.

Figure 185. Data Port example arrangement showing configuration for different port width accesses



This implementation requires that the registers are arranged in such a way as to support this range of port widths without having to split reads or writes into multiple accesses.

The SIUL2 has separate data input and data output registers for all pads, allowing the possibility of reading back an input or output value of a pad directly. This supports the ability to validate what is present on the pad rather than simply confirming the value that was written to the data register by accessing the data input registers.

The data output registers support both read and write operations to be performed.

The data input registers support read access only.

When the pad is configured to use one of its alternate functions, the data input value reflect the respective value of the pad. If a write operation is performed to the data output register for a pad configured as an alternate function (non-GPIO), this write will not be reflected by the pad value until reconfigured to GPIO.

All general purpose pads are implemented as bidirectional.

Note: In case the bidirectional operation impacts some performances (for example ADC accuracy), or is not required for a specific pad function it is acceptable to limit the functionality of some pads to "Input Only".

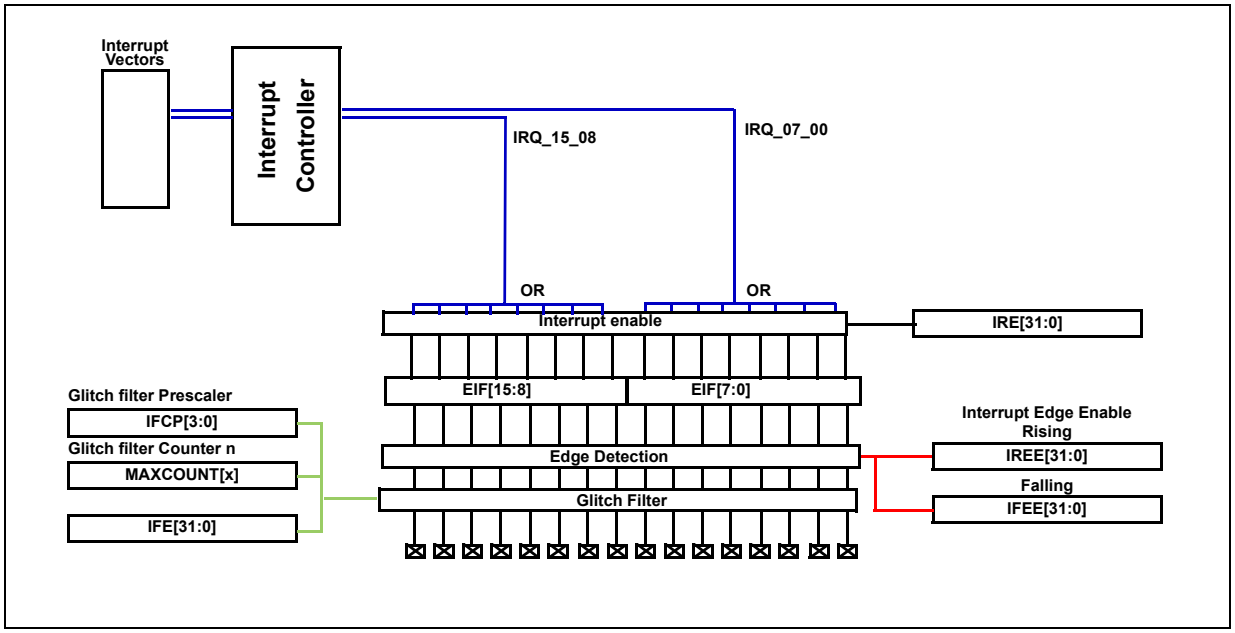
17.3.4 External interrupts/DMA requests (EIRQ pins)

The SIUL2 supports 16 external interrupts which are allocated to EIRQ pins on the device

The SIUL2 supports 2 interrupt vectors to the interrupt controller of the MCU. Each interrupt vector can support eight external interrupt sources from the device pads.

Refer to *Figure 186* for an overview of the external interrupt implementation.

Figure 186. External interrupt pad diagram



All the external interrupt pads within a single group (maximum 8 pads) have equal priority. It is the responsibility of the user software to search through the group of sources in the most appropriate way for their application.

The priority of the vectors used by the external interrupt pads is fixed based on the platform and the interrupt controller and its priority levels, but the allocation of pads to each group of interrupts can be independently configured by the MCU.

An MCU-specific number of external interrupt lines can have digital glitch filters applied to them. The supported range is 1 to 8. The glitch filters need a running internal oscillator clock to work. If no such clock is available, external interrupts will be effectively disabled when the glitch filter is enabled on an interrupt line.

17.3.4.1 External interrupt initialization

When an external interrupt pin is first enabled, it is possible to get a false interrupt flag. To prevent a false interrupt request, during pin interrupt initialization, the user must do the following:

- Mask interrupts by clearing the EIREn bits in DIRER0.
- Select the pin polarity by setting the appropriate IREEn bits in IREER0 and the appropriate IFEEn bits in IFEER0 as desired.
- Configure the appropriate bits in the MSCR[0–511] register for the external interrupt pin(s) desired as follows:
  - Clear the ODC bits to do disable output.
  - Set the IBE bit to enable the pin’s input buffer.
  - If using the internal weak pullup/pulldown, configure the appropriate WPUE and WPDE bits.

Note: MSCR_SSS bits do not need to be changed for either MSCR[0:511] or MSCR[512:1023] since the external interrupt pin input is directly connected to the SIUL for EIRQ pins.

External interrupt pins should never be configured as outputs (MSCR_ODC bits are not zeros) when external interrupt inputs are desired since false interrupts could be detected (such as from a GPIO configuration).

- Select the request desired between DMA or Interrupt by writing the appropriate DIRSn bits in DIRSR0.
- Select the desired glitch filter setup for the pins by writing the following:
  - Write the Filter Counter setting to the desired value by writing the MAXCNT[3:0] bits in the IFMCn register for the respective external interrupt that is being used.
  - Set the Filter Clock Prescaler setting from 0 to 15 to the desired value by writing the IFCP[3:0] bits in the IFCPR register.
  - Then enable the glitch filter for the desired external interrupt pins by setting the appropriate IFEn bits in IFER0.

Note: The glitch filter clock runs on the internal RC oscillator clock whereas the SIUL logic runs on the PBRIDGEx clock. The IRC clock is at a fixed frequency of 16 MHz. The PBRIDGEx clock can on higher clock (refer to the Clocking chapter)

The glitch filter is enabled on IRC clock, so make sure at least one IRC clock occurs after enabling it before trying to detect glitchless events.

- Write to EIFn bits in DISR0 to as desired to clear any flags (for DMA it is self clearing).
- Enable the interrupt pins by setting the appropriate EIREn bits in DIREER0.

17.3.4.2 External interrupt management

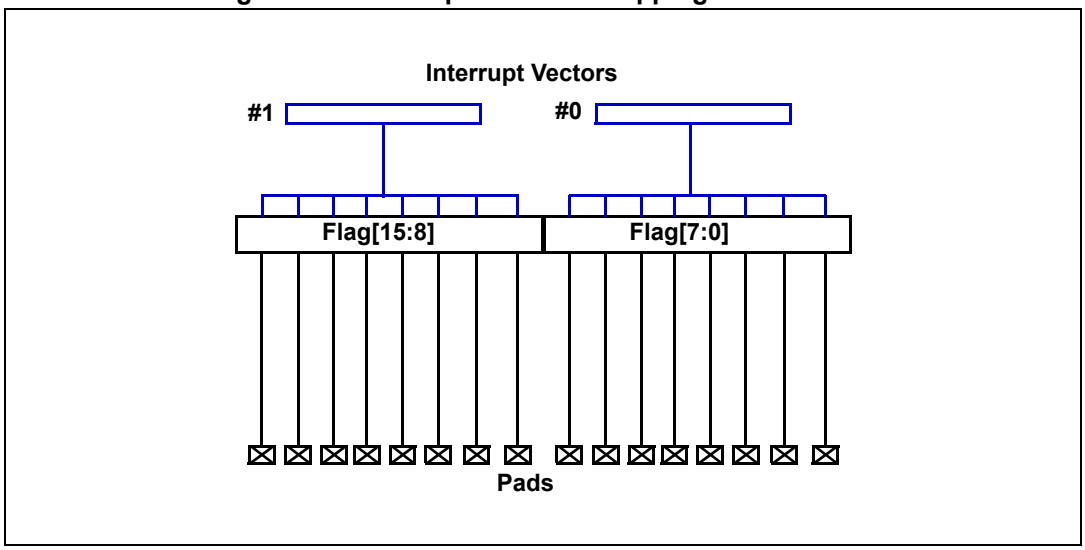
Each interrupt can be enabled or disabled independently. This can be performed using a single rolled up register (SIUL2_DIRER0). A pad defined as an external interrupt can be configured by the user to recognize interrupts with an active rising edge, an active falling edge or both edges being active. A setting of having both edge events disabled is reserved and should not be configured.

The active IRQ edge is controlled by the users through the configuration of the registers SIUL2_IREER and SIUL2_IFEER.

Each external interrupt supports an individual flag which is held in the Flag register (DISR0). This register is a clear-by-write-1 register type, preventing inadvertent overwriting of other flags in the same register.

The SPC58xEx/SPC58xGx device supports 2 SIUL2 interrupt vectors aggregating a total of 16 external interrupt sources as shown in *Figure 187*.

Figure 187. Interrupt to vector mapping at MCU level



17.3.4.3 External interrupt request

The EIRQ input pins on the device are sources for interrupt or DMA requests. The 16 EIRQ pins for the device map to 2 interrupt requests. The mapping of interrupt requests and EIRQ pins is given in *Table 288*.

Table 288. Interrupt source mapping to SIUL2 interrupt request output

Interrupt requests	EIRQ pins
0	EIRQ0 | EIRQ1 | EIRQ2 | EIRQ3 | EIRQ4 | EIRQ5 | EIRQ6 | EIRQ7
1	EIRQ8 | EIRQ9 | EIRQ10 | EIRQ11 | EIRQ12 | EIRQ13 | EIRQ14 | EIRQ15

17.3.4.4 Interrupt Vector

The 16 EIRQ pins on the device map to an Interrupt Vector in the Interrupt Controller.

Note: Refer to INTC configuration section in the Chapter Device Configuration for the Interrupt Vector table.

17.3.4.5 DMA requests

The 16 EIRQ pins on the device map to an independent DMA request channel in the DMA controller.

Note: Refer to the DMA controller configuration section in the Chapter Device Configuration for DMA request mapping table.

18 Crossbar switch (XBAR)

18.1 Introduction

This chapter provides information on the layout, configuration, and programming of the crossbar switch. The crossbar switch connects bus masters and bus slaves using a hardware interconnect matrix. This structure allows all bus masters to access different bus slaves simultaneously with no interference while providing arbitration among the bus masters when they access the same slave. A variety of bus arbitration methods and attributes may be programmed on a slave-by-slave basis.

18.2 Features

The crossbar switch includes these distinctive features:

- Symmetric crossbar bus switch implementation
  - Concurrent accesses from different masters to different slaves
  - Configurable slave arbitration attributes on a slave-by-slave basis
- 64-bit datapath width
- Support for 8-, 16-, 32-, and 64-bit single transfers
- Support for a variety of 4-, 8-, and 16-beat burst transfers including a 4-beat, 64-bit burst for cache line accesses
- Operation at one-to-one clock frequency with the bus masters
- Support for low-power park mode

18.3 Memory map and register definition

Each slave port of the crossbar switch contains configuration registers. Read and write transfers of the configuration registers require two system bus clock cycles. The registers can only be read from and written to in supervisor mode. Additionally, these registers can only be read from or written to by 32-bit accesses.

A bus error response is returned if an unimplemented location is accessed within the crossbar switch.

The slave registers also feature a bit that, when set, prevents the registers from being written. The registers remain readable, but future write attempts have no effect on the registers and are terminated with a bus error response to the master initiating the write. The core, for example, takes a data storage interrupt.

Note: This section shows the registers for all eight master and slave ports. If a master or slave is not used on this particular device, then unexpected results occur when writing to its registers. Refer to Section 7.3.3: Crossbar switch in Chapter 7: Device configuration for the exact master/slave assignments for your device. Additionally, all references to the crossbar switch registers are based on the physical port connections, not the logical port numbers.

Table 289. XBAR memory map

Address offset (hex)	Register name	Location
SLAVE channel *n* (*n* = 0 to 7)		
0x0000 + *n**0x100	XBAR Priority Register Slave *n* (XBAR_PRS*n*)	*Section 18.3.1*
0x0010 + *n**0x100	XBAR Control Register *n* (XBAR_CRS*n*)	*Section 18.3.2*
MASTER channel *n* (*n* = 0 to 7)		
0x0800 + *n**0x100	XBAR Master General Purpose Control Register n (XBAR_MGPCRn)	*Section 18.3.3*

18.3.1 XBAR Priority Registers Slave (XBAR_PRS*n*)

The priority registers (XBAR_PRS*n*) set the priority of each master port on a per-slave-port basis and reside in each slave port. The priority register can be accessed only with 32-bit accesses. The XBAR_PRS*n* register is read-only; attempts to write to it have no effect on it and result in a bus-error response to the master initiating the write.

Refer to *Section 7.3.3: Crossbar switch* in *Chapter 7: Device configuration* for XBAR_PRS*n* priority values.

Offset: 0x0000 + *n**0x100 (*n* = 0 to 7) Access: Supervisor read-only

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	M7			0	M6			0	M5			0	M4		
W																
Reset	0	0/1(1)	0/1(1)	0/1(1)	0	0/1(1)	0/1(1)	0/1(1)	0	0/1(1)	0/1(1)	0/1(1)	0	0/1(1)	0/1(1)	0/1(1)

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	M3			0	M2			0	M1			0	M0		
W																
Reset	0	0/1(1)	0/1(1)	0/1(1)	0	0/1(1)	0/1(1)	0/1(1)	0	0/1(1)	0/1(1)	0/1(1)	0	0/1(1)	0/1(1)	0/1(1)

1. Refer to *Section 7.3.3.2.3: PRS register reset values* of *Chapter 7: Device configuration*.

Figure 188. Priority Registers Slave (XBAR_PRS*n*)

Table 290. XBAR_PRS*n* field descriptions

Field	Description
1:3 M7	Master 7 priority Sets the arbitration priority for this port on the associated slave port: 000 This master has level 1 (highest) priority when accessing the slave port. 001 This master has level 2 priority when accessing the slave port. 010 This master has level 3 priority when accessing the slave port. 011 This master has level 4 priority when accessing the slave port. 100 This master has level 5 priority when accessing the slave port. 101 This master has level 6 priority when accessing the slave port. 110 This master has level 7 priority when accessing the slave port. 111 This master has level 8 (lowest) priority when accessing the slave port.
5:7 M6	Master 6 priority Sets the arbitration priority for this port on the associated slave port: 000 This master has level 1 (highest) priority when accessing the slave port. 001 This master has level 2 priority when accessing the slave port. 010 This master has level 3 priority when accessing the slave port. 011 This master has level 4 priority when accessing the slave port. 100 This master has level 5 priority when accessing the slave port. 101 This master has level 6 priority when accessing the slave port. 110 This master has level 7 priority when accessing the slave port. 111 This master has level 8 (lowest) priority when accessing the slave port.
9:11 M5	Master 5 priority Sets the arbitration priority for this port on the associated slave port: 000 This master has level 1 (highest) priority when accessing the slave port. 001 This master has level 2 priority when accessing the slave port. 010 This master has level 3 priority when accessing the slave port. 011 This master has level 4 priority when accessing the slave port. 100 This master has level 5 priority when accessing the slave port. 101 This master has level 6 priority when accessing the slave port. 110 This master has level 7 priority when accessing the slave port. 111 This master has level 8 (lowest) priority when accessing the slave port.
13:15 M4	Master 4 priority Sets the arbitration priority for this port on the associated slave port: 000 This master has level 1 (highest) priority when accessing the slave port. 001 This master has level 2 priority when accessing the slave port. 010 This master has level 3 priority when accessing the slave port. 011 This master has level 4 priority when accessing the slave port. 100 This master has level 5 priority when accessing the slave port. 101 This master has level 6 priority when accessing the slave port. 110 This master has level 7 priority when accessing the slave port. 111 This master has level 8 (lowest) priority when accessing the slave port.

Table 290. XBAR_PRS*n* field descriptions (continued)

Field	Description
17:19 M3	Master 3 priority Sets the arbitration priority for this port on the associated slave port: 000 This master has level 1 (highest) priority when accessing the slave port. 001 This master has level 2 priority when accessing the slave port. 010 This master has level 3 priority when accessing the slave port. 011 This master has level 4 priority when accessing the slave port. 100 This master has level 5 priority when accessing the slave port. 101 This master has level 6 priority when accessing the slave port. 110 This master has level 7 priority when accessing the slave port. 111 This master has level 8 (lowest) priority when accessing the slave port.
21:23 M2	Master 2 priority Sets the arbitration priority for this port on the associated slave port: 000 This master has level 1 (highest) priority when accessing the slave port. 001 This master has level 2 priority when accessing the slave port. 010 This master has level 3 priority when accessing the slave port. 011 This master has level 4 priority when accessing the slave port. 100 This master has level 5 priority when accessing the slave port. 101 This master has level 6 priority when accessing the slave port. 110 This master has level 7 priority when accessing the slave port. 111 This master has level 8 (lowest) priority when accessing the slave port.
25:27 M1	Master 1 priority Sets the arbitration priority for this port on the associated slave port: 000 This master has level 1 (highest) priority when accessing the slave port. 001 This master has level 2 priority when accessing the slave port. 010 This master has level 3 priority when accessing the slave port. 011 This master has level 4 priority when accessing the slave port. 100 This master has level 5 priority when accessing the slave port. 101 This master has level 6 priority when accessing the slave port. 110 This master has level 7 priority when accessing the slave port. 111 This master has level 8 (lowest) priority when accessing the slave port.
29:31 M0	Master 0 priority Sets the arbitration priority for this port on the associated slave port: 000 This master has level 1 (highest) priority when accessing the slave port. 001 This master has level 2 priority when accessing the slave port. 010 This master has level 3 priority when accessing the slave port. 011 This master has level 4 priority when accessing the slave port. 100 This master has level 5 priority when accessing the slave port. 101 This master has level 6 priority when accessing the slave port. 110 This master has level 7 priority when accessing the slave port. 111 This master has level 8 (lowest) priority when accessing the slave port.

18.3.2 XBAR Control Register (XBAR_CRS*n*)

Offset: 0x0010 + *n**0x100 (*n* = 0 to 7) Access: Supervisor read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	RO	HRP	0	0	0	0	0	0	HPE7	HPE6	HPE5	HPE4	HPE3	HPE2	HPE1	HPE0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	ARB		0		PCTL		0	PARK		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 189. XBAR Control Register (XBAR_CRS*n*)

Table 291. XBAR_CRS*n* field descriptions

Field	Description
0 RO	Read only Forces both of the slave port's registers to be read-only. After set, only a hardware reset clears it. 0 The slave port's registers are writable. 1 The slave port's registers are read-only and cannot be written (attempted writes have no effect on the registers and result in a bus error response).
1 HRP	Halt Request Priority Determines if a request to halt the crossbar is treated as the highest priority request, or the lowest priority request. 0 The halt request has the highest priority for arbitration on this slave port. 1 The halt request has the lowest initial priority for arbitration on this slave port.
8:15 HPE[7:0]	High Priority Enable (HPEx) Determines if Master x is able to temporarily elevate its request to the slave to high priority status. This can help reduce the amount of time the high priority requesting master must wait to gain control of the slave. 0 High Priority requests from Master x are ignored and are treated the same as regular requests. 1 High Priority requests from Master x can temporarily elevate the priority level of the master's request to the slave.
22:23 ARB	Arbitration mode Selects the arbitration policy for the slave port. 00 Fixed priority 01 Round-robin (rotating) priority 10 Reserved 11 Reserved

Table 291. XBAR_CRS*n* field descriptions (continued)

Field	Description
26:27 PCTL	Parking control Determines the slave port's parking control. The low-power park feature results in overall power savings if the slave port is not saturated; however, this forces an extra latency clock when any master tries to access the slave port while not in use because it is not parked on any master. 00 When no master makes a request, the arbiter parks the slave port on the master port defined by the PARK bit field. 01 When no master makes a request, the arbiter parks the slave port on the last master to be in control of the slave port. 10 When no master makes a request, the slave port is not parked on a master and the arbiter drives all outputs to a constant safe state. 11 Reserved
29:31 PARK	Park Determines which master port the current slave port parks on when no masters are actively making requests and the PCTL bits are cleared. **Note:** Only select master ports that are actually present on the device. If not, undefined behavior may occur. Also, take care of the hardwired connections between Master and Flash slaves ports for applying meaningful parking configuration. 000 Park on master port M0 001 Park on master port M1 010 Park on master port M2 011 Park on master port M3 100 Park on master port M4 101 Park on master port M5 110 Park on master port M6 111 Park on master port M7

18.3.3 Master General Purpose Control Register (XBAR_MGPCRn)

Offset: 0x0800 + *n**0x100 (*n* = 0 to 7) Access: Supervisor read/write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	AULB		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0/1(1)	0/1(1)	0/1(1)

1. See MGPCRn register reset values in *Section 7.3.3: Crossbar switch* of *Chapter 7: Device configuration*.

Figure 190. Master General Purpose Control Register (XBAR_MGPCRn)

Table 292. XBAR_MGPCRn field descriptions

Field	Description
29:31 AULB	Arbitrate on Undefined Length Bursts These bits are used to select the arbitration policy during undefined length bursts by the master. These bits are initialized by hardware reset. 000 No arbitration will be allowed during undefined length burst 001 Arbitration will be allowed at any time during an undefined length burst 010 Arbitration will be allowed after 4 beats of an undefined length burst 011 Arbitration will be allowed after 8 beats of an undefined length burst 100 Arbitration will be allowed after 16 beats of an undefined length burst 101 Reserved 110 Reserved 111 Reserved

18.4 Functional description

This section provides the functional details of the crossbar switch.

18.4.1 General operation

When a master sends an access to the crossbar switch, the access is immediately taken. If the targeted slave port of the access is available, then the access is immediately presented on the slave port. It is possible to make single-clock (zero wait state) accesses through the crossbar. If the targeted slave port of the access is busy or parked on a different master port, the requesting master simply sees wait states inserted until the targeted slave port can service the master's request. The latency in servicing the request depends on each master's priority level and the responding slave's access time.

Since the crossbar switch appears to be just another slave to the master device, the master device has no knowledge of whether it actually owns the slave port it is targeting. While the master does not have control of the slave port it is targeting, it simply enters a wait state.

A master is given control of the targeted slave port only after a previous access to a different slave port completes, regardless of its priority on the newly targeted slave port. This prevents deadlock from occurring when:

- A higher priority master has:
  - An outstanding request to one slave port that has a long response time and
  - A pending access to a different slave port, and
- A lower priority master is also making a request to the same slave port as the pending access of the higher priority master.

After the master has control of the slave port it is targeting, the master remains in control of that slave port until it gives up the slave port by running an IDLE cycle or by leaving that slave port for its next access. The master could also lose control of the slave port if another higher priority master makes a request to the slave port; however, if the master is running a fixed-length burst transfer it retains control of the slave port until that transfer completes. When a master has control of a given slave port, the crossbar returns all response information from the slave back to the requesting master.

The crossbar terminates all master IDLE transfers (as opposed to allowing the termination to come from one of the slave buses). Additionally, when no master is requesting access to

a slave port, the crossbar drives IDLE transfers onto the slave bus even though a default master may be granted access to the slave port.

When a slave bus is being idled by the crossbar, it can park the slave port on the master port indicated by the XBAR_CRS*n*[PARK]. This is done in an attempt to save the initial clock of arbitration delay that otherwise would be seen if the master had to arbitrate to gain control of the slave port. The slave port can also be put into low-power park mode in attempt to save power by using XBAR_CRS*n*[PCTL].

18.4.2 Register coherency

Since the content of the registers has a real-time effect on the operation of the crossbar, it is important to understand that any register modifications take effect as soon as the register is written. The values of the registers do not track with slave-port-related master accesses; instead, they track only with slave accesses.

18.4.3 Arbitration

The crossbar switch supports two arbitration schemes: a simple fixed-priority comparison algorithm and a simple round-robin fairness algorithm. The arbitration scheme is independently programmable for each slave port.

18.4.3.1 Fixed-priority operation

When operating in fixed-priority mode, each master is assigned a unique priority level in the XBAR_PRS*n* (priority registers). If two masters request access to a slave port, the master with the higher priority in the selected priority register gains control over the slave port.

When a master makes a request to a slave port, the slave port checks whether the new requesting master's priority level is higher than that of the master that currently has control over the slave port (unless the slave port is in a parked state). The slave port performs an arbitration check at every clock edge to ensure that the proper master (if any) has control of the slave port.

If the new requesting master's priority level is higher than that of the master that currently has control of the slave port, the new requesting master is granted control over the slave port at the next clock edge. The exception to this rule is if the master that currently has control over the slave port is running a fixed-length burst transfer or a locked transfer. In this case, the new requesting master must wait until the end of the burst transfer or locked transfer before it is granted control of the slave port.

If the new requesting master's priority level is lower than the master that currently has control of the slave port, the new requesting master is forced to wait until the current master runs one of the following cycles:

- An IDLE cycle
- A non-IDLE cycle to a location other than the current slave port

18.4.3.2 Round-robin priority operation

When operating in round-robin mode, each master is assigned a relative priority based on the physical master port number. This relative priority is compared to the ID (master port number) of the last master to perform a transfer on the slave bus. The highest priority requesting master becomes owner of the slave bus at the next transfer boundary (accounting for locked and fixed-length burst transfers). Priority is based on how far the ID of the requesting master is ahead of the ID of the last master.

After access is granted to a slave port, a master may perform as many transfers as desired to that port until another master makes a request to the same slave port. The next master in line is granted access to the slave port at the next transfer boundary, or possibly on the next clock cycle if the current master has no pending access request.

As an example of arbitration in round-robin mode, assume the crossbar is implemented with master ports 0, 1, 4, and 5. If the last master of the slave port was master 1, and master 0, 4, and 5 make simultaneous requests, they are serviced in the order 4, 5, and then 0.

Parking may continue to be used in a round-robin mode, but it does not affect the round-robin pointer unless the parked master actually performs a transfer. Hand-off occurs to the next master in line after one cycle of arbitration. If the slave port is put into low-power park mode, the round-robin pointer is reset to point at master port 0, giving it the highest priority.

18.5 Initialization/application information

No initialization is required by or for the crossbar switch. Hardware reset ensures all the register bits used by the crossbar switch are properly initialized to a valid state. Settings and priorities should be programmed to achieve maximum system performance.

19 Crossbar Integrity Checker (XBIC)

19.1 Overview

The Crossbar Integrity Checker (XBIC) verifies the integrity of the crossbar transfers. For forward signals (master to slave), it is done by verifying the integrity of the attribute information using an 8-bit Error Detection Code (EDC). The EDC detects any single- or double-bit errors in the attribute information and signals the Fault Collection and Control Unit (FCCU) when an error is detected. For feedback signals (slave to master), it is done by comparing the consistence of the signals during the AHB dataphase.There are three signals from slave to master, hready, hresp0, and hresp2. If any of the master signals is different from the slave signals during dataphase, the error will be reported in the Error Status Register.

During the address phase of the system bus pipelined protocol, the XBIC generates an 8-bit EDC checkbit value for every transfer, passing these checkbits through the crossbar to the targeted slave where it is compared against the 8-bit EDC checkbits calculated from the crossbar slave attributes. The EDC is calculated for 64 bits of attribute information: 32 bits of system bus attribute control plus 32 bits associated with the processor core's decorated storage attribute. The resulting (72,64) code is a distance 4 encoding using a minimum odd weight H matrix definition. The EDC definition has been optimized for timing considerations with the system bus attribute signals exclusively using weight 3 code values and the decoration attribute using a mix of weight 3 and 5 codes.

During the data phase of the system bus pipelined protocol, the XBIC compares the three signals of hready, hresp0, hresp2 from entering the crossbar on the slave side to leaving the crossbar on the master side. Data phase is often more than one clock cycle due to wait cycles. All three signals are compared clock by clock during the whole data cycle.

19.2 Features

The XBIC has the following features:

- Verification of attribute information for all crossbar transfers
  - EDC (72,64) code protects against single and double bit errors
- Verification of feedback information for each data phase during crossbar transfer
  - hready, hresp0, and hresp2 are compared from entering the crossbar on the slave side to leaving the crossbar on the master side
- Error injection for testing
  - Programmable master and slave port specifiers
  - Programmable 8-bit toggle vector to insert error in master EDC checkbit value
  - Address, EDC syndrome, master and slave port information captured on error
- Programmable integrity check enable on a per-slave-port basis
- Programmable integrity feedback check enable on a per-master-port basis

19.3 Block diagram

The crossbar transfer attribute information for all master and slave ports is routed to the XBIC, which calculates and checks the EDC parity over the attribute information as shown in *Figure 191* and *Figure 192*.

Figure 191. XBIC system block diagram

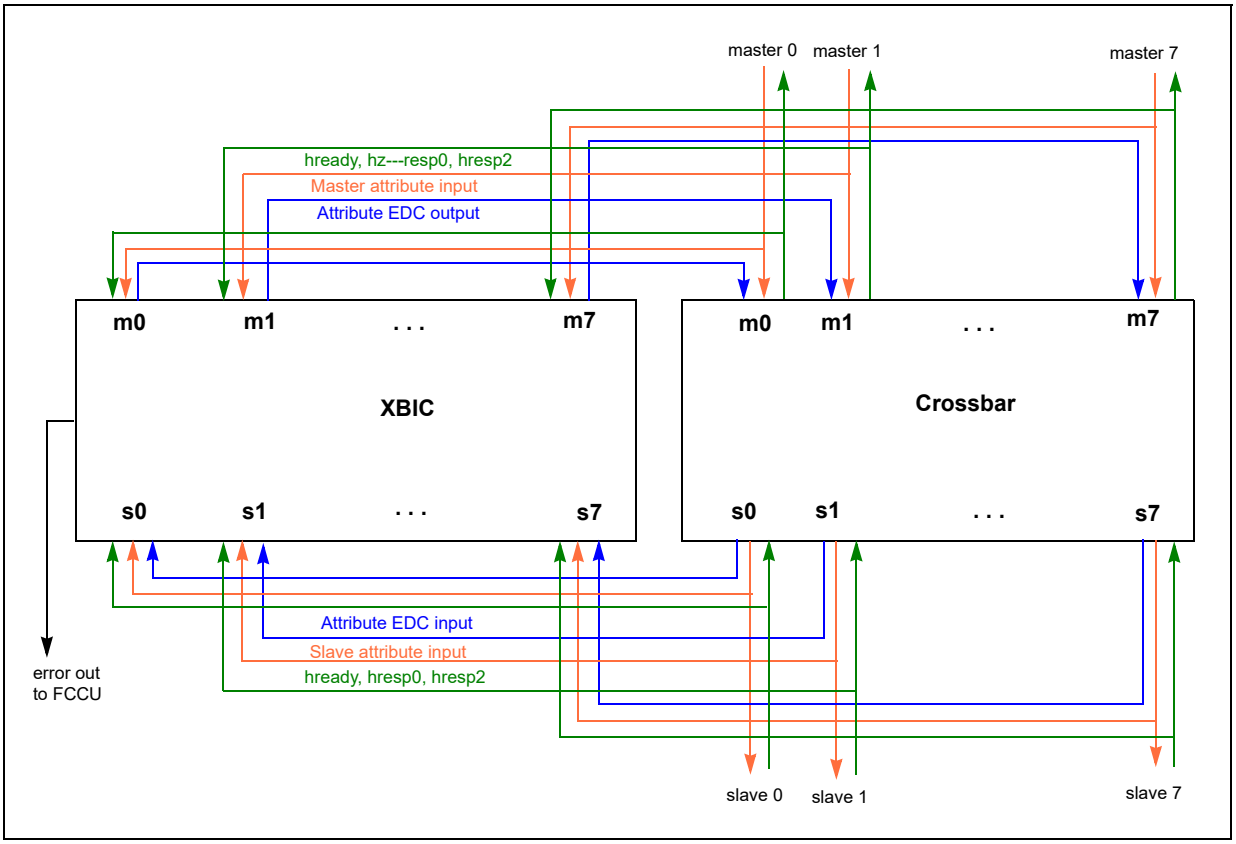
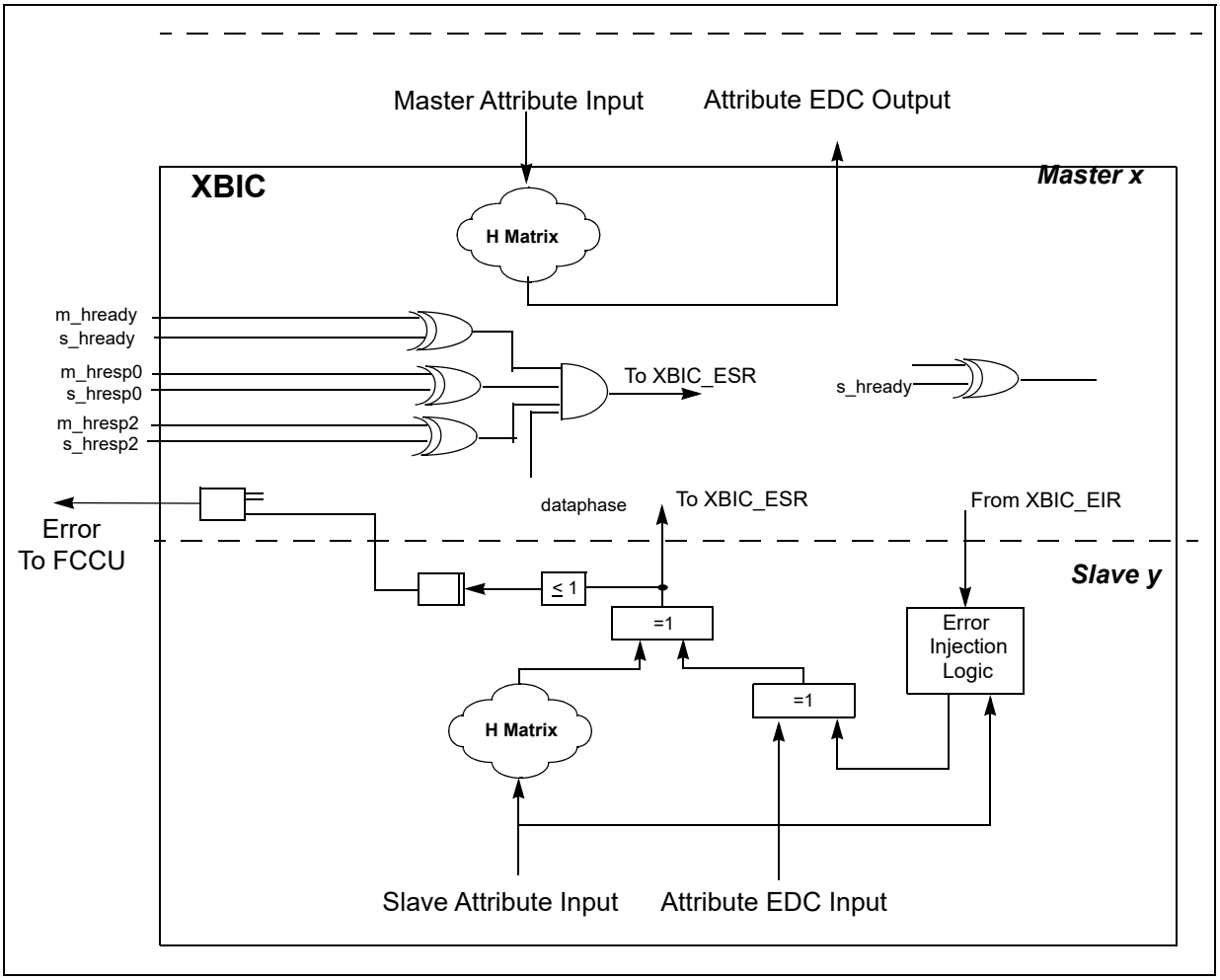


Figure 192. XBIC block diagram



19.4 External signal description

The XBIC has no external interface signals.

19.5 Memory map and register definition

The XBIC programming model has four 32-bit registers. The programming model can only be accessed in supervisor mode using 32-bit (word) accesses. Attempted references with a different access size, to an undefined (reserved) address, with a non-supported access type (a write to a read-only register), or in user mode generate an error termination.

19.5.1 Memory map

The XBIC memory map is shown in *Table 293*.

Table 293. XBIC memory map

Address offset (hex)	Register description	Location
0x0000	XBIC Module Control Register (XBIC_MCR)	*Section 19.5.2.1*
0x0004	XBIC Error Injection Register (XBIC_EIR)	*Section 19.5.2.2*
0x0008	XBIC Error Status Register (XBIC_ESR)	*Section 19.5.2.3*
0x000C	XBIC Error Address Register (XBIC_EAR)	*Section 19.5.2.4*
0x0010–0x3FFF	Reserved	

19.5.2 Register descriptions

The following sections detail the individual registers within the XBIC programming model.

19.5.2.1 XBIC Module Control Register (XBIC_MCR)

The XBIC_MCR allows attribute integrity checking to be enabled on a per-port basis.

Offset: 0x0000 Access: Supervisor Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	SE0	SE1	SE2	SE3	SE4	SE5	SE6	SE7	ME0	ME1	ME2	ME3	ME4	ME5	ME6	ME7
W																
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

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

Figure 193. XBIC Module Control Register (XBIC_MCR)

Table 294. XBIC_MCR field descriptions

Field	Description
0 SE0	Slave Port Enable for EDC error detection 0 Attribute integrity checking disabled for slave port 0 1 Attribute integrity checking enabled for slave port 0
1 SE1	Slave Port Enable for EDC error detection 0 Attribute integrity checking disabled for slave port 1 1 Attribute integrity checking enabled for slave port 1
2 SE2	Slave Port Enable for EDC error detection 0 Attribute integrity checking disabled for slave port 2 1 Attribute integrity checking enabled for slave port 2
3 SE3	Slave Port Enable for EDC error detection 0 Attribute integrity checking disabled for slave port 3 1 Attribute integrity checking enabled for slave port 3

Table 294. XBIC_MCR field descriptions (continued)

Field	Description
4 SE4	Slave Port Enable for EDC error detection 0 Attribute integrity checking disabled for slave port 4 1 Attribute integrity checking enabled for slave port 4
5 SE5	Slave Port Enable for EDC error detection 0 Attribute integrity checking disabled for slave port 5 1 Attribute integrity checking enabled for slave port 5
6 SE6	Slave Port Enable for EDC error detection 0 Attribute integrity checking disabled for slave port 6 1 Attribute integrity checking enabled for slave port 6
7 SE7	Slave Port Enable for EDC error detection 0 Attribute integrity checking disabled for slave port 7 1 Attribute integrity checking enabled for slave port 7
8 ME0	Master Port Enable for slave driven signal safety check 0 Attribute integrity checking disabled for master port 0 1 Attribute integrity checking enabled for master port 0
9 ME1	Master Port Enable for slave driven signal safety check 0 Attribute integrity checking disabled for master port 1 1 Attribute integrity checking enabled for master port 1
10 ME2	Master Port Enable for slave driven signal safety check 0 Attribute integrity checking disabled for master port 2 1 Attribute integrity checking enabled for master port 2
11 ME3	Master Port Enable for slave driven signal safety check 0 Attribute integrity checking disabled for master port 3 1 Attribute integrity checking enabled for master port 3
12 ME4	Master Port Enable for slave driven signal safety check 0 Attribute integrity checking disabled for master port 4 1 Attribute integrity checking enabled for master port 4
13 ME5	Master Port Enable for slave driven signal safety check 0 Attribute integrity checking disabled for master port 5 1 Attribute integrity checking enabled for master port 5
14 ME6	Master Port Enable for slave driven signal safety check 0 Attribute integrity checking disabled for master port 6 1 Attribute integrity checking enabled for master port 6
15 ME7	Master Port Enable for slave driven signal safety check 0 Attribute integrity checking disabled for master port 7 1 Attribute integrity checking enabled for master port 7

19.5.2.2 XBIC Error Injection Register (XBIC_EIR)

The XBIC_EIR contains fields for controlling the XBIC error injection function. When an error is injected, the EDC syndrome associated with the programmed master and slave ports is modified, causing the XBIC to assert the error indication to the FCCU and capturing the transfer information in the XBIC_ESR and XBIC_EAR registers. Otherwise, transfers are not affected by XBIC error injection.

Offset: 0x0004 Access: Supervisor Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	EIE	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	SLV			MST				SYN							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 194. XBIC Error Injection Register (XBIC_EIR)

Table 295. XBIC_EIR field descriptions

Field	Description
0 EIE	Error Injection Enable 0 Error injection disabled 1 Error injection enabled
17:19 SLV	Target Slave Port (1) Error injection is enabled for the designated slave port. Other slave ports are unaffected.
20:23 MST	Target Master ID (2) Error injection is enabled for the designated master ID. Transfers with other master IDs are not affected.
24:31 SYN	Syndrome This value is exclusive-ored with the calculated syndrome (which should be zero) to generate an error with the specified syndrome. A value of zero does not generate an error.

1. For the list of the target slave ports refer to the table ‘Crossbar switch slave assignments’ in section ‘Crossbar switch’ from ‘Device Configuration’ chapter.
2. For the list of the target master IDs refer to the table ‘SMPU logical bus master assignments’ in section ‘Crossbar switch’ from ‘Device Configuration’ chapter.

19.5.2.3 XBIC Error Status Register (XBIC_ESR)

The Error Status Register (XBIC_ESR) contains two types of error information about the *last* transfer. If an attribute integrity check error was detected, the slave port, the master port and the syndrome are captured in the SLV, MST and SYN fields. If there is a mismatch among internal signals (hready, hresp0, and hresp2) during the data phase, the slave and master port are captured in the DPSE0*x* and DPME*x* fields.

This register is cleared only on reset.

Offset: 0x0008 Access: Supervisor Read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	VLD	DPSE0	DPSE1	DPSE2	DPSE3	DPSE4	DPSE5	DPSE6	DPSE7	DPME0	DPME1	DPME2	DPME3	DPME4	DPME5	DPME6
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	DPME7	SLV			MST				SYN							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 195. XBIC Error Status Register (XBIC_ESR)

Table 296. XBIC_ESR field descriptions

Field	Description
0 VLD	Error Status Valid 0 No error detected, other fields of the ESR and EAR are not valid. 1 Error detected, all fields of the ESR and EAR are valid.
1 DPSE0	Data phase slave port error 0 No error on slave port 0 1 Data phase error on slave port 0
2 DPSE1	Data phase slave port error 0 No error on slave port 1 1 Data phase error on slave port 1
3 DPSE2	Data phase slave port error 0 No error on slave port 2 1 Data phase error on slave port 2
4 DPSE3	Data phase slave port error 0 No error on slave port 3 1 Data phase error on slave port 3
5 DPSE4	Data phase slave port error 0 No error on slave port 4 1 Data phase error on slave port 4
6 DPSE5	Data phase slave port error 0 No error on slave port 5 1 Data phase error on slave port 5
7 DPSE6	Data phase slave port error 0 No error on slave port 6 1 Data phase error on slave port 6
8 DPSE7	Data phase slave port error 0 No error on slave port 6 1 Data phase error on slave port 6

Table 296. XBIC_ESR field descriptions (continued)

Field	Description
9 DPME0	Data phase master port error 0 No error on master port 0 1 Data phase error on master port 0
10 DPME1	Data phase master port error 0 No error on master port 1 1 Data phase error on master port 1
11 DPME2	Data phase master port error 0 No error on master port 2 1 Data phase error on master port 2
12 DPME3	Data phase master port error 0 No error on master port 3 1 Data phase error on master port 3
13 DPME4	Data phase master port error 0 No error on master port 4 1 Data phase error on master port 4
14 DPME5	Data phase master port error 0 No error on master port 5 1 Data phase error on master port 5
15 DPME6	Data phase master port error 0 No error on master port 6 1 Data phase error on master port 6
16 DPME7	Data phase master port error 0 No error on master port 7 1 Data phase error on master port 7
17:19 SLV	Slave Port The slave port targeted by the last transfer with an error.
20:23 MST	Master ID The master ID value of the last transfer with an error.
24:31 SYN	Syndrome The syndrome calculated for the last transfer with an error. For single bit errors the signal in error can be determined, see *Table 298*.

19.5.2.4 XBIC Error Address Register (XBIC_EAR)

The Error Address Register (XBIC_EAR) contains the address of the last transfer in which an attribute integrity check error was detected on an enabled slave port. This register is only cleared on reset.

Offset: 0x000C Access: Supervisor Read

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

Figure 196. Error Address Register (XBIC_EAR)

Table 297. XBIC_EAR Register field descriptions

Field	Description
0:31 ADDR	The address of the last transfer with an error.

19.6 Functional description

The Crossbar Integrity Checker (XBIC) verifies the integrity of the attribute information for crossbar transfers using an 8-bit Error Detection Code (EDC). The EDC detects any single or double bit errors in the attribute information and signals the Fault Collection and Control Unit (FCCU) when an error is detected. The Module Control Register (XBIC_MCR) allows integrity checking to be enabled on a per-slave-port basis. The EDC parity is generated by the XBIC for each master port during the system bus address phase pipeline stage, routed through the crossbar via a sideband signal and checked against the slave attribute EDC generated for each enabled slave port during the first system bus data phase cycle. Detection of an error does *not* generate a bus error. The XBIC attribute integrity checking is independent from the end-to-end ECC that covers the transfer address and data.

The XBIC can be programmed to intentionally inject EDC errors for testing. Error injection is targeted toward a single slave port and a single master ID at a time as determined by the settings in the Error Injection Register (XBIC_EIR). When an error is injected, the EDC syndrome is modified which causes the XBIC to assert the error indication to the FCCU. Otherwise transfers are unaffected by XBIC error injection. This approach allows the check logic to be verified without compromising the integrity of the data transfer. Once the error injection function is enabled by setting XBIC_EIR[EIE], errors are induced on all subsequent targeted transactions until XBIC_EIR[EIE] is cleared.

Forwarding errors (master to slave) are detected during the AHB address phase. When a forwarding error is detected, due either to a hardware fault or error injection, information related to the error is captured and reported in the Error Status Register (XBIC_ESR) and Error Address Register (XBIC_EAR). The XBIC_ESR and XBIC_EAR registers contain information about the *last* transfer with a detected error and are only cleared on reset. The cause of a single-bit error can be determined by the syndrome value reported in the XBIC_ESR as shown in *Table 298*. Any other syndrome indicates a multi-bit error.

XBIC also detects backward signal (slave to master) errors during AHB data phase. For the three slave driven signals (hready, hresp0, and hresp2), XBIC performs the comparison between the input of the slave port and output of the master port during the AHB data phase. When a mismatch occurs, the error will be captured. The current slave port will be registered in XBIC_ESR bit field DPSE*x*, and the current master port will be registered in XBIC_ESR bit field DSME*x*.

Table 298. Hexadecimal attribute single-bit error syndromes

Signal	SYN	Signal	SYN	Signal	SYN	Signal	SYN
hwrite	07	hbstrb[7]	70	hdecor[31]	25	hdecor[15]	23
htrans[0]	0b	hbstrb[6]	32	hdecor[30]	68	hdecor[14]	51
hsize[2]	0d	hbstrb[5]	52	hdecor[29]	c7	hdecor[13]	54
hsize[1]	0e	hbstrb[4]	a8	hdecor[28]	83	hdecor[12]	61
hsize[0]	13	hbstrb[3]	43	hdecor[27]	85	hdecor[11]	e3
hprot[5]	15	hbstrb[2]	45	hdecor[26]	86	hdecor[10]	e6
hprot[4]	16	hbstrb[1]	4c	hdecor[25]	89	hdecor[9]	f8
hprot[3]	19	hbstrb[0]	a4	hdecor[24]	8a	hdecor[8]	38
hprot[2]	1a	hmaster[3]	a2	hdecor[23]	8c	hdecor[7]	58
hprot[1]	1c	hmaster[2]	b0	hdecor[22]	49	hdecor[6]	37
hprot[0]	91	hmaster[1]	c1	hdecor[21]	92	hdecor[5]	f1
hburst[2]	a1	hmaster[0]	c2	hdecor[20]	94	hdecor[4]	3b
hburst[1]	64	hslave[2]	c4	hdecor[19]	98	hdecor[3]	3d
hburst[0]	29	hslave[1]	c8	hdecor[18]	46	hdecor[2]	3e
hmastlock	2a	hslave[0]	d0	hdecor[17]	34	hdecor[1]	4f
hunalign	2c	hdecorated	e0	hdecor[16]	4a	hdecor[0]	6e
edc[7]	80	edc[6]	40	edc[5]	20	edc[4]	10
edc[3]	08	edc[2]	04	edc[1]	02	edc[0]	01

The H matrix for the attribute EDC is shown in *Table 299*.

Table 299. XBIC H Matrix Definition

Checkbits [7:0]	Attribute Bit																														
	hwrite	htrans[0]	hsize[2]	hsize[1]	hsize[0]	hprot[5]	hprot[4]	hprot[3]	hprot[2]	hprot[1]	hprot[0]	hburst[2]	hburst[1]	hburst[0]	hmastlock	hunalign	hbstrb[7]	hbstrb[6]	hbstrb[5]	hbstrb[4]	hbsrtb[3]	hbsrtb[2]	hbsrtb[1]	hbsrtb[0]	hmaster[3]	hmaster[2]	hmaster[1]	hmaster[0]	hslave[2]	hslave[1]	hslave[0]	hdecorated
7											*	*								*				*	*	*	*	*	*	*	*	*
6													*				*		*		*	*	*				*	*	*	*	*	*
5												*	*	*	*	*	*	*		*				*	*	*						*
4					*	*	*	*	*	*	*						*	*	*							*					*	
3		*	*	*				*	*	*				*	*	*				*			*							*		
2	*		*	*		*	*			*			*			*						*	*	*					*			
1	*	*		*	*		*		*						*			*	*		*				*			*				
0	*	*	*		*	*		*			*	*		*							*	*					*					
Checkbits [7:0]	hdecor[31]	hdecor[30]	hdecor[29]	hdecor[28]	hdecor[27]	hdecor[26]	hdecor[25]	hdecor[24]	hdecor[23]	hdecor[22]	hdecor[21]	hdecor[20]	hdecor[19]	hdecor[18]	hdecor[17]	hdecor[16]	hdecor[15]	hdecor[14]	hdecor[13]	hdecor[12]	hdecor[11]	hdecor[10]	hdecor[9]	hdecor[8]	hdecor[7]	hdecor[6]	hdecor[5]	hdecor[4]	hdecor[3]	hdecor[2]	hdecor[1]	hdecor[0]
7			*	*	*	*	*	*	*		*	*	*								*	*	*				*					
6		*	*							*				*		*		*	*	*	*	*	*		*		*				*	*
5	*	*													*		*			*	*	*	*	*		*	*	*	*	*		*
4											*	*	*		*			*	*				*	*	*	*	*	*	*	*		
3		*					*	*	*	*			*			*							*	*	*			*	*	*	*	*
2	*		*		*	*			*			*		*	*				*			*				*			*	*	*	*
1			*	*		*		*			*			*		*	*				*	*				*		*		*	*	*
0	*		*	*	*		*			*							*	*		*	*					*	*	*	*		*	

20 Peripheral Bridge (PBRIDGE)

20.1 Introduction

The peripheral bridge converts the crossbar switch interface to an interface that can access a majority of peripherals on the device.

The peripheral bridge occupies a 64 MB portion of the address space (not all peripheral slots may be used. Refer to Device Configuration chapter or the Memory Map chapter for details on slot assignment).
The bridge includes separate clock enable inputs for each of the slots, to accommodate slower peripherals.

20.1.1 Features

Key features of the peripheral bridge are:

- Supports a pair of 32-bit transactions for selected 64-bit memory accesses
- Each independently configurable peripheral includes a clock enable, which allows peripherals to operate at any speed less than the system clock rate
- Programming model that provides memory protection functionality.

20.1.2 General operation

The slave devices connected to the peripheral bridge are modules that contain readable/writable control and status registers. The system masters read and write these registers through the peripheral bridge. The peripheral bridge generates module enables, the module address, transfer attributes, byte enables, and write data as inputs to the peripherals. The peripheral bridge captures read data from the peripheral interface and drives it to the crossbar switch.

Each peripheral is typically allocated one 16-KB block (or slot) of the memory map. Usually, the peripheral bridge uses the size of the addressed peripheral to perform proper data byte lane routing only; no bus decomposition (dynamic bus sizing) is performed. However, there are the following exceptions:

- Aligned 64-bit reads are permitted to 32-bit slots.
- Aligned 64-bit writes may be targeted to 32-bit slots.
- Aligned or misaligned 64-bit instruction fetches may also be performed to 32-bit slots.

All other 64-bit accesses result in an error response.

20.2 Memory map and register definition

Table 300. Peripheral bridge memory map

Offset address	Register	Location
0x000	Master Privilege Register A (MPRA)	*Section 20.2.1*
0x004	Master Privilege Register B (MPRB)	*Section 20.2.2*
0x008–0x1F	Reserved(2)	

Table 300. Peripheral bridge memory map (continued)

Offset address	Register	Location
0x020–0x2F	Reserved(1)	
0x030–0x3F	Reserved(2)	
0x040–0xBF	Reserved(1)	
0x0C0–0xDF	Reserved(2)	
0x0E0–0xFF	Reserved(1)	
0x100	Peripheral Access Control Register A (PACRA)	*Section 20.2.3*
0x104	Peripheral Access Control Register B (PACRB)	*Section 20.2.3*
0x108	Peripheral Access Control Register C (PACRC)	*Section 20.2.3*
0x10C	Peripheral Access Control Register D (PACRD)	*Section 20.2.3*
0x110	Peripheral Access Control Register E (PACRE)	*Section 20.2.3*
0x114	Peripheral Access Control Register F (PACRF)	*Section 20.2.3*
0x118	Peripheral Access Control Register G (PACRG)	*Section 20.2.3*
0x11C	Peripheral Access Control Register H (PACRH)	*Section 20.2.3*
0x120–0x12F	Reserved(1)	
0x130–0x13F	Reserved(2)	
0x140	Off-platform Peripheral Access Control Register A (OPACRA)	*Section 20.2.4*
0x144	Off-platform Peripheral Access Control Register B (OPACRB)	*Section 20.2.4*
0x148	Off-platform Peripheral Access Control Register C (OPACRC)	*Section 20.2.4*
0x14C	Off-platform Peripheral Access Control Register D (OPACRD)	*Section 20.2.4*
0x150	Off-platform Peripheral Access Control Register E (OPACRE)	*Section 20.2.4*
0x154	Off-platform Peripheral Access Control Register F (OPACRF)	*Section 20.2.4*
0x158	Off-platform Peripheral Access Control Register G (OPACRG)	*Section 20.2.4*
0x15C	Off-platform Peripheral Access Control Register H (OPACRH)	*Section 20.2.4*
0x160	Off-platform Peripheral Access Control Register I (OPACRI)	*Section 20.2.4*
0x164	Off-platform Peripheral Access Control Register J (OPACRJ)	*Section 20.2.4*
0x168	Off-platform Peripheral Access Control Register K (OPACRK)	*Section 20.2.4*
0x16C	Off-platform Peripheral Access Control Register L (OPACRL)	*Section 20.2.4*
0x170	Off-platform Peripheral Access Control Register M (OPACRM)	*Section 20.2.4*
0x174	Off-platform Peripheral Access Control Register N (OPACRN)	*Section 20.2.4*
0x178	Off-platform Peripheral Access Control Register O (OPACRO)	*Section 20.2.4*
0x17C	Off-platform Peripheral Access Control Register P (OPACRP)	*Section 20.2.4*
0x180	Off-platform Peripheral Access Control Register Q (OPACRQ)	*Section 20.2.4*
0x184	Off-platform Peripheral Access Control Register R (OPACRR)	*Section 20.2.4*
0x188	Off-platform Peripheral Access Control Register S (OPACRS)	*Section 20.2.4*

Table 300. Peripheral bridge memory map (continued)

Offset address	Register	Location
0x18C	Off-platform Peripheral Access Control Register T (OPACRT)	Section 20.2.4
0x190	Off-platform Peripheral Access Control Register U (OPACRU)	Section 20.2.4
0x194	Off-platform Peripheral Access Control Register V (OPACRV)	Section 20.2.4
0x198	Off-platform Peripheral Access Control Register W (OPACRW)	Section 20.2.4
0x19C	Off-platform Peripheral Access Control Register X (OPACRX)	Section 20.2.4
0x1A0	Off-platform Peripheral Access Control Register Y (OPACRY)	Section 20.2.4
0x1A4	Off-platform Peripheral Access Control Register Z (OPACRZ)	Section 20.2.4
0x1A8	Off-platform Peripheral Access Control Register AA (OPACRAA)	Section 20.2.4
0x1AC	Off-platform Peripheral Access Control Register AB (OPACRAB)	Section 20.2.4
0x1B0	Off-platform Peripheral Access Control Register AC (OPACRAC)	Section 20.2.4
0x1B4	Off-platform Peripheral Access Control Register AD (OPACRAD)	Section 20.2.4
0x1B8	Off-platform Peripheral Access Control Register AE (OPACRAE)	Section 20.2.4
0x1BC	Off-platform Peripheral Access Control Register AF (OPACRAF)	Section 20.2.4

1. The following address spaces when programmed, will not terminate in an error. Software should not program these address spaces.
2. Access to this address space will terminate in an error.

20.2.1 Master Privilege Register A (MPRA)

Each MPR specifies eight 4-bit fields defining the access privilege level associated with a bus master in the device to the various peripherals. The register provides one field per bus master. Each master is assigned depending on its logical master number. Refer to Device Configuration chapter for details about the master assignments to these registers.

The MPROT*n* field is defined as shown in *Figure 197*.

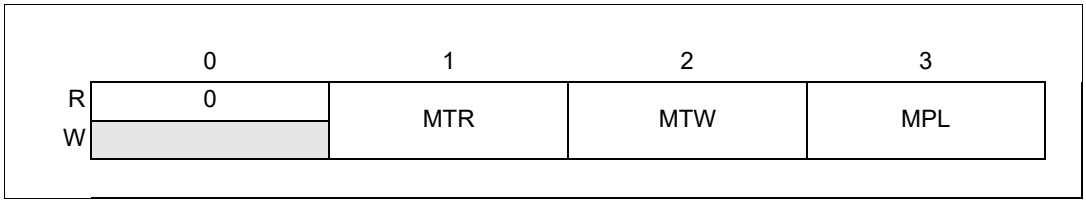


Figure 197. MPROT*n* fields

Table 301. MPROT*n* field descriptions

Field	Description
0	Reserved, should be cleared
1 MTR	Master trusted for read Determines whether the master is trusted for read accesses. 0 This master is not trusted for read accesses. 1 This master is trusted for read accesses.

Table 301. MPROT*n* field descriptions (continued)

Field	Description
2 MTW	Master trusted for writes Determines whether the master is trusted for write accesses. 0 This master is not trusted for write accesses. 1 This master is trusted for write accesses.
3 MPL	Master privilege level Determines how the privilege level of the master is determined. 0 Accesses from this master are forced to user mode. 1 Accesses from this master are not forced to user mode.

Address: 0x000 Access: Supervisor read/write

	0 1 2 3	4 5 6 7	8 9 10 11	12 13 14 15	16 17 18 19	20 21 22 23	24 25 26 27	28 29 30 31
R W	MPROT0	MPROT1	MPROT2	MPROT3	MPROT4	MPROT5	MPROT6	MPROT7
Reset	0 1 1 1	0 1 1 1	0 1 1 1	0 1 1 1	0 1 1 1	0 1 1 1	0 1 1 1	0 1 1 1

Figure 198. Master Privilege Register A (MPRA)

Table 302. MPRA field descriptions

Field	Description
0:3 MPROT0	Access privilege level associated with bus master 0
4:7 MPROT1	Access privilege level associated with bus master 1
8:11 MPROT2	Access privilege level associated with bus master 2
12:15 MPROT3	Access privilege level associated with bus master 3
16:19 MPROT4	Access privilege level associated with bus master 4
20:23 MPROT5	Access privilege level associated with bus master 5
24:27 MPROT6	Access privilege level associated with bus master 6
28:31 MPROT7	Access privilege level associated with bus master 7

20.2.2 Master Privilege Register B (MPRB)

Each MPR specifies eight 4-bit fields defining the access privilege level associated with a bus master in the device to the various peripherals. The register provides one field per bus master. Each master is assigned depending on its logical master number. Refer to Device Configuration chapter for details about the master assignments to these registers.

Address: 0x004 Access: Supervisor read/write

	0 1 2 3	4 5 6 7	8 9 10 11	12 13 14 15	16 17 18 19	20 21 22 23	24 25 26 27	28 29 30 31
R W	MPROT8	MPROT9	MPROT10	MPROT11	MPROT12	MPROT13	MPROT14	MPROT15
Reset	0 1 1 1	0 1 1 1	0 1 1 1	0 1 1 1	0 1 1 1	0 1 1 1	0 1 1 1	0 1 1 1

Figure 199. Master Privilege Register B (MPRB)

Table 303. MPRB field descriptions

Field	Description
0:3 MPROT8	Access privilege level associated with bus master 8
4:7 MPROT9	Access privilege level associated with bus master 9
8:11 MPROT10	Access privilege level associated with bus master 10
12:15 MPROT11	Access privilege level associated with bus master 11
16:19 MPROT12	Access privilege level associated with bus master 12
20:23 MPROT13	Access privilege level associated with bus master 13
24:27 MPROT14	Access privilege level associated with bus master 14
28:31 MPROT15	Access privilege level associated with bus master 15

20.2.3 Peripheral Access Control Register (PACR*x*)

Each of the on-platform peripherals has a four-bit PACR*n* field which defines the access levels supported by the given module. Eight PACR fields are grouped together to form a 32-bit PACR*x* register.

The peripheral assignments to each PACR field are defined by the memory map slot to which the peripherals are assigned. Refer to Memory Map chapter for the assignments of the device. If a peripheral is absent, the corresponding PACR field is not implemented. Reads to the location return zeros, and writes are ignored.

The PBRIDGE registers themselves are mapped into the PACR0 address space and can only be accessed in supervisor mode by trusted bus masters.

Table 304 shows the top-level structure of the PACRs.

Table 304. PACR memory map

Offset	Register	[0:3]	[4:7]	[8:11]	[12:15]	[16:19]	[20:23]	[24:27]	[28:31]
0x0100	PACRA	PACR0	PACR1	PACR2	PACR3	PACR4	PACR5	PACR6	PACR7
0x0104	PACRB	PACR8	PACR9	PACR10	PACR11	PACR12	PACR13	PACR14	PACR15
0x0108	PACRC	PACR16	PACR17	PACR18	PACR19	PACR20	PACR21	PACR22	PACR23
0x010C	PACRD	PACR24	PACR25	PACR26	PACR27	PACR28	PACR29	PACR30	PACR31
0x0110	PACRE	PACR32	PACR33	PACR34	PACR35	PACR36	PACR37	PACR38	PACR39
0x0114	PACRF	PACR40	PACR41	PACR42	PACR43	PACR44	PACR45	PACR46	PACR47
0x0118	PACRG	PACR48	PACR49	PACR50	PACR51	PACR52	PACR53	PACR54	PACR55
0x011C	PACRH	PACR56	PACR57	PACR58	PACR59	PACR60	PACR61	PACR62	PACR63

Each PACR*n* field is defined as shown in *Figure 200*.

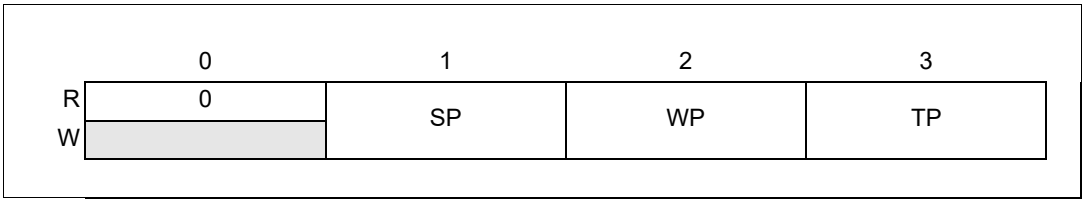


Figure 200. PACR*n* field

Table 305. PACR*n* field descriptions

Field	Description
0	Reserved, should be cleared
1 SP	Supervisor protect Determines whether the peripheral requires supervisor privilege level for access. 0 This peripheral does not require supervisor privilege level for accesses. 1 This peripheral requires supervisor privilege level for accesses. The master privilege level must indicate supervisor access attribute, and the MPROT*n*[MPL] control bit for the master must be set. If not, the access is terminated with an error response and no peripheral access is initiated.
2 WP	Write protect Determines whether the peripheral allows write accesses. 0 This peripheral allows write accesses. 1 This peripheral is write-protected. If a write access is attempted, the access is terminated with an error response and no peripheral access is initiated.
3 TP	Trusted protect Determines whether the peripheral allows accesses from an untrusted master. 0 Accesses from an untrusted master are allowed. 1 Accesses from an untrusted master are not allowed. If an access is attempted by an untrusted master, the access is terminated with an error response and no peripheral access is initiated.

Address: 0x100–0x11C (PACRA–PACRH) Access: Supervisor read/write

	0 1 2 3	4 5 6 7	8 9 10 11	12 13 14 15	16 17 18 19	20 21 22 23	24 25 26 27	28 29 30 31
R W	PACR*a*	PACRb	PACRc	PACRd	PACRe	PACRf	PACRg	PACRh
Reset	0 1 0 0	0 1 0 0	0 1 0 0	0 1 0 0	0 1 0 0	0 1 0 0	0 1 0 0	0 1 0 0

Note: The reset value of PACR0 is 0101b.

Figure 201. Peripheral Access Control Registers (PACR*x*)

Table 306. PACR*x* field descriptions

Field	Description
0:3 PACRa	Access level associated with module a
4:7 PACRb	Access level associated with module b
8:11 PACRc	Access level associated with module c
12:15 PACRd	Access level associated with module d
16:19 PACRe	Access level associated with module e
20:23 PACRf	Access level associated with module f
24:27 PACRg	Access level associated with module g
28:31 PACRh	Access level associated with module h

20.2.4 Off-Platform Peripheral Access Control Registers (OPACR*x*)

Each of the off-platform peripherals has a four-bit OPACR*n* field which defines the access levels supported by the given module. Eight OPACRs are grouped together to form a 32-bit OPACR*x* register.

The peripheral assignments to each OPACR field are defined by the memory map slot to which the peripherals are assigned. Refer to Memory Map chapter for the assignments of the device. If a peripheral is absent, the corresponding OPACR is not implemented. Reads to the location return zeros and writes are ignored.

Table 307 shows the top-level structure of the OPACR registers.

Table 307. OPACR memory map

Offset	Register	[0:3]	[4:7]	[8:11]	[12:15]	[16:19]	[20:23]	[24:27]	[28:31]
0x0140	OPACRA	OPACR 0	OPACR 1	OPACR 2	OPACR 3	OPACR 4	OPACR 5	OPACR 6	OPACR 7
0x0144	OPACRB	OPACR 8	OPACR 9	OPACR 10	OPACR 11	OPACR 12	OPACR 13	OPACR 14	OPACR 15
0x0148	OPACRC	OPACR 16	OPACR 17	OPACR 18	OPACR 19	OPACR 20	OPACR 21	OPACR 22	OPACR 23
0x014C	OPACRD	OPACR 24	OPACR 25	OPACR 26	OPACR 27	OPACR 28	OPACR 29	OPACR 30	OPACR 31
0x0150	OPACRE	OPACR 32	OPACR 33	OPACR 34	OPACR 35	OPACR 36	OPACR 37	OPACR 38	OPACR 39
0x0154	OPACRF	OPACR 40	OPACR 41	OPACR 42	OPACR 43	OPACR 44	OPACR 45	OPACR 46	OPACR 47
0x0158	OPACRG	OPACR 48	OPACR 49	OPACR 50	OPACR 51	OPACR 52	OPACR 53	OPACR 54	OPACR 55
0x015C	OPACRH	OPACR 56	OPACR 57	OPACR 58	OPACR 59	OPACR 60	OPACR 61	OPACR 62	OPACR 63
0x0160	OPACRI	OPACR 64	OPACR 65	OPACR 66	OPACR 67	OPACR 68	OPACR 69	OPACR 70	OPACR 71
0x0164	OPACRJ	OPACR 72	OPACR 73	OPACR 74	OPACR 75	OPACR 76	OPACR 77	OPACR 78	OPACR 79
0x0168	OPACRK	OPACR 80	OPACR 81	OPACR 82	OPACR 83	OPACR 84	OPACR 85	OPACR 86	OPACR 87
0x016C	OPACRL	OPACR 88	OPACR 89	OPACR 90	OPACR 91	OPACR 92	OPACR 93	OPACR 94	OPACR 95
0x0170	OPACRM	OPACR 96	OPACR 97	OPACR 98	OPACR 99	OPACR 100	OPACR 101	OPACR 102	OPACR 103
0x0174	OPACRN	OPACR 104	OPACR 105	OPACR 106	OPACR 107	OPACR 108	OPACR 109	OPACR 110	OPACR 111
0x0178	OPACRO	OPACR 112	OPACR 113	OPACR 114	OPACR 115	OPACR 116	OPACR 117	OPACR 118	OPACR 119
0x017C	OPACRP	OPACR 120	OPACR 121	OPACR 122	OPACR 123	OPACR 124	OPACR 125	OPACR 126	OPACR 127
0x0180	OPACRQ	OPACR 128	OPACR 129	OPACR 130	OPACR 131	OPACR 132	OPACR 133	OPACR 134	OPACR 135
0x0184	OPACRR	OPACR 136	OPACR 137	OPACR 138	OPACR 139	OPACR 140	OPACR 141	OPACR 142	OPACR 143
0x0188	OPACRS	OPACR 144	OPACR 145	OPACR 146	OPACR 147	OPACR 148	OPACR 149	OPACR 150	OPACR 151

Table 307. OPACR memory map (continued)

Offset	Register	[0:3]	[4:7]	[8:11]	[12:15]	[16:19]	[20:23]	[24:27]	[28:31]
0x018C	OPACRT	OPACR 152	OPACR 153	OPACR 154	OPACR 155	OPACR 156	OPACR 157	OPACR 158	OPACR 159
0x0190	OPACRU	OPACR 160	OPACR 161	OPACR 162	OPACR 163	OPACR 164	OPACR 165	OPACR 166	OPACR 167
0x0194	OPACRV	OPACR 168	OPACR 169	OPACR 170	OPACR 171	OPACR 172	OPACR 173	OPACR 174	OPACR 175
0x0198	OPACRW	OPACR 176	OPACR 177	OPACR 178	OPACR 179	OPACR 180	OPACR 181	OPACR 182	OPACR 183
0x019C	OPACRX	OPACR 184	OPACR 185	OPACR 186	OPACR 187	OPACR 188	OPACR 189	OPACR 190	OPACR 191
0x01A0	OPACRY	OPACR 192	OPACR 193	OPACR 194	OPACR 195	OPACR 196	OPACR 197	OPACR 198	OPACR 199
0x01A4	OPACRZ	OPACR 200	OPACR 201	OPACR 202	OPACR 203	OPACR 204	OPACR 205	OPACR 206	OPACR 207
0x01A8	OPACRAA	OPACR 208	OPACR 209	OPACR 210	OPACR 211	OPACR 212	OPACR 213	OPACR 214	OPACR 215
0x01AC	OPACRAB	OPACR 216	OPACR 217	OPACR 218	OPACR 219	OPACR 220	OPACR 221	OPACR 222	OPACR 223
0x01B0	OPACRAC	OPACR 224	OPACR 225	OPACR 226	OPACR 227	OPACR 228	OPACR 229	OPACR 230	OPACR 231
0x01B4	OPACRAD	OPACR 232	OPACR 233	OPACR 234	OPACR 235	OPACR 236	OPACR 237	OPACR 238	OPACR 239
0x01B8	OPACRAE	OPACR 240	OPACR 241	OPACR 242	OPACR 243	OPACR 244	OPACR 245	OPACR 246	OPACR 247
0x01BC	OPACRAF	OPACR 248	OPACR 249	OPACR 250	OPACR 251	OPACR 252	OPACR 253	OPACR 254	OPACR 255

The OPACR*n* field is defined as in *Figure 202*.

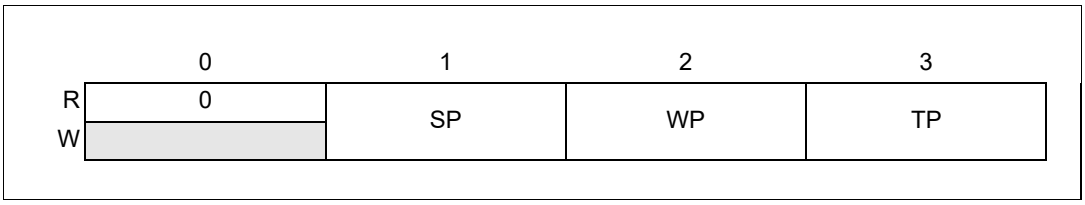


Figure 202. OPACR*n* fields

Table 308. OPACR*n* field descriptions

Field	Description
0	Reserved, should be cleared
1 SP	Supervisor protect Determines whether the peripheral requires supervisor privilege level for access. 0 This peripheral does not require supervisor privilege level for accesses. 1 This peripheral requires supervisor privilege level for accesses. The master privilege level must indicate supervisor access attribute, and the MPROT*n*[MPL] control bit for the master must be set. If not, the access is terminated with an error response and no peripheral access is initiated.
2 WP	Write protect Determines whether the peripheral allows write accesses. 0 This peripheral allows write accesses. 1 This peripheral is write-protected. If a write access is attempted, the access is terminated with an error response and no peripheral access is initiated.
3 TP	Trusted protect Determines whether the peripheral allows accesses from an untrusted master. 0 Accesses from an untrusted master are allowed. 1 Accesses from an untrusted master are not allowed. If an access is attempted by an untrusted master, the access is terminated with an error response and no peripheral access is initiated.

Address: 0x140–0x1BC (OPACRA–OPACRAF) Access: Supervisor read/write

	0 1 2 3	4 5 6 7	8 9 10 11	12 13 14 15	16 17 18 19	20 21 22 23	24 25 26 27	28 29 30 31
R W	OPACR*a*	OPACRb	OPACRc	OPACRd	OPACRe	OPACRf	OPACRg	OPACRh
Reset	0 1 0 0	0 1 0 0	0 1 0 0	0 1 0 0	0 1 0 0	0 1 0 0	0 1 0 0	0 1 0 0

Figure 203. Off-Platform Peripheral Access Control Registers (OPACR*x*)

Table 309. OPACR*x* field descriptions

Field	Description
0:3 OPACRa	Access level associated with module a
4:7 OPACRb	Access level associated with module b
8:11 OPACRc	Access level associated with module c
12:15 OPACRd	Access level associated with module d
16:19 OPACRe	Access level associated with module e
20:23 OPACRf	Access level associated with module f

Table 309. OPACR*x* field descriptions (continued)

Field	Description
24:27 OPACRg	Access level associated with module g
28:31 OPACRh	Access level associated with module h

20.3 Functional description

The peripheral bridge serves as an interface between the crossbar switch and the slave peripheral bus. It functions as a protocol translator. Support is provided for generating a pair of 32-bit slave accesses when performing certain 64-bit peripheral accesses.

Accesses which fall within the address space of the peripheral bridge are decoded to provide individual module selects for peripheral devices on the slave bus interface.

20.3.1 Access support

Aligned and misaligned 32-bit and 16-bit accesses, as well as byte accesses, are supported for 32-bit peripherals. Misaligned accesses are supported to allow memory to be placed on the slave peripheral bus. Peripheral registers must not be misaligned, although no explicit checking is performed by the peripheral bridge. The peripheral bridge performs two slave peripheral bus transfers for allowed 64-bit transactions, to 32-bit sized peripheral slots only. All other accesses are performed with a single transfer.

All accesses to the peripheral slots must be sized less than or equal to the designated peripheral slot size. If an access is attempted which is larger (in size) than the targeted port, an error response is generated without a peripheral access.

21 System Memory Protection Unit (SMPU)

21.1 Overview

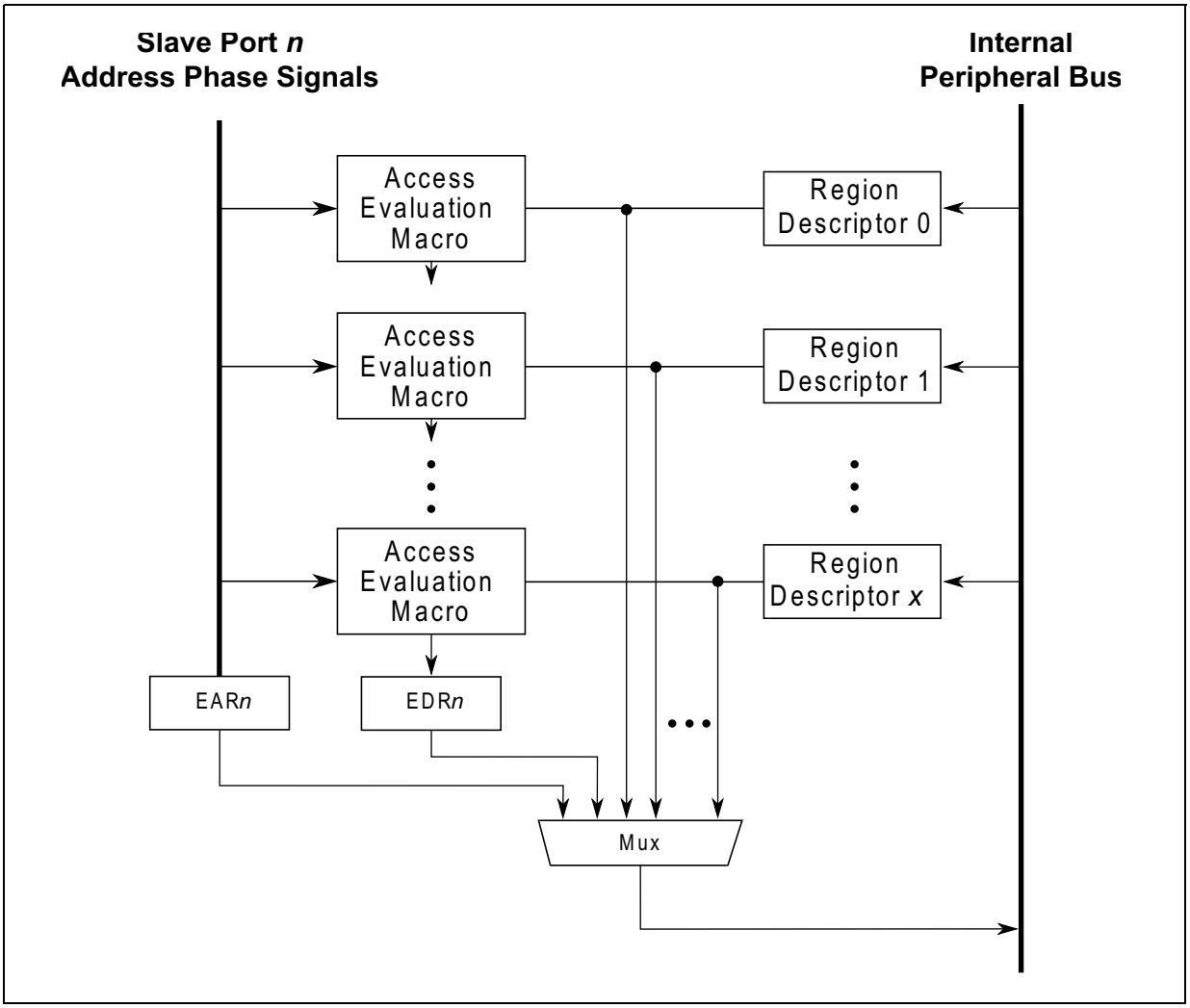
The System Memory Protection Unit (SMPU) provides hardware access control for system bus memory references. The SMPU concurrently monitors and evaluates system bus transactions using preprogrammed region descriptors that define memory spaces and their access rights. Memory references that have sufficient access control rights are allowed to complete, while references that are not mapped to any region descriptor or have insufficient rights are terminated with an access error response.

21.2 Block diagram

A simplified block diagram of the SMPU module is shown in *Figure 204*. The hardware's two-dimensional connection matrix is clearly visible with the basic access evaluation macro shown as the replicated submodule block. The crossbar switch slave ports are shown on the left, the region descriptor registers are in the middle, and the peripheral bus interface is on the right side. The evaluation macro contains two magnitude comparators connected to the start and end address registers from each region descriptor as well as the combinational logic blocks to determine the region hit and detect protection violations.

For details of the access evaluation macro, refer to *Section 21.5.1: Access evaluation macro*.

Figure 204. SMPU block diagram



21.3 Features

The SMPU feature set includes:

- Supports up to 24 program-visible 128-bit region descriptors, accessible as four 32-bit words each. (Specific module instances may support fewer than 24)
  - Each region descriptor defines an arbitrarily sized space, aligned anywhere in memory
    Region sizes can vary from a minimum of 1 byte to a maximum of (4 GB minus 1 byte)
  - Read/write access control permissions defined in region descriptor
  - Cache-inhibit attribute indicator per region descriptor
  - Hardware-assisted maintenance of the descriptor valid bit minimizes coherency issues
  - Priority given to granting permission over denying access for overlapping region descriptors
- Supports as many as eight crossbar slave ports
- Detects access errors if a memory reference does not hit in any memory region, or if the reference is illegal in all hit memory regions. If an access error occurs, the reference is terminated with an error response, and the SMPU inhibits the bus cycle being sent to the targeted slave device.
- Aborted core accesses generate instruction storage (ISI) or data storage (DSI) interrupts (refer to the documentation related to the core for details).
- Error registers (per bus master ID) capture the last faulting address, attributes, and other information.
- Global SMPU enable/disable control bit

21.4 Memory map and register definition

21.4.1 Memory map

The programing model is partitioned into three groups: control registers, error capture registers, and the data structure containing the region descriptors.

The programing model can only be referenced using 32-bit accesses. Attempted references using different access sizes, to undefined (reserved) addresses, or with a non-supported access type (a write to a read-only register, or a read of a write-only register) generate an error termination.

Note: Refer to the device configuration details for any chip-specific register information for this module. For example, a specific instance of a module may support only up to 8 region descriptors.

Table 310. SMPU memory map

Address offset	Register name	Section
0x0000	CESR0	*Section 21.4.2.1*
0x0004	CESR1	*Section 21.4.2.2*
(n=0 to 15)		
0x8*n+0x0100	EARn	*Section 21.4.2.3*

Table 310. SMPU memory map (continued)

Address offset	Register name	Section
0x8*n+0x0104	EDRn	*Section 21.4.2.4*
(n=0 to 23)		
0x10*n+0x0400	RGDn_WORD0	*Section 21.4.2.5*
0x10*n+0x0404	RGDn_WORD1	*Section 21.4.2.6*
0x10*n+0x0408	RGDn_WORD2_FMT0	*Section 21.4.2.7*
0x10*n+0x040C	RGDn_WORD3	*Section 21.4.2.8*

21.4.2 Register descriptions

21.4.2.1 Control/Error Status Register 0 (CESR0)

Address: 0x0000 Access: Supervisor read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	MERR															
W	w1c															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	1	0	0	0	0	0	0	0	0	0	0	0	HRL			GVLD
W																
Reset	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0

Figure 205. Control/Error Status Register 0 (CESR0)

Table 311. CESR0 field descriptions

Field	Description
0:15 MERR	Master *n* error, where the bus master number matches the bit number Indicates a captured error in EAR*n* and EDR*n*. A bit is set when the hardware detects an error and records the faulting address and attributes. It is cleared by writing '1' to it. If another error is captured at the exact same cycle as the write, the flag remains set. A find-first-one instruction (or equivalent) can detect the presence of a captured error. 0 No error has occurred for bus master *n*. 1 An error has occurred for bus master *n*.
16	Reserved This read-only bit is reserved and always has the value '1'.
28:30 HRL	Hardware revision level Specifies the SMPU's hardware and definition revision level. It can be read by software to determine the functional definition of the module.
31 GVLD	Global Valid (global enable/disable for the SMPU) 0 SMPU is disabled. All accesses from all bus masters are allowed. 1 SMPU is enabled.

21.4.2.2 Control/Error Status Register 1 (CESR1)

Address: 0x0004 Access: Supervisor read-only

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	MEOVR															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	1	0	0	0	0	0	0	0	0	0	0	0	NRGD			
W																
Reset	1	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0

Figure 206. Control/Error Status Register 1 (CESR1)

Table 312. CESR1 field descriptions

Field	Description
0:15 MEOVR	Master *n* error overrun, where the bus master number matches the bit number Each bit in this field signals that another SMPU error for bus master *n* has occurred before the previous error was processed. The details of the first error are recorded in the EAR*n* and EDR*n* registers, and no information on subsequent errors is recorded until the associated CESR0.MERR flag is cleared. 0 No error overrun condition has been detected for bus master *n.* 1 An error overrun condition has been detected for bus master *n.*
16	Reserved This read-only bit is reserved and always has the value ‘1’.
28:31 NRGD	Number of region descriptors Indicates the number of region descriptors implemented in the SMPU. 0001 4 region descriptors 0010 8 region descriptors 0011 12 region descriptors 0100 16 region descriptors 0101 20 region descriptors 0110 24 region descriptors

21.4.2.3 Error Address Register, Bus Master *n* (EAR*n*)

When the SMPU detects an access error on bus master *n*, the 32-bit reference address is captured in this read-only register and the corresponding bit in CESR0.MERR is set. Additional information about the faulting access is captured in the corresponding EDR*n* at the same time.

Note: The corresponding EARn and EDRn registers contain information on the original access error; subsequent errors associated with the given master are recorded as overruns in the CESR1.MEOVR field until the original error is processed.

Address: 0x0100 + *n**0x8 (*n* = 0 to 15) Access: Supervisor read-only

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	EADDR															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	EADDR															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 207. Error Address Register, Bus Master *n* (EAR*n*)

Table 313. EAR*n* field descriptions

Field	Description
0:31 EADDR	Error address Indicates the reference address from bus master *n* that generated the access error.

21.4.2.4 Error Detail Register, Bus Master *n* (EDR*n*)

When the SMPU detects an access error associated with bus master *n*, 32 bits of error detail are captured in this read-only register and the corresponding bit in CESR0.MERR is set. Information on the faulting address is captured in the corresponding EAR*n* register at the same time.

Note: The corresponding EARn and EDRn registers contain information on the original access error; subsequent errors associated with the given master are recorded as overruns in the CESR1.MEOVR field until the original error is processed.

Address: 0x0104 + *n**0x8 (*n* = 0 to 15) Access: Supervisor read-only

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	EACD															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	EACD								1	EATTR		ERW	EMN			
W																
Reset	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0

Figure 208. Error Detail Register, Bus Master *n* (EDR*n*)

Table 314. EDR*n* field descriptions

Field	Description
0:23 EACD	Error access control detail Indicates the region descriptor with the access error, where the region descriptor number matches the bit number. If EDR*n* contains a captured error and EACD is all zeroes, an access did not hit in any region descriptor. If only a single EACD bit is set, the access error was caused by a single non-overlapping region descriptor. If two or more EACD bits are set, the access error was caused by an overlapping set of region descriptors.
24	Reserved This read-only bit is reserved and always has the value ‘1’.
25:26 EATTR	Error attributes Indicates attribute information about the faulting reference. 00 User mode, instruction access 01 User mode, data access 10 Supervisor mode, instruction access 11 Supervisor mode, data access
27 ERW	Error read/write Indicates the access type of the faulting reference. 0 Read 1 Write
28:31 EMN	Error master number Indicates the logical bus master number of the faulting reference. This field is used to determine the bus master that generated the access error.

21.4.2.5 Region Descriptor *n*, Word 0 (RGD*n*_WORD0)

Address: 0x0400 + *n**0x10 (*n* = 0 to 23) Access: Supervisor read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	SRTADDR															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	SRTADDR															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 209. Region Descriptor *n*, Word 0 (RGD*n*_WORD0)

Table 315. RGD*n*_WORD0 field descriptions

Field	Description
0:31 SRTADDR	Start address Defines the byte start address of the memory region.

21.4.2.6 Region Descriptor *n*, Word 1 (RGD*n*_WORD1)

Address: 0x0404 + *n**0x10 (*n* = 0 to 23) Access: Supervisor read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	ENDADDR															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	ENDADDR															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 210. Region Descriptor *n*, Word 1 (RGD*n*_WORD1)

Table 316. RGD*n*_WORD1 field descriptions

Field	Description
0:31 ENDADDR	End address Defines the byte end address of the memory region. **Note:** The SMPU does not verify that ENDADDR ≥ SRTADDR.

21.4.2.7 Region Descriptor *n*, Word 2 Format 0 (RGD*n*_WORD2_FMT0)

RGD_WORD2 has two formats as determined by the RGD_WORD3.FMT field.

Note: RGD_WORD2_FMT0 applies when RGD_WORD3[FMT] = 0.

RGD_WORD2_FMT0 defines the access control rights of the memory region on a per master basis. The access control rights are defined by separate read and write permissions. For these fields, the bus master number refers to the *logical* bus master number.

For the access control rights, there are two flags per logical bus master:

- Read (r) permission refers to the ability to access the referenced memory address using an operand (data) fetch or an instruction fetch.
- Write (w) permission refers to the ability to update the referenced memory address using a store (data) instruction.

Each field consists of the two flags, with (r) being in the more significant position. For example, M0P has (r) as bit 0 and (w) as bit 1.

The bit settings are as follows:

- If set, the corresponding flag allows the specific access type (r = memory read, w = memory write).
- If cleared, the specific access type is not allowed.

Writes to this word clear the region descriptor's valid bit.

Address: 0x0408 + *n**0x10 (*n* = 0 to 23) Access: Supervisor read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	M0P		M1P		M2P		M3P		M4P		M5P		M6P		M7P	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	M8P		M9P		M10P		M11P		M12P		M13P		M14P		M15P	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 211. Region Descriptor *n*, Word 2 Format 0 (RGD*n*_WORD2_FMT0)

Table 317. RGD*n*_WORD2_FMT0 field descriptions

Field	Description
0:1 M0P	Bus master 0 permissions
2:3 M1P	Bus master 1 permissions
4:5 M2P	Bus master 2 permissions
6:7 M3P	Bus master 3 permissions
8:9 M4P	Bus master 4 permissions
10:11 M5P	Bus master 5 permissions
12:13 M6P	Bus master 6 permissions
14:15 M7P	Bus master 7 permissions
16:17 M8P	Bus master 8 permissions
18:19 M9P	Bus master 9 permissions
20:21 M10P	Bus master 10 permissions
22:23 M11P	Bus master 11 permissions
24:25 M12P	Bus master 12 permissions
26:27 M13P	Bus master 13 permissions

Table 317. RGD*n*_WORD2_FMT0 field descriptions (continued)

Field	Description
28:29 M14P	Bus master 14 permissions
30:31 M15P	Bus master 15 permissions

21.4.2.8 Region Descriptor *n*, Word 3 (RGD*n*_WORD3)

The final word of the SMPU region descriptor contains the valid bit, the format select (FMT), plus other configuration bits.

Address: 0x040C + *n**0x10 (*n* = 0 to 23) Access: Supervisor read/write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	FMT	RO	0	CI	VLD
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 212. Region Descriptor n, Word 3 (RGDn_WORD3)

Table 318. RGD*n*_WORD3 field descriptions

Field	Description
27 FMT	Region Descriptor Format This bit selects the configuration format (FMT0 or FMT1) for this region descriptor. **Note:** A specific module instance of the SMPU may support only the FMT0 format. If so, the FMT field is read-only with a fixed value of 0 and only RGD_WORD2_FMT0 applies. 0 Use format 0 (RGD_WORD2_FMT0) 1 Use format 1 (RGD_WORD2_FMT1)
28 RO	Read-Only This bit is intended to prevent accidental writes of an SMPU region descriptor. **Note:** Setting RO in an RGD locks all four words of the RGD until a system reset; the valid bit of the RGD and the global valid bit have no effect. 0 The region descriptor can be read or written. 1 Attempted writes to any location in the region descriptor are ignored with an error-free data transfer termination.

Table 318. RGD*n*_WORD3 field descriptions (continued)

Field	Description
30 CI	Cache Inhibit Defines the cacheability attribute of the memory region. This bit is returned to the bus master that initiated the memory reference. **Note:** An address range specified in an SMPU region descriptor for a cacheable space (that is, CI = 0) must be defined with a starting address aligned on a 0-modulo-32 byte address and with a multiple of the 32 byte cache line size. 0 References to this region can be cached. 1 References to this region cannot be cached.
31 VLD	Valid Signals that the region descriptor is valid. Any write to RGD*n*_WORD0–2 clears this bit. 0 Region descriptor is invalid. 1 Region descriptor is valid.

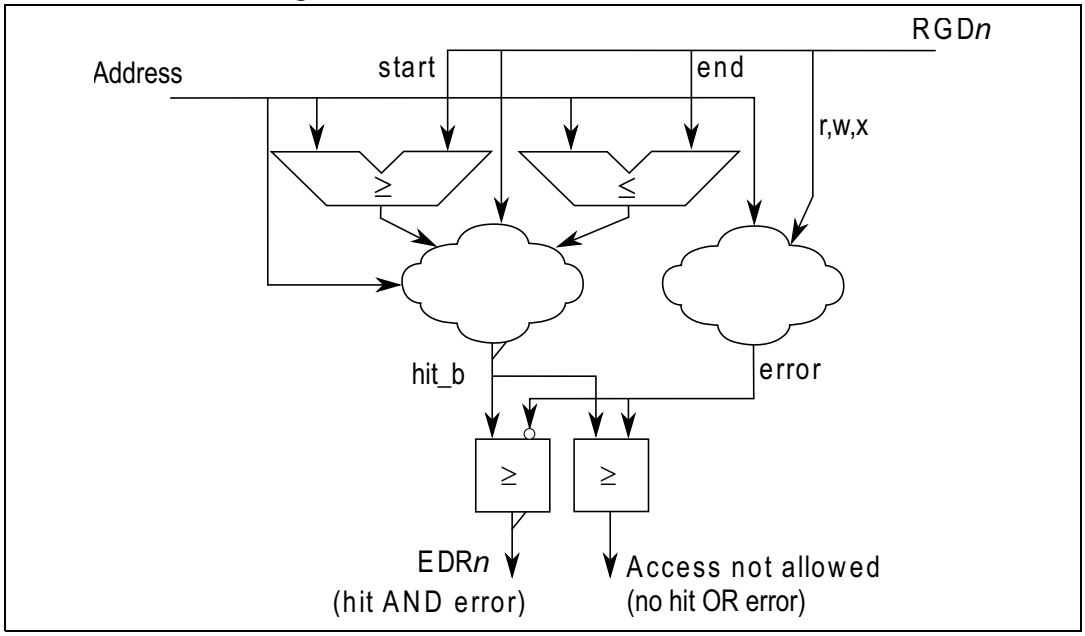
21.5 Functional description

In this section, the functional operation of the SMPU is detailed, including the operation of the access evaluation macro and the handling of error-terminated bus cycles.

21.5.1 Access evaluation macro

The basic operation of the SMPU is performed in the access evaluation macro, a hardware structure replicated in the two-dimensional connection matrix. As shown in *Figure 213*, the access evaluation macro inputs the crossbar bus address phase signals and the contents of a region descriptor (RGD*n*), then performs two major functions: region hit determination and detection of an access protection violation. *Figure 213* shows a functional block diagram.

Figure 213. SMPU access evaluation macro



21.5.1.1 Hit determination

To determine if the current reference hits in the given region, two magnitude comparators are used with the region's start and end addresses. The boolean equation for this portion of the hit determination is:

```
region_hit = ((addr[0:31] >= RGDn_Word0[SRTADDR]) &
      (addr[0:31] <= RGDn_Word1[ENDADDR])) &
RGDn_Word3[VLD]
```

Where addr is the current reference address, RGDn_Word0.SRTADDR and RGDn_Word1.ENDADDR are the start and end addresses, and RGDn_Word3.VLD is the valid bit.

Note: The SMPU does not verify that ENDADDR ≥ SRTADDR.

Region hit determination is based only on an address comparison of the first byte being accessed; that is, the SMPU does not check the size of the access to make sure it fits within an allowed region.

21.5.1.2 Privilege violation determination

While the access evaluation macro is determining the region hit, the logic is also evaluating whether the current access is allowed by the permissions defined in the region descriptor. Using the master signal, a set of effective permissions is generated from the relevant fields in the region descriptor. The protection violation logic then evaluates the access against the effective permissions, using the specification shown in *Table 319*.

Table 319. Protection violation definition

Access type	Access permissions		Protection violation?
	r	w	
Instruction fetch read	0	—	Yes, no read permission
	1	—	No, access is allowed
Data read	0	—	Yes, no read permission
	1	—	No, access is allowed
Data write	—	0	Yes, no write permission
	—	1	No, access is allowed

21.5.2 Putting it all together and error terminations

For each slave port monitored, the SMPU performs a reduction-AND of all the individual terms from each access evaluation macro. This expression then terminates the bus cycle with an error and reports an access error for three conditions:

1. The access does not hit in any region descriptor.
2. The access hits in a single region descriptor and that region has a protection violation.
3. The access hits in multiple (overlapping) regions and all regions have protection violations.

As shown in the third condition, granting permission is a higher priority than denying access for overlapping regions. This approach is more flexible to system software in region descriptor assignments. For an example of the use of overlapping region descriptors, refer to *Section 21.7*.

21.6 Initialization information

At system startup, load the required number of region descriptors, including setting RGD*n*_Word3.VLD. Setting CESR0.GVLD enables the module.

If the system requires that all the loaded region descriptors be enabled simultaneously, first ensure that the entire SMPU is disabled (CESR0.GVLD = 0). Next, load the appropriate region descriptors (RGD*n*), including the setting of the RGDn_Word3.VLD bits. Finally, set CESR0.GVLD to enable the entire module.

A region descriptor must be set to allow access to the SMPU registers if further changes are needed.

21.7 Application information

In an operational system, interfacing with the SMPU is generally classified into the following activities:

- Creating a new memory region: Load the appropriate region descriptor into an available RGD*n*, using four sequential 32-bit writes. The hardware assists in the maintenance of the valid bit, so if this approach is followed, there are no coherency issues with the multi-cycle descriptor writes.
- Modifying a region descriptor: Load the updates into the region descriptor using sequential 32-bit writes. Writing to Word3 re-enables the region descriptor valid bit. The hardware assists in the maintenance of the valid bit, so if this approach is followed, there are no coherency issues with the multi-cycle descriptor writes.
- Removing a region descriptor: Clearing RGD*n*_Word3.VLD deletes an existing region descriptor.
- Accessing the SMPU: Allocate a region descriptor to restrict SMPU access to supervisor mode from a specific master.
- Detecting an access error: The current bus cycle is terminated with an error response and EAR*n* and EDR*n* capture information on the faulting reference. The error-terminated bus cycle typically initiates an error response in the originating bus master. For example, a processor core may respond with a bus error exception, while a data movement bus master may respond with an error interrupt. The processor can retrieve the captured error address and detail information simply by reading E{A,D}R*n*. CESR0.MERR signals. The error registers contain the captured fault data.
- Overlapping region descriptors: Applying overlapping regions often reduces the number of descriptors required for a given set of access controls. In the overlapping memory space, the protection rights of the corresponding region descriptors are logically summed together (the boolean OR operator).

The following dual-core system example contains four bus masters: the two processors (CP0, CP1) and two DMA engines (DMA1, a traditional data movement engine transferring data between RAM and peripherals, and DMA2, a second engine transferring data to/from the RAM only). Consider the following region descriptor assignments:

Table 320. Overlapping region descriptor example

Region description	RGDn		CP0	CP1	DMA1	DMA2	System memory map space
CP0 code	0		rwx	r--	—	—	Flash
CP1 code	1		r--	rwx	—	—	
CP0 data & stack	2		rw-	—	—	—	RAM
CP0 → CP1 shared data	2	3	r--	r--	—	—	
CP1 → CP0 shared data	4						
CP1 data & stack	4		—	rw-	—	—	
Shared DMA data	5		rw-	rw-	rw	rw	
SMPU	6		rw-	rw-	—	—	Peripheral space
Peripherals	7		rw-	rw-	rw	—	

In this example, there are eight descriptors used to span nine regions in the three main spaces of the system memory map (flash, RAM, and peripheral space). Each region indicates the specific permissions for each of the four bus masters, and this definition provides an assigned set of shared, private, and executable memory spaces.

Of particular interest are the two overlapping spaces: region descriptors 2 & 3 and 3 & 4.

The space defined by RGD2 with no overlap is a private data and stack area that provides read/write access to CP0 only. The overlapping space between RGD2 and RGD3 defines a shared data space for passing data from CP0 to CP1, and the access controls are defined by the logical OR of the two region descriptors. Thus, CP0 has (rw- | r--) = (rw-) permissions, while CP1 has (--- | r--) = (r--) permission in this space. Both DMA engines are excluded from this shared processor data region. The overlapping spaces between RGD3 and RGD4 defines another shared data space, this one for passing data from CP1 to CP0. For this overlapping space, CP0 has (r-- | ---) = (r--) permission, while CP1 has (rw- | r--) = (rw-) permission. The non-overlapped space of RGD4 defines a private data and stack area for CP1 only.

The space defined by RGD5 is a shared data region, accessible by all four bus masters. Finally, the slave peripheral space mapped onto the peripheral bus is partitioned into two regions: one containing the SMPU's programming model accessible only to the two processor cores and the remaining peripheral region accessible to both processors and the traditional DMA1 master.

This simple example is intended to show one possible application of the capabilities of the SMPU in a typical system.

22 Intelligent AHB Gasket (IAHBG)

22.1 Introduction

This section details the intelligent bridging gasket between the fast and slow clock domains. The intelligent operation of the gasket initiates pending reads early and optimizes bursts to recover some of the performance that is lost on a registered interface between different clock domains. Pending reads begin even if there is not an hready to the master. Non-bursted writes only begin when the slave is IDLE, pending writes wait until IDLE.

The IAHBG complies with the AHBV6lite protocol, including an extension to support masters that do address retraction during the error termination sequence or abort a burst read due to an error termination or another non-error termination like cache inhibit termination.

Table 321 lists the available operation modes.

Table 321. Operation modes available

Operating mode (Master:Slave)	Comments
0:0	Flowthru (no register wall)
1:1	Master and slave at same operating frequency; gasket performance optimizations on by default
2:1	Master at twice frequency of slave; gasket performance optimizations on by default
1:2	Slave at twice frequency of master; gasket performance optimizations on by default
4:1	Master at four times frequency of slave; gasket performance optimizations disabled at this time
1:4	Slave at four times frequency of master; gasket performance optimizations disabled at this time

22.2 Timing modes

22.2.1 1:1

In the 1:1 timing mode the master is running at the same clock frequency as the slave. The pending read can either be a single or burst transaction.

22.2.2 2:1

In the 2:1 timing mode the master is running at twice the clock frequency of the slave. The 4:1 mode is very similar to the 2:1 mode. The pending read can either be a single or burst transaction.

22.2.3 1:2

In the 1:2 timing mode, the slave is running at twice the clock frequency of the master.The 1:4 mode is very similar to the 1:2 mode. The pending read can either be a single or burst transaction.

Because the slave is operating at a faster clock frequency than the master, the transactions from the master must be stored into a queue of data and response attribute registers. At a certain point during the master burst transactions, the queue is either emptied into the master for a burst read or it is emptied into the slave for a burst write.

23 Semaphores2 (SEMA42)

23.1 Introduction

The peripheral platform shell includes a memory-mapped module that provides robust hardware support needed in multi-core systems for implementing semaphores and provides a simple mechanism to achieve “lock and unlock” operations via a single write access. The hardware semaphore module is an architecture-neutral solution that provides hardware-enforced gates as well as other useful system functions related to the gating mechanisms.

23.1.1 Multi-core programming

Multi-processor systems require a function that can be used to safely and easily provide a locking mechanism that is then used by system software to control access to shared data structures, shared hardware resources, and so on. These gating mechanisms are used by the software to serialize (and synchronize) writes to shared data or resources, or both to prevent race conditions and preserve memory coherency between different processes and processors.

Consider the following description of a typical use-case. Processor X enters a section of code where shared data values are to be updated. It must first acquire a semaphore; this can be considered to be locking (or closing) a software gate. Once the gate has been locked, a properly architected software system does not allow other processes (or processors) to execute the same code segment or modify the shared data structure protected by the gate; in other words, other processes/processors are locked out. Many software implementations include a spin-wait loop within the lock function until the locking of the gate is accomplished. Once the lock has been obtained, processor X continues execution and updates the data values protected by the particular lock. Once the updates are complete, processor X unlocks (or opens) the software gate, allowing other processes/processors access to the updated data values.

There are three important rules that must be followed for a correctly implemented system solution:

1. All writes to shared data values or shared hardware resources must be protected by a “gate” variable.
2. Once a processor locks a gate, accesses to the shared data or resources by other processes/processors must be blocked. This is enforced by software conventions.
3. The processor that locks a particular gate is the only processor that can open (unlock) that gate.

Information in the hardware gate identifying the locking processor are useful for system-level debugging.

23.1.2 Features

The Semaphores module implements hardware-enforced semaphores as an IPS-mapped slave peripheral device. The feature set includes:

- Module definition supporting 16, 32, or 64 hardware-enforced gates in a multi-processor configuration, where up to 15 processors can be supported; cp*X* is meant to represent core processor *X*.
  - Gates appear as an *n*-entry byte-size array with read and write accesses (*n* = 16, 32, 64).
    - Processors lock gates by writing “processor_number+1” to the appropriate gate and must read back the gate value to verify the lock operation was successful.
    - Once locked, the gate is unlocked by a write of zeroes from the locking processor.
    - The number of implemented gates is specified by a hardware configuration define.
  - Each hardware gate appears as a 16-state, 4-bit state machine.
    - 16-state implementation

      if gate = 0x0, then state = unlocked

      if gate = 0x1, then state = locked by processor (master) 0

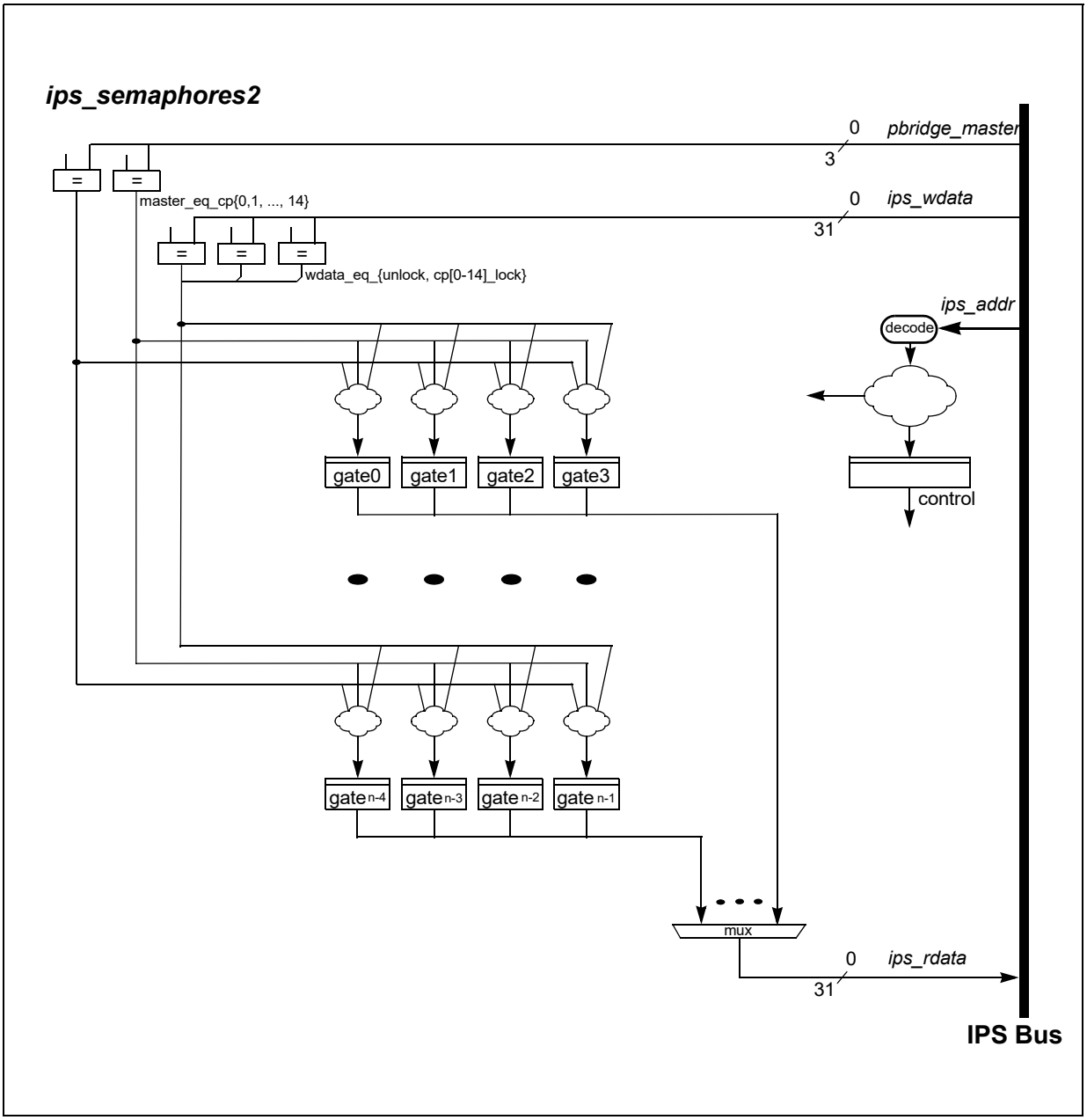
      if gate = 0x2, then state = locked by processor (master) 1

      ...

      if gate = 0xF, then state = locked by processor (master) 14
    - Uses the logical bus master number as a reference attribute plus the specified data patterns to validate all write operations.
    - Once locked, the gate can (and must) be unlocked by a write of zeroes from the locking processor.
  - Secure reset mechanisms are supported to clear the contents of individual gates, as well as a clear_all capability.
- Memory-mapped IPS slave peripheral platform module
  - Interface to the IPS bus for programming-model accesses

A simplified block diagram of the Semaphores module is shown in *Figure 214*. In the diagram, the register blocks named gate0, gate1,..., gate (*n* – 2), gate (*n* – 1) include the finite state machines implementing the semaphore gates.

Figure 214. Semaphores2 block diagram



23.2 Memory map/register definition

Table 322. Memory map/register definition

Address offset	Register name	Access	Reset value	Section/page
0x0000 + n*0x1	Semaphores2 Gate 0 – Semaphores2 Gate 15 (SEMA4_GATE0–SEMA4_GATE15)	R/W	0x00	*Section 23.2.1*
0x0010–0x003F	Reserved			

Table 322. Memory map/register definition (continued)

Address offset	Register name	Access	Reset value	Section/page
0x0040	Semaphores2 Reset Gate (SEMA4_RSTGT)	R/W	0x0000	*Section 23.2.2*
0x0042–0xFFFF	Reserved			

23.2.1 Semaphores2 Gate *n* Register (SEMA4_GATE*n*)

Each semaphore gate is implemented in a 4-bit finite state machine, right-justified in a byte data structure. The hardware uses the logical bus master number in conjunction with the data patterns to validate all attempted write operations. Only processor bus masters can modify the gate registers. Once locked, a gate can (and must) be opened (unlocked) by the locking processor core.

Multiple gate values can be read in a single access, but only a single gate can be updated via a write operation at a time. Attempted writes with a data value that is neither the unlock value nor the appropriate lock value (processor_number + 1) are simply treated as “no operation” and do not affect any gate state. Attempts to write multiple gates in a single aligned access with a size larger than an 8-bit (byte) reference generate an error termination and do not allow any gate state changes.

Offset: 0x0000 + *n**0x1 (*n* = 0 to 15) Access: Supervisor Read/Write

	0	1	2	3	4	5	6	7
R	0	0	0	0	GTFSM			
W								
Reset	0	0	0	0	0	0	0	0

Figure 215. Semaphores2 Gate *n* Register (SEMA4_GATE*n*)

Table 323. SEMA4_GATE*n* field descriptions

Field	Description
4:7 GTFSM	Gate Finite State Machine The hardware gate is maintained in a 16-state implementation, defined as: 0000 The gate is unlocked (free). 0001 The gate has been locked by processor 0. 0010 The gate has been locked by processor 1. 0011 The gate has been locked by processor 2. 0100 The gate has been locked by processor 3. 0101 The gate has been locked by processor 4. 0110 The gate has been locked by processor 5. 0111 The gate has been locked by processor 6. 1000 The gate has been locked by processor 7. 1001 The gate has been locked by processor 8. 1010 The gate has been locked by processor 9. 1011 The gate has been locked by processor 10. 1100 The gate has been locked by processor 11. 1101 The gate has been locked by processor 12. 1110 The gate has been locked by processor 13. 1111 The gate has been locked by processor 14. **Note:** The state of the gate reflects the last processor that locked it, which can be useful during system debug. For more details, see *Section 23.2.1: Semaphores2 Gate n Register (SEMA4_GATEn)*.

23.2.2 Semaphores2 (Secure) Reset Gate (SEMA4_RSTGT)

Although the intent of the hardware gate implementation specifies a protocol where the locking processor must unlock the gate, it is recognized that system operation may require a reset function to re-initialize the state of any gate(s) without requiring a system-level reset.

To support this special gate reset requirement, the Semaphores module implements a “secure” reset mechanism that allows a hardware gate (or all the gates) to be initialized by following a specific dual-write access pattern. Using a technique similar to that required for the servicing of a software watchdog timer, the secure gate reset requires two consecutive writes with predefined data patterns from the same processor to force the clearing of the specified gate(s). The required access pattern is:

1. A processor performs a 16-bit write to the SEMA4_RSTGT memory location. The most significant byte (SEMA4_RSTGT[RSTGDP]) must be 0xE2; the least significant byte is a “don’t_care” for this reference.
2. The same processor then performs a second 16-bit write to the SEMA4_RSTGT location. For this write, the upper byte (SEMA4_RSTGT[RSTGDP]) is the logical complement of the first data pattern (0x1D) and the lower byte (SEMA4_RSTGT[RSTGTN]) specifies the gate(s) to be reset. This gate field can specify a single gate be cleared, or else that all gates are to be cleared. If the same processor writes incorrect data on the second access or another processor performs the second write access, the special gate reset sequence is aborted and no error signal will be asserted.
3. Reads of the SEMA4_RSTGT location return information on the 2-bit state machine (SEMA4_RSTGT[RSTGSM]) that implements this function, the bus master performing the reset (SEMA4_RSTGT[RSTGMS]), and the gate number(s) last cleared

(SEMA4_RSTGT[RSTGTN]). Reads of the SEMA4_RSTGT register do not affect the secure reset finite state machine in any manner.

Offset: 0x0040 Access: Supervisor Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	RSTGSM		RSTGMS				RSTGTN							
W	RSTGDP															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 216. Semaphores2 (Secure) Reset Gate (SEMA4_RSTGT)

Table 324. SEMA4_RSTGT field descriptions

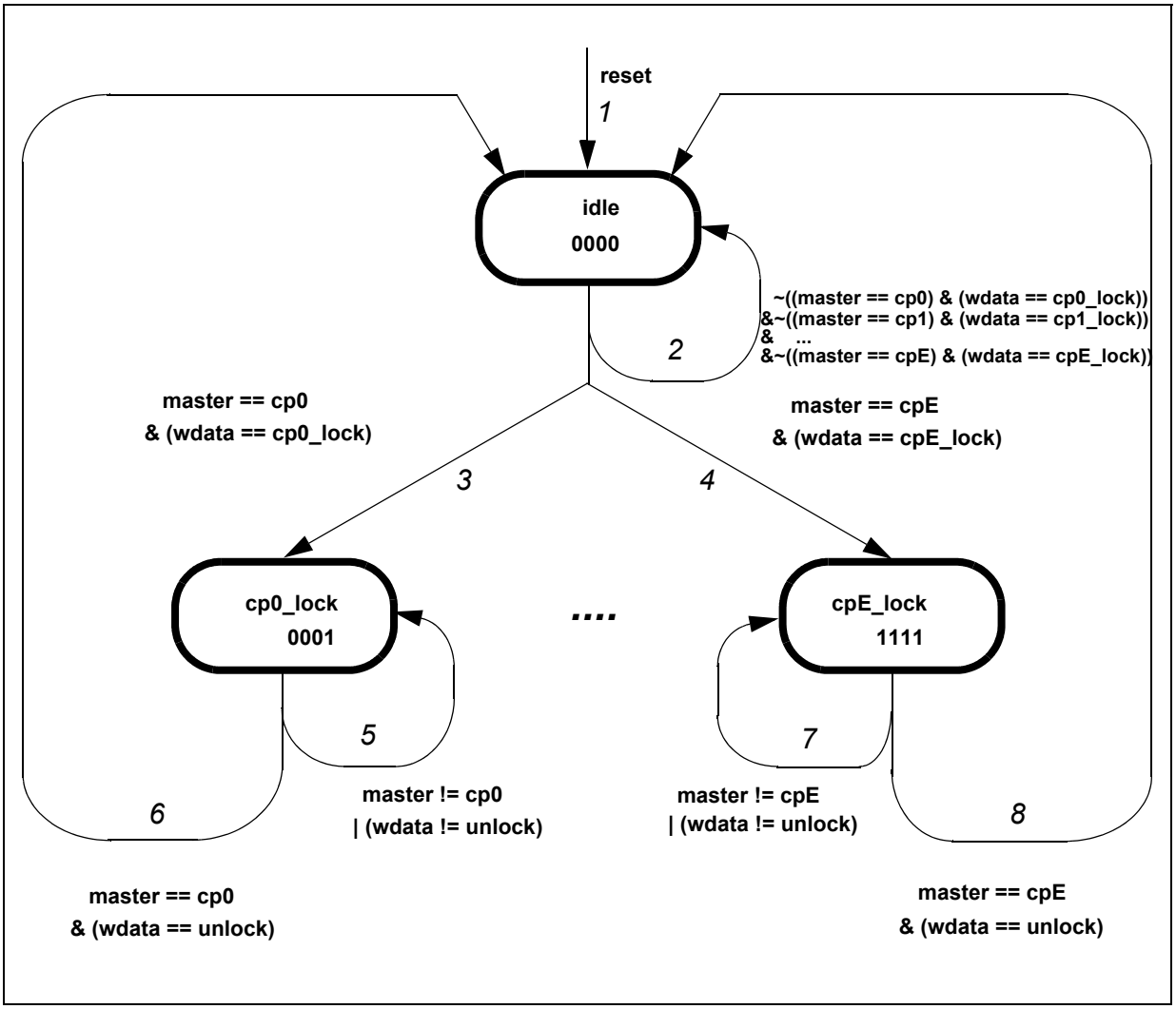
Field	Description
0:7 RSTGDP	Reset Gate Data Pattern This write-only field is accessed with the specified data patterns on the two consecutive writes to enable the gate reset mechanism. For the first write, RSTGDP=0xE2 while the second write requires RSTGDP=0x1D.
2:3 RSTGSM	Reset Gate Finite State Machine The reset state machine is maintained in a 2-bit, 3-state implementation, defined as: 00 Idle, waiting for the first data pattern write. 01 Waiting for the second data pattern write. 10 The 2-write sequence has completed. Generate the specified gate reset(s). After the reset is performed, this machine returns to the idle (waiting for first data pattern write) state. The “10” state persists for only one clock cycle. Software will never be able to observe this state. 11 This state encoding is never used and therefore reserved. Reads of the SEMA4_RSTGT register return the encoded state machine value. Note that the RSTGSM=10 state is valid for only a single machine cycle, so it is impossible for a read to return this value.
4:7 RSTGMS	Reset Gate Bus Master This 4-bit read-only field records the logical number of the bus master performing the gate reset function. The reset function requires that the two consecutive writes to this register must be initiated by the same bus master to succeed. This field is updated each time a write to this register occurs. The association between system bus master port numbers, the associated bus master device, and the logical processor number is SoC-specific. Consult the device reference manual for this information.
8:15 RSTGTN	Reset Gate Number This 8-bit field specifies the specific hardware gate to be reset. This field is updated by the second write. If RSTGTN < 64, then *reset the single gate* defined by RSTGTN, else *reset all the gates*.

23.3 Functional description

Functional operation of the Semaphores module and specific details of the state machines of the SEMA4_GATE*n* registers are provided as follows.

As described previously, each of the SEMA4_GATE*n* registers implements a 4-bit, 16-state machine. A *simplified* diagram of the state transitions for each gate is shown in *Figure 217*.

Figure 217. SEMA4_GATE*n* state machine



The bus master number is used to identify core processor *X* (cp*X*) where the notation cpE is used to represent core processor 14 (= 0xE). The platform passes the AHB bus master number through the PBRIDGE controller and drives the signal to the Semaphores module as an IPS sideband signal.

The state transitions for SEMA4_GATE*n* are defined in *Table 325*.

Table 325. SEMA4_GATE*n* state transitions

Current State	Next State	Transition	Description
—	idle	1	Any reset, whether a system reset or a software-initiated gate reset, unconditionally forces the gate into the idle state.
idle	idle	2	Unless a write of the appropriate lock value from the corresponding processor occurs, the gate remains in the idle state.

Table 325. SEMA4_GATE*n* state transitions (continued)

Current State	Next State	Transition	Description
idle	cp0_lock	3	When a write of the “cp0_lock” data value is initiated by processor 0, the gate transitions into the cp0_lock state.
idle	cpE_lock	4	When a write of the “cpE_lock” value is initiated by processor 0xE, the gate transitions into the cpE_lock state.
cp0_lock	cp0_lock	5	Once in this state, the gate remains here if any attempted write is not from cp0 with the unlock data value.
cp0_lock	idle	6	The gate returns to the idle (unlocked) state once a write from cp0 with the unlock data value occurs.
cpE_lock	cpE_lock	7	Once in this state, the gate remains here if any attempted write is not from cpE with the unlock data value.
cpE_lock	idle	8	The gate returns to the idle (unlocked) state once a write from cpE with the unlock data value occurs.

As previously noted, the following gate data values are used:

- The lock data value is (processor number + 1) where the processor number and the platform bus master number are the same.
- The unlock data value is 0x00.

24 Interrupt Controller (INTC)

24.1 Introduction

Note: For the chip-specific implementation details of this module's instances, refer to the Section 7.3.6: Interrupt controller (INTC) configuration.

The INTC:

- Provides priority-based preemptive scheduling of interrupt requests
- Schedules interrupt requests (IRQs) from software and internal peripherals to one or more processors (PRCs)
- Provides interrupt prioritization and preemption, interrupt masking, interrupt priority elevation, and protocol support

This scheduling scheme is suitable for statically scheduled hard real-time systems.

The INTC is targeted to work with a Power Architecture processor and is capable of processing high-demand interrupt sources where the Interrupt Service Routines (ISRs) nest to multiple levels, but it also can be used with other processors and applications.

For high-priority interrupt requests in these target applications, it is necessary to minimize the interval between the assertion of the interrupt request from the peripheral and the point at which the processor is performing useful work to service the interrupt request. The INTC supports this goal by providing a unique vector for each interrupt request source. It also provides 64 priorities so that lower priority ISRs do not delay the execution of higher priority ISRs. Since each individual application will have different priorities for each source of interrupt request, the priority of each interrupt request is configurable.

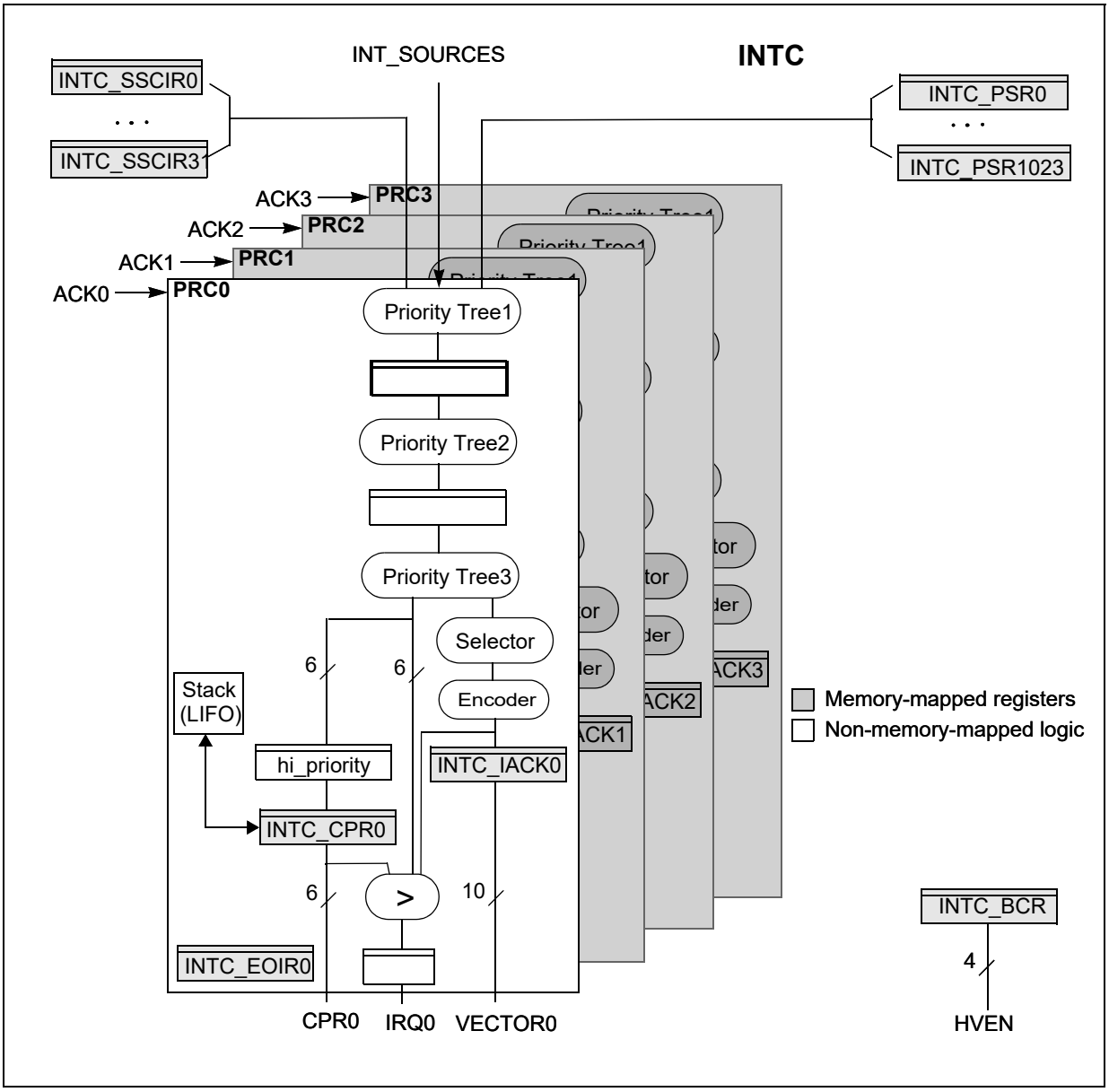
When multiple tasks share a resource, coherent accesses to that resource need to be supported. The INTC supports the Priority Ceiling Protocol for coherent accesses. By providing a modifiable priority mask, the priority can be raised temporarily so that all tasks which share the resource are unable to preempt each other.

Multiple processors can assert interrupt requests to each other through software-settable interrupt requests. These same software-settable interrupt requests can also be used to separate the work involved in servicing an interrupt request into two parts, a high-priority portion and a low-priority portion. The high-priority portion is initiated by a peripheral interrupt request, but then the ISR can assert a software-settable interrupt request to finish the servicing in a lower priority ISR. Therefore these software-settable interrupt requests can be used instead of having the peripheral ISR schedule a task through the RTOS.

24.2 Block diagram

The block diagram for the INTC is shown in *Figure 218*.

Figure 218. INTC block diagram



24.3 Features

- Each peripheral interrupt source is software-steerable to processors 0, 1 or 2 or any combination of interrupt request outputs
- 32 software-settable interrupt request sources
- 10-bit vector
  - Unique vector for each interrupt request source
  - Hardware connection to processor or read from register
- Each interrupt source can be programmed to one of 64 priorities
- Each interrupt source can be triggered by software
- Preemption
  - Preemptive prioritized interrupt requests to processor
  - ISR with higher priority preempts ISRs or tasks with lower priorities
  - Automatic pushing or popping of preempted priority to or from a LIFO
  - Ability to modify the ISR or task priority; modifying the priority can be used to implement the Priority Ceiling Protocol for accessing shared resources
  - 3 INTC clock cycles from interrupt request into interrupt request to CPU
- Low latency
  - 3 INTC clock cycles from receipt of interrupt request from peripheral to interrupt request to processor; four clock cycles from receipt of software request

24.4 Modes of operation

The interrupt controller operates with the following handshaking modes: software vector mode and hardware vector mode. The state of the hardware vector enable bit, INTC_BCR[HVEN*x*], determines which mode is used. In Power Architecture devices, software vector mode uses the autovector mode of the Processor for a single entry point for the ISR (16 bytes of vector space available), whereas the hardware vector mode uses the non-autovector mode where the vector offset from the INTC is provided to the Processor (4 bytes for each IRQ). In debug mode the interrupt controller operation is identical to its normal operation of software vector mode or hardware vector mode.

24.4.1 Software vector mode

In software vector mode, software (that is, the interrupt exception handler) must read a register in the INTC to obtain the vector associated with the interrupt request to the processor. The INTC will use software vector mode for a given processor when its associated HVEN*x* bit in INTC_BCR (refer to *Section 24.5.2.1*) is negated. The hardware vector enable signal to any processor is driven as negated when its associated HVEN*x* bit is negated. The vector is read from the INTC_IACKR*x* registers (refer to *Section 24.5.2.4*). Reading the INTC_IACKR*x* negates the interrupt request to the associated processor. Even if a higher priority interrupt request has arrived while waiting for this interrupt acknowledge, the interrupt request to the processor will negate for at least one clock. The reading also pushes the PRI value in INTC_CPR*x* (refer to *Section 24.5.2.3*) onto the associated LIFO, and updates PRI in the associated INTC_CPR*x* with the new priority.

24.4.2 Hardware vector mode

In hardware vector mode, the hardware causes the first instruction that will be executed (when handling the interrupt request to the processor) to be an instruction that is specific to that vector. Therefore the interrupt exception handler is specific to a peripheral or software-settable interrupt request, rather than being common to all of them. The INTC will use hardware vector mode for a given processor when its associated HVEN*x* bit in INTC_BCR is asserted. The hardware vector enable signal to the associated processor is driven as asserted.

When the interrupt request to the associated processor asserts, the interrupt vector signal is updated. The value of that interrupt vector is the unique vector associated with the preempting peripheral or software-settable interrupt request. The vector value matches the value of the INTVEC field in the INTC_IACKR*x*, depending on which processor was assigned to handle a given interrupt source.

The assertion of the interrupt acknowledge signal for a given processor pushes the associated PRI value in the associated INTC_CPR*x* register onto the associated LIFO and updates the associated PRI in the associated INTC_CPR*x* register with the new priority. This pushing of the PRI value onto the associated LIFO and updating PRI in the associated INTC_CPR*x* does not occur when the associated interrupt acknowledge signal asserts and the INTC_SSCIRs (refer to *Section 24.5.2.6: INTC Software Set/Clear Interrupt Register n (INTC_SSCIRn)*) is written at a time such that the PRI value in the associated INTC_CPR*x* register would need to be pushed and the previously last pushed PRI value would need to be popped simultaneously. In this case, PRI in the associated INTC_CPR*x* is updated with the new priority, and the associated LIFO is neither pushed nor popped.

24.5 Memory map and register definition

24.5.1 Memory map

Table 326 is the INTC memory map.

The actual availability and number of registers are chip-specific. For this information, refer to the chapter that describes how modules are configured and connected.

Table 326. INTC memory map

Address offset	Register	Location
0x0000	INTC Block Configuration Register (INTC_BCR)	*Section 24.5.2.1*
0x0004	INTC Master Protection Register (INTC_MPROT)	*Section 24.5.2.2*
0x0008–0x000F	Reserved	
0x0010	INTC Current Priority Register for Processor 0 (INTC_CPR0)	*Section 24.5.2.3*
0x0014	INTC Current Priority Register for Processor 1 (INTC_CPR1)	*Section 24.5.2.3*
0x0018	INTC Current Priority Register for Processor 2 (INTC_CPR2)	*Section 24.5.2.3*
0x001C–0x001F	Reserved	
0x0020	INTC Interrupt Acknowledge Register for Processor 0 (INTC_IACKR0)	*Section 24.5.2.4*
0x0024	INTC Interrupt Acknowledge Register for Processor 1 (INTC_IACKR1)	*Section 24.5.2.4*

Table 326. INTC memory map (continued)

Address offset	Register	Location
0x0028	INTC Interrupt Acknowledge Register for Processor 2 (INTC_IACKR2)	*Section 24.5.2.4*
0x002C–0x002F	Reserved	
0x0030	INTC End Of Interrupt Register for Processor 0 (INTC_EOIR0)	*Section 24.5.2.5*
0x0034	INTC End Of Interrupt Register for Processor 1 (INTC_EOIR1)	*Section 24.5.2.5*
0x0038	INTC End Of Interrupt Register for Processor 2 (INTC_EOIR2)	*Section 24.5.2.5*
0x003C–0x003F	Reserved	
0x0040–0x005F	INTC Software Set/Clear Interrupt Register 0–31 (INTC_SSCIR0–INTC_SSCIR31)	*Section 24.5.2.6*
0x0060–0x085F	INTC Priority Select Registers 0–1023 (INTC_PSR0–INTC_PSR1023)	*Section 24.5.2.7*
0x0860–0x0FFF	Reserved	

Note: A transfer error will be asserted if an access is attempted outside of the memory map or for any unimplemented registers. For example, if the design has only 512 interrupt sources and a source > 512 is accessed, a transfer error is asserted.

24.5.2 Register descriptions

With the exception of the INTC_SSCIR*n* and INTC_PSR*n* registers, all of the registers are 32-bit. Any combination of accessing the four bytes of a register with a single access is supported, provided that the access does not cross a register boundary. These supported accesses include types and sizes of 8 bits, aligned 16 bits, misaligned 16 bits to the middle two bytes, and aligned 32 bits.

Although INTC_SSIR*n* are 8 bits wide and INTC_PSR*n* are 16 bits wide, they can be accessed with a single 16-bit or 32-bit access, provided that the access does not cross a 32-bit boundary.

Some registers have specific exceptions to these rules, as outlined in their definitions.

In software vector mode, the effects of a read of the INTC Interrupt Acknowledge register (INTC_IACKR*x*) are the same regardless of the size of the read. In software or hardware vector mode, the size of a write to the INTC end-of-interrupt register (INTC_EOIR*x*) does not affect the operation of the write.

24.5.2.1 INTC Block Configuration Register (INTC_BCR)

The Block Configuration Register is used to configure options of the INTC.

The master ID is used to determine if this register can be written by a given processor. Only the processor with master ID zero will be allowed write access, otherwise, a termination error is asserted.

If INTC_MPROT[MPROT] = 0 (disabled), all processors have write access to this register.

If INTC_MPROT[MPROT] = 1 (enabled), the processor with master ID 0, 1 or 2 = INTC_MPROT[ID] has write access, otherwise, a termination error is asserted.

Offset: 0x0000 Access: User read/write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	Reserved	0	0	0	HVEN2	0	0	0	HVEN1	0	0	0	HVEN0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 219. INTC Block Configuration Register (INTC_BCR)

Table 327. INTC_BCR field descriptions

Field	Description
23 HVEN2	Hardware vector enable. Controls whether the INTC is in hardware vector mode or software vector mode. Refer to *Section 24.4*, for the details of the handshaking with the processor in each mode. 0 Software vector mode 1 Hardware vector mode
27 HVEN1	Hardware vector enable. Controls whether the INTC is in hardware vector mode or software vector mode. Refer to *Section 24.4*, for the details of the handshaking with the processor in each mode. 0 Software vector mode 1 Hardware vector mode
31 HVEN0	Hardware vector enable Controls whether the INTC is in hardware vector mode or software vector mode. Refer to *Section 24.4*, for the details of the handshaking with the processor in each mode. 0 Software vector mode 1 Hardware vector mode

24.5.2.2 INTC Master Protection Register (INTC_MPROT)

The Master Protection Register is used to protect cores from writing to other cores INTC registers.

Offset: 0x0004 Access: User read/write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	ID		0	0	0	MPROT
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 220. INTC Master Protection Register (INTC_MPROT)

Table 328. INTC_MPROT field descriptions

Field	Description
26:27 ID	ID This register can only be updated if MPROT=0 or if processor master ID=ID. When MPROT is set, the ID should also be updated to indicate which master can write to any register in the INTC memory map. 00 master0 (reset default) 01 master1 10 master2
31 MPROT	Master Protection Controls what programming space a given master can write to. If MPROT is disabled, then all masters can write to any valid entry of the INTC memory map. If MPROT is enabled, INTC_MPROT[ID] can write to any valid entry of the INTC memory map but other masters can only write to their assigned resources. Once MPROT is enabled by a master, only that master may modify INTC_MPROT. 0 Master programming space protection disabled 1 Master programming space protection enabled

24.5.2.3 INTC Current Priority Register for Processor *x* (INTC_CPR*x*)

The INTC_CPR*x* masks any peripheral or software-settable interrupt request that is set at the same or lower priority as the current value of the INTC_CPR*x*[PRI] field from generating an interrupt request to the processor. When the INTC Interrupt Acknowledge register (INTC_IACKR*x*) is read in software vector mode or the interrupt acknowledge signal from the processor is asserted in hardware vector mode, the value of PRI is pushed onto the LIFO, and PRI is updated with the priority of the preempting interrupt request. When the INTC end-of-interrupt register (INTC_EOIR*x*) is written, the LIFO is popped into the INTC_CPR*x* PRI field. An exception case in hardware vector mode to this behavior is described in *Section 24.4.2*.

The masking priority can be raised or lowered by writing to the PRI field, supporting the priority ceiling protocol. Refer to *Section 24.7.5*.

The master ID is used to determine if this register can be written by a given processor. The processor with master ID zero will be allowed write access. Only the corresponding processor will have write access, otherwise, a termination error is asserted.

If INTC_MPROT[MPROT] = 0 (disabled), all processors have write access to this register.

If INTC_MPROT[MPROT] = 1 (enabled), the processor with master ID 0, 1 or 2 = INTC_MPROT[ID] or the corresponding processor has write access, otherwise, a termination error is asserted.

Note: A store to modify the PRI field that closely precedes or follows an access to a shared resource may result in a non-coherent access to that resource. Refer to Section 24.7.5.2, for example code to ensure coherency.

0x0010 (INTC_CPR0)
Offset: 0x0014 (INTC_CPR1) Access: User read/write
0x0018 (INTC_CPR2)

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	PRI					
W																
Reset	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1

Figure 221. INTC Current Priority Register for Processor *x* (INTC_CPR*x*)

Table 329. INTC_CPR*x* field descriptions

Field	Description
26:31 PRI	Priority of the currently executing ISR, according to the field values 63 (highest priority) down to 0 (lowest priority).

24.5.2.4 INTC Interrupt Acknowledge Register for Processor *x* (INTC_IACKR*x*)

The Interrupt Acknowledge Register for Processor *x* provides a value which can be used to load the address of an ISR from a vector table. The vector table can be composed of addresses of the ISRs specific to their respective interrupt vectors.

Also, in software vector mode, the INTC_IACKR*x* has side effects from reads unless INTC_MPROT[MPROT] = 1 and processor with master ID 0, 1 or 2 != INTC_MPROT[ID]. Therefore, it must not be read speculatively while in this mode. The side effects are the same regardless of the size of the read. Reading the INTC_IACKR*x* does not have side effects in hardware vector mode.

The master ID is used to determine if this register can be written by a given processor. The processor with master ID zero will be allowed write access. Only the corresponding processor will have write access, otherwise, a termination error is asserted.

If INTC_MPROT[MPROT] = 0 (disabled), all processors have read and write access to this register.

If INTC_MPROT[MPROT] = 1 (enabled), the processor with master ID 0, 1 or 2 = INTC_MPROT[ID] or the corresponding processor has read and write access, otherwise, a termination error is asserted.

Offset: 0x0020 (INTC_IACKR0) / 0x0024 (INTC_IACKR1) / 0x0028 (INTC_IACKR2) — Access: User write always; non-speculative read(1)

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	VTBA															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	VTBA				INTVEC										0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. When the corresponding INTC_BCR[HVEN*x*] = 1, a read of the INTC_IACKR*x* has no side effects.

Figure 222. INTC Interrupt Acknowledge Register for Processor *x* (INTC_IACKR*x*)

Table 330. INTC_IACKR*x* field descriptions

Field	Description
0:19 VTBA	Vector table base address Must be set with the base address of the vector table prior to enabling interrupts.
20:29 INTVEC	Interrupt vector Vector of the peripheral or software-settable interrupt request that caused the interrupt request to the processor. When the interrupt request to the processor asserts, the INTVEC is updated, whether the INTC is in software or hardware vector mode.

24.5.2.5 INTC End of Interrupt Register for Processor *x* (INTC_EOIR*x*)

Writing to the INTC_EOIR*x* signals the end of the servicing of the interrupt request. When the INTC_EOIR*x* is written, the priority last pushed on the LIFO is popped into INTC_CPR*x*. The values and size of data written to the INTC_EOIR*x* are ignored. Those values and sizes written to this register neither update the INTC_EOIR*x* contents nor affect whether the LIFO pops. Typically, when you write to this register, write four all-zero bytes.

The master ID is used to determine if this register can be written by a given processor. The processor with master ID zero will be allowed write access. Only the corresponding processor will have write access, otherwise, a termination error is asserted.

If INTC_MPROT[MPROT] = 0 (disabled), all processors have write access to this register.

If INTC_MPROT[MPROT] = 1 (enabled), the processor with master ID 0, 1 or 2 = INTC_MPROT[ID] or the corresponding processor has write access, otherwise, a termination error is asserted.

Offset: 0x0030 (INTC_EOIR0) 0x0034 (INTC_EOIR1) 0x0038 (INTC_EOIR2) Access: User write-only

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R																
W	EOI															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R																
W	EOI															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 223. INTC End of Interrupt Register for Processor *x* (INTC_EOIR*x*)

Table 331. INTC_EOIR*x* field descriptions

Field	Description
0:31 EOI	End of Interrupt Write four all-zero bytes to this field to signal the end of the servicing of an interrupt request.

24.5.2.6 INTC Software Set/Clear Interrupt Register *n* (INTC_SSCIR*n*)

The INTC_SSCIR registers support the setting or clearing of software-settable interrupt requests. These registers contain independent sets of bits to set and clear a corresponding flag bit by software. Unlike the SWT bit in the PSRs, the INTC_SSCIR registers do not require a read-modify-write. With the exception of being set by software, this flag bit behaves the same as a flag bit set within a peripheral. This flag bit generates an interrupt request within the INTC just like a peripheral interrupt request. Writing a 1 to SET will leave SET unchanged at 0 but will set CLR. Writing a 0 to SET will have no effect. CLR is the flag bit. Writing a 1 to CLR will clear it. Writing a 0 to CLR will have no effect. If a 1 is written to a pair of SET and CLR bits at the same time, CLR is asserted, regardless of whether CLR was asserted before the write.

Figure 224 shows the structure of the first INTC_SSCIR. The other registers follow the same structure.

Any processor will be allowed to write to any of the SET bits in the INTC_SSCIR*n* registers. The CLR bit will only be writable by the processor which has been assigned to handle that interrupt request, as determined by the setting of the INTC_PSR*n*[PRC_SEL*x*]. An attempt by any other processor to write the CLR bit and not the SET bit will result in a termination error.

Note: Since the four software settable interrupts in a word can be assigned to different processors, the cores with non-zero master IDs must only write the INTC_SSCIRn registers using a byte operation. A write of any other size will result in a termination error.

Any processor will be allowed to write to any of the SET bits in the INTC_SSCIR*n* registers.

If INTC_MPROT[MPROT] = 0 (disabled), all processors can write to the CLR bit.

If INTC_MPROT[MPROT] = 1 (enabled), the processor with master ID 0, 1 or 2 = INTC_MPROT[ID] or the corresponding processor can write to the CLR bit and not the SET bit, otherwise, a termination error is asserted.

Note: If INTC_MPROT[MPROT] is enabled, processors with master ID != INTC_MPROT[ID] must only write the INTC_SSCIRn registers using a byte operation. A write of any other size will result in a termination error.

Offset: 0x0040 + *n**0x1 (*n* = 0 to 31) Access: User read/write

	0	1	2	3	4	5	6	7
R	0	0	0	0	0	0	0	CLR
W							SET	w1c
Reset	0	0	0	0	0	0	0	0

Figure 224. INTC Software Set/Clear Interrupt Register *n* (INTC_SSCIR*n*)

Table 332. INTC_SSCIR*n* field descriptions

Field	Description
6 SET	Set flag bit Writing a 1 will set the corresponding CLR bit. Writing a 0 will have no effect. Each SET is read as a 0.
7 CLR	Clear flag bit CLR is the flag bit. Writing a 1 to CLR will clear it provided that a 1 is not written simultaneously to its corresponding SET bit. Writing a 0 to CLR will have no effect. 0 Interrupt request not pending within INTC. 1 Interrupt request pending within INTC.

24.5.2.7 INTC Priority Select Register *n* (INTC_PSR*n*)

The Priority Select Registers support the selection of an individual priority for each source of interrupt request, and the routing of the request to one or more processors. The unique vector of each peripheral or software-settable interrupt request determines which INTC_PSR*n* is assigned to that interrupt request. The software-settable interrupt requests are assigned the lowest numbered vectors, and their priorities are configured in the lowest numbered INTC_PSR registers. The peripheral interrupt requests are assigned to vectors immediately following the software-settable interrupt requests, and their priorities are configured in the immediately following INTC_PSR registers. The peripheral interrupt requests are assigned vectors 32–1023, and their priorities are configured in INTC_PSR32–1023, respectively.

Figure 225 and *Figure 226* show the structure and numbering scheme of the first and last PSRs, respectively. The other registers follow the same numbering scheme.

Only the processor with master ID zero will be allowed to write the INTC_PSR*n*[PRC_SEL*x*,SWT]. When writing the INTC_PSR[PRI], only the processor with master ID 0 and the processor whose master ID matches the setting of the INTC_PSR*n*[PRC_SEL*x*] will be allowed to write the INTC_PSR*n*[PRI]. An attempt by any other processor to write the INTC_PSR*n*[PRI] will result in a termination error.

If INTC_MPROT[MPROT] = 0 (disabled), all processors have write access.

Note: If INTC_MPROT[MPROT] = 1 (enabled), the processor with master ID 0, 1 or 2 = INTC_MPROT[ID] or the corresponding processor has write access to INTC_PSRn[PRC_SELx,SWT], otherwise, a termination error is asserted. When writing the INTC_PSRn[PRI], only the processor with master ID 0, 1 or 2 = INTC_MPROT[ID] and the processor who's master ID 0, 1 or 2 matches the setting of the INTC_PSRn[PRC_SELx] will be allowed to write the INTC_PSRn[PRI]. An attempt by any other processor to write the INTC_PSRn[PRI] will result in a termination error.Unlike the peripheral interrupt request PSRs, there is no SWT bit in the PSRs corresponding to the software-settable interrupt requests.

Interrupts are not detected while the PSRs are being written.

Note: If INTC_MPROT[MPROT] is enabled, the core with master ID 0, 1 or 2 = INTC_MPROT[ID] will be able to write the INTC_PSRn with either byte, half-word or word operations. All other master IDs are restricted to byte operations to the INTC_PSRn[PRI] bits (the odd addresses). Any other sized write operation, or a byte operation to the INTC_PSRn[PRC_SELx] (the even byte) will result in a termination error.

Note: The core with master ID 0 will be able to write the INTC_PSRn with either byte, half-word or word operations. All other master IDs are restricted to byte operations to the INTC_PSRn[PRI] bits (the odd addresses). Any other sized write operation, or a byte operation to the INTC_PSRn[PRC_SELx] (the even byte) will result in a termination error.

Offset: 0x0060 + *n* * 0x2 (*n* = 0 to 31) Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	PRC_SEL0	PRC_SEL1	PRC_SEL2	Reserved	0	0	0	0	0	0	PRI					
W																
Reset	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 225. INTC Priority Select Register *n* (INTC_PSR*n*)

Table 333. INTC_PSR*n* field descriptions

Field	Description
0 PRC_SEL0	Processor 0 select bit If an interrupt source is enabled, PRC_SEL0 selects whether the interrupt request is to be sent to processor 0.
1 PRC_SEL1	Processor 1 select bit If an interrupt source is enabled, PRC_SEL1 selects whether the interrupt request is to be sent to processor 1.
2 PRC_SEL2	Processor 2 select bit If an interrupt source is enabled, PRC_SEL2 selects whether the interrupt request is to be sent to processor 2.
10:15 PRI	Priority select Selects the priority for the interrupt requests. PRI has values 63 (highest priority) down to 0 (lowest priority).

Offset: 0x0060 + *n* * 0x2 (*n*=32 to 1023) Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	PRC_SEL0	PRC_SEL1	PRC_SEL2	Reserved	0	0	0	SWT	0	0	PRI					
W																
Reset	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 226. INTC Priority Select Register *n* (INTC_PSR*n*)

Table 334. INTC_PSR*n* field descriptions

Field	Description
0 PRC_SEL0	Processor 0 select bit If an interrupt source is enabled, PRC_SEL0 selects whether the interrupt request is to be sent to processor 0.
1 PRC_SEL1	Processor 1 select bit If an interrupt source is enabled, PRC_SEL1 selects whether the interrupt request is to be sent to processor 1.
2 PRC_SEL2	Processor 2 select bit If an interrupt source is enabled, PRC_SEL2 selects whether the interrupt request is to be sent to processor 2.
7 SWT	This bit is only available in PSRs that do not correspond to the SSCIRs. The SWT bit supports triggering an interrupt by software rather than hardware. This is an alternative to setting a flag bit in a peripheral. This flag bit generates an interrupt request within the INTC just like a peripheral interrupt request. Writing a '1' to SWT*n* sets the bit. On the interrupt acknowledge cycle, the SWT*n* bit is cleared. The SWT function works both in hardware and software vector mode. It is only cleared by the enabled Processor(s) defined in the PRC_SEL*n* field of the PSR.
10:15 PRI	Priority select Selects the priority for the interrupt requests. PRI has values 63 (highest priority) down to 0 (lowest priority).

24.6 Functional description

The functional description involves the areas of interrupt request sources, priority management, and handshaking with the processor. In addition, spaces in the memory map have been reserved for other possible implementations of the INTC.

24.6.1 Interrupt request sources

The INTC has two types of interrupt requests, peripheral and software-settable. These interrupt requests can assert on any clock cycle.

The INTC has no spurious vector support. Therefore, if an asserted peripheral or software-settable interrupt request, whose PRI value in INTC_PSR*n* is higher than the PRI value in INTC_CPR*x*, negates before the interrupt request to the processor for that peripheral or software-settable interrupt request is acknowledged, then the interrupt request to the processor still can assert (or will remain asserted) for that peripheral or software-settable interrupt request. In this case, the interrupt vector will correspond to that peripheral or

software-settable interrupt request. Also, the PRI value in the associated INTC_CPR*x* is updated with the corresponding PRI value in INTC_PSR*n*.

Furthermore, clearing the peripheral interrupt request's enable bit in the peripheral, or (alternatively) setting its mask bit, has the same consequences as clearing its flag bit. Setting its enable bit or clearing its mask bit while its flag bit is asserted has the same effect on the INTC as an interrupt event setting the flag bit.

24.6.1.1 Peripheral interrupt requests

An interrupt event in a peripheral's hardware sets a flag bit which resides in that peripheral. The interrupt request from the peripheral is driven by that flag bit.

The time from when the peripheral starts to drive its peripheral interrupt request to the INTC, to the time that the INTC starts to drive the interrupt request to the processor, is three clocks.

24.6.1.2 Software-settable interrupt requests

The software set/clear interrupt registers (INTC_SSCIR*n*) support the setting or clearing of software-settable interrupt requests. These registers contain independent sets of bits to set and clear a corresponding flag bit by software. An interrupt request is triggered by software by writing a 1 to a SET bit in INTC_SSCIR*n*. This write sets the corresponding CLR bit, which is a flag bit, resulting in the interrupt request. The interrupt request is cleared by writing a 1 to the CLR bit. Specific behavior includes the following:

- Writing a 1 to SET leaves SET unchanged at 0 but sets the flag bit (which is the CLR bit).
- Writing a 0 to SET has no effect.
- Writing a 1 to CLR clears the flag (CLR) bit.
- Writing a 0 to CLR has no effect.
- If a 1 is written to a pair of SET and CLR bits at the same time, the flag (CLR) is set, regardless of whether CLR was asserted before the write.

The time from the write to the SET bit, to the time that the INTC starts to drive the interrupt request to the processor, is four clocks.

Any processor x will be allowed to write to any of the SET bits in the Software Set/Clear Interrupt registers. The CLR bit will only be writable by the processor x which has been assigned to handle that interrupt request, as determined by the setting of the INTC_PSR*n*[PRC_SEL*x*] bits.

24.6.1.3 Unique vector for each interrupt request source

Each peripheral and software-settable interrupt request is assigned a hardwired unique 10-bit vector. Software-settable interrupts 0–31 are assigned vectors 0–31, respectively. The peripheral interrupt requests are assigned vectors from 32 to a number as high as needed to cover all peripheral interrupt requests.

24.6.2 Priority management

The asserted interrupt requests are compared to each other based on their PRI*n* and PRC_SEL*x* values set in INTC_PSR*n*. The result of that comparison also is compared to PRI in the associated INTC_CPR*x*. The results of those comparisons are used to manage

the priority of the ISR being executed by the associated processor. The associated LIFO also assists in managing that priority.

24.6.2.1 Current priority and preemption

The priority arbitrator, selector, encoder, and comparator logic shown in *Figure 218* are used to compare the priority of the asserted interrupt requests to the current priority. If the priority of any asserted peripheral or software-settable interrupt request is higher than the current priority for a given processor, then the interrupt request to the processor is asserted. Also, a unique vector for the preempting peripheral or software-settable interrupt request is generated for the associated INTC_IACKR*x*, and if in hardware vector mode, for the interrupt vector provided to the processor.

24.6.2.1.1 Priority tree

The priority tree for each processor compares all the priorities of all of the asserted interrupt requests assigned to that processor, both peripheral and software-settable. The output of the priority tree is the highest of those priorities assigned to a given processor. Also, any interrupt requests which have this highest priority are output as asserted interrupt requests to the associated selector logic.

24.6.2.1.2 Selector

If only one interrupt request from the associated priority tree is asserted, then it is passed as asserted to the associated encoder logic. If multiple interrupt requests from the associated priority tree are asserted, then only the one with the lowest vector is passed as asserted to the associated encoder logic. The lower vector is chosen regardless of the time order of the assertions of the peripheral or software-settable interrupt requests.

24.6.2.1.3 Encoder

The encoder logic generates the unique 10-bit vector for the asserted interrupt request from the request selector subblock for the associated processor.

24.6.2.1.4 Comparator

The comparator logic compares the highest priority output from the associated priority arbitrator subblock with PRI in the associated INTC_CPR*x*. If the comparator detects that this highest priority is higher than the current priority, then it asserts the interrupt request to the associated processor. This interrupt request to the processor asserts whether this highest priority is raised above the value of PRI in the associated INTC_CPR*x*, or the PRI value in the associated INTC_CPR*x* is lowered below this highest priority. This highest priority then becomes the new priority which is written to PRI in the associated INTC_CPR*x* when the interrupt request to the processor is acknowledged. Interrupt requests whose PRI in INTC_PSR*n* are zero will not cause a preemption because their PRI will not be higher than PRI in the associated INTC_CPR*x*.

Another function of the comparator is to signal an update of the INTC_IACKR*x* with the vector number of the first interrupt that arrives that has a priority higher than the current priority. Once the vector number and priority are captured, they cannot be superseded by a higher priority interrupt until an update of the INTC_CPR*x* occurs. In software vector mode, higher priority interrupts can supersede the previously captured interrupt vector number and priority until the time a hardware or software interrupt acknowledge is processed.

24.6.2.2 Stack (LIFO)

The LIFO stores the preempted PRI values from the associated INTC_CPR*x*. Therefore, because these priorities are stacked within the INTC, if interrupts need to be enabled during the ISR, at the beginning of the interrupt exception handler the PRI value in the associated INTC_CPR*x* does not need to be loaded from the associated INTC_CPR*x* and stored onto the context stack. Likewise at the end of the interrupt exception handler, the priority does not need to be loaded from the context stack and stored in the associated INTC_CPR*x*.

The PRI value in the associated INTC_CPR*x* is pushed onto the LIFO when the associated INTC_IACKR*x* is read in software vector mode or the interrupt acknowledge signal from the associated processor is asserted in hardware vector mode. The priority is popped into PRI in the associated INTC_CPR*x* whenever the associated INTC_EOIR*x* is written. An exception case in hardware vector mode to this behavior is described in *Section 24.4.2*.

Although the INTC supports up to 64 priorities, an ISR executing with PRI in the INTC_CPR*x* equal to 63 will not be preempted. Therefore, the LIFO supports the stacking of 63 priorities. However, the LIFO is only 62 entries deep. An entry for a priority of 0 is not needed because of the ways that pushing onto a full LIFO and popping an empty LIFO are treated. If the LIFO is pushed 63 or more times than it is popped, the priorities first pushed are overwritten. A priority of 0 would be an overwritten priority. However, the LIFO will pop zeroes if it is popped more times than it is pushed. Therefore, although a priority of 0 was overwritten, it is regenerated with the popping of an empty LIFO.

24.6.3 Handshaking with processor

24.6.3.1 Software vector mode handshaking

24.6.3.1.1 Acknowledging interrupt request to processor

The software vector mode handshaking can be used with processors that only support an interrupt request to them, or processors that support both an interrupt request by itself as well as an interrupt request with an interrupt vector. The software vector mode handshaking even supports processors that always expect an interrupt vector with the interrupt request to them. Refer to *Figure 227*.

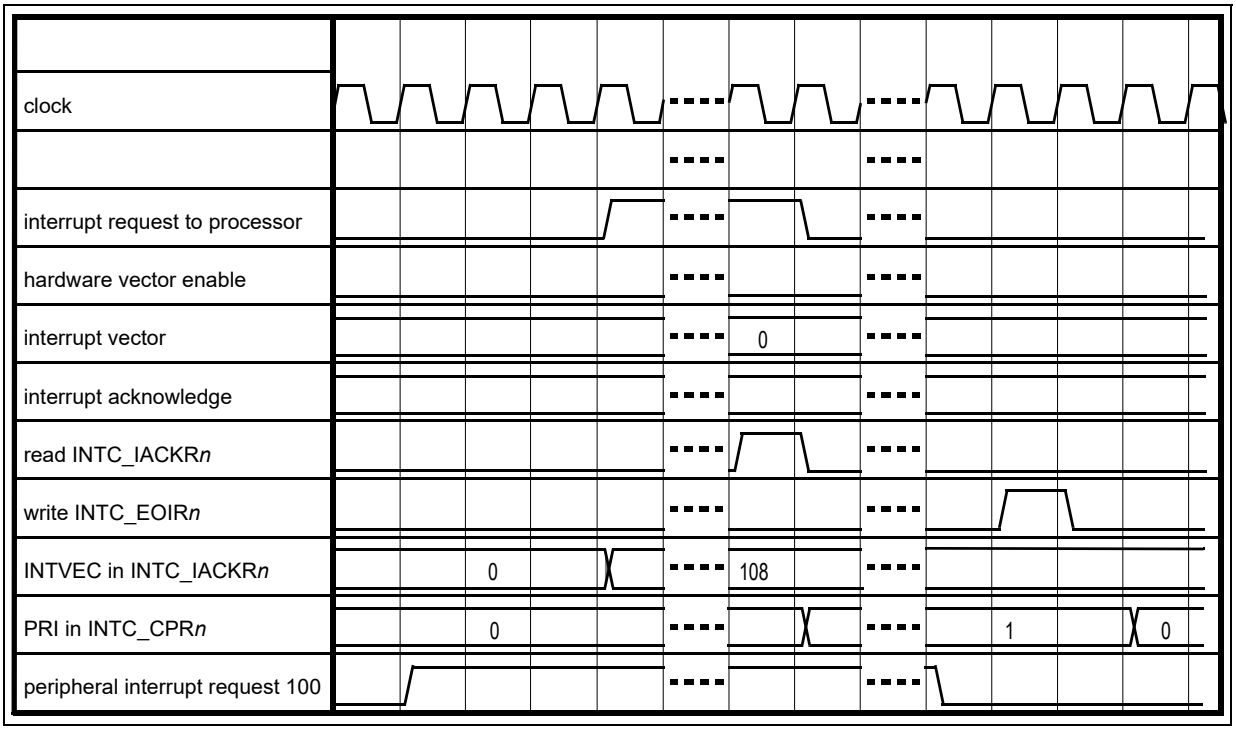
A timing diagram of the interrupt request and acknowledge handshaking in software vector mode, along with the handshaking near the end of the interrupt exception handler, is shown in *Figure 227*. The INTC examines the peripheral and software-settable interrupt requests. When it finds an asserted peripheral or software-settable interrupt request with a higher priority than PRI in the associated INTC_CPR*x*, it asserts the interrupt request to the associated processor. The INTVEC field in the associated INTC_IACKR*x* is updated with the preempting interrupt request's vector when the interrupt request to the processor is asserted. The INTVEC field retains that value until the next time the interrupt request to the processor is asserted. The rest of the handshaking is described in *Section 24.4.1*.

24.6.3.1.2 End of interrupt exception handler

Before the interrupt exception handling completes, INTC_EOIR*x* must be written. When it is written, the associated LIFO is popped so that the preempted priority is restored into PRI of the associated INTC_CPR*x*. Before it is written, the peripheral or software-settable flag bit must be cleared so that the peripheral or software-settable interrupt request is negated.

When returning from the preemption, the INTC does not search for the peripheral or software-settable interrupt request whose ISR was preempted. Depending on how much the ISR has progressed, that interrupt request may no longer even be asserted. When PRI in the associated INTC_CPR*x* is lowered to the priority of the preempted ISR, the interrupt request for the preempted ISR or any other asserted peripheral or software-settable interrupt request at or below that priority will not cause a preemption. Instead, after the restoration of the preempted context, the processor will return to the instruction address that it had been going to execute next, before it was preempted. This next instruction is part of the preempted ISR or the interrupt exception handler's prolog or epilog.

Figure 227. Timing diagram of software vector mode handshaking

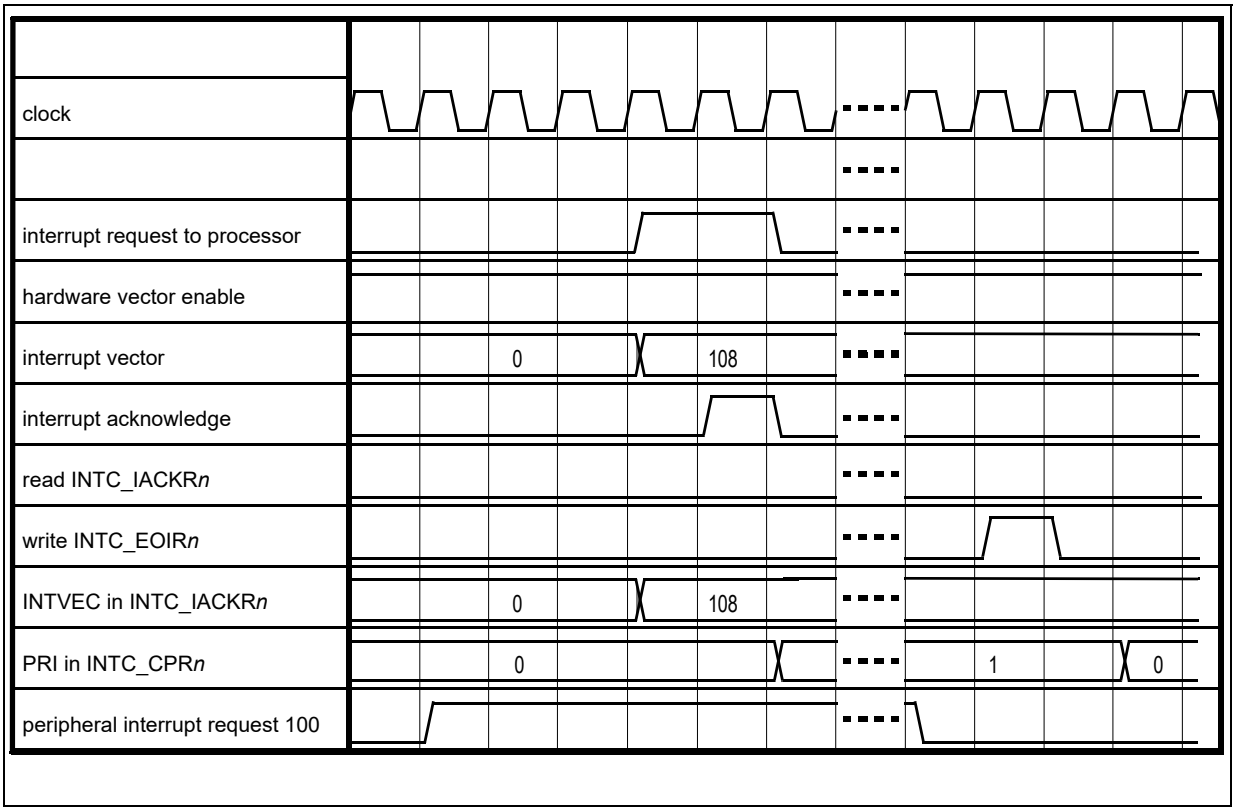


24.6.3.2 Hardware vector mode handshaking

A timing diagram of the interrupt request and acknowledge handshaking in hardware vector mode, along with the handshaking near the end of the interrupt exception handler, is shown in *Figure 228*. As in software vector mode, the INTC examines the peripheral and software-settable interrupt requests, and when it finds an asserted interrupt request with a higher priority than PRI in the associated INTC_CPR*x*, it asserts the interrupt request to the associated processor. The INTVEC field in the associated INTC_IACKR*x* is updated with the preempting peripheral or software-settable interrupt request's vector when the interrupt request to the processor is asserted. The INTVEC field retains that value until the next time the interrupt request to the associated processor is asserted. In addition, the value of the interrupt vector to the associated processor matches the value of the INTVEC field in the associated INTC_IACKR*x*. The rest of the handshaking is described in *Section 24.4.2*.

The handshaking near the end of the interrupt exception handler, that is the writing to the associated INTC_EOIR*x*, is the same as in software vector mode. Refer to *Section 24.6.3.1.2: End of interrupt exception handler*.

Figure 228. Timing diagram for hardware vector mode handshaking



24.7 Initialization/application information

24.7.1 Initialization flow

After exiting reset, all of the PRI and PRC_SEL*x* fields in INTC_PSR*n* are zero, and PRI in the INTC_CPR*x* registers is 31. These reset values will prevent the INTC from asserting the interrupt request to the processors. The enable or mask bits in the peripherals are reset such that the peripheral interrupt requests are negated. An initialization sequence for allowing the peripheral and software-settable interrupt requests to cause an interrupt request to the processor is:

- Interrupt_request_initialization:
  - Configure HVEN*x* in INTC_BCR.
  - Configure VTBA in INTC_IACKR*x.*
  - Raise the PRI fields and set the PRC_SEL*x* fields to the desired processor in INTC_PSR*n*.
  - Set the enable bits or clear the mask bits for the peripheral interrupt requests.
  - Lower PRI in INTC_CPR*x* to zero.
  - Enable processor(s) recognition of interrupts.

24.7.2 Interrupt exception handler

These example interrupt exception handlers excerpts use Power Architecture assembly code.

24.7.2.1 Software vector mode

In software vector mode for Power Architecture, there are 16 bytes of vector space available at the single ISR entry point.

```
interrupt_exception_handler:
b  interrupt_exception_handler_continued
   # 16 bytes available, branch to continue

interrupt_exception_handler_continued:
code to create stack frame, save working register, and save SRR0 and SRR1
(Power Architecture Save/Restore registers) (not shown)

lis     r3,INTC_IACKRx@ha    # form adjusted upper half of INTC_IACKRx
                             # address
lwz     r3,INTC_IACKRx@l(r3) # load INTC_IACKRx, which clears request to
                             # processor
lwz     r3,0x0(r3)           # load address of ISR from vector table
wrteei  1                    # enable processor recognition of interrupts

code to save rest of context required by Power Architecture EABI (Embedded
Binary Application Interface) (not shown)

mtlr    r3           # move the INTC_IACKRx address into the link register
blrl                 # branch to ISR; link register updated with epilog
                     # address

epilog:
code to restore most of context required by Power Architecture EABI (not
shown)

# Popping the LIFO after the restoration of most of the context and the
# disabling of processor recognition of interrupts eases the calculation of
# the maximum stack depth at the cost of postponing the servicing of the
# next interrupt request.
mbar                         # ensure store to clear flag bit has completed
lis     r3,INTC_EOIRx@ha     # form adjusted upper half of INTC_EOIRx
                             # address
li      r4,0x0               # form 0 to write to INTC_EOIRx
wrteei  0                    # disable processor recognition of interrupts
stw     r4,INTC_EOIRx@l(r3)  # store to INTC_EOIRx, informing INTC to lower
                             # priority
```

```
code to restore SRR0 and SRR1, restore working registers, and delete stack
frame (not shown)

rfi

vector_table_base_address:
address of ISR for interrupt with vector 0
address of ISR for interrupt with vector 1
   .
   .
   .
address of ISR for interrupt with vector 1022
address of ISR for interrupt with vector 1023

ISRn:
code to service the interrupt event (not shown)
code to clear flag bit which drives interrupt request to INTC (not shown)

blr       # return to epilog
```

24.7.2.2 Hardware vector mode

This interrupt exception handler is useful with processor and system bus implementations that support a hardware vector.

```
interrupt_exception_handlern:
b  interrupt_exception_handler_continuedn # 16 bytes available, branch to
                                          # continue

interrupt_exception_handler_continuedn:
code to create stack frame, save working register, and save SRR0 and SRR1
(not shown)

wrteei  1       # enable processor recognition of interrupts

code to save rest of context required by Power Architecture EABI (not shown)

bl      ISRn    # branch to ISR for interrupt with vector n

epilog:
code to restore most of context required by Power Architecture EABI (not
shown)

# Popping the LIFO after the restoration of most of the context and the
# disabling of processor recognition of interrupts eases the calculation of
# the maximum stack depth at the cost of postponing the servicing of the
# next interrupt request.
mbar                        # ensure store to clear flag bit has completed
```

```
lis     r3,INTC_EOIRx@ha      # form adjusted upper half of INTC_EOIRx
                              # address
li      r4,0x0                # form 0 to write to INTC_EOIRx
wrteei  0                     # disable processor recognition of interrupts
stw     r4,INTC_EOIRx@l(r3)   # store to INTC_EOIRn, informing INTC to lower
priority

SRR0 and SRR1, restore working registers, and delete stack frame (not shown)

rfi

ISRn:
code to service the interrupt event
code to clear flag bit which drives interrupt request to INTC

blr# branch to epilog
```

24.7.3 ISR, RTOS, and task hierarchy

The RTOS and all of the tasks under its control typically execute with PRI in INTC_CPR*x* having a value of 0. The RTOS will execute the tasks according to whatever priority scheme it may have, but that priority scheme is independent and has a lower priority of execution than the priority scheme of the INTC. In other words, the ISRs execute above INTC_CPR*x* priority 0 and outside the control of the RTOS while RTOS executes at INTC_CPR*x* priority 0, and while the tasks execute at different priorities under the control of the RTOS, they also execute at INTC_CPR*x* priority 0.

If a task shares a resource with an ISR and the Priority Ceiling Protocol (PCP) is being used to manage that shared resource, then the task's priority can be elevated in the INTC_CPR*x* while the shared resource is being accessed.

An ISR whose PRI in INTC_PSR*n* has a value of 0 will not cause an interrupt request to the selected processor, even if its peripheral or software-settable interrupt request is asserted. For a peripheral interrupt request, not setting its enable bit or disabling the mask bit will cause it to remain negated, which consequently also will not cause an interrupt request to the processor. Since the ISRs are outside the control of the RTOS, this ISR will not run unless called by another ISR or the interrupt exception handler, perhaps after executing another ISR.

24.7.4 Order of execution

An ISR with a higher priority can preempt an ISR with a lower priority, regardless of the unique vectors associated with each of their peripheral or software-settable interrupt requests. However, if multiple peripheral or software-settable interrupt requests are asserted, more than one of these interrupt requests has the highest priority and that priority is high enough to cause preemption, the INTC selects the one with the lowest unique vector regardless of the order in time that they asserted. However, the ability to meet deadlines with this scheduling scheme is no less than if the ISRs execute in the time order that their peripheral or software-settable interrupt requests asserted.

The example in *Table 335* shows the order of execution of both cases, ISRs with different priorities and ISRs with the same priority.

Table 335. Order of ISR execution example

		Code executing at end of step						
Step#	**Step Description**	**RTOS**	**ISR108[1]**	**ISR208**	**ISR308**	**ISR408**	**interrupt exception handler**	**PRI in INTC_CPR at end of step**
1	RTOS at priority 0 is executing.	X						0
2	Peripheral interrupt request 100 at priority 1 asserts. Interrupt taken.		X					1
3	Peripheral interrupt request 400 at priority 4 asserts. Interrupt taken.					X		4
4	Peripheral interrupt request 300 at priority 3 asserts.					X		4
5	Peripheral interrupt request 200 at priority 3 asserts.					X		4
6	ISR408 completes. Interrupt exception handler writes to INTC_EOIR*x*.						X	1
7	Interrupt taken. ISR208 starts to execute, even though peripheral interrupt request 300 asserted first.			X				3
8	ISR208 completes. Interrupt exception handler writes to INTC_EOIR*x*.						X	1
9	Interrupt taken. ISR308 starts to execute.				X			3
10	ISR308 completes. Interrupt exception handler writes to INTC_EOIR*x*.						X	1
11	ISR108 completes. Interrupt exception handler writes to INTC_EOIR*x*.						X	0
12	RTOS continues execution.	X						0

1. ISR108 executes for peripheral interrupt request 100 because the first eight ISRs are for software-settable interrupt requests.

24.7.5 Priority ceiling protocol

24.7.5.1 Elevating priority

The PRI field in INTC current priority register (INTC_CPR*x*) is elevated in the OSEK PCP to the ceiling of all of the priorities of the ISRs that share a resource. This protocol therefore allows coherent accesses of the ISRs to that shared resource.

For example, ISR1 has a priority of 1, ISR2 has a priority of 2, and ISR3 has a priority of 3. They all share the same resource. Before ISR1 or ISR2 can access that resource, they must raise the PRI value in INTC_CPR0 to *x*, the ceiling of all of the ISR priorities. After they release the resource, they must lower the PRI value in INTC_CPR*x* to prevent further priority inversion. If they do not raise their priority, then ISR2 can preempt ISR1, and ISR3 can preempt ISR1 or ISR2, possibly corrupting the shared resource. Another possible failure mechanism is deadlock if the higher priority ISR needs the lower priority ISR to release the resource before it can continue, but the lower priority ISR cannot release the resource until the higher priority ISR completes and execution returns to the lower priority ISR.

Using the PCP instead of disabling processor recognition of all interrupts eliminates the time when accessing a shared resource that all higher priority interrupts are blocked. For example, while ISR3 cannot preempt ISR1 while it is accessing the shared resource, all of the ISRs with a priority higher than 3 can preempt ISR1.

24.7.5.2 Ensuring coherency

24.7.5.2.1 Interrupt with blocked priority

A scenario can exist that can cause non-coherent accesses to the shared resource. As an example, ISR1 and ISR2 both share a resource. ISR1 has a lower priority than ISR2. ISR1 is executing, and it writes to the INTC_CPR*x*. The instruction following this store is a store to a value in a shared coherent data block. Either just before or at the same time as the first store, the INTC asserts the interrupt request to the processor because the peripheral interrupt request for ISR2 has asserted. As the processor is responding to the interrupt request from the INTC, and as it is aborting transactions and flushing its pipeline, it is possible (in some implementations) that both of these stores are executed. ISR2 thereby assumes that it can access the data block coherently, but the data block has been corrupted.

OSEK uses the GetResource and ReleaseResource system services to manage access to a shared resource. To prevent this corruption of a coherent data block, modifications to PRI in INTC_CPR*x* can be made by those system services with the code sequence:

Figure 229. OSEK Code Sequence

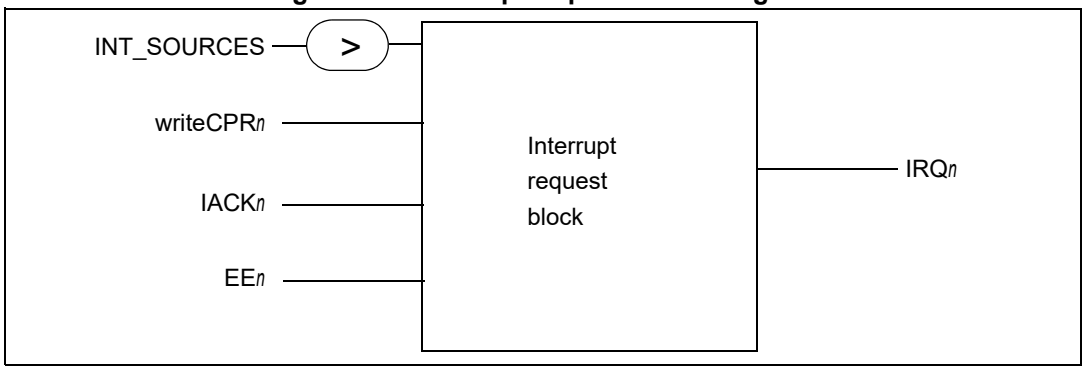
```
GetResource:
   wrteei  0       # disable external interrupts to the Processor
   raise   PRI     # Write to CPR, cache inhibit, guarded
   mbar            # flush out writes from store buffer
   wrteei  1       # enable external interrupts to the Processor
   isync           # re-fetch Processor pipeline

ReleaseResource:
   mbar            # flush out writes from store buffer
   wrteei  0       # disable external interrupts to the Processor
   lower   PRI     # Write to CPR, cache inhibit, guarded
   wrteei  1       # enable external interrupts to the Processor
```

24.7.5.2.1.1 Interrupt request to processor

Referencing *Figure 218*, the interrupt request logic to the processor is shown in *Figure 230*.

Figure 230. Interrupt request block diagram



Assuming that there are no IRQ*n* deasserted (no pending interrupt requests to processor *x*), if one of the external interrupt sources (INT_SOURCES) has a priority higher than the INTC_CPR*x*, then interrupt request to the processor (IRQ*n*) is asserted. It will stay asserted until one of the following conditions is true:

- Interrupt acknowledge (IACKR*x*) is asserted
- If the EE*n* bit has been cleared by the wrteei instruction (refer to *Figure 229*), IRQ*n* will be re-evaluated when processor *x* writes to the INTC_CPR*x* while processor *x* ignores IRQ*n*

This provides a safe way to guarantee that no interrupts will be recognized by processor x while updating the INTC_CPR*x*.

24.7.5.2.2 Raised priority preserved

Before the instruction and after the GetResource system service executes, all pending transactions are complete. These pending transactions can include an ISR for a peripheral or software-settable interrupt request whose priority was equal to or lower than the raised priority. Also, during the epilog of the interrupt exception handler for this preempting ISR, the raised priority has been restored from the LIFO to PRI in INTC_CPR*x*. The shared coherent data block now can be accessed coherently. *Figure 231* shows the timing diagram for this scenario, and *Table 336* explains the events. The example is for software vector mode. Except for the method of retrieving the vector and acknowledging the interrupt request to the processor, hardware vector mode is identical.

Figure 231. Timing diagram of raised priority preserved

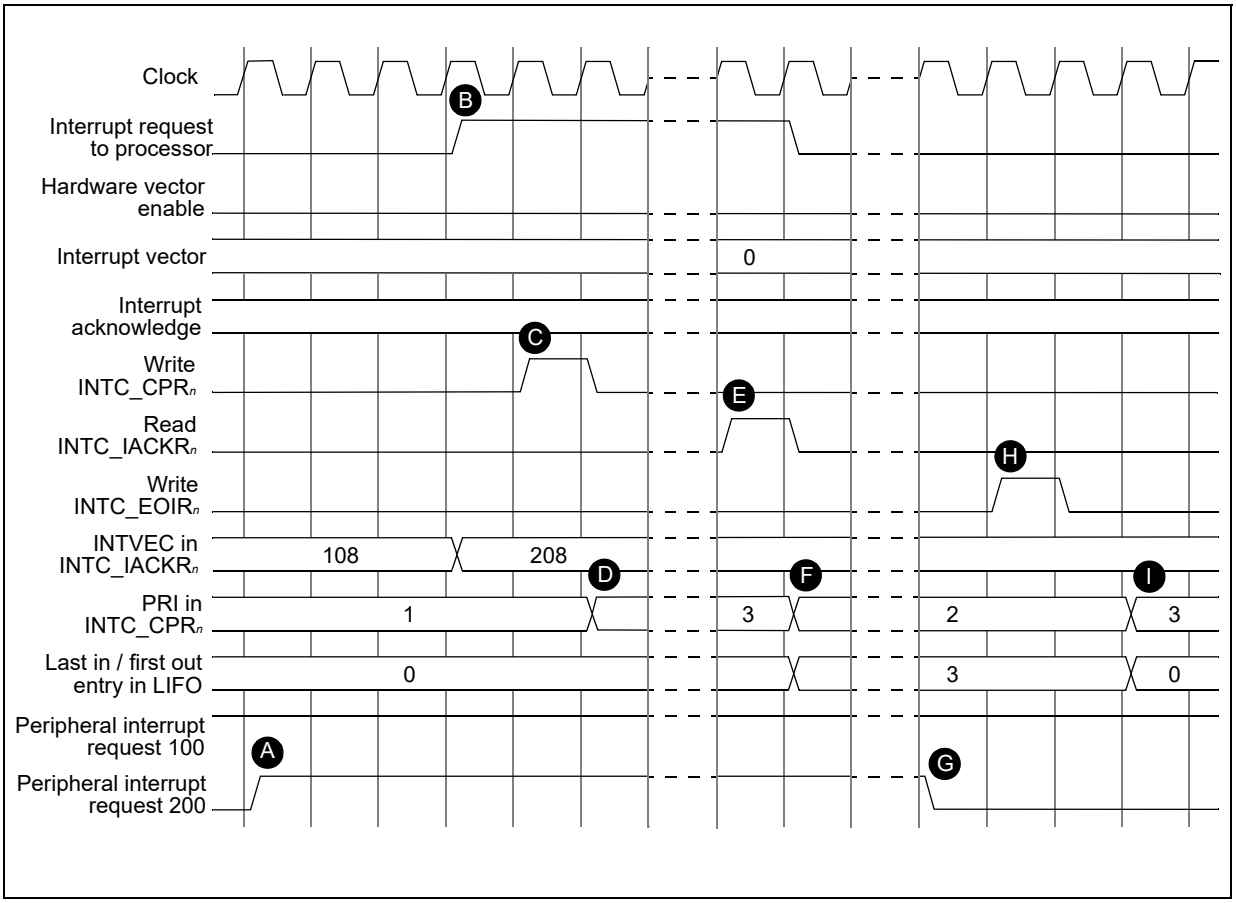


Table 336. Raised priority preserved events

Event	Description
A	Peripheral interrupt request with name equal to 200, asserts during execution of ISR108 running at priority 1.
B	Interrupt request to processor asserts. INTVEC in INTC_IACKR*x* updates with vector for that peripheral interrupt request.
C	ISR108 writes to INTC_CPR*x* to raise priority to 3 before accessing shared coherent data block.
D	PRI in INTC_CPR*x* now at 3, reflecting the write. This write, just before accessing data block, is the last instruction the processor executes before being interrupted.
E	Interrupt exception handler prolog acknowledges interrupt by reading INTC_IACKR*x*.
F	PRI of 3 pushed onto LIFO. PRI in INTC_CPR*x* updates to 2, the priority of ISR208.
G	ISR208 clears its flag bit, deasserting its peripheral interrupt request.
H	Interrupt exception handler epilog writes to INTC_EOIR*x*.
I	LIFO pops 3, restoring the raised priority onto PRI in INTC_CPR*x*. Next value to pop from LIFO is the priority from before peripheral interrupt request with name equal to 100 interrupted. ISR108 now can access data block coherently after interrupt exception handler executes rfi instruction.

24.7.6 Selecting priorities according to request rates and deadlines

The selection of the priorities for the ISRs can be made using Rate Monotonic Scheduling (RMS) or a superset of it, Deadline Monotonic Scheduling (DMS). In RMS, the ISRs which have higher request rates have higher priorities. In DMS, if the deadline is before the next time the ISR is requested, then the ISR is assigned a priority according to the time from the request for the ISR to the deadline, not from the time of the request for the ISR to the next request for it.

For example, ISR1 executes every 100 µs, ISR2 executes every 200 µs, and ISR3 executes every 300 µs. ISR1 has a higher priority than ISR2 which has a higher priority than ISR3. However, if ISR3 has a deadline of 150 µs, then it has a higher priority than ISR2.

The INTC supports 64 priorities, which may be many fewer than the number of ISRs. In this case, the ISRs should be grouped with other ISRs that have similar deadlines. For example, a priority could be allocated for every time the request rate doubles. ISRs with request rates around 1 ms would share a priority, ISRs with request rates around 500 µs would share a priority, ISRs with request rates around 250 µs would share a priority, and so on. With this approach, a range of ISR request rates of 2^{16} could be covered, regardless of the number of ISRs.

Reducing the number of priorities does cause some priority inversion, which reduces the processor's ability to meet its deadlines. However, reducing the number of priorities can reduce the size and latency through the interrupt controller. It also allows easier management of ISRs with similar deadlines that share a resource. They can be placed at the same priority without any further priority inversion, and they do not need to use the PCP to access the shared resource.

24.7.7 Software-settable interrupt requests

The software-settable interrupt requests can be used in two ways. They can be used to schedule a lower priority portion of an ISR and for processors to interrupt other processors in a multiple processor system.

24.7.7.1 Scheduling a lower priority portion of an ISR

A portion of an ISR needs to be executed at the PRI value in INTC_PSR*n*, which becomes the PRI value in INTC_CPR*x* with the interrupt acknowledge. The ISR, however, can have a portion of it which need not be executed at this higher priority. Therefore, executing this later portion which need not be executed at this higher priority can prevent the execution of ISRs which do not have a higher priority than the earlier portion of the ISR but do have a higher priority than the later portion of the ISR needs. This preemptive scheduling inefficiency reduces the processor's ability to meet its deadlines.

One option is for the ISR to complete the earlier higher priority portion, but then schedule through the RTOS a task to execute the later lower priority portion. However, some RTOSs can require a large amount of time for an ISR to schedule a task. Therefore, a second option is for the ISR, after completing the higher priority portion, to set a SET bit in INTC_SSCIR*n*. Writing a 1 to SET causes a software-settable interrupt request. This software-settable interrupt request, which usually will have a lower PRI value in the INTC_PSR*n*, therefore will not cause preemptive scheduling inefficiencies.

24.7.7.2 Scheduling an ISR on another processor

Since the SET bits in the INTC_SSCIR*n* are memory mapped, processors in multiple processor systems can schedule ISRs on the other processors. One possible application is if one processor simply wants to command another processor to perform a piece of work, and the initiating processor does not need to use the results of that work. If the initiating processor needs to know if the processor executing the software-settable ISR has not completed the work before asking it to again execute that ISR, it can check if the corresponding CLR bit in INTC_SSCIR*n* is asserted before again writing a 1 to the SET bit.

Another application is the sharing of a block of data. For example, a first processor has completed accessing a block of data and wants a second processor to then access it. Furthermore, after the second processor has completed accessing the block of data, the first processor again wants to access it. The accesses to the block of data must be done coherently. The procedure is that the first processor writes a 1 to a SET bit on the second processor. The second processor, after accessing the block of data, clears the corresponding CLR bit and then writes 1 to a SET bit on the first processor, informing it that it now can access the block of data.

24.7.8 Lowering priority within an ISR

In implementations without the software-settable interrupt requests in INTC_SSCIR*n*, one way (besides scheduling a task through an RTOS) to prevent preemptive scheduling inefficiencies with an ISR whose work spans multiple priorities (as described in *Section 24.7.7.1*) is to lower the current priority. However, the INTC has a LIFO whose depth is determined by the number of priorities.

Note: Lowering the PRI value in INTC_CPRx within an ISR to below the ISR's corresponding PRI value in INTC_PSRn allows more preemptions than the depth of the LIFO can support.

Therefore, through its use of the LIFO the INTC does not support lowering the current priority within an ISR as a way to avoid preemptive scheduling inefficiencies.

24.7.9 Negating an interrupt request outside of its ISR

24.7.9.1 Negating an interrupt request as a side effect of an ISR

Some peripherals have flag bits which can be cleared as a side effect of servicing a peripheral interrupt request. For example, reading a specific register can clear the flag bits, and consequently their corresponding interrupt requests too. This clearing as a side effect of servicing a peripheral interrupt request can cause the negation of other peripheral interrupt requests besides the peripheral interrupt request whose ISR presently is executing. This negating of a peripheral interrupt request outside of its ISR can be a desired effect.

24.7.9.2 Negating multiple interrupt requests in one ISR

An ISR can clear other flag bits besides its own flag bit. One reason that an ISR clears multiple flag bits is because it serviced those other flag bits, and therefore the ISRs for these other flag bits do not need to be executed.

24.7.9.3 Proper setting of interrupt request priority

Whether an interrupt request negates outside of its own ISR due to the side effect of an ISR execution or the intentional clearing of a flag bit, the priorities of the peripheral or software-settable interrupt requests for these other flag bits must be selected properly. Their PRI

values in INTC_PSR*n* must be selected to be at or lower than the priority of the ISR that cleared their flag bits. Otherwise, those flag bits still can cause the interrupt request to the processor to assert. Furthermore, the clearing of these other flag bits also has the same timing relationship to the writing to INTC_SSCIR*n* as the clearing of the flag bit that caused the present ISR to be executed. Refer to *Section 24.6.3.1.2* for more information.

A flag bit whose enable bit or mask bit is negating its peripheral interrupt request can be cleared at any time, regardless of the peripheral interrupt request's PRI value in INTC_PSR*n*.

24.7.10 Examining LIFO contents

Normally the user does not need to know the contents of the LIFO, or even how deep the LIFO is nested. Although the LIFO contents are not memory-mapped, the user can read the contents by popping the LIFO and reading the PRI field in the INTC current priority register (INTC_CPR*x*). Disabling processor recognition of interrupts while examining the LIFO contents provides a coherent view of the preempted priorities. The code sequence is:

```
pop_lifo:
   store to INTC_EOIRx
   load INTC_CPRx, examine PRI, and store onto stack
   if PRI is not zero or value when interrupts were enabled, branch to
pop_lifo
```

When the examination is complete, the LIFO can be restored using this code sequence:

```
push_lifo:
   load stacked PRI value and store to INTC_CPRx
   load INTC_IACKRx
   if stacked PRI values are not depleted, branch to push_lifo
```

Note: Reading the INTC_IACKRx acknowledges the interrupt request to the processor and updates the INTC_CPRx[PRI] with the priority of the preempting interrupt request. If the processor recognition of interrupts is disabled during the LIFO restoration, interrupt requests to the processor can go undetected. However, because the peripheral or software-settable interrupt requests are not cleared, the peripheral interrupt request to the processor re-asserts when INTC_CPRx[PRI] is lower than the priorities of those peripheral or software-settable interrupt requests.

24.8 Interrupt sources

The list of interrupt sources is chip-specific. For this list, refer to the chapter that describes how modules are configured and connected.

25 Enhanced Direct Memory Access (eDMA)

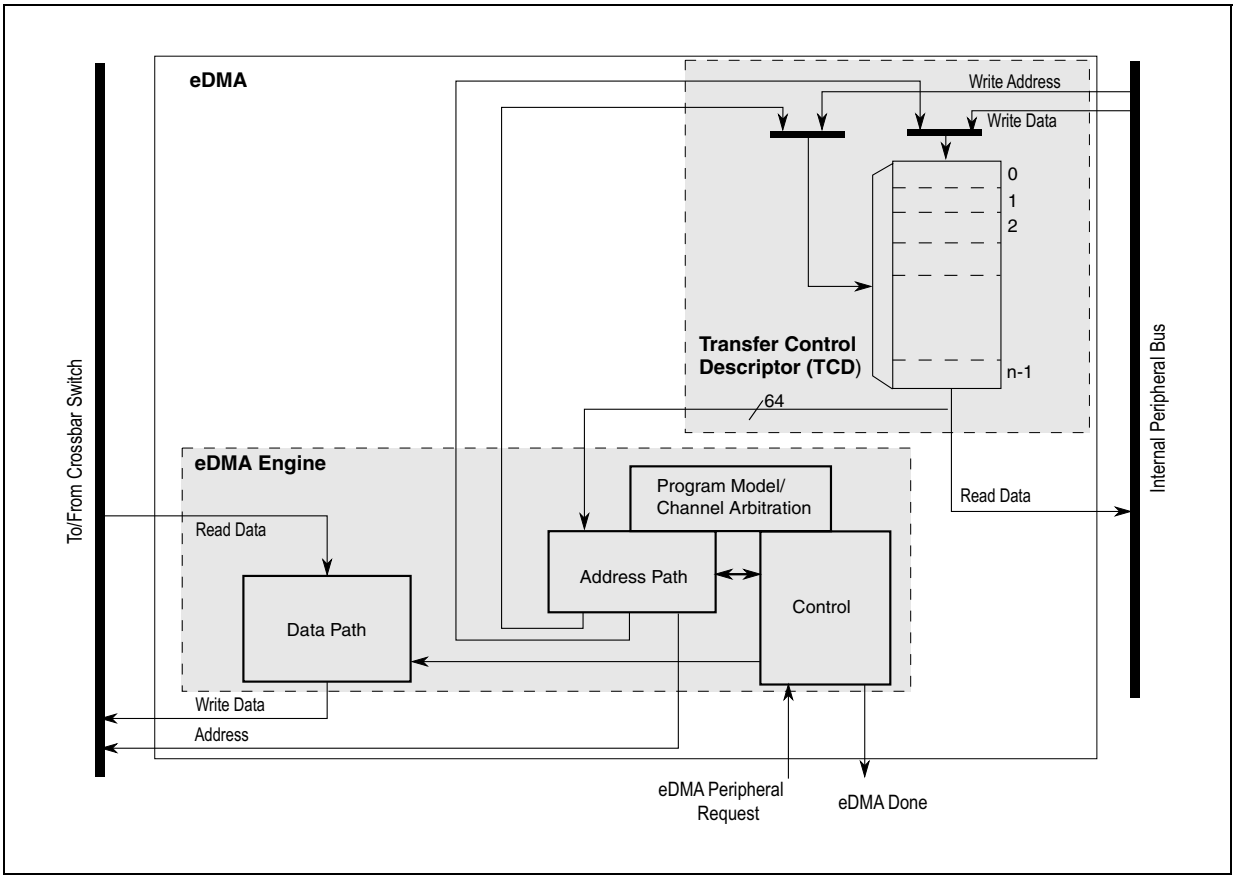
Note: For the chip-specific implementation details of this module's instances, please refer to the device configuration section.

25.1 Introduction

The enhanced direct memory access (eDMA) controller is a second-generation module capable of performing complex data transfers with minimal intervention from a host processor. The hardware microarchitecture includes a DMA engine that performs source- and destination-address calculations, and the actual data-movement operations, along with local memory containing transfer control descriptors for each of the 64 channels.

Figure 232 illustrates the eDMA module.

Figure 232. eDMA block diagram



25.1.1 Features

The eDMA is a highly-programmable data-transfer engine optimized to minimize the intervention from the host processor. It is intended for use in applications where the data

size to be transferred is statically known and not defined within the data packet itself. The eDMA module features:

- Data movements via dual-address transfers: read from source, write to destination:
  - Programmable source and destination addresses and transfer size, plus support for enhanced addressing modes.
- 64-channel implementation that performs complex data transfers with minimal intervention from a host processor.
  - Internal data buffer, used as temporary storage to support 16- and 32-byte burst transfers.
  - Connections to the crossbar switch for bus mastering the data movements.
- Transfer control descriptor (TCD) organized to support two-deep, nested transfer operations.
  - 32-byte TCD stored in local memory for each channel.
  - An inner data transfer loop defined by a minor byte transfer count.
  - An outer data transfer loop defined by a major iteration count.
- Channel activation via one of three methods:
  - Explicit software initiation.
  - Initiation via a channel-to-channel linking mechanism for continuous transfers.
  - Peripheral-paced hardware requests (one per channel).
- Fixed-priority and round-robin channel arbitration.
- Channel completion reported via optional interrupt requests.
  - One interrupt per channel, optionally asserted at completion of major iteration count.
  - Optional error terminations per channel and logically summed together to form one error interrupt to the interrupt controller.
- Optional support for scatter/gather DMA processing.
- Support for complex data structures.
- Support to cancel transfers via software.

Throughout this chapter, *n* is used to reference the channel number.

25.2 Modes of operation

The eDMA supports the following modes of operation.

25.2.1 Normal mode

In Normal mode, the eDMA transfers data between a source and a destination. The source and destination can be a memory block or an I/O block capable of operation with the eDMA.

A service request initiates a transfer of a specific number of bytes (NBYTES) as specified in the transfer control descriptor (TCD). The minor loop is the sequence of read-write operations that transfers these NBYTES per service request. Each service request executes one iteration of the major loop, which transfers NBYTES of data.

25.2.2 Debug mode

DMA operation is configurable in Debug mode via the control register:

- If CR[EDBG] is cleared, the DMA continues to operate.
- If CR[EDBG] is set, the eDMA stops transferring data. If Debug mode is entered while a channel is active, the eDMA continues operation until the channel retires.

25.2.3 Wait mode

Before entering Wait mode the DMA attempts to complete its current transfer. After the transfer completes the device enters Wait mode.

25.3 Memory map/register definition

The eDMA's programming model is partitioned into two regions:

- The first region defines a number of registers providing control functions.
- The second region corresponds to the local transfer control descriptor memory.

Some registers are implemented as two 32-bit registers, and include an H and L suffix, signaling the high and low portions of the control function. *Table 337* provides a 32-bit view of the eDMA's memory map.

Each channel requires a 32-byte transfer control descriptor for defining the desired data movement operation. The channel descriptors are stored in the local memory in sequential order: channel 0 to channel 63. Each TCD*n* definition is presented as 11 registers of 16 or 32 bits.

Reading reserved bits in a register returns the value of zero and writes to reserved bits in a register are ignored. Reading or writing a reserved memory location generates a bus error.

Table 337. DMA memory map

Address offset	Register name	Location
0x0000	Control Register (eDMA_CR)	*Section 25.3.1*
0x0004	Error Status Register (eDMA_ES)	*Section 25.3.2*
0x0008	Enable Request Register High (eDMA_ERQH)	*Section 25.3.3*
0x000C	Enable Request Register Low (eDMA_ERQL)	*Section 25.3.4*
0x010	Enable Error Interrupt Register High (eDMA_EEIH)	*Section 25.3.5*
0x0014	Enable Error Interrupt Register Low (eDMA_EEIL)	*Section 25.3.6*
0x0018	Set Enable Request Register (eDMA_SERQ)	*Section 25.3.7*
0x0019	Clear Enable Request Register (eDMA_CERQ)	*Section 25.3.8*
0x001A	Set Enable Error Interrupt Register (eDMA_SEEI)	*Section 25.3.9*
0x001B	Clear Enable Error Interrupt Register (eDMA_CEEI)	*Section 25.3.10*
0x001C	Clear Interrupt Request Register (eDMA_CINT)	*Section 25.3.11*
0x001D	Clear Error Register (eDMA_CERR)	*Section 25.3.12*
0x001E	Set START Bit Register (eDMA_SSRT)	*Section 25.3.13*

Table 337. DMA memory map (continued)

Address offset	Register name	Location
0x001F	Clear DONE Status Bit Register (eDMA_CDNE)	*Section 25.3.14*
0x0020	Interrupt Request Register High (eDMA_INTH)	*Section 25.3.15*
0x0024	Interrupt Request Register Low (eDMA_INTL)	*Section 25.3.16*
0x0028	Error Register High (eDMA_ERRH)	*Section 25.3.17*
0x002C	Error Register Low (eDMA_ERRL)	*Section 25.3.18*
0x0030	Hardware Request Status Register High (eDMA_HRSH)	*Section 25.3.19*
0x0034	Hardware Request Status Register Low (eDMA_HRSL)	*Section 25.3.20*
0x0038–0x00FF	Reserved	
eDMA channel registers per channel (*n* = 0 to 63)		
0x0100 + *n**0x1	Channel *n* Priority Register (eDMA_DCHPRI*n*)	*Section 25.3.21*
0x0140 + *n**0x1	Channel *n* Master ID Register (eDMA_DCHMID*n*)	*Section 25.3.22*
0x0180–0x0FFF	Reserved	
0x1000+*n**0x20	TCD Source Address (eDMA_TCD*n*_SADDR)	*Section 25.3.23*
0x1004+*n**0x20	TCD Transfer Attributes (eDMA_TCD*n*_ATTR)	*Section 25.3.24*
0x1006+*n**0x20	TCD Signed Source Address Offset (eDMA_TCD*n*_SOFF)	*Section 25.3.25*
0x1008+*n**0x20	TCD Minor Byte Count (Minor Loop Disabled) (eDMA_TCD*n*_NBYTES_MLNO)	*Section 25.3.26*
	TCD Signed Minor Loop Offset (Minor Loop Enabled and Offset Disabled) (eDMA_TCD*n*_NBYTES_MLOFFNO)	*Section 25.3.27*
	TCD Signed Minor Loop Offset (Minor Loop and Offset Enabled) (eDMA_TCD*n*_NBYTES_MLOFFYES)	*Section 25.3.28*
0x100C+*n**0x20	TCD Last Source Address Adjustment (eDMA_TCD*n*_SLAST)	*Section 25.3.29*
0x1010+*n**0x20	TCD Destination Address (eDMA_TCD*n*_DADDR)	*Section 25.3.30*
0x1014+*n**0x20	TCD Current Minor Loop Link, Major Loop Count (Channel Linking Enabled) (eDMA_TCD*n*_CITER_ELINKYES)	*Section 25.3.31*
	TCD Current Minor Loop Link, Major Loop Count (Channel Linking Disabled) (eDMA_TCD*n*_CITER_ELINKNO)	*Section 25.3.32*
0x1016+*n**0x20	TCD Signed Destination Address Offset (eDMA_TCD*n*_DOFF)	*Section 25.3.33*
0x1018+*n**0x20	TCD Last Destination Address Adjustment/Scatter Gather Address (eDMA_TCD*n*_DLASTSGA)	*Section 25.3.34*
0x101C+*n**0x20	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Enabled) (eDMA_TCD*n*_BITER_ELINKYES)	*Section 25.3.35*
	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Disabled) (eDMA_TCD*n*_BITER_ELINKNO)	*Section 25.3.36*
0x101E+*n**0x20	TCD Control and Status (eDMA_TCD*n*_CSR)	*Section 25.3.37*

25.3.1 Control Register (eDMA_CR)

The CR defines the basic operating configuration of the DMA.

The DMA arbitrates channel service requests in four groups (0, 1, 2, 3) of 16 channels each. Group 3 contains channels 63–48; group 2 contains channels 47–32; group 1 contains channels 31–16; and group 0 contains channels 15–0.

Arbitration within a group can be configured to use a fixed-priority or a round-robin scheme. In fixed-priority arbitration, the highest priority channel requesting service is selected to execute. The channel priority registers assign the priorities (refer to the DCHPRI*n* registers). In Round-Robin Arbitration mode, the channel priorities are ignored, and channels within each group are cycled through without regard to priority.

Note: Writes to the DMA_CR must be performed only when the DMA channels are inactive (TCDn_CSR[ACTIVE] bits are cleared).

The group priorities operate in a similar fashion. In Group Fixed-Priority Arbitration mode, channel service requests in the highest priority group are executed first where priority level 3 is the highest and priority level 0 is the lowest. The group priorities are assigned in the GRP*n*PRI fields of the DMA Control Register (CR). All group priorities must have unique values prior to the occurrence of any channel service requests; otherwise, a configuration error will be reported. In Group Round-Robin mode, the group priorities are ignored and the groups are cycled through without regard to priority.

Address: 0x0000 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0														CX	ECX
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	GRP3PRI		GRP2PRI		GRP1PRI		GRP0PRI		EMLM	CLM	HALT	HOE	ERGA	ERCA	EDBG	0
W																
Reset	0/1(1)	0/1(1)	0/1(1)	0/1(1)	0/1(1)	0/1(1)	0/1(1)	0/1(1)	0	0	0	0	0	0	0	0

1. The number of channel groups and the reset value of eDMA_CR register are eDMA module specific. eDMA_0_CR = 0x0000_0400 and eDMA_1_CR = 0x0000_E400. Refer to the Device Configuration chapter for details.

Figure 233. Control Register (eDMA_CR)

Table 338. eDMA_CR field descriptions

Field	Description
14 CX	Cancel transfer 0 Normal operation 1 Cancel the remaining data transfer. Stop the executing channel and force the minor loop to finish. The cancel takes effect after the last write of the current read/write sequence. The CXFR bit clears itself after the cancel has been honored. This cancel retires the channel normally as if the minor loop was completed.
15 ECX	Error cancel transfer 0 Normal operation 1 Cancel the remaining data transfer in the same fashion as the CX bit. Stop the executing channel and force the minor loop to finish. The cancel takes effect after the last write of the current read/write sequence. The ECX bit clears itself after the cancel is honored. In addition to canceling the transfer, ECX treats the cancel as an error condition; thus updating the ES register and generating an optional error interrupt.
16:17 GRP3PRI	Channel group 3 priority Group 3 priority level when fixed-priority group arbitration is enabled. **Note:** This bit-field is valid and applicable only for eDMA_1 module.
18:19 GRP2PRI	Channel group 2 priority Group 2 priority level when fixed-priority group arbitration is enabled. **Note:** This bit-field is valid and applicable only for eDMA_1 module.
20:21 GRP1PRI	Channel group 1 priority Group 1 priority level when fixed-priority group arbitration is enabled.
22:23 GRP0PRI	Channel group 0 priority Group 0 priority level when fixed-priority group arbitration is enabled.
24 EMLM	Enable minor loop mapping 0 Disabled. TCD*n*.word2 is defined as a 32-bit NBYTES field. 1 Enabled. TCD*n*.word2 is redefined to include individual enable fields, an offset field, and the NBYTES field. The individual enable fields allow the minor loop offset to be applied to the source address, the destination address, or both. The NBYTES field is reduced when either offset is enabled.
25 CLM	Continuous Link mode 0 A minor loop channel link made to itself goes through channel arbitration before being activated again. 1 A minor loop channel link made to itself does not go through channel arbitration before being activated again. Upon minor loop completion, the channel activates again if that channel has a minor loop channel link enabled and the link channel is itself. This effectively applies the minor loop offsets and restarts the next minor loop.
26 HALT	Halt DMA operations 0 Normal operation 1 Stall the start of any new channels. Executing channels are allowed to complete. Channel execution resumes when this bit is cleared.
27 HOE	Halt on error 0 Normal operation 1 Any error causes the HALT bit to set. Subsequently, all service requests are ignored until the HALT bit is cleared.

Table 338. eDMA_CR field descriptions (continued)

Field	Description
28 ERGA	Enable round-robin group arbitration 0 Fixed-priority arbitration is used for selection among the groups. 1 Round-robin arbitration is used for selection among the groups.
29 ERCA	Enable round-robin channel arbitration 0 Fixed-priority arbitration is used for channel selection within each group. 1 Round-robin arbitration is used for channel selection within each group.
30 EDBG	Enable debug 0 When in Debug mode the DMA continues to operate. 1 When in Debug mode, the DMA stalls the start of a new channel. Executing channels are allowed to complete. Channel execution resumes when the system or the EDBG bit is cleared.

25.3.2 Error Status Register (eDMA_ES)

The ES provides information concerning the last recorded channel error. Channel errors can be caused by a configuration error (an illegal setting in the transfer-control descriptor or an illegal priority register setting in Fixed Arbitration mode) or an error termination to a bus master read or write cycle.

Refer to *Section 25.4.3*, for more details.

Address: 0x0004 Access: User read-write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	VLD	0	0	0	0	0	0	0	0	0	0	0	0	0	UCE	ECX
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	GPE	CPE	ERRCHN						SAE	SOE	DAE	DOE	NCE	SGE	SBE	DBE
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 234. Error Status Register (eDMA_ES)

Table 339. eDMA_ES field descriptions

Field	Description
0 VLD	ERRH and ERRL status bits 0 No ERR bits are set. 1 At least one ERR bit is set indicating a valid error exists that has not been cleared.
14 UCE	Uncorrectable ECC error 0 No uncorrectable ECC error 1 The last recorded error was an uncorrectable TCD RAM error.
15 ECX	Transfer canceled 0 No canceled transfers. 1 The last recorded entry was a zeroed transfer by the error cancel transfer input.

Table 339. eDMA_ES field descriptions (continued)

Field	Description
16 GPE	Group priority error 0 No group priority error 1 The last recorded error was a configuration error among the group priorities. All group priorities are not unique.
17 CPE	Channel priority error 0 No channel priority error 1 The last recorded error was a configuration error in the channel priorities. Channel priorities are not unique.
18:23 ERRCHN	Error channel number or canceled channel number GPE or last recorded error canceled transfer.
24 SAE	Source address error 0 No source address configuration error. 1 The last recorded error was a configuration error detected in the TCD*n*_SADDR field. TCD*n*_SADDR is inconsistent with TCD*n*_ATTR[SSIZE].
25 SOE	Source offset error 0 No source offset configuration error 1 The last recorded error was a configuration error detected in the TCD*n*_SOFF field. TCD*n*_SOFF is inconsistent with TCD*n*_ATTR[SSIZE].
26 DAE	Destination address error 0 No destination address configuration error 1 The last recorded error was a configuration error detected in the TCD*n*_DADDR field. TCD*n*_DADDR is inconsistent with TCD*n*_ATTR[DSIZE].
27 DOE	Destination offset error 0 No destination offset configuration error 1 The last recorded error was a configuration error detected in the TCD*n*_DOFF field. TCD*n*_DOFF is inconsistent with TCD*n*_ATTR[DSIZE].
28 NCE	NBYTES/CITER configuration error 0 No NBYTES/CITER configuration error 1 The last recorded error was a configuration error detected in the TCD*n*_NBYTES or TCD*n*_CITER fields.
29 SGE	Scatter/gather configuration error 0 No scatter/gather configuration error 1 The last recorded error was a configuration error detected in the TCD*n*_DLASTSGA field. This field is checked at the beginning of a scatter/gather operation after major loop completion if TCD*n*_CSR[ESG] is enabled. TCD*n*_DLASTSGA is not on a 32-byte boundary.
30 SBE	Source bus error 0 No source bus error 1 The last recorded error was a bus error on a source read.
31 DBE	Destination bus error 0 No destination bus error 1 The last recorded error was a bus error on a destination write.

25.3.3 Enable Request Register High (eDMA_ERQH)

The ERQ{H,L} registers provide a bitmap for the 64 implemented channels to enable the request signal for each channel. ERQH supports channels 63–32, while EQRL covers

channels 31–00. The state of any given channel enable is directly affected by writes to this register; it is also affected by writes to the SERQ and CERQ. The {S,C}ERQ registers are provided so the request enable for a single channel can easily be modified without needing to perform a read-modify-write sequence to the ERQ{H,L}.

DMA request input signals and this enable request flag must be asserted before a channel's hardware service request is accepted. The state of the eDMA enable request flag does not affect a channel service request made explicitly through software or a linked channel request.

Address: 0x0008 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	ERQ 63	ERQ 62	ERQ 61	ERQ 60	ERQ 59	ERQ 58	ERQ 57	ERQ 56	ERQ 55	ERQ 54	ERQ 53	ERQ 52	ERQ 51	ERQ 50	ERQ 49	ERQ 48
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	ERQ 47	ERQ 46	ERQ 45	ERQ 44	ERQ 43	ERQ 42	ERQ 41	ERQ 40	ERQ 39	ERQ 38	ERQ 37	ERQ 36	ERQ 35	ERQ 34	ERQ 33	ERQ 32
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 235. Enable Request Register High (eDMA_ERQH)

Table 340. eDMA_ERQH field descriptions

Field	Description
0:31 ERQ[63:32]	Enable DMA Request x 0 The DMA request signal for the corresponding channel is disabled. 1 The DMA request signal for the corresponding channel is enabled.

25.3.4 Enable Request Register Low (eDMA_ERQL)

The ERQ{H,L} registers provide a bit map for the 64 implemented channels to enable the request signal for each channel. ERQH supports channels 63–32, while EQRL covers channels 31–00.

The state of any given channel enable is directly affected by writes to this register; it is also affected by writes to the SERQ and CERQ. The {S,C}ERQ registers are provided so the request enable for a single channel can easily be modified without needing to perform a read-modify-write sequence to the ERQ{H,L}.

DMA request input signals and this enable request flag must be asserted before a channel's hardware service request is accepted. The state of the DMA enable request flag does not affect a channel service request made explicitly through software or a linked channel request.

Address: 0x000C Access: User read/write

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

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

Figure 236. Enable Request Register Low (eDMA_ERQL)

Table 341. eDMA_ERQL field descriptions

Field	Description
0:31 ERQ[31:0]	Enable DMA Request x 0 The DMA request signal for the corresponding channel is disabled. 1 The DMA request signal for the corresponding channel is enabled.

25.3.5 Enable Error Interrupt Register High (eDMA_EEIH)

The EEI{H,L} registers provide a bit map for the 64 channels to enable the error interrupt signal for each channel. EEIH supports channels 63–32, while EEIL covers channels 31–00. The state of any given channel's error interrupt enable is directly affected by writes to this register; it is also affected by writes to the SEEI and CEEI. The {S,C}EEI are provided so the error interrupt enable for a single channel can easily be modified without the need to perform a read-modify-write sequence to the EEI{H,L} registers.

The DMA error indicator and the error interrupt enable flag must be asserted before an error interrupt request for a given channel is asserted to the interrupt controller.

Address: 0x0010 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	EEI	EEI	EEI	EEI	EEI	EEI	EEI	EEI	EEI	EEI	EEI	EEI	EEI	EEI	EEI	EEI
W	63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	EEI	EEI	EEI	EEI	EEI	EEI	EEI	EEI	EEI	EEI	EEI	EEI	EEI	EEI	EEI	EEI
W	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 237. Enable Error Interrupt Register High (eDMA_EEIH)

Table 342. eDMA_EEIH field descriptions

Field	Description
0:31 EEI[63:32]	Enable error interrupt x 0 The error signal for corresponding channel does not generate an error interrupt. 1 The assertion of the error signal for corresponding channel generates an error interrupt request.

25.3.6 Enable Error Interrupt Register Low (eDMA_EEIL)

The Enable Error Interrupt Register Low (DMA_EEIL) provides a bit map for the 64 channels to enable the error interrupt signal for each channel. EEIH supports channels 63–32, while EEIL covers channels 31–00. The state of any given channel's error interrupt enable is directly affected by writes to this register; it is also affected by writes to the SEEI and CEEI. The {S,C}EEI are provided so the error interrupt enable for a single channel can easily be modified without the need to perform a read-modify-write sequence to the EEI{H,L} registers.

The DMA error indicator and the error interrupt enable flag must be asserted before an error interrupt request for a given channel is asserted to the interrupt controller.

Address: 0x0014 Access: User read/write

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

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

Figure 238. Enable Error Interrupt Register Low (eDMA_EEIL)

Table 343. eDMA_EEIL field descriptions

Field	Description
0:31 EEI[31:0]	Enable error interrupt x 0 The error signal for corresponding channel does not generate an error interrupt. 1 The assertion of the error signal for corresponding channel generates an error interrupt request.

25.3.7 Set Enable Request Register (eDMA_SERQ)

The Set Enable Request Register (DMA_SERQ) provides a simple memory-mapped mechanism to set a given bit in the ERQ{H,L} to enable the DMA request for a given channel. The data value on a register write causes the corresponding bit in the ERQ{H,L} to be set. Setting the SAER bit provides a global set function, forcing the entire contents of ERQ{H,L} to be set. If the NOP bit is set, the command is ignored. This allows you to write multiple-byte registers as a 32-bit word. Reads of this register return all zeroes.

Address: 0x0018 Access: User write-only

	0	1	2	3	4	5	6	7
R	0	0	0					
W	NOP	SAER	SERQ					
Reset	0	0	0	0	0	0	0	0

Figure 239. Set Enable Request Register (eDMA_SERQ)

Table 344. eDMA_SERQ field descriptions

Field	Description
0 NOP	0 Normal operation 1 No operation, ignore bits 1–7 of this register
1 SAER	Set all enable requests 0 Set only those ERQ bits specified in the SERQ field 1 Set all bits in ERQ{H,L}
2:7 SERQ	Set enable request Sets the corresponding bit in ERQ{H,L}

25.3.8 Clear Enable Request Register (eDMA_CERQ)

The Clear Enable Request Register (DMA_CERQ) provides a simple memory-mapped mechanism to clear a given bit in the ERQ{H,L} to disable the DMA request for a given channel. The data value on a register write causes the corresponding bit in the ERQ{H,L} to be cleared. Setting the CAER bit provides a global clear function, forcing the entire contents of the ERQ{H,L} to be cleared, disabling all DMA request inputs.

If NOP is set, the command is ignored. This allows you to write multiple-byte registers as a 32-bit word. Reads of this register return all zeroes.

Address: 0x0019 Access: User write-only

	0	1	2	3	4	5	6	7
R	0	0	0					
W	NOP	CAER	CERQ					
Reset	0	0	0	0	0	0	0	0

Figure 240. Clear Enable Request Register (eDMA_CERQ)

Table 345. eDMA_CERQ field descriptions

Field	Description
0 NOP	0 Normal operation 1 No operation, ignore bits 1–7 of this register
1 CAER	Clear all enable requests 0 Clear only those ERQ bits specified in the CERQ field 1 Clear all bits in ERQ
2:7 CERQ	Clear enable request Clears the corresponding bit in ERQ{H,L}

25.3.9 Set Enable Error Interrupt Register (eDMA_SEEI)

The Set Enable Error Interrupt Register (DMA_SEEI) provides a simple memory-mapped mechanism to set a given bit in the EEI{H,L} to enable the error interrupt for a given channel. The data value on a register write causes the corresponding bit in the EEI{H,L} to be set. Setting the SAEE bit provides a global set function, forcing the entire EEI{H,L} contents to be set. If the NOP bit is set, the command is ignored. This allows you to write multiple-byte registers as a 32-bit word. Reads of this register return all zeroes.

Address: 0x001A Access: User write-only

	0	1	2	3	4	5	6	7
R	0	0	0					
W	NOP	SAEE	SEEI					
Reset	0	0	0	0	0	0	0	0

Figure 241. Set Enable Error Interrupt Register (eDMA_SEEI)

Table 346. eDMA_SEEI field descriptions

Field	Description
0 NOP	0 Normal operation 1 No operation, ignore bits 1–7 of this register
1 SAEE	Sets all enable error interrupts 0 Set only those EEI bits specified in the SEEI field. 1 Sets all bits in EEI.
2:7 SEEI	Set enable error interrupt Sets the corresponding bit in EEI{H,L}

25.3.10 Clear Enable Error Interrupt Register (eDMA_CEEI)

The Clear Enable Error Interrupt Register (DMA_CEEI) provides a simple memory-mapped mechanism to clear a given bit in the EEI{H,L} to disable the error interrupt for a given channel. The data value on a register write causes the corresponding bit in the EEI{H,L} to be cleared. Setting the CAEE bit provides a global clear function, forcing the EEI{H,L} contents to be cleared, disabling all DMA request inputs.

If the NOP bit is set, the command is ignored. This allows you to write multiple-byte registers as a 32-bit word. Reads of this register return all zeroes.

Address: 0x001B Access: User write-only

	0	1	2	3	4	5	6	7
R	0	0	0					
W	NOP	CAEE	CEEI					
Reset	0	0	0	0	0	0	0	0

Figure 242. Clear Enable Error Interrupt Register (eDMA_CEEI)

Table 347. eDMA_CEEI field descriptions

Field	Description
0 NOP	0 Normal operation 1 No operation, ignore bits 1–7 of this register
1 CAEE	Clear all enable error interrupts 0 Clear only those EEI bits specified in the CEEI field 1 Clear all bits in EEI
2:7 CEEI	Clear enable error interrupt Clears the corresponding bit in EEI{H,L}

25.3.11 Clear Interrupt Request Register (eDMA_CINT)

The Clear Interrupt Request Register (DMA_CINT) provides a simple, memory-mapped mechanism to clear a given bit in the INT{H,L} to disable the interrupt request for a given channel. The given value on a register write causes the corresponding bit in the INT{H,L} to be cleared. Setting the CAIR bit provides a global clear function, forcing the entire contents of the INT{H,L} to be cleared, disabling all DMA interrupt requests.

If the NOP bit is set, the command is ignored. This allows you to write multiple-byte registers as a 32-bit word. Reads of this register return all zeroes.

Address: 0x001C Access: User write-only

	0	1	2	3	4	5	6	7
R	0	0	0					
W	NOP	CAIR	CINT					
Reset	0	0	0	0	0	0	0	0

Figure 243. Clear Interrupt Request Register (eDMA_CINT)

Table 348. eDMA_CINT field descriptions

Field	Description
0 NOP	0 Normal operation 1 No operation, ignore bits 1–7 of this register
1 CAIR	Clear all interrupt requests 0 Clear only those INT bits specified in the CINT field 1 Clear all bits in INT{H,L}
2:7 CINT	Clear interrupt request Clears the corresponding bit in INT{H,L}

25.3.12 Clear Error Register (eDMA_CERR)

The CERR provides a simple memory-mapped mechanism to clear a given bit in the ERR{H,L} to disable the error condition flag for a given channel. The given value on a register write causes the corresponding bit in the ERR{H,L} to be cleared. Setting the CAEI bit provides a global clear function, forcing the ERR{H,L} contents to be cleared, clearing all channel error indicators. If the NOP bit is set, the command is ignored. This allows you to write multiple-byte registers as a 32-bit word. Reads of this register return all zeroes.

Address: 0x001D Access: User write-only

	0	1	2	3	4	5	6	7
R	0	0	0					
W	NOP	CAEI	CERR					
Reset	0	0	0	0	0	0	0	0

Figure 244. Clear Error Register (eDMA_CERR)

Table 349. eDMA_CERR field descriptions

Field	Description
0 NOP	0 Normal operation 1 No operation, ignore bits 1–7 of this register
1 CAEI	Clear all error indicators 0 Clear only those ERR bits specified in the CERR field 1 Clear all bits in ERR
2:7 CERR	Clear error indicator Clears the corresponding bit in ERR{H,L}

25.3.13 Set START Bit Register (eDMA_SSRT)

The Set START Bit Register (DMA_SSRT) provides a simple memory-mapped mechanism to set the START bit in the TCD of the given channel. The data value on a register write causes the START bit in the corresponding transfer control descriptor to be set. Setting the SAST bit provides a global set function, forcing all START bits to be set. If the NOP bit is set, the command is ignored. This allows you to write multiple-byte registers as a 32-bit word. Reads of this register return all zeroes.

Address: 0x001E Access: User write-only

	0	1	2	3	4	5	6	7
R	0	0	0					
W	NOP	SAST	SSRT					
Reset	0	0	0	0	0	0	0	0

Figure 245. Set START Bit Register (eDMA_SSRT)

Table 350. eDMA_SSRT field descriptions

Field	Description
0 NOP	0 Normal operation 1 No operation, ignore bits 1–7 of this register
1 SAST	Set all START bits (activates all channels) 0 Set only those TCD*n*_CSR[START] bits specified in the SSRT field 1 Set all bits in TCD*n*_CSR[START]
2:7 SSRT	Set START bit Sets the corresponding bit in TCD*n*_CSR[START]

25.3.14 Clear DONE Status Bit Register (eDMA_CDNE)

The Clear DONE Status Bit Register (DMA_CDNE) provides a simple memory-mapped mechanism to clear the DONE bit in the TCD of the given channel. The data value on a register write causes the DONE bit in the corresponding transfer control descriptor to be cleared. Setting the CADN bit provides a global clear function, forcing all DONE bits to be cleared. If the NOP bit is set, the command is ignored. This allows you to write multiple-byte registers as a 32-bit word. Reads of this register return all zeroes.

Address: 0x001F Access: User write-only

	0	1	2	3	4	5	6	7
R	0	0	0					
W	NOP	CADN	CDNE					
Reset	0	0	0	0	0	0	0	0

Figure 246. Clear DONE Status Bit Register (eDMA_CDNE)

Table 351. eDMA_CDNE field descriptions

Field	Description
0 NOP	0 Normal operation 1 No operation, ignore bits 1–7 of this register
1 CADN	Clears all DONE bits 0 Clears only those TCD*n*_CSR[DONE] bits specified in the CDNE field 1 Clears all bits in TCD*n*_CSR[DONE]
2:7 CDNE	Clear DONE bit Clears the corresponding bit in TCD*n*_CSR[DONE]

25.3.15 Interrupt Request Register High (eDMA_INTH)

The Interrupt Request Register High (DMA_INTH) provides a bit map for the upper half of 64 channels signaling the presence of an interrupt request for each channel. Depending on the appropriate bit setting in the transfer-control descriptions, the eDMA engine generates an interrupt on a data transfer completion. The outputs of this register are directly routed to the interrupt controller (INTC). During the interrupt-service routine associated with any given channel, it is the software's responsibility to clear the appropriate bit, negating the interrupt request. Typically, a write to the CINT in the interrupt service routine is used for this purpose.

The state of any given channel's interrupt request is directly affected by writes to this register; it is also affected by writes to the CINT. On writes to the INT, a '1' in any bit position clears the corresponding channel's interrupt request. A zero in any bit position has no effect on the corresponding channel's current interrupt status. The CINT is provided so the interrupt request for a single channel can easily be cleared without the need to perform a read-modify-write sequence to the INT{H,L}.

Address: 0x0020 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	INT 63	INT 62	INT 61	INT 60	INT 59	INT 58	INT 57	INT 56	INT 55	INT 54	INT 53	INT 52	INT 51	INT 50	INT 49	INT 48
W	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	INT 47	INT 46	INT 45	INT 44	INT 43	INT 42	INT 41	INT 40	INT 39	INT 38	INT 37	INT 36	INT 35	INT 34	INT 33	INT 32
W	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 247. Interrupt Request Register High (eDMA_INTH)

Table 352. eDMA_INTH field descriptions

Field	Description
0:31 INT[63:32]	Interrupt request x 0 The interrupt request for corresponding channel is cleared. 1 The interrupt request for corresponding channel is active.

25.3.16 Interrupt Request Register Low (eDMA_INTL)

The Interrupt Request Register Low (DMA_INTL) provides a bit map for the lower half of the 64 channels signaling the presence of an interrupt request for each channel. Depending on the appropriate bit setting in the transfer-control descriptors, the eDMA engine generates an interrupt at data transfer completion. The outputs of this register are directly routed to the interrupt controller (INTC). During the interrupt-service routine associated with any given channel, it is the software's responsibility to clear the appropriate bit, negating the interrupt request. Typically, a write to the CINT in the interrupt service routine is used for this purpose.

The state of any given channel's interrupt request is directly affected by writes to this register; it is also affected by writes to the CINT. On writes to the INT, a '1' in any bit position clears the corresponding channel's interrupt request. A zero in any bit position has no effect on the corresponding channel's current interrupt status. The CINT is provided so the interrupt request for a single channel can easily be cleared without the need to perform a read-modify-write sequence to the INT{H,L}.

Address: 0x0024 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	INT 31	INT 30	INT 29	INT 28	INT 27	INT 26	INT 25	INT 24	INT 23	INT 22	INT 21	INT 20	INT 19	INT 18	INT 17	INT 16
W	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	INT 15	INT 14	INT 13	INT 12	INT 11	INT 10	INT 9	INT 8	INT 7	INT 6	INT 5	INT 4	INT 3	INT 2	INT 1	INT 0
W	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 248. Interrupt Request Register Low (eDMA_INTL)

Table 353. eDMA_INTL field descriptions

Field	Description
0:31 INT[31:0]	Interrupt request x 0 The interrupt request for corresponding channel is cleared. 1 The interrupt request for corresponding channel is active.

25.3.17 Error Register High (eDMA_ERRH)

The ERR{H,L} provides a bit map for the 64 channels, signaling the presence of an error for each channel. ERRH supports channels 63–32, while ERRL covers channels 31–00. The eDMA engine signals the occurrence of a error condition by setting the appropriate bit in this register. The outputs of this register are enabled by the contents of the EEI, then logically summed across groups of 16, 32, and 64 channels to form several group error interrupt requests that are then routed to the interrupt controller. During the execution of the interrupt-service routine associated with any DMA errors, it is software's responsibility to clear the appropriate bit, negating the error-interrupt request. Typically, a write to the CERR in the interrupt-service routine is used for this purpose. The normal DMA channel completion indicators (setting the transfer control descriptor DONE flag and the possible assertion of an interrupt request) are not affected when an error is detected.

The content of this register can also be polled because a non-zero value indicates the presence of a channel error regardless of the state of the EEI. The state of any given channel's error indicators is affected by writes to this register; it is also affected by writes to the CERR. On writes to the ERR, a one in any bit position clears the corresponding channel's error status. A zero in any bit position has no effect on the corresponding channel's current error status. The CERR is provided so the error indicator for a single channel can easily be cleared.

Address: 0x0028 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	ERR 63	ERR 62	ERR 61	ERR 60	ERR 59	ERR 58	ERR 57	ERR 56	ERR 55	ERR 54	ERR 53	ERR 52	ERR 51	ERR 50	ERR 49	ERR 48
W	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	ERR 47	ERR 46	ERR 45	ERR 44	ERR 43	ERR 42	ERR 41	ERR 40	ERR 39	ERR 38	ERR 37	ERR 36	ERR 35	ERR 34	ERR 33	ERR 32
W	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 249. Error Register High (eDMA_ERRH)

Table 354. eDMA_ERRH field descriptions

Field	Description
0:31 ERR[63:32]	Error in channel x 0 An error in the corresponding channel has not occurred. 1 An error in the corresponding channel has occurred.

25.3.18 Error Register Low (eDMA_ERRL)

The ERR{H,L} provides a bit map for the 64 channels, signaling the presence of an error for each channel. ERRH supports channels 63–32, while ERRL covers channels 31–00. The eDMA engine signals the occurrence of an error condition by setting the appropriate bit in this register. The outputs of this register are enabled by the contents of the EEI, then logically summed across groups of 16 , 32, and 64 channels to form several group error interrupt requests that are then routed to the interrupt controller. During the execution of the interrupt-service routine associated with any DMA errors, it is software's responsibility to clear the appropriate bit, negating the error-interrupt request. Typically, a write to the CERR in the interrupt-service routine is used for this purpose. The normal DMA channel completion indicators (setting the transfer control descriptor DONE flag and the possible assertion of an interrupt request) are not affected when an error is detected.

The contents of this register can also be polled because a non-zero value indicates the presence of a channel error regardless of the state of the EEI. The state of any given channel's error indicators is affected by writes to this register; it is also affected by writes to the CERR. On writes to the ERR, a one in any bit position clears the corresponding channel's error status. A zero in any bit position has no effect on the corresponding channel's current error status. The CERR is provided so the error indicator for a single channel can easily be cleared.

Address: 0x002C Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	ERR 31	ERR 30	ERR 29	ERR 28	ERR 27	ERR 26	ERR 25	ERR 24	ERR 23	ERR 22	ERR 21	ERR 20	ERR 19	ERR 18	ERR 17	ERR 16
W	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	ERR 15	ERR 14	ERR 13	ERR 12	ERR 11	ERR 10	ERR 9	ERR 8	ERR 7	ERR 6	ERR 5	ERR 4	ERR 3	ERR 2	ERR 1	ERR 0
W	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 250. Error Register Low (eDMA_ERRL)

Table 355. eDMA_ERRL field descriptions

Field	Description
0:31 ERR[31:0]	Error in channel x 0 An error in the corresponding channel has not occurred. 1 An error in the corresponding channel has occurred.

25.3.19 Hardware Request Status Register High (eDMA_HRSH)

The HRS{H,L} provides a bit map for the DMA channels, signaling the presence of a qualified hardware service request for each channel. HRSH supports channels 63–32, while HRSL covers channels 31–00. The Hardware Request Status bits reflect the current state of the register and qualified (via the ERQ{H,L} fields) DMA request signals as seen by the DMA's arbitration logic. This view into the hardware request signals may be used for debug purposes.

Note: These bits reflect the state of the request as seen by the arbitration logic. Therefore, this status is affected by the ERQ{H,L} bits.

Address: 0x0030 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	HRS 63	HRS 62	HRS 61	HRS 60	HRS 59	HRS 58	HRS 57	HRS 56	HRS 55	HRS 54	HRS 53	HRS 52	HRS 51	HRS 50	HRS 49	HRS 48
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	HRS 47	HRS 46	HRS 45	HRS 44	HRS 43	HRS 42	HRS 41	HRS 40	HRS 39	HRS 38	HRS 37	HRS 36	HRS 35	HRS 34	HRS 33	HRS 32
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 251. Hardware Request Status Register High (eDMA_HRSH)

Table 356. eDMA_HRSH field descriptions

Field	Description
0:31 HRS[63:32]	Hardware request status channel x 0 A hardware service request for the corresponding channel is not present. 1 A hardware service request for the corresponding channel is present.

25.3.20 Hardware Request Status Register Low (eDMA_HRSL)

The HRS{H,L} provides a bit map for the DMA channels, signaling the presence of a qualified hardware service request for each channel. HRSH supports channels 63–32, while HRSL covers channels 31–00. The Hardware Request Status bits reflect the current state of the register and qualified (via the ERQ{H,L} fields) DMA request signals as seen by the DMA's arbitration logic. This view into the hardware request signals may be used for debug purposes.

Note: These bits reflect the state of the request as seen by the arbitration logic. Therefore, this status is affected by the ERQ{H,L} bits.

Address: 0x0034 Access: User read/write

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

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

Figure 252. Hardware Request Status Register Low (eDMA_HRSL)

Table 357. eDMA_HRSL field descriptions

Field	Description
0:31 HRS[31:0]	Hardware request status channel x 0 A hardware service request for the corresponding channel is not present. 1 A hardware service request for the corresponding channel is present.

25.3.21 Channel *n* Priority Register (eDMA_DCHPRI*n*)

When the Fixed-Priority Channel Arbitration mode is enabled (CR[ERCA] = 0), the contents of these registers define the unique priorities associated with each channel within a group. The channel priorities are evaluated by numeric value; for example, 0 is the lowest priority, 1 is the next higher priority, then 2, 3, and so on. Software must program the channel priorities with unique values. Otherwise, a configuration error is reported. The range of the priority value is limited to the values of 0 through 15.

When read, the GRPPRI bits of the DCHPRI*n* register reflect the current priority level of the group of channels in which the corresponding channel resides. GRPPRI bits are not

affected by writes to the DCHPRI*n* registers. The group priority is assigned in the DMA control register.

Address: 0x0100 + *n**0x1 (*n* = 0 to 63) Access: User read/write

	0	1	2	3	4	5	6	7
R	ECP	DPA	GRPPRI		CHPRI			
W								
Reset(1)	0	0	x	x	0	0	0	0

1. x = Undefined at reset.

Figure 253. Channel *n* Priority Register (eDMA_DCHPRI*n*)

Table 358. eDMA_DCHPRI*n* field descriptions

Field	Description
0 ECP	Enable channel preemption This bit resets to zero. 0 Channel *n* cannot be suspended by a higher priority channel's service request. 1 Channel *n* can be temporarily suspended by the service request of a higher priority channel.
1 DPA	Disable preempt ability This bit resets to zero. 0 Channel *n* can suspend a lower priority channel. 1 Channel *n* cannot suspend any channel, regardless of channel priority.
2:3 GRPPRI	Channel *n* current group priority Group priority assigned to this channel group when fixed-priority arbitration is enabled. These two bits are read only; writes are ignored. **Note:** Reset value for the group and channel priority fields, GRPPRI and CHPRI, is equal to the corresponding channel number for each priority register, that is, DCHPRI31[GRPPRI] = 0b01 and DCHPRI31[CHPRI] equals 0b1111.
4:7 CHPRI	Channel *n* arbitration priority Channel priority when fixed-priority arbitration is enabled. **Note:** Reset value for the group and channel priority fields, GRPPRI and CHPRI, is equal to the corresponding channel number for each priority register, that is, DCHPRI31[GRPPRI] = 0b01 and DCHPRI31[CHPRI] equals 0b1111.

25.3.22 Channel *n* Master ID Register (eDMA_DCHMID*n*)

The DMA Master ID Replication registers allow the DMA to use the same protection level and AHB system bus ID of the master programming the DMA's TCD. When enabled, the DMA uses the master ID and protection level stored in the DCHMID register instead of the DMA's default values. When a master (a core, for example) programs a TCD, its master ID and protection level are captured when the TCD Word 7 control attributes are written. Although the scatter/gather operation can change the contents of TCD Word 7, that operation does not affect the DCHMID*n* registers.

Address: 0x140 + *n**0x1 (*n* = 0 to 63) Access: User read/write

	0	1	2	3	4	5	6	7
R	EMI	PAL	0		MID			
W								
Reset	0	0	0	0	0	0	0	0

Figure 254. Channel *n* Master ID Register (eDMA_DCHMID*n*)

Table 359. eDMA_DCHMID*n* field descriptions

Field	Description
0 EMI	Enable Master ID replication 0 Master ID replication is disabled. 1 Master ID replication is enabled.
1 PAL	Privileged Access Level The protection level captured in this register reflects the level used when writing the channel's control attributes; lower byte of TCD Word 7. 0 User protection level for DMA transfers 1 Privileged protection level for DMA transfers
4:7 MID	Master ID DMA's master ID when channel *n* is active and master ID replication is enabled. **Note:** The master ID captured in this register reflects the ID used when writing the channel's control attributes; lower byte of TCD Word 7.

25.3.23 TCD Source Address (eDMA_TCD*n*_SADDR)

Refer to *Figure 255* and *Table 360* for details.

Address: 0x1000 + *n**0x20 (*n* = 0 to 63) Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	SADDR															
W																
Reset(1)	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	SADDR															
W																
Reset(1)	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x

1. x = Undefined at reset.

Figure 255. TCD Source Address (eDMA_TCD*n*_SADDR)

Table 360. eDMA_TCD*n*_SADDR field descriptions

Field	Description
0:31 SADDR	Source address Memory address pointing to the source data.

25.3.24 TCD Transfer Attributes (eDMA_TCD*n*_ATTR)

Refer to *Figure 256* and *Table 361* for details.

Address: 0x1004 + *n**0x20 (*n* = 0 to 63) Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	SMOD				SSIZE				DMOD				DSIZE			
Reset(1)	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x

1. x = Undefined at reset.

Figure 256. TCD Transfer Attributes (eDMA_TCD*n*_ATTR)

Table 361. eDMA_TCD*n*_ATTR field descriptions

Field	Description
0:4 SMOD	Source address modulo 0 Source address modulo feature is disabled. Not 0 This value defines a specific address range specified to be the value after SADDR + SOFF calculation is performed on the original register value. Setting this field provides the ability to implement a circular data queue easily. For data queues requiring power-of-2 size bytes, the queue should start at a 0-modulo-size address and the SMOD field should be set to the appropriate value for the queue, freezing the desired number of upper address bits. The value programmed into this field specifies the number of lower address bits allowed to change. For a circular queue application, the SOFF is typically set to the transfer size to implement post-increment addressing with the SMOD function constraining the addresses to a 0-modulo-size range.
5:7 SSIZE	Source data transfer size Using a reserved encoding causes a configuration error. 000 8-bit 001 16-bit 010 32-bit 011 64-bit 100 Reserved 101 32-byte 110 Reserved 111 Reserved
8:12 DMOD	Destination address modulo See the SMOD definition.
13:15 DSIZE	Destination data transfer size See the SSIZE definition.

25.3.25 TCD Signed Source Address Offset (eDMA_TCD*n*_SOFF)

Refer to *Figure 257* and *Table 362* for details.

Address: 0x1006 + *n**0x20 (*n* = 0 to 63) Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	SOFF															
Reset(1)	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x

Figure 257. TCD Signed Source Address Offset (eDMA_TCD*n*_SOFF)

1. x = Undefined at reset.

Table 362. eDMA_TCD*n*_SOFF field descriptions

Field	Description
0:15 SOFF	Source address signed offset Sign-extended offset applied to the current source address to form the next-state value as each source read is completed.

25.3.26 TCD Minor Byte Count (Minor Loop Disabled) (eDMA_TCD*n*_NBYTES_MLNO)

Refer to *Figure 258* and *Table 363* for details.

If minor loop mapping is disabled (CR[EMLM] = 0), TCD word 2 is defined as follows.

Address: 0x1008+ *n**0x20 (*n* = 0 to 63) Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	NBYTES															
Reset(1)	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	NBYTES															
Reset(1)	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x

1. x = Undefined at reset.

Figure 258. TCD Minor Byte Count (Minor Loop Disabled) (eDMA_TCD*n*_NBYTES_MLNO)

Table 363. eDMA_TCD*n*_NBYTES_MLNO field descriptions

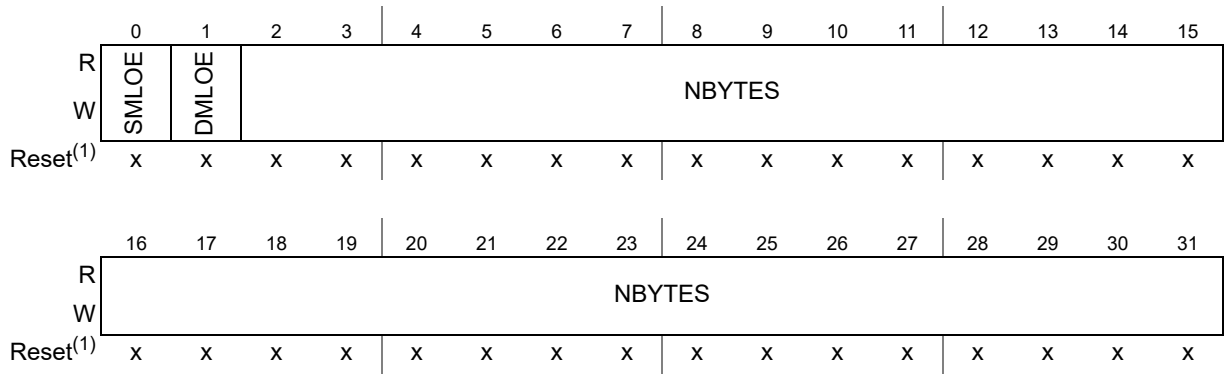
Field	Description
0:31 NBYTES	Minor byte transfer count Number of bytes to be transferred in each service request of the channel. As a channel activates, the appropriate TCD contents load into the eDMA engine, and the appropriate reads and writes perform until the minor byte transfer count has transferred. This is an indivisible operation and cannot be halted. (Although, it may be stalled by using the bandwidth control field, or via preemption.) After the minor count is exhausted, the SADDR and DADDR values are written back into the TCD memory, the major iteration count is decremented and restored to the TCD memory. If the major iteration count is completed, additional processing is performed. **Note:** An NBYTES value of 0x0000_0000 is interpreted as a 4 GB transfer.

25.3.27 TCD Signed Minor Loop Offset (Minor Loop Enabled and Offset Disabled) (eDMA_TCD*n*_NBYTES_MLOFFNO)

TCD Word 2 is defined as follows if:

- Minor loop mapping is enabled (CR[EMLM] = 1) and
- SMLOE = 0 and DMLOE = 0

Address: 0x1008 + *n**0x20 (*n* = 0 to 63) Access: User read/write



1. x = Undefined at reset.

Figure 259. TCD Signed Minor Loop Offset (Minor Loop Enabled and Offset Disabled) (eDMA_TCD*n*_NBYTES_MLOFFNO)

Table 364. eDMA_TCD*n*_NBYTES_MLOFFNO field descriptions

Field	Description
0 SMLOE	Source minor loop offset enable Selects whether the minor loop offset is applied to the source address upon minor loop completion. 0 The minor loop offset is not applied to the SADDR. 1 The minor loop offset is applied to the SADDR.
1 DMLOE	Destination minor loop offset enable Selects whether the minor loop offset is applied to the destination address upon minor loop completion. 0 The minor loop offset is not applied to the DADDR. 1 The minor loop offset is applied to the DADDR.
2:31 NBYTES	Minor byte transfer count Number of bytes to be transferred in each service request of the channel. As a channel activates, the appropriate TCD contents load into the eDMA engine, and the appropriate reads and writes perform until the minor byte transfer count has transferred. This is an indivisible operation and cannot be halted. (Although, it may be stalled by using the bandwidth control field, or via preemption.) After the minor count is exhausted, the SADDR and DADDR values are written back into the TCD memory, the major iteration count is decremented and restored to the TCD memory. If the major iteration count is completed, additional processing is performed.

25.3.28 TCD Signed Minor Loop Offset (Minor Loop and Offset Enabled) (eDMA_TCD*n*_NBYTES_MLOFFYES)

TCD Word 2 is defined as follows if:

- Minor loop mapping is enabled (CR[EMLM] = 1) and
- Minor loop offset enabled (SMLOE or DMLOE = 1)

Address: 0x1008+ *n**0x20 (*n* = 0 to 63) Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	SMLOE	DMLOE	MLOFF													
W																
Reset(1)	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	MLOFF						NBYTES									
W																
Reset(1)	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x

1. x = Undefined at reset.

Figure 260. TCD Signed Minor Loop Offset (Minor Loop and Offset Enabled) (eDMA_TCD*n*_NBYTES_MLOFFYES)

Table 365. eDMA_TCD*n*_NBYTES_MLOFFYES field descriptions

Field	Description
0 SMLOE	Source minor loop offset enable Selects whether the minor loop offset is applied to the source address upon minor loop completion. 0 The minor loop offset is not applied to the SADDR. 1 The minor loop offset is applied to the SADDR.
1 DMLOE	Destination minor loop offset enable Selects whether the minor loop offset is applied to the destination address upon minor loop completion. 0 The minor loop offset is not applied to the DADDR. 1 The minor loop offset is applied to the DADDR.
2:21 MLOFF	If SMLOE or DMLOE is set, this field represents a sign-extended offset applied to the source or destination address to form the next-state value after the minor loop completes.
22:31 NBYTES	Minor byte transfer count Number of bytes to be transferred in each service request of the channel. As a channel activates, the appropriate TCD contents load into the eDMA engine, and the appropriate reads and writes perform until the minor byte transfer count has transferred. This is an indivisible operation and cannot be halted. (Although, it may be stalled by using the bandwidth control field, or via preemption.) After the minor count is exhausted, the SADDR and DADDR values are written back into the TCD memory, the major iteration count is decremented and restored to the TCD memory. If the major iteration count is completed, additional processing is performed.

25.3.29 TCD Last Source Address Adjustment (eDMA_TCD*n*_SLAST)

Address: 0x100C + *n**0x20 (*n* = 0 to 63) Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	SLAST															
Reset(1)	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	SLAST															
Reset(1)	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x

1. = Undefined at reset.

Figure 261. TCD Last Source Address Adjustment (eDMA_TCD*n*_SLAST)

Table 366. eDMA_TCD*n*_SLAST field descriptions

Field	Description
0:31 SLAST	Last source address adjustment Adjustment value added to the source address at the completion of the major iteration count. This value can be applied to restore the source address to the initial value, or adjust the address to reference the next data structure.

25.3.30 TCD Destination Address (eDMA_TCD*n*_DADDR)

See TCD Destination Address register figure and DMA_TCD*n*_DADDR field descriptions table as follows.

Address: 0x1010 + *n**0x20 (*n* = 0 to 63) Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	DADDR															
Reset(1)	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	DADDR															
Reset(1)	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x

1. x = Undefined at reset.

Figure 262. TCD Destination Address (eDMA_TCD*n*_DADDR)

Table 367. eDMA_TCD*n*_DADDR field descriptions

Field	Description
0:31 DADDR	Destination address Memory address pointing to the destination data.

25.3.31 TCD Current Minor Loop Link, Major Loop Count (Channel Linking Enabled) (eDMA_TCD*n*_CITER_ELINKYES)

See TCD Current Minor Loop Link, Major Loop Count (Channel Linking Enabled) register figure and DMA_TCDn_CITER_ELINKYES field descriptions table as follows.

Address: 0x1014 + *n**0x20 (*n* = 0 to 63) Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	ELINK	LINKCH							CITER							
Reset(1)	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x

1. x = Undefined at reset.

Figure 263. TCD Current Minor Loop Link, Major Loop Count (Channel Linking Enabled) (eDMA_TCD*n*_CITER_ELINKYES)

Table 368. eDMA_TCD*n*_CITER_ELINKYES field descriptions

Field	Description
0 ELINK	Enable channel-to-channel linking on minor-loop complete As the channel completes the minor loop, this flag enables linking to another channel, defined by the LINKCH field. The link target channel initiates a channel service request via an internal mechanism that sets the TCD*n*_CSR[START] bit of the specified channel. If channel linking is disabled, the CITER value is extended to 15 bits in place of a link channel number. If the major loop is exhausted, this link mechanism is suppressed in favor of the MAJORELINK channel linking. **Note:** This bit must be equal to the BITER[ELINK] bit. Otherwise, a configuration error is reported. 0 The channel-to-channel linking is disabled. 1 The channel-to-channel linking is enabled.
1:6 LINKCH	Link channel number If channel-to-channel linking is enabled (ELINK = 1), then after the minor loop is exhausted, the eDMA engine initiates a channel service request to the channel defined by these six bits by setting that channel's TCD*n*_CSR[START] bit.
7:15 CITER	Current major iteration count This 9-bit (ELINK = 1) or 15-bit (ELINK = 0) count represents the current major loop count for the channel. It is decremented each time the minor loop is completed and updated in the transfer control descriptor memory. After the major iteration count is exhausted, the channel performs a number of operations (for example: final source and destination address calculations), optionally generating an interrupt to signal channel completion before reloading the CITER field from the beginning iteration count (BITER) field. **Note:** When the CITER field is initially loaded by software, it must be set to the same value as that contained in the BITER field. If the channel is configured to execute a single service request, the initial values of BITER and CITER should be 0x0001.

25.3.32 TCD Current Minor Loop Link, Major Loop Count (Channel Linking Disabled) (eDMA_TCD*n*_CITER_ELINKNO)

See TCD Current Minor Loop Link, Major Loop Count (Channel Linking Disabled) register figure and DMA_TCDn_CITER_ELINKNO field descriptions table as follows.

Address: 0x1014 + *n**0x20 (*n* = 0 to 63) Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	ELINK	CITER														
Reset(1)	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x

1. x = Undefined at reset.

Figure 264. TCD Current Minor Loop Link, Major Loop Count (Channel Linking Disabled) (eDMA_TCD*n*_CITER_ELINKNO)

Table 369. eDMA_TCD*n*_CITER_ELINKNO field descriptions

Field	Description
0 ELINK	Enable channel-to-channel linking on minor-loop complete As the channel completes the minor loop, this flag enables linking to another channel, defined by the LINKCH field. The link target channel initiates a channel service request via an internal mechanism that sets the TCD*n*_CSR[START] bit of the specified channel.If channel linking is disabled, the CITER value is extended to 15 bits in place of a link channel number. If the major loop is exhausted, this link mechanism is suppressed in favor of the MAJORELINK channel linking. **Note:** This bit must be equal to the BITER[ELINK] bit. Otherwise, a configuration error is reported. 0 The channel-to-channel linking is disabled. 1 The channel-to-channel linking is enabled.
1:15 CITER	Current major iteration count This 9-bit (ELINK = 1) or 15-bit (ELINK = 0) count represents the current major loop count for the channel. It is decremented each time the minor loop is completed and updated in the transfer control descriptor memory. After the major iteration count is exhausted, the channel performs a number of operations (for example, final source and destination address calculations), optionally generating an interrupt to signal channel completion before reloading the CITER field from the beginning iteration count (BITER) field. **Note:** When the CITER field is initially loaded by software, it must be set to the same value as that contained in the BITER field. If the channel is configured to execute a single service request, the initial values of BITER and CITER should be 0x0001.

25.3.33 TCD Signed Destination Address Offset (eDMA_TCD*n*_DOFF)

See TCD Signed Destination Address Offset register figure and DMA_TCD*n*_DOFF field descriptions table as follows.

Address: 0x1016 + *n**0x20 (*n* = 0 to 63) Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	DOFF															
Reset(1)	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x

1. x = Undefined at reset.

Figure 265. TCD Signed Destination Address Offset (eDMA_TCD*n*_DOFF)

Table 370. eDMA_TCD*n*_DOFF field descriptions

Field	Description
0:15 DOFF	Destination address signed offset Sign-extended offset applied to the current destination address to form the next-state value as each destination write is completed.

25.3.34 TCD Last Destination Address Adjustment/Scatter Gather Address (eDMA_TCD*n*_DLASTSGA)

Address: 0x1018 + *n**0x20 (*n* = 0 to 63) Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	DLASTSGA															
Reset(1)	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	DLASTSGA															
Reset(1)	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x

1. x = Undefined at reset

Figure 266. TCD Last Destination Address Adjustment/Scatter Gather Address (eDMA_TCD*n*_DLASTSGA)

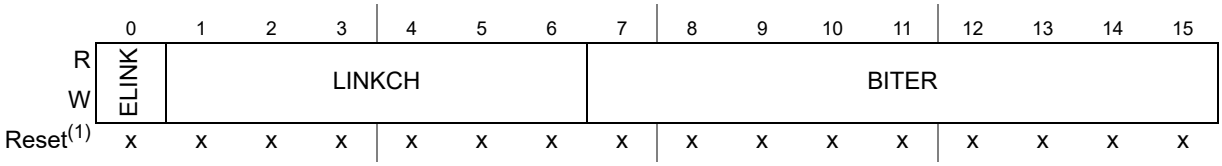
Table 371. eDMA_TCD*n*_DLASTSGA field descriptions

Field	Description

0:31
DLASTSGA
	Destination last address adjustment or the memory address for the next transfer control descriptor to be loaded into this channel (scatter/gather).
	If (TCD*n*_CSR[ESG] = 0) then
	– Adjustment value added to the destination address at the completion of the major iteration count. This value can apply to restore the destination address to the initial value or adjust the address to reference the next data structure.
	else
	– This address points to the beginning of a 0-modulo-32-byte region containing the next transfer control descriptor to be loaded into this channel. This channel reload is performed as the major iteration count completes. The scatter/gather address must be 0-modulo-32-byte, else a configuration error is reported.

25.3.35 TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Enabled) (eDMA_TCD*n*_BITER_ELINKYES)

Address: 0x101C + *n**0x20 (*n* = 0 to 63) Access: User read/write



1. x = Undefined at reset.

Figure 267. TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Enabled) (eDMA_TCD*n*_BITER_ELINKYES)

Table 372. eDMA_TCD*n*_BITER_ELINKYES field descriptions

Field	Description
0 ELINK	Enables channel-to-channel linking on minor loop complete As the channel completes the minor loop, this flag enables the linking to another channel, defined by BITER[LINKCH]. The link target channel initiates a channel service request via an internal mechanism that sets the TCD*n*_CSR[START] bit of the specified channel. If channel linking disables, the BITER value extends to 15 bits in place of a link channel number. If the major loop is exhausted, this link mechanism is suppressed in favor of the MAJORELINK channel linking. **Note:** When the software loads the TCD, this field must be set equal to the corresponding CITER field. Otherwise, a configuration error is reported. As the major iteration count is exhausted, the contents of this field is reloaded into the CITER field. 0 The channel-to-channel linking is disabled. 1 The channel-to-channel linking is enabled.
1:6 LINKCH	Link channel number If channel-to-channel linking is enabled (ELINK = 1), then after the minor loop is exhausted, the eDMA engine initiates a channel service request at the channel defined by these six bits by setting that channel's TCD*n*_CSR[START] bit. **Note:** When the software loads the TCD, this field must be set equal to the corresponding CITER field. Otherwise, a configuration error is reported. As the major iteration count is exhausted, the contents of this field is reloaded into the CITER field.
7:15 BITER	Starting major iteration count As the transfer control descriptor is first loaded by software, this 9-bit (ELINK = 1) or 15-bit (ELINK = 0) field must be equal to the value in the CITER field. As the major iteration count is exhausted, the contents of this field are reloaded into the CITER field. **Note:** When the software loads the TCD, this field must be set equal to the corresponding CITER field. Otherwise, a configuration error is reported. As the major iteration count is exhausted, the contents of this field is reloaded into the CITER field. If the channel is configured to execute a single service request, the initial values of BITER and CITER should be 0x0001.

25.3.36 TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Disabled) (eDMA_TCD*n*_BITER_ELINKNO)

Address: 0x101C + *n**0x20 (*n* = 0 to 63) Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	ELINK	BITER														
W																
Reset(1)	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x

1. x = Undefined at reset.

Figure 268. TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Disabled) (eDMA_TCD*n*_BITER_ELINKNO)

Table 373. eDMA_TCD*n*_BITER_ELINKNO field descriptions

Field	Description
0 ELINK	Enables channel-to-channel linking on minor loop complete As the channel completes the minor loop, this flag enables the linking to another channel, defined by BITER[LINKCH]. The link target channel initiates a channel service request via an internal mechanism that sets the TCD*n*_CSR[START] bit of the specified channel. If channel linking disables, the BITER value extends to 15 bits in place of a link channel number. If the major loop is exhausted, this link mechanism is suppressed in favor of the MAJORELINK channel linking. **Note:** When the software loads the TCD, this field must be set equal to the corresponding CITER field. Otherwise, a configuration error is reported. As the major iteration count is exhausted, the contents of this field is reloaded into the CITER field. 0 The channel-to-channel linking is disabled. 1 The channel-to-channel linking is enabled.
1:15 BITER	Starting major iteration count As the transfer control descriptor is first loaded by software, this 9-bit (ELINK = 1) or 15-bit (ELINK = 0) field must be equal to the value in the CITER field. As the major iteration count is exhausted, the contents of this field are reloaded into the CITER field. **Note:** When the software loads the TCD, this field must be set equal to the corresponding CITER field. Otherwise, a configuration error is reported. As the major iteration count is exhausted, the contents of this field is reloaded into the CITER field. If the channel is configured to execute a single service request, the initial values of BITER and CITER should be 0x0001.

25.3.37 TCD Control and Status (eDMA_TCD*n*_CSR)

See TCD Control and Status register figure and eDMA_TCD*n*_CSR field descriptions table as follows.

Address: 0x101E + *n**0x20 (*n* = 0 to 63) Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	BWC		MAJORLINKCH						DONE	ACTIVE	MAJORELINK	ESG	DREQ	INTHALF	INTMAJOR	START
Reset(1)	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x

1. x = Undefined at reset.

Figure 269. TCD Control and Status (eDMA_TCD*n*_CSR)

Table 374. eDMA_TCD*n*_CSR field descriptions

Field	Description
0:1 BWC	Bandwidth control Throttles the amount of bus bandwidth consumed by the eDMA. In general, as the eDMA processes the minor loop, it continuously generates read/write sequences until the minor count is exhausted. This field forces the eDMA to stall after the completion of each read/write access to control the bus request bandwidth seen by the crossbar switch (XBS). **Note:** If the source and destination sizes are equal, this field is ignored between the first and second transfers and after the last write of each minor loop. This behavior is a side effect of reducing start-up latency. 00 No eDMA engine stalls 01 Reserved 10 eDMA engine stalls for 4 cycles after each r/w 11 eDMA engine stalls for 8 cycles after each r/w

2:7
MAJORLINKCH
	Link channel number
	If (MAJORELINK = 0) then
	– No channel-to-channel linking (or chaining) is performed after the major loop counter is exhausted.
	else
	– After the major loop counter is exhausted, the eDMA engine initiates a channel service request at the channel defined by these six bits by setting that channel's TCD*n*_CSR[START] bit.

8 DONE	Channel done This flag indicates the eDMA has completed the major loop. The eDMA engine sets it as the CITER count reaches zero; The software clears it, or the hardware when the channel is activated. **Note:** This bit must be cleared to write the MAJORELINK or ESG bits. This bit resets to zero.
9 ACTIVE	Channel active This flag signals the channel is currently in execution. It is set when channel service begins, and the eDMA clears it as the minor loop completes or if any error condition is detected.

Table 374. eDMA_TCD*n*_CSR field descriptions (continued)

Field	Description
10 MAJORELINK	Enable channel-to-channel linking on major loop complete As the channel completes the major loop, this flag enables the linking to another channel, defined by MAJORLINKCH. The link target channel initiates a channel service request via an internal mechanism that sets the TCD*n*_CSR[START] bit of the specified channel. **Note:** To support the dynamic linking coherency model, this field is forced to zero when written to while the TCD*n*_CSR[DONE] bit is set. 0 The channel-to-channel linking is disabled. 1 The channel-to-channel linking is enabled.
11 ESG	Enable scatter/gather processing As the channel completes the major loop, this flag enables scatter/gather processing in the current channel. If enabled, the eDMA engine uses DLASTSGA as a memory pointer to a 0-modulo-32 address containing a 32-byte data structure loaded as the transfer control descriptor into the local memory. **Note:** To support the dynamic scatter/gather coherency model, this field is forced to zero when written to while the TCD*n*_CSR[DONE] bit is set. 0 The current channel's TCD is normal format. 1 The current channel's TCD specifies a scatter gather format. The DLASTSGA field provides a memory pointer to the next TCD to be loaded into this channel after the major loop completes its execution.
12 DREQ	Disable request {H,L} 0 The channel's ERQ bit is not affected. 1 The channel's ERQ bit is cleared when the major loop is complete.
13 INTHALF	Enable an interrupt when major counter is half complete If this flag is set, the channel generates an interrupt request by setting the appropriate bit in the INT when the current major iteration count reaches the halfway point. Specifically, the comparison performed by the eDMA engine is (CITER == (BITER >> 1)). This halfway point interrupt request is provided to support double-buffered (aka ping-pong) schemes or other types of data movement where the processor needs an early indication of the transfer's progress. **Note:** If BITER is set to 1, do not use INTHALF. Use INTMAJOR instead. 0 The half-point interrupt is disabled. 1 The half-point interrupt is enabled.
14 INTMAJOR	Enable an interrupt when major iteration count completes If this flag is set, the channel generates an interrupt request by setting the appropriate bit in the INT when the current major iteration count reaches zero. 0 The end-of-major loop interrupt is disabled 1 The end-of-major loop interrupt is enabled
15 START	Channel start If this flag is set, the channel is requesting service. The eDMA hardware automatically clears this flag after the channel begins execution. This bit resets to zero. 0 The channel is not explicitly started. 1 The channel is explicitly started via a software initiated service request.

25.4 Functional description

This section provides a description of the DMA request sources, an overview of the microarchitecture, and functional operation of the eDMA module.

25.4.1 eDMA microarchitecture

The eDMA module is partitioned into two major modules: the eDMA engine and the transfer-control descriptor local memory. Additionally, the eDMA engine is further partitioned into four submodules:

- eDMA engine
  - Address path:

    This block implements registered versions of two channel transfer control descriptors, channel x and channel y, and manages all master bus-address calculations. All the channels provide the same functionality. This structure allows data transfers associated with one channel to be preempted after the completion of a read/write sequence if a higher priority channel activation is asserted while the first channel is active. After a channel is activated, it runs until the minor loop is completed, unless preempted by a higher priority channel. This provides a mechanism (enabled by DCHPRI*n*[ECP]) where a large data move operation can be preempted to minimize the time another channel is blocked from execution.

    When any channel is selected to execute, the contents of its TCD are read from local memory and loaded into the address path channel x registers for a normal start and into channel y registers for a preemption start. After the minor loop completes execution, the address path hardware writes the new values for the TCD*n* _{SADDR, DADDR, CITER} back to local memory. If the major iteration count is exhausted, additional processing is performed, including updates to the final address pointer, reloading the TCD*n* _CITER field, and a possible fetch of the next TCD*n* from memory as part of a scatter/gather operation.
  - Data path:

    This block implements the bus master read/write datapath. It includes 32 bytes of register storage and the necessary multiplex logic to support any data alignment. The internal read data bus is the primary input, and the internal write data bus is the primary output.

    The address and data path modules directly support the two-stage pipelined internal bus. The address path module represents the first stage of the bus pipeline (address phase), while the data path module implements the second stage of the pipeline (data phase).
  - Program model/channel arbitration:

    This block implements the first section of the eDMA programming model as well as the channel arbitration logic. The programming model registers are connected to the internal peripheral bus (not shown). The eDMA peripheral request inputs and interrupt request outputs are also connected to this block (via control logic).
  - Control:

    This block provides all the control functions for the eDMA engine. For data transfers where the source and destination sizes are equal, the eDMA engine performs a series of source read/destination write operations until the number of bytes specified in the minor loop byte count has moved. For descriptors where the sizes are not equal, multiple accesses of the smaller size data are required for each reference of the larger size. As an example, if the source size references 16-

bit data and the destination is 32-bit data, two reads are performed, then one 32-bit write.

- Transfer control descriptor memory
  - Memory controller:

    This logic implements the dual-ported controller, managing accesses from the eDMA engine as well as references from the internal peripheral bus.

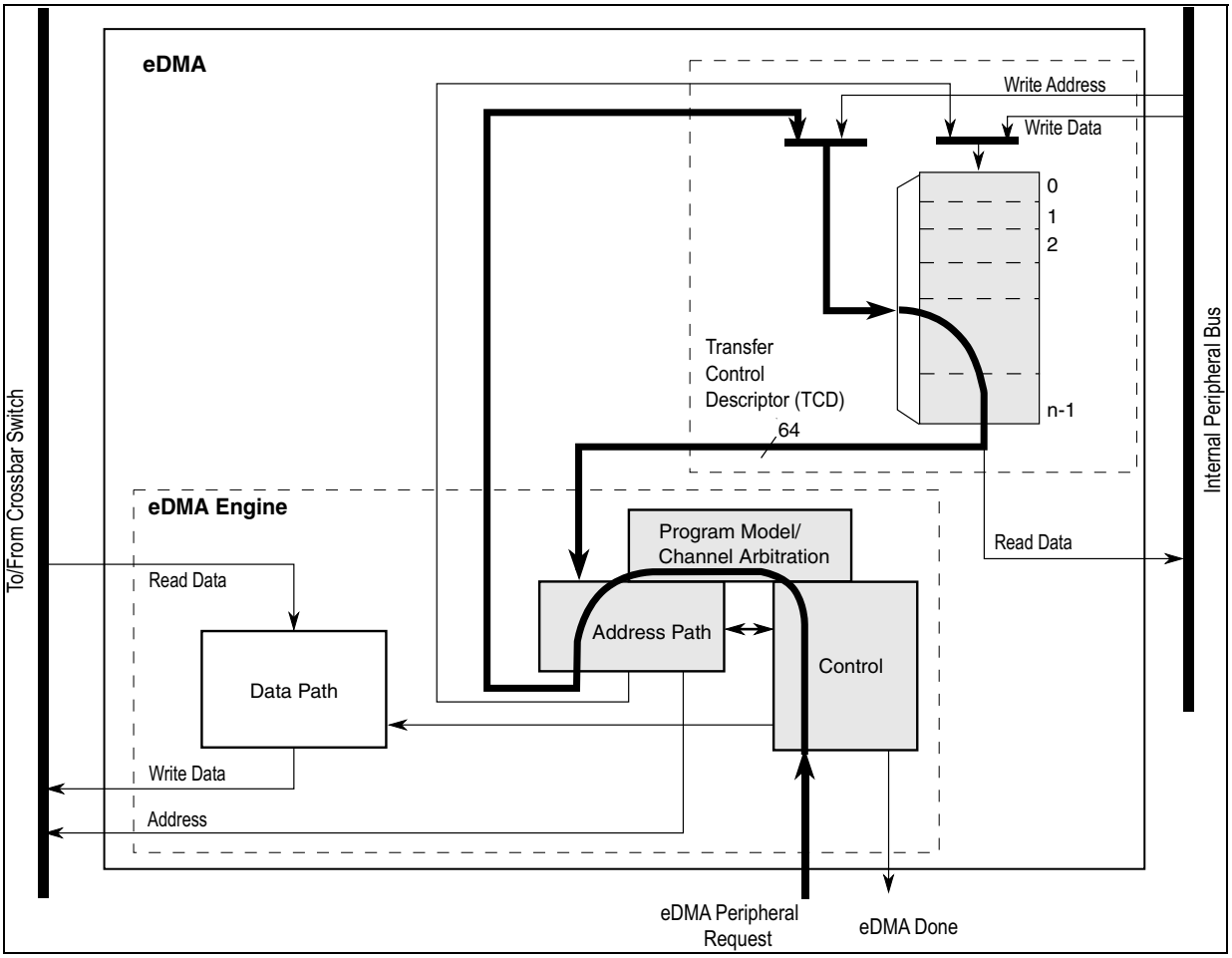
    As noted earlier, in the event of simultaneous accesses, the eDMA engine is given priority and the peripheral transaction is stalled.
  - Memory array:

    TCD storage is implemented using a single-port, synchronous RAM array.

25.4.2 eDMA basic data flow

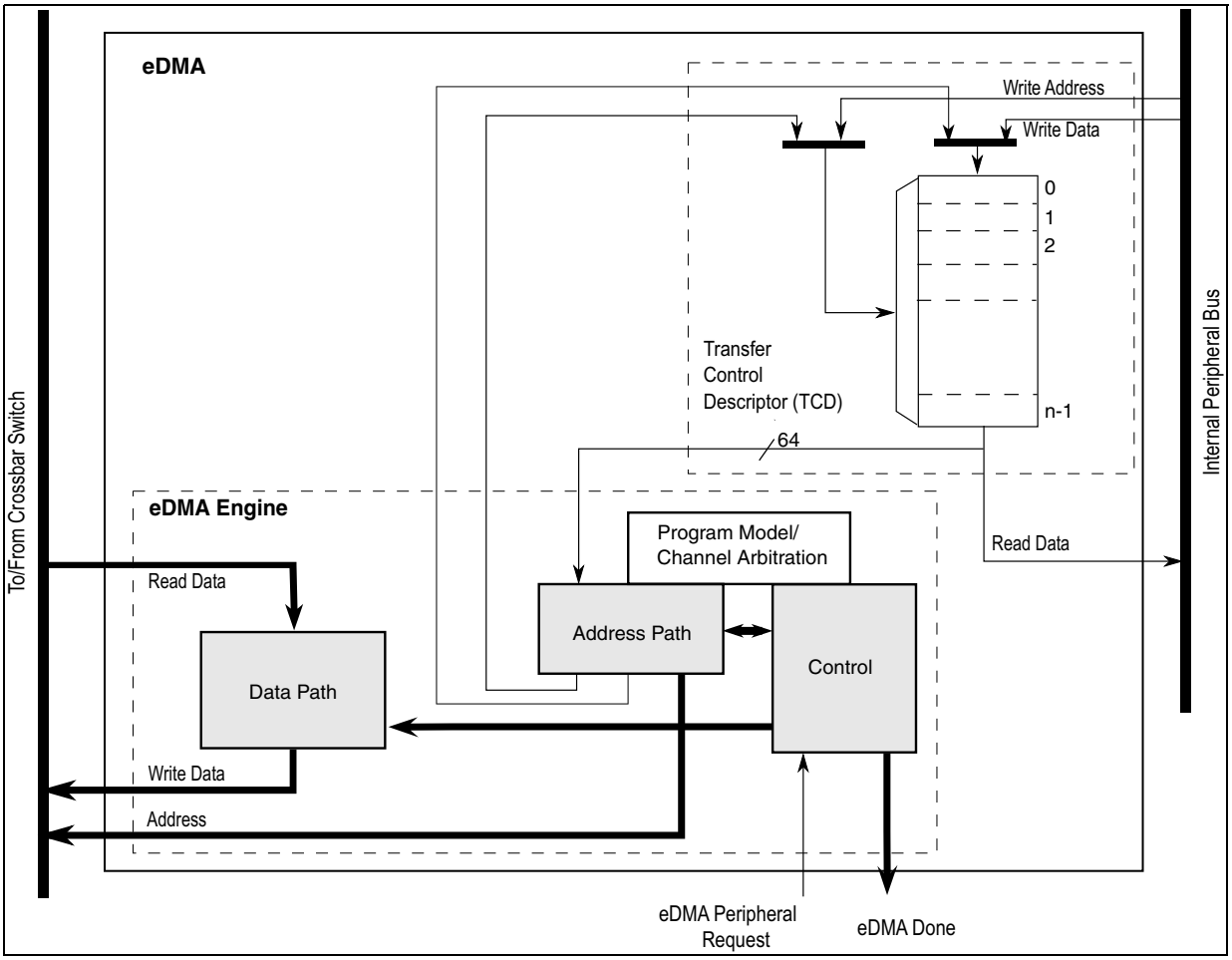
The basic flow of a data transfer can be partitioned into three segments. As shown in *Figure 270*, the first segment involves the channel activation. In the diagram, this example uses the assertion of the eDMA peripheral request signal to request service for channel *n*. Channel activation via software and the TCD*n*_CSR[START] bit follows the same basic flow as peripheral requests. The eDMA request input signal is registered internally and then routed through the eDMA engine: first through the control module, then into the program model and channel arbitration. In the next cycle, channel arbitration is performed, using the fixed-priority or round-robin algorithm. After arbitration is complete, the activated channel number is sent through the address path and converted into the address to access the local memory for TCD*n*. Next, the TCD memory is accessed and the descriptor read from the local memory and loaded into the eDMA engine address path channel x or y registers. The TCD memory is 64 bits wide to minimize the time needed to fetch the activated channel descriptor and load it into the address path channel x or y registers.

Figure 270. DMA operation, part 1



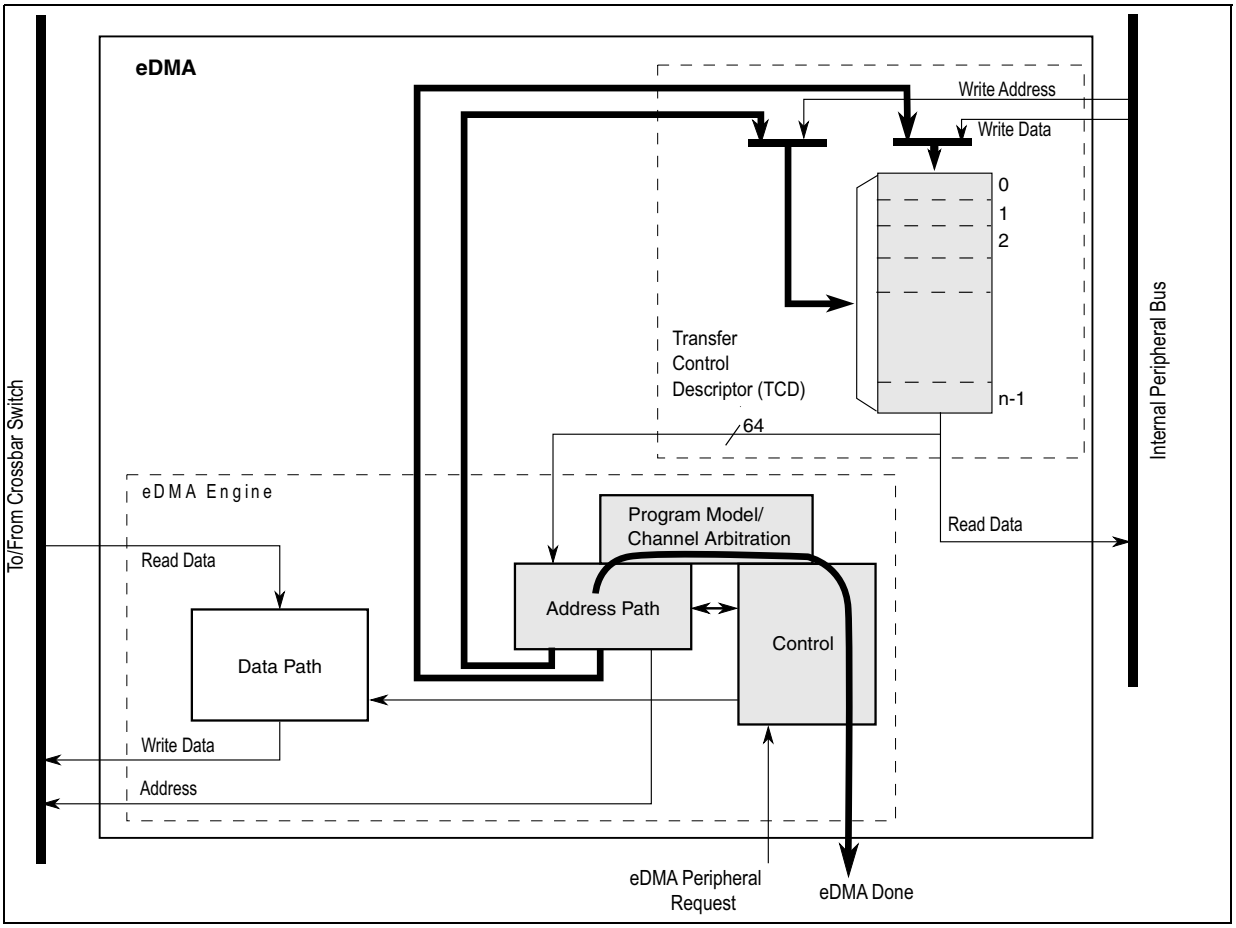
In the second part of the basic data flow as shown in *Figure 271* the modules associated with the data transfer (address path, data path, and control) sequence through source reads and destination writes to perform the actual data movement. The source reads are initiated and the fetched data is temporarily stored in the data path block until it is gated onto the internal bus during the destination write. This source read/destination write processing continues until the minor byte count has transferred.

Figure 271. DMA operation, part 2



After the minor byte count has moved, the final phase of the basic data flow is performed. In this segment, the address path logic performs updates to certain fields in the appropriate TCD, for example, SADDR, DADDR, and CITER. If the major iteration count is exhausted, additional operations are performed. These include the final address adjustments and reloading of the BITER field into the CITER. Assertion of an optional interrupt request also occurs at this time, as a possible fetch of a new TCD from memory does using the scatter/gather address pointer included in the descriptor. The updates to the TCD memory and the assertion of an interrupt request are shown in *Figure 272*.

Figure 272. DMA operation, part 3



25.4.3 Error reporting and handling

Channel errors are reported in the ES register and can be caused by:

- A configuration error (an illegal setting in the transfer-control descriptor or an illegal priority register setting in Fixed Arbitration mode),
- An uncorrectable ECC error, or
- An error termination to a bus master read or write cycle.

A configuration error is reported when the starting source or destination address, source or destination offsets, minor loop byte count, or the transfer size represent an inconsistent state. Each of these possible causes is detailed in the list below:

- The addresses and offsets must be aligned on 0-modulo-transfer-size boundaries.
- The minor loop byte count must be a multiple of the source and destination transfer sizes.
- All source reads and destination writes must be configured to the natural boundary of the programmed transfer size, respectively.
- In Fixed Arbitration mode, a configuration error is caused by any two channel priorities being equal within a group, or any group priority levels being equal among the groups.

All channel priority levels within a group must be unique and all group priority levels among the groups must be unique when Fixed Arbitration mode is enabled.

- If a scatter/gather operation is enabled upon channel completion, a configuration error is reported if the scatter/gather address (DLAST_SGA) is not aligned on a 32-byte boundary.
- If minor loop channel linking is enabled upon channel completion, a configuration error is reported when the link is attempted if the TCD*n*_CITER[E_LINK] bit does not equal the TCD*n*_BITER[E_LINK] bit.

If enabled, all configuration error conditions, except the scatter/gather and minor-loop link errors, report as the channel activates and asserts an error interrupt request. A scatter/gather configuration error is reported when the scatter/gather operation begins at major loop completion when properly enabled. A minor loop channel link configuration error is reported when the link operation is serviced at minor loop completion.

If a system bus read or write is terminated with an error, the data transfer is stopped and the appropriate bus error flag set. In this case, the state of the channel's transfer control descriptor is updated by the eDMA engine with the current source address, destination address, and current iteration count at the point of the fault. When a system-bus error occurs, the channel terminates after the read or write transaction (which is already pipelined after errant access) has completed. If a bus error occurs on the last read prior to beginning the write sequence, the write executes using the data captured during the bus error. If a bus error occurs on the last write prior to switching to the next read sequence, the read sequence executes before the channel terminates due to the destination bus error.

If an uncorrectable ECC error occurs during a peripheral bus access, the access is terminated with a bus error. The occurrence of an uncorrectable ECC error during an eDMA engine read causes the eDMA engine to stop the active channel immediately. The ECC error and channel number are recorded in the eDMA's Error Status register.

A transfer may be canceled by software with the CR[CX] bit. When a cancel transfer request is recognized, the DMA engine stops processing the channel. The current read-write sequence is allowed to finish. If the cancel occurs on the last read-write sequence of a major or minor loop, the cancel request is discarded and the channel retires normally.

The error cancel transfer is the same as a cancel transfer except the ES register is updated with the canceled channel number and ECX is set. The TCD of a canceled channel contains the source and destination addresses of the last transfer saved in the TCD. If the channel needs to be restarted, you must re-initialize the TCD since the aforementioned fields no longer represent the original parameters. When a transfer is canceled by the error cancel transfer mechanism, the channel number is loaded into DMA_ES[ERRCHN] and ECX and VLD are set. In addition, an error interrupt may be generated if enabled.

The occurrence of any error causes the eDMA engine to stop the active channel immediately, and the appropriate channel bit in the eDMA error register is asserted. At the same time, the details of the error condition are loaded into the ES register. The major loop complete indicators, setting the transfer control descriptor DONE flag and the possible assertion of an interrupt request, are not affected when an error is detected. After the error status has been updated, the eDMA engine continues operating by servicing the next appropriate channel. A channel that experiences an error condition is not automatically disabled. If a channel is terminated by an error and then issues another service request before the error is fixed, that channel executes and terminates with the same error condition.

25.4.4 Channel preemption

Channel preemption is enabled on a per-channel basis by setting the DCHPRI*n*[ECP] bit. Channel preemption allows the executing channel's data transfers to temporarily suspend in favor of starting a higher priority channel. After the preempting channel has completed all its minor loop data transfers, the preempted channel is restored and resumes execution. After the restored channel completes one read/write sequence, it is again eligible for preemption. If any higher priority channel is requesting service, the restored channel is suspended and the higher priority channel is serviced. Nested preemption (attempting to preempt a preempting channel) is not supported. After a preempting channel begins execution, it cannot be preempted. Preemption is available only when fixed arbitration is selected for both group and channel arbitration modes.

A channel's ability to preempt another channel can be disabled by setting DCHPRI*n*[DPA]. When a channel's preempt ability is disabled, that channel cannot suspend a lower priority channel's data transfer; regardless of the lower priority channel's ECP setting. This allows for a pool of low-priority, large data-moving channels to be defined. These low-priority channels can be configured so that they do not preempt each other, thus preventing a low-priority channel from consuming the preempt slot normally available to a true, high-priority channel.

25.4.5 eDMA performance

This section addresses the performance of the eDMA module, focusing on two separate metrics. In the traditional data movement context, performance is best expressed as the peak data transfer rates achieved using the eDMA. In most implementations, this transfer rate is limited by the speed of the source and destination address spaces. In a second context where device-paced movement of single data values to/from peripherals is dominant, a measure of the requests that can be serviced in a fixed time is a more interesting metric. In this environment, the speed of the source and destination address spaces remains important, but the microarchitecture of the eDMA also factors significantly into the resulting metric.

The peak transfer rates for several different source and destination transfers are shown in *Table 375* and *Table 376*. The following assumptions apply to those tables:

- Internal SRAM can be accessed with zero wait states when viewed from the system bus data phase.
- All internal peripheral bus reads require one wait-state, and internal peripheral bus writes one wait-state, again viewed from the system bus data phase.
- All internal peripheral bus accesses are 32 bits in size.
- All internal SRAM accesses are 64 bits in size.

Table 375 presents a peak transfer rate comparison, measured in megabytes per second.

Table 375. eDMA peak transfer rates (Mbyte/s)

System speed, width	From internal SRAM to internal SRAM	From 32-bit internal peripheral bus to internal SRAM	From internal SRAM to 32-bit internal peripheral bus
100 MHz, 64 bits	400.0	160.0	160.0
150 MHz, 64 bits	600.0	240.0	240.0
200 MHz, 64 bits	800.0	320.0	320.0

The second performance metric is a measure of the number of DMA requests that can be serviced in a given amount of time. For this metric, it is assumed the peripheral request causes the channel to move a single internal peripheral bus-mapped operand to/from internal SRAM. The same timing assumptions used in the previous example apply to this calculation. In particular, this metric also reflects the time required to activate the channel. The eDMA design supports the following hardware service request sequence:

- Cycle 1: eDMA peripheral request is asserted.
- Cycle 2: The eDMA peripheral request is registered locally in the eDMA module and qualified. (TCD*n* _CSR[START] bit initiated requests start at this point with the registering of the user write to TCD*n* Word 7).
- Cycle 3: Channel arbitration begins.
- Cycle 4: Channel arbitration completes. The transfer control descriptor local memory read is initiated.
- Cycles 5–6: The first two parts of the activated channel's TCD is read from the local memory. The memory width to the eDMA engine is 64 bits, so the entire descriptor can be accessed in four cycles.
- Cycle 7: The first system bus read cycle is initiated, as the third part of the channel's TCD is read from the local memory. Depending on the state of the crossbar switch, arbitration at the system bus may insert an additional cycle of delay here.
- Cycles 8 to $x^{(11)}$: The last part of the TCD is read in. This cycle represents the first data phase for the read, and the address phase for the destination write.

  The exact timing from this point is a function of the transfer size and response times for the channel's read and write accesses. In this case of two internal peripheral bus reads and a single 64-bit internal SRAM write, the combined data phase time is five cycles. For a single SRAM read and two 64-bit internal peripheral bus writes, the combined data phase time is five cycles.
- Cycle $x^{(11)}$ + 1: This cycle represents the data phase of the last destination write.
- Cycle $x^{(11)}$ + 2: The eDMA engine completes the execution of the inner minor loop and prepares to write back the required TCD*n* fields into the local memory. The TCD*n* Word 7 is read and checked for channel linking or scatter/gather requests.
- Cycle $x^{(11)}$ + 3: The appropriate fields in the first part of the TCD*n* are written back into the local memory.
- Cycle $x^{(11)}$ + 4: The fields in the second part of the TCD*n* are written back into the local memory. This cycle coincides with the next channel arbitration cycle start.
- Cycle $x^{(11)}$ + 5: The next channel to be activated performs the read of the first part of its TCD from the local memory. This is equivalent to Cycle 4 for the first channel's service request.

Assuming zero wait states on the system bus, DMA requests can be processed every nine cycles. Assuming an average of the access times associated with internal peripheral bus-to-SRAM (three cycles) and SRAM-to-internal peripheral bus (three cycles), DMA requests can be processed every 10 cycles (4 + (3 + 3)/2 + 3). This is the time from Cycle 4 to Cycle *x* + 5. The resulting peak request rate, as a function of the system frequency, is shown in *Table 376*. This metric represents millions of requests per second. A single read/write operation with same transfer size for the source and destination is assumed.

11. "*x*" represents an undefined cycle value dependent on transfer sizes and response times for the channel's read and write accesses.

Table 376. eDMA peak request rate (MReq/s)

System frequency (MHz)	Request rate	
	zero wait states	with wait states
100.0	11.1	10.0
150.0	16.6	15.0
200.0	22.2	20.0

A general formula to compute the peak request rate (with overlapping requests) is:

Equation 1 PEAKreq = freq / [entry + (1 + read_ws) + (1 + write_ws) + exit]

where:

- PEAKreq = peak request rate
- freq = system frequency
- entry = channel startup (4 cycles)
- read_ws = wait states seen during the system bus read data phase
- write_ws = wait states seen during the system bus write data phase
- exit = channel shutdown (3 cycles)

For example: consider a system with the following characteristics:

- Internal SRAM can be accessed with zero wait states when viewed from the system bus data phase.
- All internal peripheral bus reads require one wait state, and internal peripheral bus writes one wait-state, again viewed from the system bus data phase.
- System operates at 150 MHz.

For an SRAM to internal peripheral bus transfer,

Equation 2 PEAKreq = 150 MHz / [4 + (1 + 0) + (1 + 1) + 3] cycles = 15.0 Mreq/s

For an internal peripheral bus-to-SRAM transfer,

Equation 3 PEAKreq = 150 MHz / [4 + (1 + 1) + (1 + 0) + 3] cycles = 15.0 Mreq/s

The minimum number of cycles to perform a single read/write, with zero wait states on the system bus, from a cold start (where no channel is executing and the eDMA is idle):

- 11 cycles for a software (TCD*n*_CSR[START] bit) request
- 12 cycles for a hardware (eDMA peripheral request signal) request

Two cycles account for the arbitration pipeline and one extra cycle on the hardware request resulting from the internal registering of the eDMA peripheral request signals. For the peak request rate calculations above, the arbitration and request registering are absorbed in or overlap the previous executing channel.

When channel linking or scatter/gather is enabled, a two cycle delay is imposed on the next channel selection and startup. This allows the link channel or the scatter/gather channel to be eligible and considered in the arbitration pool for next channel selection.

25.5 Initialization/application information

The following sections discuss initialization of the eDMA and programming considerations.

25.5.1 eDMA initialization

A typical initialization of the eDMA has the following sequence:

1. Write the CR register if a configuration other than the default is desired.
2. Write the channel priority levels into the DCHPRI*n* registers if a configuration other than the default is desired.
3. Enable error interrupts in the EEI registers if so desired.
4. Write the 32-byte TCD for each channel that may request service.
5. Enable any hardware service requests via the ERQ register.
6. Request channel service by software (setting the TCD*n*_CSR[START] bit) or hardware (slave device asserting its eDMA peripheral request signal).

After any channel requests service, a channel is selected for execution based on the arbitration and priority levels written into the programmer's model. The eDMA engine reads the entire TCD, including the TCD control and status fields (shown in *Table 377*) for the selected channel into its internal address path module. As the TCD is read, the first transfer is initiated on the internal bus unless a configuration error is detected. Transfers from the source (as defined by the source address, TCD*n*_SADDR) to the destination (as defined by the destination address, TCD*n*_DADDR) continue until the specified number of bytes (TCD*n*_NBYTES) are transferred. When the transfer is complete, the eDMA engine's local TCD*n*_SADDR, TCD*n*_DADDR, and TCD*n*_CITER are written back to the main TCD memory and any minor loop channel linking is performed, if enabled. If the major loop is exhausted, further post processing executes (interrupts, major loop channel linking, and scatter/gather operations) if enabled.

Table 377. TCD control and status fields

TCD*n*_CSR field name	Description
START	Control bit to start channel explicitly when using a software initiated DMA service (automatically cleared by hardware)
ACTIVE	Status bit indicating the channel is currently in execution
DONE	Status bit indicating major loop completion (cleared by software when using a software initiated DMA service)
D_REQ	Control bit to disable DMA request at end of major loop completion when using a hardware initiated DMA service
BWC	Control bits for throttling bandwidth control of a channel
E_SG	Control bit to enable scatter-gather feature
INT_HALF	Control bit to enable interrupt when major loop is half complete
INT_MAJ	Control bit to enable interrupt when major loop completes

Figure 273 shows how each DMA request initiates one minor-loop transfer (iteration) without CPU intervention. DMA arbitration can occur after each minor loop, and one level of

minor loop DMA preemption is allowed. The number of minor loops in a major loop is specified by the beginning iteration count (BITER).

Figure 273. Example of multiple loop iterations

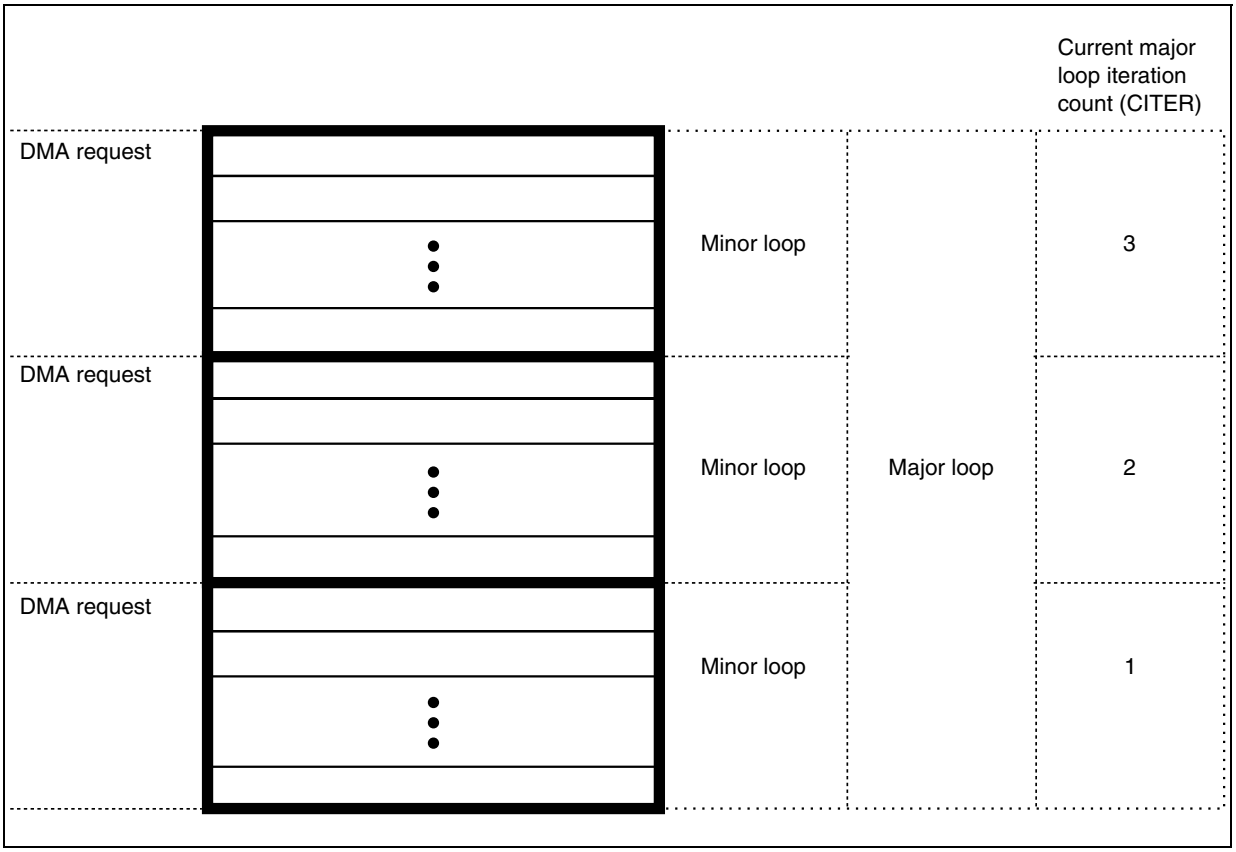
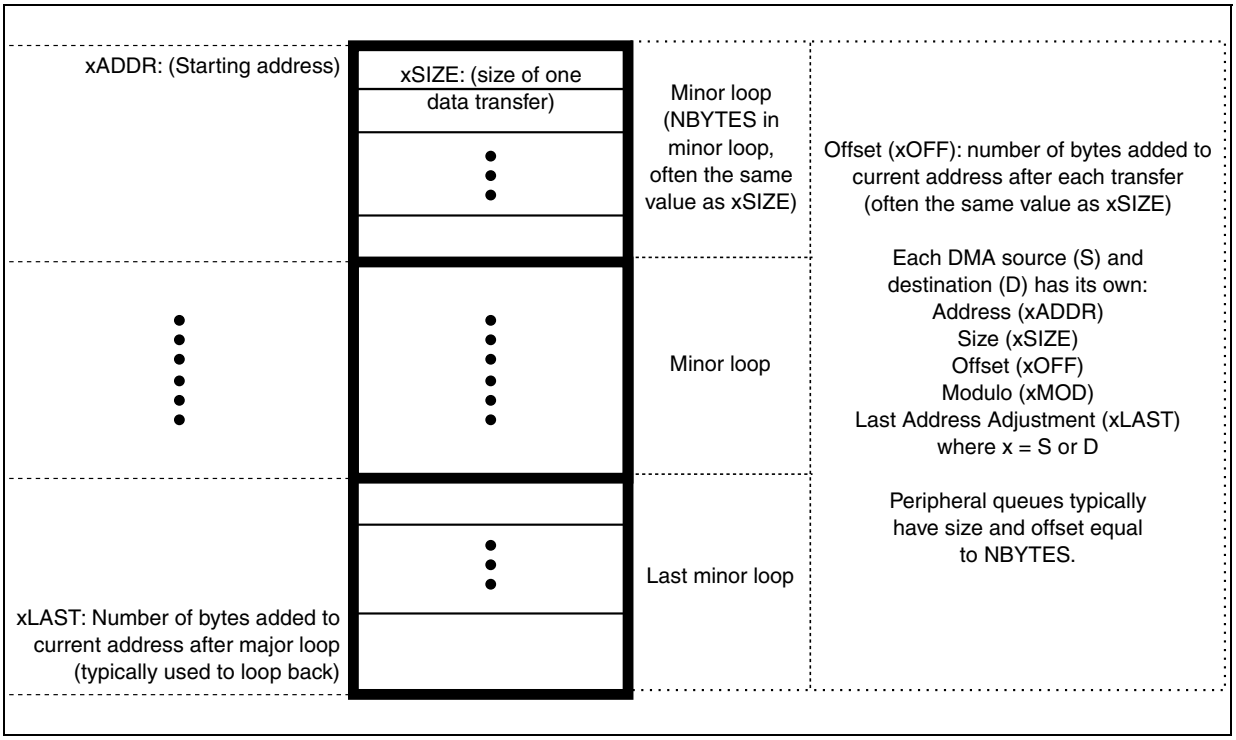


Figure 274 lists the memory array terms and how the TCD settings interrelate.

Figure 274. Memory array terms



25.5.2 DMA programming errors

The eDMA performs various tests on the transfer control descriptor to verify consistency in the descriptor data. Most programming errors are reported on a per-channel basis with the exception of two errors: the group priority error (ES[GPE]) and the channel priority error (ES[CPE]).

For all error types other than group or channel priority errors, the channel number causing the error is recorded in the ES register. If the error source is not removed before the next activation of the problem channel, the error is detected and recorded again.

Channel priority errors are identified within a group once that group has been selected as the active group. For example:

1. The eDMA is configured for fixed-group and fixed-channel arbitration modes.
2. Group 3 is the highest priority and all channels are unique in that group.
3. Group 2 is the next highest priority and has two channels with the same priority level.
4. If Group 3 has any service requests, those requests will be executed.
5. Once all of Group 3 requests have completed, Group 2 will be the next active group.
6. If Group 2 has a service request, then an undefined channel in Group 2 will be selected and a channel priority error will occur.
7. This repeats until all of the Group 2 requests have been removed or a higher priority Group 3 request comes in.

In this sequence, for item 2, the eDMA Acknowledge lines will assert only if the selected channel is requesting service via the eDMA peripheral request signal. If interrupts are enabled for all channels, the user will get an error interrupt, but the channel number for the ERR register and the error interrupt request line may be wrong because they reflect the

selected channel. A group priority error is global and any request in any group will cause a group priority error.

If priority levels are not unique, when any channel requests service, a channel priority error is reported. The highest (channel/group) priority with an active request is selected, but the lowest numbered (channel/group) with that priority is selected by arbitration and executed by the eDMA engine. The hardware service request handshake signals, error interrupts, and error reporting is associated with the selected channel.

25.5.3 DMA Arbitration mode considerations

This section discusses arbitration considerations for the eDMA.

25.5.3.1 Fixed-group arbitration, fixed-channel arbitration

In this mode, the channel service request from the highest priority channel in the highest priority group is selected to execute. If the eDMA is programmed so the channels within one group use “fixed” priorities, and that group is assigned the highest “fixed” priority of all groups, it is possible for that group to take all the bandwidth of the eDMA controller—that is, no other groups will be serviced if there is always at least one DMA request pending on a channel in the highest priority group when the controller arbitrates the next DMA request. The advantage of this scenario is that latency can be small for channels that need to be serviced quickly. Preemption is available in this scenario only.

25.5.3.2 Round-robin group arbitration, fixed-channel arbitration

The occurrence of one or more DMA requests from one or more groups, the channel with the highest priority from a specific group will be serviced first. Groups are serviced starting with the highest group number with a service request and rotating through to the lowest group number containing a service request.

Once the channel request is serviced, the group round-robin algorithm will select the highest pending request from the next group in the round-robin sequence. Servicing continues round-robin, always servicing the highest priority channel in the next group in the sequence, or just skipping a group if it has no pending requests.

If a channel requests service at a rate that equals or exceeds the round-robin service rate, then that channel will always be serviced before lower priority channels in the same group, and thus the lower priority channels will never be serviced. The advantage of this scenario is that no one group will consume all the eDMA bandwidth. The highest priority channel selection latency is potentially greater than fixed/fixed arbitration. Excessive request rates on high priority channels could prevent the servicing of lower priority channels in the same group.

25.5.3.3 Round-robin group arbitration, round-robin channel arbitration

Groups will be serviced as described in *Section 25.5.3.2: Round-robin group arbitration, fixed-channel arbitration*, but this time channels will be serviced in channel number order. Only one channel is serviced from each requesting group for each round-robin pass through the groups.

Within each group, channels are serviced starting with the highest channel number and rotating through to the lowest channel number without regard to channel priority levels.

Because channels are serviced in round-robin manner, any channel that generates DMA requests faster than a combination of the group round-robin service rate and the channel

service rate for its group will not prevent the servicing of other channels in its group. Any DMA requests that are not serviced are simply lost, but at least one channel will be serviced.

This scenario ensures that all channels will be guaranteed service at some point, regardless of the request rates. However, the potential latency could be quite high. All channels are treated equally. Priority levels are not used in round-robin/round-robin mode.

25.5.3.4 Fixed-group arbitration, round-robin channel arbitration

The highest priority group with a request will be serviced. Lower priority groups will be serviced if no pending requests exist in the higher priority groups.

Within each group, channels are serviced starting with the highest channel number and rotating through to the lowest channel number without regard to the channel priority levels assigned within the group.

This scenario could cause the same bandwidth consumption problem as indicated in *Section 25.5.3.1: Fixed-group arbitration, fixed-channel arbitration* but all the channels in the highest priority group will get serviced. Service latency will be short on the highest priority group, but could potentially get much longer as the group priority decreases.

25.5.4 DMA transfer

This section discusses how to perform DMA transfers with the eDMA.

25.5.4.1 Single request

To perform a simple transfer of *n* bytes of data with one activation, set the major loop to one (TCD*n*_CITER = TCD*n*_BITER = 1). The data transfer begins after the channel service request is acknowledged and the channel is selected to execute. After the transfer is complete, the TCD*n*_CSR[DONE] bit is set and an interrupt generates if properly enabled.

For example, the following TCD entry is configured to transfer 16 bytes of data. The eDMA is programmed for one iteration of the major loop, transferring 16 bytes per iteration. The source memory has a byte wide memory port located at 0x1000. The destination memory has a 32-bit-wide port located at 0x2000. The address offsets are programmed in increments to match the transfer size: one byte for the source and four bytes for the destination. The final source and destination addresses are adjusted to return to their beginning values.

```
TCDn_CITER = TCDn_BITER = 1
TCDn_NBYTES = 16
TCDn_SADDR = 0x1000
TCDn_SOFF = 1
TCDn_ATTR[SSIZE] = 0
TCDn_SLAST = -16
TCDn_DADDR = 0x2000
TCDn_DOFF = 4
TCDn_ATTR[DSIZE] = 2
TCDn_DLAST_SGA= -16
TCDn_CSR[INT_MAJ] = 1
TCDn_CSR[START] = 1 (Should be written last after all other fields have
 been initialized)
All other TCDn fields = 0
```

This generates the following event sequence:

1. User write to the TCD*n*_CSR[START] bit requests channel service.
2. The channel is selected by arbitration for servicing.
3. eDMA engine writes: TCD*n*_CSR[DONE] = 0, TCD*n*_CSR[START] = 0, TCD*n*_CSR[ACTIVE] = 1.
4. eDMA engine reads: channel TCD data from local memory to internal register file.
5. The source-to-destination transfers are executed as follows:
   a) Read byte from location 0x1000, read byte from location 0x1001, read byte from 0x1002, read byte from 0x1003.
   b) Write 32 bits to location 0x2000 → first iteration of the minor loop.
   c) Read byte from location 0x1004, read byte from location 0x1005, read byte from 0x1006, read byte from 0x1007.
   d) Write 32 bits to location 0x2004 → second iteration of the minor loop.
   e) Read byte from location 0x1008, read byte from location 0x1009, read byte from 0x100A, read byte from 0x100B.
   f) Write 32 bits to location 0x2008 → third iteration of the minor loop.
   g) Read byte from location 0x100C, read byte from location 0x100D, read byte from 0x100E, read byte from 0x100F.
   h) Write 32 bits to location 0x200C → last iteration of the minor loop → major loop complete.
6. The eDMA engine writes: TCD*n*_SADDR = 0x1000, TCD*n*_DADDR = 0x2000, TCD*n*_CITER = 1 (TCD*n*_BITER).
7. The eDMA engine writes: TCD*n*_CSR[ACTIVE] = 0, TCD*n*_CSR[DONE] = 1, INT[*n*] = 1.
8. The channel retires and the eDMA goes idle or services the next channel.

25.5.4.2 Multiple requests

Besides transferring 32 bytes via two hardware requests, the next example is the same as the previous one. The only fields that change are the major loop iteration count and the final address offsets. The eDMA is programmed for two iterations of the major loop transferring 16 bytes per iteration. After the channel's hardware requests are enabled in the ERQ register, the slave device initiates channel service requests.

```
TCDn_CITER = TCDn_BITER = 2
TCDn_SLAST = -32
TCDn_DLAST_SGA = -32
```

This would generate the following sequence of events:

1. First hardware (eDMA peripheral) request for channel service.
2. The channel is selected by arbitration for servicing.
3. eDMA engine writes: TCD*n*_CSR[DONE] = 0, TCD*n*_CSR[START] = 0, TCD*n*_CSR[ACTIVE] = 1.
4. eDMA engine reads: channel TCD*n* data from local memory to internal register file.
5. The source to destination transfers are executed as follows:
   a) Read byte from location 0x1000, read byte from location 0x1001, read byte from 0x1002, read byte from 0x1003.
   b) Write 32 bits to location 0x2000 → first iteration of the minor loop.

c) Read byte from location 0x1004, read byte from location 0x1005, read byte from 0x1006, read byte from 0x1007.
d) Write 32 bits to location 0x2004 → second iteration of the minor loop.
e) Read byte from location 0x1008, read byte from location 0x1009, read byte from 0x100A, read byte from 0x100B.
f) Write 32 bits to location 0x2008 → third iteration of the minor loop.
g) Read byte from location 0x100C, read byte from location 0x100D, read byte from 0x100E, read byte from 0x100F.
h) Write 32 bits to location 0x200C → last iteration of the minor loop.

6. eDMA engine writes: TCD*n*_SADDR = 0x1010, TCD*n*_DADDR = 0x2010, TCD*n*_CITER = 1.
7. eDMA engine writes: TCD*n*_CSR[ACTIVE] = 0.
8. The channel retires → one iteration of the major loop. The eDMA goes idle or services the next channel.
9. Second hardware (eDMA peripheral) requests channel service.
10. The channel is selected by arbitration for servicing.
11. eDMA engine writes: TCD*n*_CSR[DONE] = 0, TCD*n*_CSR[START] = 0, TCD*n*_CSR[ACTIVE] = 1.
12. eDMA engine reads: channel TCD data from local memory to internal register file.
13. The source to destination transfers are executed as follows:
    a) Read byte from location 0x1010, read byte from location 0x1011, read byte from 0x1012, read byte from 0x1013.
    b) Write 32 bits to location 0x2010 → first iteration of the minor loop.
    c) Read byte from location 0x1014, read byte from location 0x1015, read byte from 0x1016, read byte from 0x1017.
    d) Write 32 bits to location 0x2014 → second iteration of the minor loop.
    e) Read byte from location 0x1018, read byte from location 0x1019, read byte from 0x101A, read byte from 0x101B.
    f) Write 32 bits to location 0x2018 → third iteration of the minor loop.
    g) Read byte from location 0x101C, read byte from location 0x101D, read byte from 0x101E, read byte from 0x101F.
    h) Write 32 bits to location 0x201C → last iteration of the minor loop → major loop complete.
14. eDMA engine writes: TCD*n*_SADDR = 0x1000, TCD*n*_DADDR = 0x2000, TCD*n*_CITER = 2(TCD*n*_BITER).
15. eDMA engine writes: TCD*n*_CSR[ACTIVE] = 0, TCD*n*_CSR[DONE] = 1, INT[*n*] = 1.
16. The channel retires → major loop complete. The eDMA goes idle or services the next channel.

25.5.4.3 Modulo feature

The modulo feature of the eDMA provides the ability to implement a circular data queue in which the size of the queue is a power of 2. MOD is a 5-bit field for the source and destination in the TCD, and it specifies which lower address bits increment from their original value after the address + offset calculation. All upper address bits remain the same as in the original value. A setting of 0 for this field disables the modulo feature.

Table 378 shows how the transfer addresses are specified based on the setting of the MOD field. Here a circular buffer is created where the address wraps to the original value while the 28 upper address bits (0x1234567x) retain their original value. In this example the source address is set to 0x12345670, the offset is set to 4 bytes, and the MOD field is set to 4, allowing for a 16-byte size queue.

Table 378. Modulo feature example

Transfer number	Address
1	0x12345670
2	0x12345674
3	0x12345678
4	0x1234567C
5	0x12345670
6	0x12345674

25.5.5 eDMA TCD*n* status monitoring

This section discusses how to monitor eDMA status.

25.5.5.1 Minor loop complete

There are two methods to test for minor loop completion when using software initiated service requests. The first is to read the TCD*n*_CITER field and test for a change. Another method may be extracted from the sequence shown below. The second method is to test the TCD*n*_CSR[START] bit and the TCD*n*_CSR[ACTIVE] bit. The minor-loop-complete condition is indicated by both bits reading zero after the TCD*n*_CSR[START] was set. Polling the TCD*n*_CSR[ACTIVE] bit may be inconclusive, because the active status may be missed if the channel execution is short in duration.

The TCD status bits execute a sequence for a software activated channel as shown in *Table 379*.

Table 379. TCD status sequence – software activated channel

Step	TCD*n*_CSR bits			State
	START	ACTIVE	DONE	
1	1	0	0	Channel service request via software
2	0	1	0	Channel is executing
3a	0	0	0	Channel has completed the minor loop and is idle
3b	0	0	1	Channel has completed the major loop and is idle

The best method to test for minor-loop completion when using hardware (peripheral) initiated service requests is to read the TCD*n*_CITER field and test for a change. The hardware request and acknowledge handshake signals are not visible in the programmer's model.

The TCD status bits execute a sequence for a hardware-activated channel as shown in *Table 380*.

Table 380. TCD status sequence – hardware activated channel

Step	TCD*n*_CSR bits			State
	START	ACTIVE	DONE	
1	0	0	0	Channel service request via hardware (peripheral request asserted)
2	0	1	0	Channel is executing
3a	0	0	0	Channel has completed the minor loop and is idle
3b	0	0	1	Channel has completed the major loop and is idle

For both activation types, the major-loop-complete status is explicitly indicated via the TCD*n*_CSR[DONE] bit. The TCD*n*_CSR[START] bit is cleared automatically when the channel begins execution regardless of how the channel activates.

25.5.5.2 Active channel TCD*n* reads

The eDMA reads back the true TCD*n*_SADDR, TCD*n*_DADDR, and TCD*n*_NBYTES values if read while a channel executes. The true values of the SADDR, DADDR, and NBYTES are the values the eDMA engine currently uses in its internal register file and not the values in the TCD local memory for that channel. The addresses (SADDR and DADDR) and NBYTES (decrements to zero as the transfer progresses) can give an indication of the progress of the transfer. All other values are read back from the TCD local memory.

25.5.5.3 Preemption status

Preemption is available only when fixed arbitration is selected for both group and channel arbitration modes. A preemptive situation is one in which a preempt-enabled channel runs and a higher priority request becomes active. When the eDMA engine is not operating in Fixed-Group, Fixed-Channel Arbitration mode, the determination of the actively running relative priority outstanding requests become undefined. Channel and group priorities are treated as equal (constantly rotating) when Round-Robin Arbitration mode is selected.

The TCD*n*_CSR[ACTIVE] bit for the preempted channel remains asserted throughout the preemption. The preempted channel is temporarily suspended while the preempting channel executes one major loop iteration. If two TCD*n*_CSR[ACTIVE] bits are set simultaneously in the global TCD map, a higher priority channel is actively preempting a lower priority channel.

25.5.6 Channel linking

Channel linking (or chaining) is a mechanism where one channel sets the TCD*n*_CSR[START] bit of another channel (or itself), therefore initiating a service request for that channel. When properly enabled, the EDMA engine automatically performs this operation at the major or minor loop completion.

The minor loop channel linking occurs at the completion of the minor loop (or one iteration of the major loop). The TCD*n*_CITER[E_LINK] field determines whether a minor loop link is requested. When enabled, the channel link is made after each iteration of the major loop

except for the last. When the major loop is exhausted, only the major loop channel link fields are used to determine if a channel link should be made. For example, the initial fields of:

```
TCDn_CITER[E_LINK] = 1
TCDn_CITER[LINKCH] = 0xC
TCDn_CITER[CITER] value = 0x4
TCDn_CSR[MAJOR_E_LINK] = 1
TCDn_CSR[MAJOR_LINKCH] = 0x7
```

executes as:

1. Minor loop done → set TCD12_CSR[START] bit
2. Minor loop done → set TCD12_CSR[START] bit
3. Minor loop done → set TCD12_CSR[START] bit
4. Minor loop done, major loop done → set TCD7_CSR[START] bit

When minor loop linking is enabled (TCD*n*_CITER[E_LINK] = 1), the TCD*n*_CITER[CITER] field uses a 9-bit vector to form the current iteration count. When minor loop linking is disabled (TCD*n*_CITER[E_LINK] = 0), the TCD*n*_CITER[CITER] field uses a 15-bit vector to form the current iteration count. The bits associated with the TCD*n*_CITER[LINKCH] field are concatenated onto the CITER value to increase the range of the CITER.

Note: The TCDn_CITER[E_LINK] bit and the TCDn_BITER[E_LINK] bit must equal or a configuration error is reported. The CITER and BITER vector widths must be equal to calculate the major loop, half-way done interrupt point.

Table 381 summarizes how a DMA channel can link to another DMA channel (that is, how it can use another channel's TCD) at the end of a loop.

Table 381. Channel linking parameters

Desired link behavior	TCD control field name	Description
Link at end of minor loop	CITER[E_LINK]	Enable channel-to-channel linking on minor loop completion (current iteration)
	CITER[LINKCH] Link channel number when linking at end of minor loop (current iteration)	
Link at end of major loop	CSR[MAJOR_E_LINK]	Enable channel-to-channel linking on major loop completion
	CSR[MAJOR_LINKCH] Link channel number when linking at end of major loop	

25.5.7 Dynamic programming

25.5.7.1 The following sections describe dynamic programming.

25.5.7.2 Dynamic priority changing

The following two options are recommended for dynamically changing channel priority levels:

1. Switch to Round-Robin Channel Arbitration mode, change the channel priorities, then switch back to Fixed Arbitration mode.
2. Disable all the channels within a group, then change the channel priorities within that group only, then enable the appropriate channels.

The following two options are available for dynamically changing group priority levels:

1. Switch to Round-Robin Group Arbitration mode, change the group priorities, then switch back to Fixed Arbitration mode.
2. Disable all channels, change the group priorities, then enable the appropriate channels.

25.5.7.3 Dynamic channel linking

Dynamic channel linking is the process of setting the TCD.major.e_link bit during channel execution. This bit is read from the TCD local memory at the end of channel execution, thus allowing the user to enable the feature during channel execution.

Because the user is allowed to change the configuration during execution, a coherency model is needed. Consider the scenario where the user attempts to execute a dynamic channel link by enabling the TCD.major.e_link bit at the same time the eDMA engine is retiring the channel. The TCD.major.e_link would be set in the programmer's model, but it would be unclear whether the actual link was made before the channel retired.

The coherency model in *Table 382* is recommended when executing a dynamic channel link request.

Table 382. Coherency model for a dynamic channel link request

Step	Action
1	Write 1b to the TCD.major.e_link bit.
2	Read back the TCD.major.e_link bit.
3	Test the TCD.major.e_link request status: – If TCD.major.e_link = 1b, the dynamic link attempt was successful. – If TCD.major.e_link = 0b, the attempted dynamic link did not succeed (the channel was already retiring).

For this request, the TCD local memory controller forces the TCD.major.e_link bit to zero on any writes to a channel's TCD.word7 after that channel's TCD.done bit is set, indicating the major loop is complete.

Note: The user must clear the TCD.done bit before writing the TCD.major.e_link bit. The TCD.done bit is cleared automatically by the eDMA engine after a channel begins execution.

25.5.7.4 Dynamic scatter/gather

Dynamic scatter/gather is the process of setting the TCD.e_sg bit during channel execution. This bit is read from the TCD local memory at the end of channel execution, thus allowing the user to enable the feature during channel execution.

Because the user is allowed to change the configuration during execution, a coherency model is needed. Consider the scenario where the user attempts to execute a dynamic scatter/gather operation by enabling the TCD.e_sg bit at the same time the eDMA engine is

retiring the channel. The TCD.e_sg would be set in the programmer's model, but it would be unclear whether the actual scatter/gather request was honored before the channel retired.

Two methods for this coherency model are shown in the following subsections. Method 1 has the advantage of reading the major.linkch field and the e_sg bit with a single read. For both dynamic channel linking and scatter/gather requests, the TCD local memory controller forces the TCD.major.e_link and TCD.e_sg bits to zero on any writes to a channel's TCD.word7 if that channel's TCD.done bit is set indicating the major loop is complete.

Note: The user must clear the TCD.done bit before writing the TCD.major.e_link or TCD.e_sg bits. The TCD.done bit is cleared automatically by the eDMA engine after a channel begins execution.

25.5.7.4.1 Method 1 (channel not using major loop channel linking)

For a channel not using major loop channel linking, the coherency model in *Table 383* may be used for a dynamic scatter/gather request.

When the TCD.major.e_link bit is zero, the TCD.major.linkch field is not used by the eDMA. In this case, the TCD.major.linkch bits may be used for other purposes. This method uses the TCD.major.linkch field as a TCD identification (ID).

Table 383. Coherency model for method 1

Step	Action
1	When the descriptors are built, write a unique TCD ID in the TCD.major.linkch field for each TCD associated with a channel using dynamic scatter/gather.
2	Write 1b to theTCD.d_req bit. *Note: Should a dynamic scatter/gather attempt fail, setting the d_req bit will prevent a future hardware activation of this channel. This stops the channel from executing with a destination address (daddr) that was calculated using a scatter/gather address (written in the next step) instead of a dlast final offset value.*
3	Write theTCD.dlast_sga field with the scatter/gather address.
4	Write 1b to the TCD.e_sg bit.
5	Read back the 16 bit TCD control/status field.
6	Test the TCD.e_sg request status and TCD.major.linkch value: – If e_sg = 1b, the dynamic link attempt was successful. – If e_sg = 0b and the major.linkch (ID) did not change, the attempted dynamic link did not succeed (the channel was already retiring). – If e_sg = 0b and the major.linkch (ID) changed, the dynamic link attempt was successful (the new TCD's e_sg value cleared the e_sg bit).

25.5.7.4.2 Method 2 (channel using major loop linking)

For a channel using major loop channel linking, the coherency model in *Table 384* may be used for a dynamic scatter/gather request. This method uses the TCD.dlast_sga field as a TCD identification (ID).

Table 384. Coherency model for method 2

Step	Action
1	Write 1b to theTCD.d_req bit. **Note:** Should a dynamic scatter/gather attempt fail, setting the d_req bit will prevent a future hardware activation of this channel. This stops the channel from executing with a destination address (daddr) that was calculated using a scatter/gather address (written in the next step) instead of a dlast final offset value.
2	Write theTCD.dlast_sga field with the scatter/gather address.
3	Write 1b to the TCD.e_sg bit.
4	Read back the TCD.e_sg bit.
5	Test the TCD.e_sg request status: – If e_sg = 1b, the dynamic link attempt was successful. – If e_sg = 0b, read the 32 bit TCD dlast_sga field. – If e_sg = 0b and the dlast_sga did not change, the attempted dynamic link did not succeed (the channel was already retiring). – If e_sg = 0b and the dlast_sga changed, the dynamic link attempt was successful (the new TCD's e_sg value cleared the e_sg bit).

26 DMA channel multiplexer (DMACHMUX)

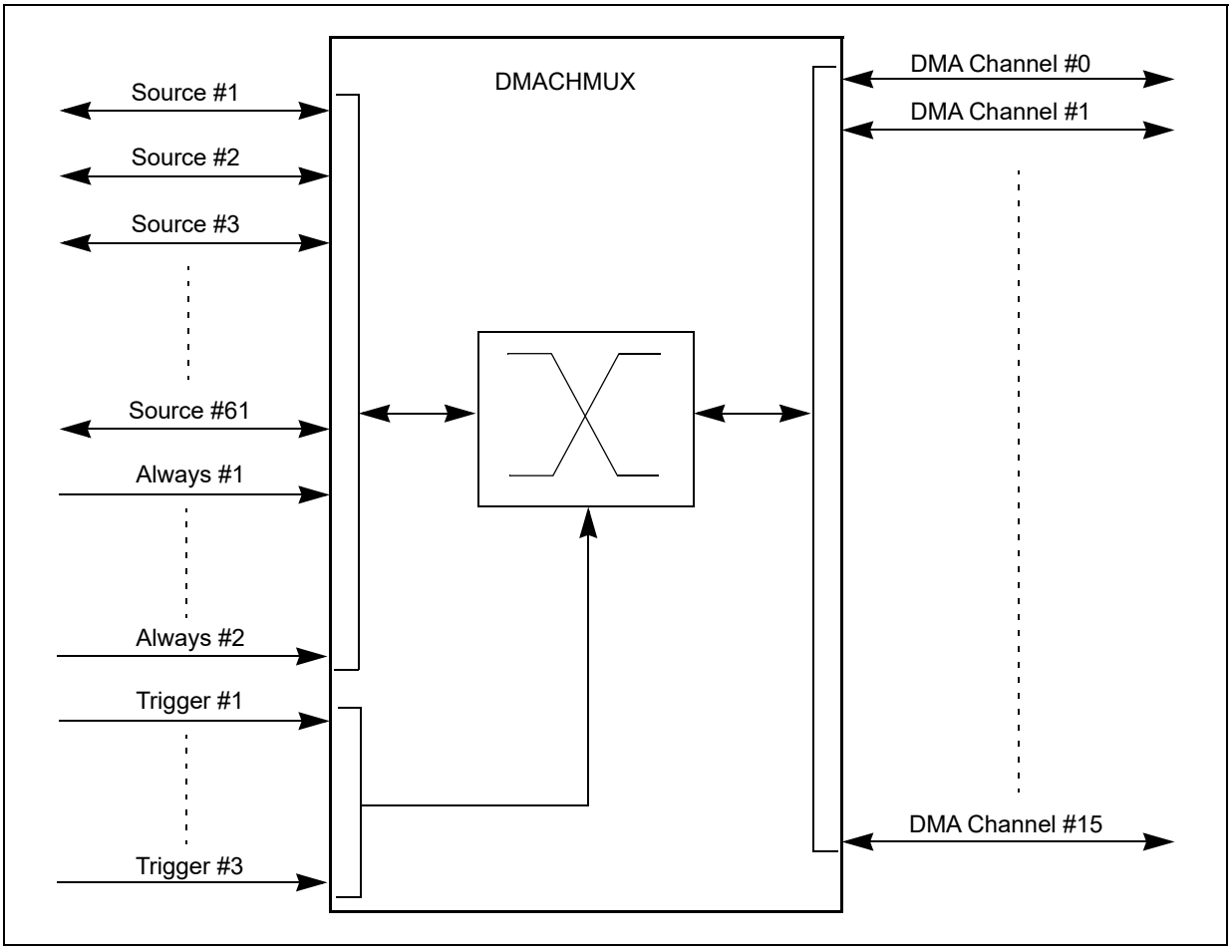
26.1 Introduction

Note: For the chip-specific implementation details of this module's instances, refer to the chip configuration information.

26.1.1 Overview

The DMA channel multiplexer (DMACHMUX) allows to route 61 DMA peripheral sources (called slots) to 16 DMA channels. This is illustrated in *Figure 275*.

Figure 275. DMACHMUX block diagram



26.1.2 Features

The DMACHMUX provides these features.

- 61 peripheral slots + 2 always-on slots can be routed to 16 channels.
- 16 independently selectable DMA channels routers.
  - The first 3 channels additionally provide a trigger functionality.
- Each channel router can be assigned to one of 61 possible peripheral DMA slots or to one of the 2 always-on slots.

26.1.3 Modes of operation

The following operation modes are available.

- Disabled Mode
  In this mode, the DMA channel is disabled. Since disabling and enabling of DMA channels is done primarily via the DMA configuration registers, this mode is used mainly as the reset state for a DMA channel in the DMACHMUX. It may also be used to temporarily suspend a DMA channel while reconfiguration of the system takes place (for example, changing the period of a DMA trigger).
- Normal Mode
  In this mode, a DMA source (such as DSPI transmit or DSPI receive for example) is routed directly to the specified DMA channel. The operation of the DMACHMUX in this mode is completely transparent to the system.
- Periodic Trigger Mode
  In this mode, a DMA source may only request a DMA transfer (such as when a transmit buffer becomes empty or a receive buffer becomes full) periodically. Configuration of the period is done in the registers of the Periodic Interrupt Timer (PIT). This mode is only available for channels 0–2 of each DMACHMUX instance.

26.2 External signal description

26.2.1 Overview

The DMACHMUX has no external pins.

26.3 Memory map and register definition

Table 385 lists the DMACHMUX memory-mapped registers. All addresses are offsets; the absolute address can be computed by adding the specified offset to the base address of the DMACHMUX.

Table 385. DMACHMUX memory map

Address offset	Register	Access	Location
eDMA channel registers per channel (*n* = 0 to 15)			
0x0000 + *n* * 0x1	Channel *n* Configuration registers (CHCFG*n*)	R/W	*Section 26.3.1.1*

All registers are accessible via 8-bit, 16-bit, or 32-bit accesses. However, 16-bit accesses must be aligned to 16-bit boundaries, and 32-bit accesses must be aligned to 32-bit boundaries. As an example, CHCFG0 through CHCFG3 are accessible by a 32-bit READ/WRITE to address 'Base + 0x00', but performing a 32-bit access to address 'Base + 0x01' is illegal.

26.3.1 Register descriptions

This section provides a detailed description of the memory-mapped registers in the DMACHMUX.

26.3.1.1 Channel *n* Configuration registers (CHCFG*n*)

Each of the DMA channels can be independently enabled/disabled and associated with one of the DMACHMUX slots (peripheral slots or always-on slots) in the system.

Address: 0x0000 + *n* * 0x1 (*n* = 0 to 15) Access: User read/write

	0	1	2	3	4	5	6	7
R W	ENBL	TRIG	SOURCE					
Reset	0	0	0	0	0	0	0	0

Figure 276. Channel Configuration registers (CHCFG*n*)

Table 386. CHCFG*n* field descriptions

Field	Description
0 ENBL	DMA Channel Enable ENBL enables the DMA Channel. 0 DMA channel is disabled. This mode is primarily used during configuration of the DMACHMUX. The DMA has separate channel enables/disables, which should be used to disable or reconfigure a DMA channel 1 DMA channel is enabled
1 TRIG	DMA Channel Trigger Enable (for triggered channels only) TRIG enables the periodic trigger capability for the DMA Channel. 0 Triggering is disabled. If triggering is disabled, and the ENBL bit is set, the DMA Channel will simply route the specified source to the DMA channel 1 Triggering is enabled
2:7 SOURCE	DMA Channel Source (DMACHMUX slot) SOURCE specifies which DMA source, if any, is routed to a particular DMA channel. Refer to the *Chapter 7: Device configuration* for further details about the peripherals and their slot numbers

Table 387. Channel and trigger enabling

ENBL	TRIG	Function	Mode
0	X	DMA Channel is disabled	Disabled Mode
1	0	DMA Channel is enabled with no triggering (transparent)	Normal Mode
1	1	DMA Channel is enabled with triggering	Periodic Trigger Mode

Note: Setting multiple CHCFG registers with the same source value will result in unpredictable behavior.

Before changing the trigger or source settings a DMA channel must be disabled via the CHCFG[n].ENBL bit.

26.4 Functional description

This section provides a functional description of the DMACHMUX. The primary purpose of the DMACHMUX is to provide flexibility in the system's use of the available DMA channels. As such, configuration of the DMACHMUX is intended to be a static procedure done during execution of the system boot code. However, if the procedure outlined in *Section 26.5.2: Enabling and configuring sources* is followed, the configuration of the DMACHMUX may be changed during the normal operation of the system.

Functionally, the DMACHMUX channels may be divided into two classes.

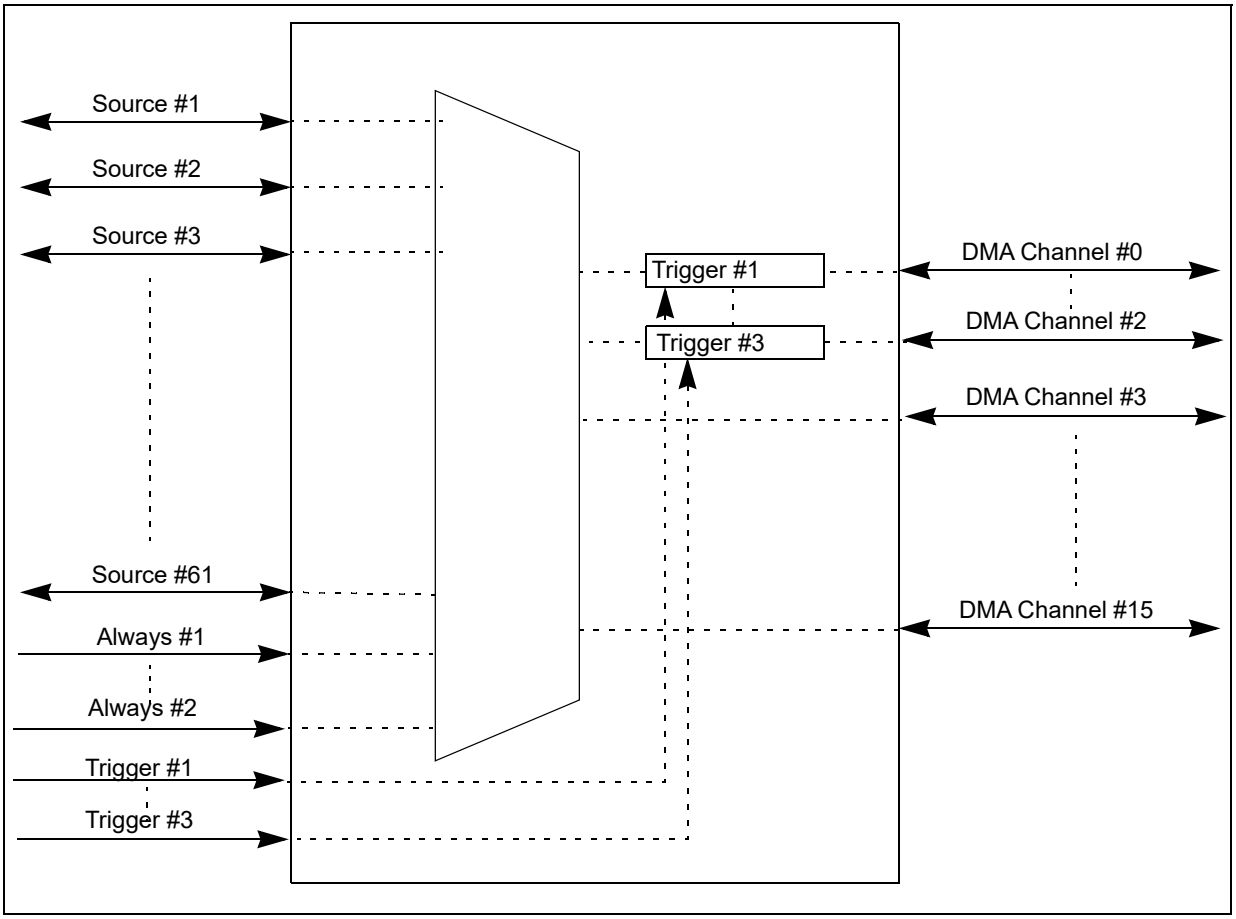
- Channels which implement the normal routing functionality plus periodic triggering capability.
- Channels which implement only the normal routing functionality.

26.4.1 DMA channels with periodic triggering capability

Besides the normal routing functionality, the first 3 channels of the DMACHMUX provide a special periodic triggering capability that can be used to provide an automatic mechanism to transmit bytes, frames or packets at fixed intervals without the need for processor intervention. The trigger is generated by the Periodic Interrupt Timer (PIT): such as the configuration of the periodic triggering interval is done via configuration registers in the PIT. Refer to *Chapter 50: Periodic Interrupt Timer (PIT)* for more information on this topic.

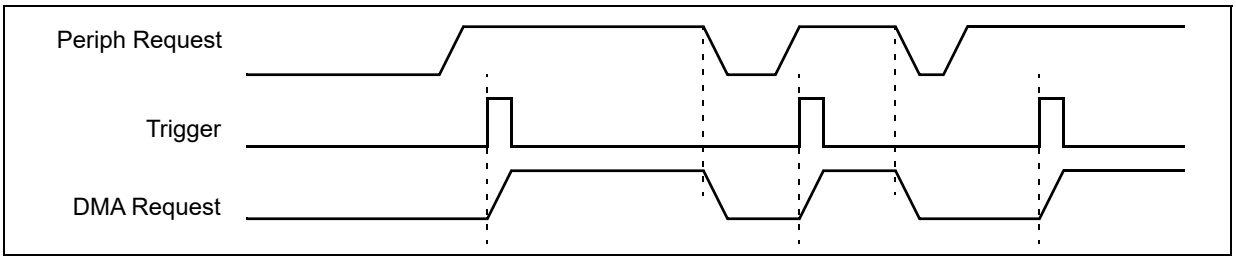
Note: Because of the dynamic nature of the system (such as DMA channel priorities, bus arbitration, interrupt service routine lengths), the number of clock cycles between a trigger and the actual DMA transfer cannot be guaranteed.

Figure 277. DMACHMUX triggered channels



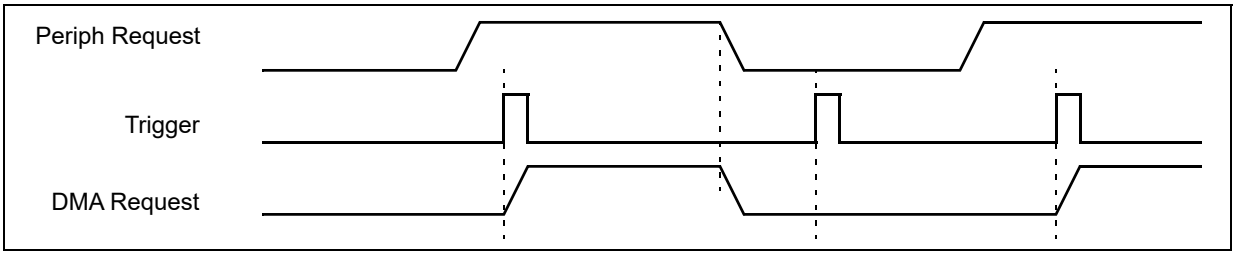
The DMA channel triggering capability allows the system to schedule regular DMA transfers, usually on the transmit side of certain peripherals, without the intervention of the processor. This trigger works by gating the request from the Peripheral to the DMA until a trigger event has been seen. This is illustrated in *Figure 278*.

Figure 278. DMACHMUX channel triggering: normal operation



Once the DMA request has been serviced, the peripheral will negate its request, effectively resetting the gating mechanism until the peripheral re-asserts its request AND the next trigger event is seen. This means that if a trigger is seen, but the peripheral is not requesting a transfer, that trigger will be ignored. This situation is illustrated in *Figure 279*.

Figure 279. DMACHMUX channel triggering: ignored trigger



This triggering capability may be used with any peripheral that supports DMA transfers, and is most useful for two types of situations.

- Periodically polling external devices on a particular bus. As an example, the transmit side of an SPI is assigned to a DMA channel with a trigger, as described above. Once setup, the SPI will request DMA transfers (presumably from memory) as long as its transmit buffer is empty. By using a trigger on this channel, the SPI transfers can be automatically performed every 5 µs (as an example). On the receive side of the SPI, the SPI and DMA can be configured to transfer receive data into memory, effectively implementing a method to periodically read data from external devices and transfer the results into memory without processor intervention.
- Using the GPIO Ports to drive or sample waveforms. By configuring the DMA to transfer data to one or more GPIO ports, it is possible to create complex waveforms using tabular data stored in on-chip memory. Conversely, using the DMA to periodically transfer data from one or more GPIO ports, it is possible to sample complex waveforms and store the results in tabular form in on-chip memory.

A more detailed description of the capability of each trigger (such as resolution, range of values) may be found in *Chapter 50: Periodic Interrupt Timer (PIT)* and in the *Chapter 56: Ethernet*.

26.4.2 DMA channels with no triggering capability

The other channels of the DMACHMUX provide the normal routing functionality as described in *Section 26.1.3: Modes of operation*.

26.4.3 “Always Enabled” DMA sources

In addition to the peripherals that can be used as DMA sources, there are 2 additional DMA sources that are “always enabled”. Unlike the peripheral DMA sources, where the peripheral controls the flow of data during DMA transfers, the *“always enabled”* sources provide no such “throttling” of the data transfers. These sources are most useful in the following cases:

- Doing DMA transfers to/from GPIO: Moving data from/to one or more GPIO pins, either unthrottled (that is, as fast as possible), or periodically (using the DMA triggering capability).
- Doing DMA transfers from memory to memory: Moving data from memory to memory, typically as fast as possible, sometimes with software activation.
- Doing DMA transfers from memory to the external bus (or vice-versa): Similar to memory to memory transfers, this is typically done as quickly as possible.
- Any DMA transfer that requires software activation: Any DMA transfer that should be explicitly started by software.

In cases where software should initiate the start of a DMA transfer, a *“always enabled”* DMA source can be used to provide maximum flexibility. When activating a DMA channel via

software, subsequent executions of the minor loop require a new "*start*" event be sent. This can either be a new software activation, or a transfer request from the DMACHMUX. The options for doing this are:

- Transfer all data in a single minor loop. By configuring the DMA to transfer all of the data in a single minor loop (that is major loop counter = 1), no reactivation of the channel is necessary. The disadvantage to this option is the reduced granularity in determining the load that the DMA transfer will incur on the system. For this option, the DMA channel should be disabled in the DMACHMUX.
- Use explicit software reactivation. In this option, the DMA is configured to transfer the data using both minor and major loops, but the processor is required to reactivate the channel (by writing to the DMA registers) *after every minor loop*. For this option, the DMA channel should be disabled in the DMACHMUX.
- Use a *"always enabled"* DMA source. In this option, the DMA is configured to transfer the data using both minor and major loops, and the DMACHMUX does the channel reactivation. For this option, the DMA channel should be enabled and pointing to an *"always enabled"* source. Note that the reactivation of the channel can be continuous (DMA triggering is disabled) or can use the DMA triggering capability. In this manner, it is possible to execute periodic transfers of packets of data from one source to another, without processor intervention.

26.5 Initialization/application information

26.5.1 Reset

The reset state of each individual bit is shown within the register description section (refer to *Section 26.3.1: Register descriptions*). In summary, after reset, all channels are disabled and must be explicitly enabled before use.

26.5.2 Enabling and configuring sources

To enable a source with periodic triggering:

1. Determine with which DMA channel the source will be associated. Note that only the first 2 DMA channels have periodic triggering capability.
2. Clear the CHCFG[ENBL] and CHCFG[TRIG] fields of the DMA channel.
3. Ensure that the DMA channel is properly configured in the DMA. The DMA channel may be enabled at this point.
4. Configure the corresponding timer.
5. Select the source to be routed to the DMA channel. Write to the corresponding CHCFG register, ensuring that the CHCFG[ENBL] and CHCFG[TRIG] fields are set.

Note: The following is an example. Refer to the chip configuration details for the number of this device's DMA channels that have triggering capability.

Example 1 Configure source #5 Transmit for use with DMA Channel 2, with periodic triggering capability:

1. Write 0x00 to CHCFG2 (Base Address + 0x02).
2. Configure Channel 2 in the DMA, including enabling the channel.
3. Configure a timer for the desired trigger interval.
4. Write 0xC5 to CHCFG2 (Base Address + 0x02).

The following code example illustrates steps #1 and #4 above:

In file **registers.h:**

```
#define DMAMUX_BASE_ADDR     0xFC084000/* Example only ! */
/* Following example assumes char is 8-bits */
volatile unsigned char *CHCFG0 = (volatile unsigned char *)
(DMAMUX_BASE_ADDR+0x0000);
volatile unsigned char *CHCFG1 = (volatile unsigned char *)
(DMAMUX_BASE_ADDR+0x0001);
volatile unsigned char *CHCFG2 = (volatile unsigned char *)
(DMAMUX_BASE_ADDR+0x0002);
volatile unsigned char *CHCFG3 = (volatile unsigned char *)
(DMAMUX_BASE_ADDR+0x0003);
volatile unsigned char *CHCFG4 = (volatile unsigned char *)
(DMAMUX_BASE_ADDR+0x0004);
volatile unsigned char *CHCFG5 = (volatile unsigned char *)
(DMAMUX_BASE_ADDR+0x0005);
volatile unsigned char *CHCFG6 = (volatile unsigned char *)
(DMAMUX_BASE_ADDR+0x0006);
volatile unsigned char *CHCFG7 = (volatile unsigned char *)
(DMAMUX_BASE_ADDR+0x0007);
volatile unsigned char *CHCFG8 = (volatile unsigned char *)
(DMAMUX_BASE_ADDR+0x0008);
volatile unsigned char *CHCFG9 = (volatile unsigned char *)
(DMAMUX_BASE_ADDR+0x0009);
volatile unsigned char *CHCFG10= (volatile unsigned char *)
(DMAMUX_BASE_ADDR+0x000A);
volatile unsigned char *CHCFG11= (volatile unsigned char *)
(DMAMUX_BASE_ADDR+0x000B);
volatile unsigned char *CHCFG12= (volatile unsigned char *)
(DMAMUX_BASE_ADDR+0x000C);
volatile unsigned char *CHCFG13= (volatile unsigned char *)
(DMAMUX_BASE_ADDR+0x000D);
volatile unsigned char *CHCFG14= (volatile unsigned char *)
(DMAMUX_BASE_ADDR+0x000E);
volatile unsigned char *CHCFG15= (volatile unsigned char *)
(DMAMUX_BASE_ADDR+0x000F);
```

In File **main.c:**

```
#include "registers.h"
   :
   :
*CHCFG2 = 0x00;
*CHCFG2 = 0xC5;
```

To enable a source without periodic triggering:

1. Determine with which DMA channel the source will be associated. Note that only the first 3 DMA channels have periodic triggering capability.
2. Clear the CHCFG[ENBL] and CHCFG[TRIG] fields of the DMA channel.
3. Ensure that the DMA channel is properly configured in the DMA. The DMA channel may be enabled at this point.
4. Select the source to be routed to the DMA channel. Write to the corresponding CHCFG register, ensuring that CHCFG[ENBL] is set while CHCFG[TRIG] is cleared.

Note: The following is an example. Refer to the chip configuration details for the number of this device's DMA channels that have triggering capability.

Example 2 Configure source #5 Transmit for use with DMA Channel 2, with no periodic triggering capability:

1. Write 0x00 to CHCFG2 (Base Address + 0x02).
2. Configure Channel 2 in the DMA, including enabling the channel.
3. Write 0x85 to CHCFG2 (Base Address + 0x02).

The following code example illustrates steps #1 and #3 above:

In File **registers.h:**

```
  #define DMAMUX_BASE_ADDR     0xFC084000/* Example only ! */
  /* Following example assumes char is 8-bits */
  volatile unsigned char *CHCFG0 = (volatile unsigned char *)
(DMAMUX_BASE_ADDR+0x0000);
  volatile unsigned char *CHCFG1 = (volatile unsigned char *)
(DMAMUX_BASE_ADDR+0x0001);
  volatile unsigned char *CHCFG2 = (volatile unsigned char *)
(DMAMUX_BASE_ADDR+0x0002);
  volatile unsigned char *CHCFG3 = (volatile unsigned char *)
(DMAMUX_BASE_ADDR+0x0003);
  volatile unsigned char *CHCFG4 = (volatile unsigned char *)
(DMAMUX_BASE_ADDR+0x0004);
  volatile unsigned char *CHCFG5 = (volatile unsigned char *)
(DMAMUX_BASE_ADDR+0x0005);
  volatile unsigned char *CHCFG6 = (volatile unsigned char *)
(DMAMUX_BASE_ADDR+0x0006);
  volatile unsigned char *CHCFG7 = (volatile unsigned char *)
(DMAMUX_BASE_ADDR+0x0007);
  volatile unsigned char *CHCFG8 = (volatile unsigned char *)
(DMAMUX_BASE_ADDR+0x0008);
  volatile unsigned char *CHCFG9 = (volatile unsigned char *)
(DMAMUX_BASE_ADDR+0x0009);
  volatile unsigned char *CHCFG10= (volatile unsigned char *)
(DMAMUX_BASE_ADDR+0x000A);
  volatile unsigned char *CHCFG11= (volatile unsigned char *)
(DMAMUX_BASE_ADDR+0x000B);
  volatile unsigned char *CHCFG12= (volatile unsigned char *)
(DMAMUX_BASE_ADDR+0x000C);
```

```
  volatile unsigned char *CHCFG13= (volatile unsigned char *)
(DMAMUX_BASE_ADDR+0x000D);
  volatile unsigned char *CHCFG14= (volatile unsigned char *)
(DMAMUX_BASE_ADDR+0x000E);
  volatile unsigned char *CHCFG15= (volatile unsigned char *)
(DMAMUX_BASE_ADDR+0x000F);
```

In File **main.c:**

```
  #include "registers.h"
     :
     :
  *CHCFG2 = 0x00;
  *CHCFG2 = 0x85;
```

Disabling a source:

A particular DMA source may be disabled by not writing the corresponding source value into any of the CHCFG registers. Additionally, some module specific configuration may be necessary. Refer to the appropriate section for more details.

Switching the source of a DMA Channel:

1. Disable the DMA channel in the DMA and reconfigure the channel for the new source.
2. Clear the ENBL and TRIG bits of the DMA channel.
3. Select the source to be routed to the DMA channel. Write to the corresponding CHCFG register, ensuring that the ENBL and TRIG bits are set.

Example 3 Switch DMA Channel 8 from source #5 transmit to source #7 transmit:

1. In the DMA configuration registers, disable DMA channel 8 and reconfigure it to handle the transfers to peripheral slot 7. This example assumes channel 8 does not have triggering capability.
2. Write 0x00 to CHCFG8 (Base Address + 0x08).
3. Write 0x87 to CHCFG8 (Base Address + 0x08). (In this example, setting the TRIG bit would have no effect, due to the assumption that channels 8 does not support the periodic triggering functionality).

The following code example illustrates steps #2 and #3 above.

In File **registers.h:**

```
  #define DMAMUX_BASE_ADDR     0xFC084000/* Example only ! */
  /* Following example assumes char is 8-bits */
  volatile unsigned char *CHCFG0 = (volatile unsigned char *)
(DMAMUX_BASE_ADDR+0x0000);
  volatile unsigned char *CHCFG1 = (volatile unsigned char *)
(DMAMUX_BASE_ADDR+0x0001);
  volatile unsigned char *CHCFG2 = (volatile unsigned char *)
(DMAMUX_BASE_ADDR+0x0002);
  volatile unsigned char *CHCFG3 = (volatile unsigned char *)
(DMAMUX_BASE_ADDR+0x0003);
  volatile unsigned char *CHCFG4 = (volatile unsigned char *)
(DMAMUX_BASE_ADDR+0x0004);
```

```
  volatile unsigned char *CHCFG5 = (volatile unsigned char *)
(DMAMUX_BASE_ADDR+0x0005);
  volatile unsigned char *CHCFG6 = (volatile unsigned char *)
(DMAMUX_BASE_ADDR+0x0006);
  volatile unsigned char *CHCFG7 = (volatile unsigned char *)
(DMAMUX_BASE_ADDR+0x0007);
  volatile unsigned char *CHCFG8 = (volatile unsigned char *)
(DMAMUX_BASE_ADDR+0x0008);
  volatile unsigned char *CHCFG9 = (volatile unsigned char *)
(DMAMUX_BASE_ADDR+0x0009);
  volatile unsigned char *CHCFG10= (volatile unsigned char *)
(DMAMUX_BASE_ADDR+0x000A);
  volatile unsigned char *CHCFG11= (volatile unsigned char *)
(DMAMUX_BASE_ADDR+0x000B);
  volatile unsigned char *CHCFG12= (volatile unsigned char *)
(DMAMUX_BASE_ADDR+0x000C);
  volatile unsigned char *CHCFG13= (volatile unsigned char *)
(DMAMUX_BASE_ADDR+0x000D);
  volatile unsigned char *CHCFG14= (volatile unsigned char *)
(DMAMUX_BASE_ADDR+0x000E);
  volatile unsigned char *CHCFG15= (volatile unsigned char *)
(DMAMUX_BASE_ADDR+0x000F);
```

In File **main.c:**

```
  #include "registers.h"
     :
     :
  *CHCFG8 = 0x00;
  *CHCFG8 = 0x87;
```

27 Clocking

27.1 Introduction

This chapter describes the architecture of the system level clocks and includes the following information:

- System clock specifics
- Clock architecture
- Clock sources
- Clock monitoring
- Programmable clock dividers
- Clock control registers
- Progressive Clock Switching (PCS)

Note: This chapter only covers clocks that are generated on the device. Protocol clocks that are provided by external devices are covered in the respective chapters where they are used.

The system clock (SYS_CLK) services the computational shell modules (include the main CPUs Main Core_0/1/2 and Checker Core_0s/2s), that run in the high-speed domain, memories and debug logic. Its independence from the peripheral clocks allows SYS_CLK to be a frequency modulated (FM) clock with reduced electromagnetic emissions while having a precise clock for the peripheral timer and communication functions. It can be reduced during low computational activity periods to lower power consumption while providing a fixed frequency to the communication interfaces and timers.

The internal 16 MHz RC oscillator (IRCOSC) services device boot up and may be enabled via the Mode Entry Module in the event of PLL or external oscillator failure. See the “Mode Entry Module (MC_ME)” chapter for details.

The alternative clock sources are:

- External oscillator/crystal (XOSC)
- Internal 16 MHz RC oscillator (IRCOSC)
- PLL0 (PLL0_PHI)
- PLL1 (PLL1_PHI)

Auxiliary Clock Selectors in the Clock Generation Module (MC_CGM) allow developers to select an independent clock source for each system peripheral. There is an additional System Clock Selector used exclusively for the system clock.

The outputs of the module clock selectors have individual dividers that can divide the clock selector outputs by up to 1024 for a given peripheral.

Clock Monitor Units (CMUs) are connected to outputs of clock dividers to verify that clock frequencies stay within operating limits and can be configured to signal an interrupt or initiate a system reset when an issue with a clock is found.

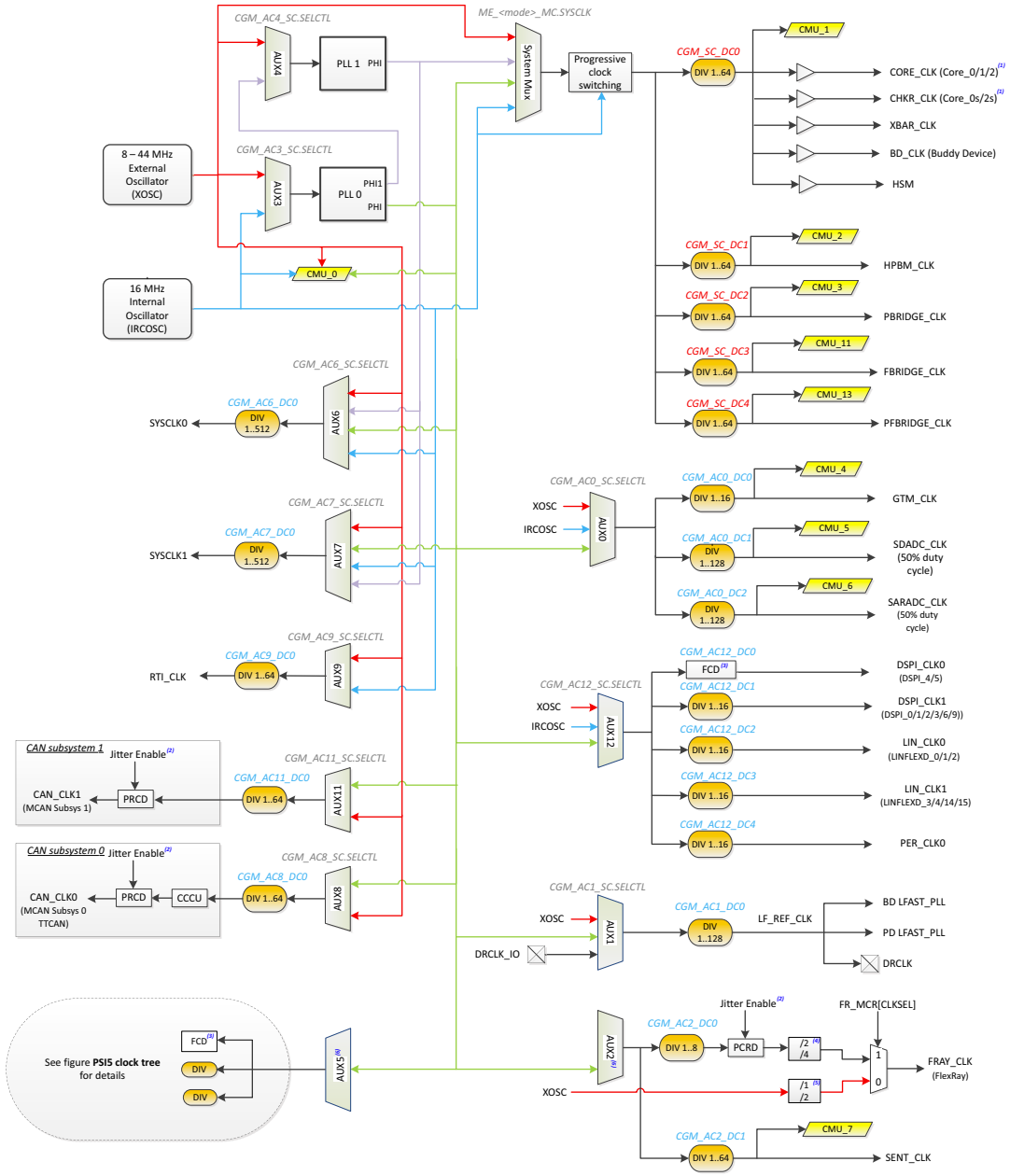
27.2 Clock generation

The top-level clock generation architecture can be seen in *Figure 280* and *Figure 282*. See *Chapter 30: Clock Generation Module (MC_CGM)* for details on the Fractional Clock Dividers. All clock selectors and dividers are located and programmed in the MC_CGM, with

the exception of the Hardware Security Module (HSM) clock divider. All dividers are integer dividers (1, 2, 3, …, *n*) with the ranges shown in the divider blocks in *Figure 280* and *Figure 281*. All dividers connected to the System Clock Selector must have 50 % duty cycle outputs.

Note: Clock Monitoring Units CMU1 to CMU13 in Figure 280 and *Figure 281 use IRCOSC as their reference clock. The user can select either IRCOSC or XOSC as the reference clock for CMU_0.*

Figure 280. Clock generation[1][2] (part 1 of 2)



All dividers shown in the diagram are integer dividers with a range of 1, 2, 3,...., n. FCD are fractional clock dividers. All clock dividers are 50% duty cycle.

Caption

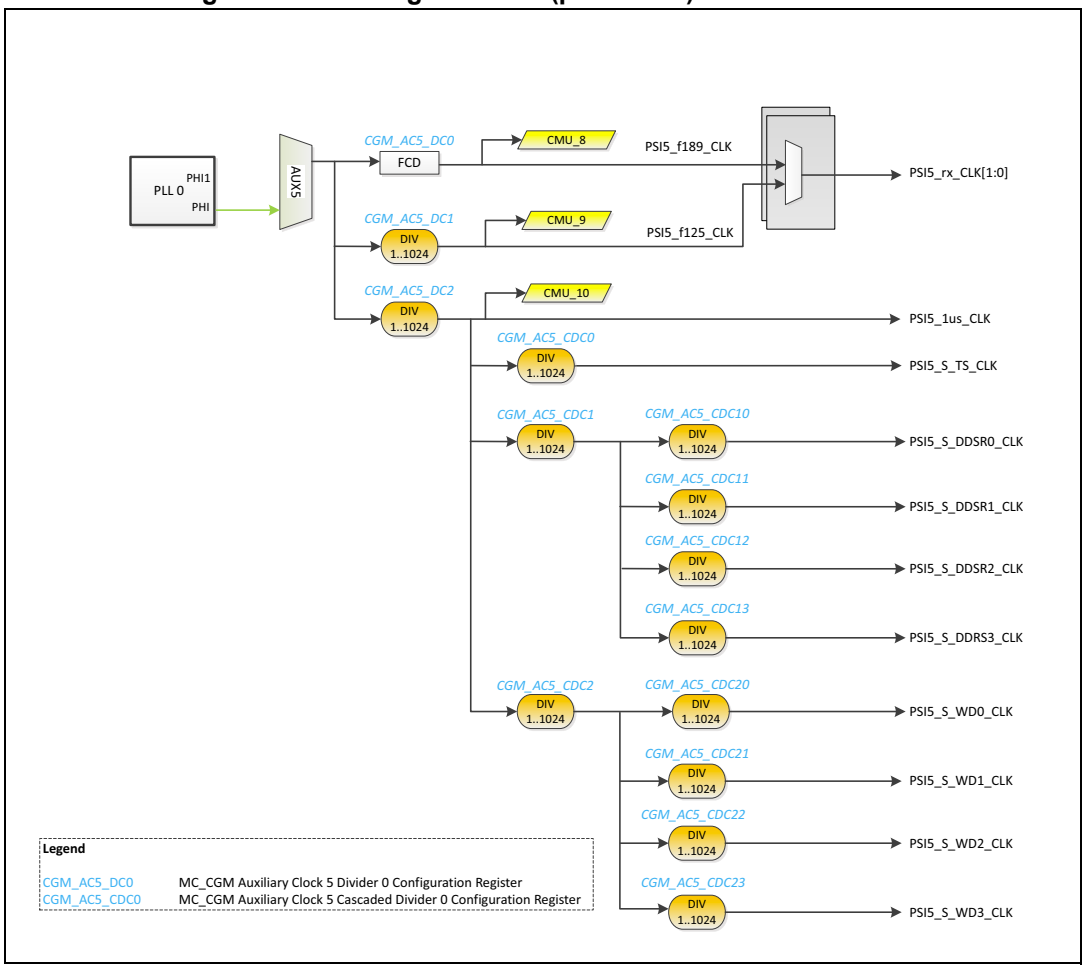
CGM_SC_DC0 MC_CGM System Clock Divider 0 Configuration Register
CGM_AC1_DC0 MC_CGM Auxiliary Clock 1 Divider 0 Configuration Register
CGM_AC1_SC MC_CGM Auxiliary Clock 1 Select Control Register

XOSC
IRCOSC
PLL0:PHI
PLL1:PHI

1. The CORE_CLK and CHKR_CLK clock trees are isolated downstream of System Clock Divider 0. Isolation is maintained in the physical design of the clock trees.
2. See chapter Password and Device Security Module (PASS) for jitter enable details.
3. See chapter Clock Generation Module (MC_CGM) for fractional divider details.
4. Divide by 2 in case FR_MCR[BITRATE] is set to 10/8/5 Mbit/s; divide by 4 in case FR_MCR[BITRATE] is set to 2.5 Mbit/s.
5. Divide by 1 in case FR_MCR[BITRATE] is set to 10/8/5 Mbit/s; divide by 2 in case FR_MCR[BITRATE] is set to 2.5 Mbit/s.
6. Since Aux2 and Aux5 have a single source(PLL0), SW has to ensure to disable PCTL of peripherals working on these two auxes, before turning off PLL0 in any mode of MC_ME.

1. See *Figure 284* for LFAST clocking details
2. See *Figure 283* for the clock distribution

Figure 281. Clock generation (part 2 of 2) - PSI5 clock tree



27.2.1 MC_CGM registers

Table 388 shows the relationship between the clocks in *Figure 280*, *Figure 281* and the MC_CGM registers.

Table 388. MC_CGM relationship to clocks

MC_CGM register(s)	Register description	Output clock	CMU(1)
CGM_SC_DC0	System Clock Divider Configuration 0	CHKR_CLK (Checker Core_0s/2s) CORE_CLK (Core_0/1/2) XBAR_CLK HSM_CLK BD_CLK (Buddy Device)	CMU_1_CORE_XBAR_BD
CGM_SC_DC1	System Clock Divider Configuration 1	HPBM_CLK (DMA0/1, SIPI0, INTC, GTM)	CMU_2_HPBM

Table 388. MC_CGM relationship to clocks (continued)

MC_CGM register(s)	Register description	Output clock	CMU[1]
CGM_SC_DC2	System Clock Divider Configuration 2	PBRIDGE_CLK (All peripheral bridges and peripherals not connected to any other system clock divider)	CMU_3_PBRIDGE
CGM_SC_DC3	System Clock Divider Configuration 3	FBRIDGE_CLK (SWT, STM, SEMA42)	CMU_11_FBRIDGE
CGM_SC_DC4	System Clock Divider Configuration 4	PFBRIDGE_CLK (CAN)	CMU_13_PFBRIDGE
CGM_AC0_DC0 CGM_AC0_SC	Aux Clock 0 Divider Configuration 0 Aux Clock 0 Select Control	GTM_CLK	CMU_4_GTM
CGM_AC0_DC1 CGM_AC0_SC	Aux Clock 0 Divider Configuration 1 Aux Clock 0 Select Control	SDADC_CLK	CMU_5_SDADC
CGM_AC0_DC2 CGM_AC0_SC	Aux Clock 0 Divider Configuration 2 Aux Clock 0 Select Control	SARADC_CLK	CMU_6_SARADC
CGM_AC0_DC3 CGM_AC0_SC	Aux Clock 0 Divider Configuration 3 Aux Clock 0 Select Control	EMIOS_CLK	CMU_12_EMIOS
CGM_AC1_DC0 CGM_AC1_SC	Aux Clock 1 Divider Configuration 0 Aux Clock 1 Select Control	LF_REF_CLK (BD LFAST PLL, PD LFAST PLL, DRCLK)	—
CGM_AC2_DC0	Aux Clock 2 Divider Configuration 0	FRAY_CLK	—
CGM_AC2_DC1	Aux Clock 2 Divider Configuration 1	SENT_CLK	CMU_7_SENT
CGM_AC3_SC	Aux Clock 3 Select Control	PLL0	—
CGM_AC4_SC	Aux Clock 4 Select Control	PLL1	—
CGM_AC5_DC0	Aux Clock 5 Divider Configuration 0	PSI5_f189_CLK	CMU_8_PSI5_F189
CGM_AC5_DC1	Aux Clock 5 Divider Configuration 1	PSI5_f125_CLK	CMU_9_PSI5_F125
CGM_AC5_DC2	Aux Clock 5 Divider Configuration 2	PSI5_1µs_CLK	CMU_10_PSI5_1us
CGM_AC5_CDC0	Aux Clock 5 Cascaded Divider Configuration 0	PSI5_S_TS_CLK	—
CGM_AC5_CDC1	Aux Clock 5 Cascaded Divider Configuration 1	PSI5_S_DDSR_CLK	—
CGM_AC5_CDC2	Aux Clock 5 Cascaded Divider Configuration 2	PSI5_S_WD_CLK	—
CGM_AC5_CDC10	Aux Clock 5 Cascaded Divider Configuration 10	PSI5_S_DDSR0_CLK	—
CGM_AC5_CDC11	Aux Clock 5 Cascaded Divider Configuration 11	PSI5_S_DDSR1_CLK	—
CGM_AC5_CDC12	Aux Clock 5 Cascaded Divider Configuration 12	PSI5_S_DDSR2_CLK	—
CGM_AC5_CDC13	Aux Clock 5 Cascaded Divider Configuration 13	PSI5_S_DDSR3_CLK	—

Table 388. MC_CGM relationship to clocks (continued)

MC_CGM register(s)	Register description	Output clock	CMU[1]
CGM_AC5_CDC20	Aux Clock 5 Cascaded Divider Configuration 20	PSI5_S_WD0_CLK	—
CGM_AC5_CDC21	Aux Clock 5 Cascaded Divider Configuration 21	PSI5_S_WD1_CLK	—
CGM_AC5_CDC22	Aux Clock 5 Cascaded Divider Configuration 22	PSI5_S_WD2_CLK	—
CGM_AC5_CDC23	Aux Clock 5 Cascaded Divider Configuration 23	PSI5_S_WD3_CLK	—
CGM_AC6_DC0 CGM_AC6_SC	Aux Clock 6 Divider Configuration 0 Aux Clock 6 Select Control	SYSCLK0	—
CGM_AC7_DC0 CGM_AC7_SC	Aux Clock 7 Divider Configuration 0 Aux Clock 7 Select Control	SYSCLK1	—
CGM_AC8_DC0 CGM_AC8_SC	Aux Clock 8 Divider Configuration 0 Aux Clock 8 Select Control	CAN_CLK0	—
CGM_AC9_DC0 CGM_AC9_SC	Aux Clock 9 Divider Configuration 0 Aux Clock 9 Select Control	RTI_CLK	—
CGM_AC11_DC0 CGM_AC11_SC	Aux Clock 11 Divider Configuration 0 Aux Clock 11 Select Control	CAN_CLK1	—
CGM_AC12_DC0 CGM_AC12_SC	Aux Clock 12 Divider Configuration 0 Aux Clock 12 Select Control	DSPI_CLK0 (DSPI4/5)	—
CGM_AC12_DC1 CGM_AC12_SC	Aux Clock 12 Divider Configuration 1 Aux Clock 12 Select Control	DSPI_CLK1 (DSPI0/1/2/3/6/7/8/9)	—
CGM_AC12_DC2 CGM_AC12_SC	Aux Clock 12 Divider Configuration 2 Aux Clock 12 Select Control	LIN_CLK0 (LIN0/1/2/16/17)	—
CGM_AC12_DC3 CGM_AC12_SC	Aux Clock 12 Divider Configuration 3 Aux Clock 12 Select Control	LIN_CLK1 (LIN3/4/14/15)	—
CGM_AC12_DC4 CGM_AC12_SC	Aux Clock 12 Divider Configuration 4 Aux Clock 12 Select Control	PER_CLK0	—
Note: See the Clock Generation Module (MC_CGM) for specifics on register implementation.			

1. CMU_0 is not shown in the table since it does not monitor an output clock from the MC_CGM.

27.3 System clock frequency limitations

The maximum frequency of operation for the device system level clocks is given in *Table 389*.

Table 389. Maximum system level clock frequencies

Clock Selector	Nominal programmed max output frequency for input to dividers (MHz)	SC_DC number	Divider logic	Divider setting when SYS_CLK is running on PLL clock (fast clock mode)	Divider setting when SYS_CLK is running on XOSC or IRCOSC (slow clock mode)
System clock selector(1)	180	0	CORE_CLK	SYS_CLK / 1	SYS_CLK / 1
System clock selector(1)	180	1	HPBM_CLK	SYS_CLK / 2	SYS_CLK / 1
System clock selector(1)	180	2	PBRIDGE_CLK	SYS_CLK / 4	SYS_CLK / 1
System clock selector(1)	180	3	FBRIDGE_CLK	SYS_CLK / 1	SYS_CLK / 1
System clock selector(1)	180	4	PFBRIDGE_CLK	SYS_CLK / 1 (2) or SYS_CLK / 2 or SYS_CLK / 4	SYS_CLK / 1
System clock selector(1)	180	—	HSM_CLK (3)	SYS_CLK / 2	

Note: The user is responsible to verify that these values are not exceeded.

Note: Refer to *Figure 280* and *Figure 281* for clocking architecture overview.

1. System clock selection is a component of the device run mode switching process. Refer to the Mode Entry Module (MC_ME) chapter for details.
2. PFBRIDGE_CLK max frequency supported is 160MHz. So SYS_CLK/1 is possible only if System clock frequency is less than or equal to 160MHz.
3. Maximum frequency is related to HSM core clock (HSM_CLK). Refer to *Figure 283* for details.

Table 390. Maximum auxiliary level clock frequencies

Clock Selector	Nominal programmed max output frequency for input to dividers	Divider	Divider logic	Nominal programmed divider max output frequency (MHz)
Aux Clock 0 Selector	400	0	GTM_CLK	100
Aux Clock 0 Selector	400	1	SDADC_CLK	16
Aux Clock 0 Selector	400	2	SARADC_CLK	16
Aux Clock 0 Selector	400	3	EMIOS_CLK	100
Aux Clock 1 Selector	400	0	BD LFAST PLL PD LFAST PLL DRCLK	26

Table 390. Maximum auxiliary level clock frequencies (continued)

Clock Selector	Nominal programmed max output frequency for input to dividers	Divider	Divider logic	Nominal programmed divider max output frequency (MHz)
Aux Clock 2 Selector	400	0	FRAY_CLK(1)	80
Aux Clock 2 Selector	400	1	SENT_CLK	100
Aux Clock 3 Selector	Refer to the Data Sheet	—	PLL0	—
Aux Clock 4 Selector	Refer to the Data Sheet	—	PLL1	—
Aux Clock 5 Selector	400	0	PSI5_f189_CLK (FCD output)	6.048
Aux Clock 5 Selector	400	1	PSI5_f125_CLK	4
Aux Clock 5 Selector	400	2	PSI5_1us_CLK	1
Aux Clock 5 Selector	400	CDC0	PSI5_S_TS_CLK	25
Aux Clock 5 Selector	400	CDC1	PSI5_S_DDSR_CLK	25
Aux Clock 5 Selector	400	CDC10	PSI5_S_DDSR0_CLK	25
Aux Clock 5 Selector	400	CDC11	PSI5_S_DDSR1_CLK	25
Aux Clock 5 Selector	400	CDC12	PSI5_S_DDSR2_CLK	25
Aux Clock 5 Selector	400	CDC13	PSI5_S_DDSR3_CLK	25
Aux Clock 5 Selector	400	CDC2	PSI5_S_WD_CLK	25
Aux Clock 5 Selector	400	CDC20	PSI5_S_WD_CLK0	25
Aux Clock 5 Selector	400	CDC21	PSI5_S_WD_CLK1	25
Aux Clock 5 Selector	400	CDC22	PSI5_S_WD_CLK2	25
Aux Clock 5 Selector	400	CDC23	PSI5_S_WD_CLK3	25
Aux Clock 6 Selector	400	0	SYSCLK0	Refer to the Data Sheet, section “I/O output DC characteristics”

Table 390. Maximum auxiliary level clock frequencies (continued)

Clock Selector	Nominal programmed max output frequency for input to dividers	Divider	Divider logic	Nominal programmed divider max output frequency (MHz)
Aux Clock 7 Selector	400	0	SYSCLK1	Refer to the Data Sheet, section “I/O output DC characteristics”
Aux Clock 8 Selector	400	0	CAN_CLK0	If CCCU is enabled: 400 If CCCU is bypassed: 80
Aux Clock 9 Selector	40	0	RTI_CLK	40
Aux Clock 11 Selector	400	0	CAN_CLK1	80
Aux Clock 12 Selector	400	0	DSPI_CLK0(2)	100
Aux Clock 12 Selector	400	1	DSPI_CLK1(2)	100
Aux Clock 12 Selector	400	2	LIN_CLK0	100
Aux Clock 12 Selector	400	3	LIN_CLK1	100
Aux Clock 12 Selector	400	4	PER_CLK0	100
Note: The user is responsible to verify that these values are not exceeded. **Note:** Refer to the *Figure 280* and *Figure 281* for clocking architecture overview.				

1. 80 MHz maximum frequency is at aux divider output. The maximum frequency of FRAY_CLK after the multiplexer is 40 MHz.
2. DSPI_CLK0/1 frequency should always be greater than or equal to 1.25 times of PBRIDGE_CLK.

27.3.1 JTAG frequencies

Table 391 shows the maximum frequencies of the JTAG interface.

Table 391. JTAG frequencies

Device	Configuration	e200z4 max. internal JTAG frequency	Buddy device max. RWA internal frequency	Max. external JTAG frequency	Notes
PD(1)	Unconfigured clock	—	N/A	Refer to electrical characteristics for maximum usable external JTAG frequency	Assumes 16 MHz clock on PD (IRC) with +/- 1.5% tolerance, and JTAG usable at 1/2 CPU frequency
	Clock configured to 200 MHz	100 MHz			JTAG usable at 1/2 CPU frequency
ED(2)	BD only powered	N/A	22.4 MHz		BD clocked by internal 64 MHz RC clock (RCOSC64) with +/- 30% tolerance and BD RWA usable at /2 frequency
	PD and BD powered, unconfigured clock	—	—		ED clocked by internal 8 MHz RC clock (IRC) with +/- 1.5% tolerance, and BD RWA usable at /2 frequency
	PD and BD powered, PD clock configured clock to 200 MHz	100 MHz	100 MHz		100 MHz JTAG achievable using LFAST mode debug, or s/w mode JTAGM. JTAG messages are generated internally by JTAGM and stay within BD and PD domain. The external tool interface is LFAST only

1. PD = Production Die
2. ED = Emulation and Debug Device = Production Die + Buddy Die

27.4 Default clock configuration

At power up, the IRCOSC is the default clock for the entire system. All system clock dividers are set to ‘1’ at power on reset, except for the HSM, which is set to its maximum value.

27.5 Clock sources

The following clock sources are described in this section:

- PLL0
- PLL1 (SSCG - Spread Spectrum Clock Generator (FM))
- External oscillator (XOSC)
- External Clock (EXTAL Bypass)
- 16 MHz internal RC oscillator (IRCOSC)

IRCOSC is also the system clock available during the STANDBY and it is placed in STANDBY power domain.

Additionally, there are two low power oscillators in STANDBY domain:

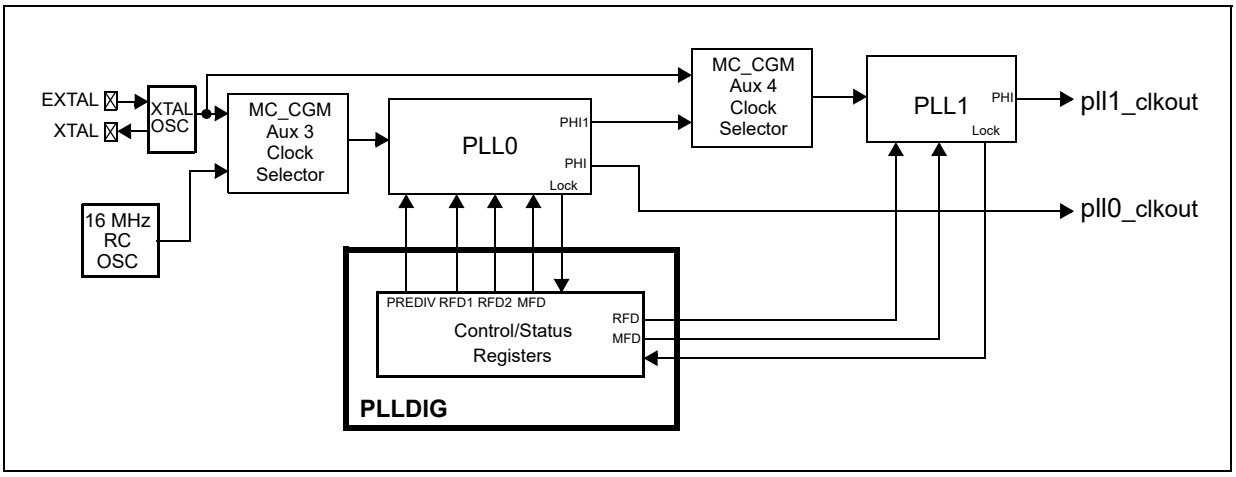
- Low power RC oscillator 1024 KHz (LPRC)
- Slow external crystal oscillator 32 KHz (SXOSC)

These two low power oscillators can be used as clock for counters of API/RTC.

27.5.1 PLL

The Dual PLL provides separate system and peripheral clocks as shown in *Figure 280*. The PLLs are disabled after power is cycled and must be enabled by software. The block diagram for the Dual PLL is shown in *Figure 282*.

Figure 282. SPC58xEx/SPC58xGx PLL digital interface block diagram



27.5.1.1 PLL0: base PLL (non-FM)

This primary PLL provides a clock that is not Frequency Modulated to the peripheral IPs and the reference clock to PLL1.

The input clock sources for PLL0 are:

- XOSC
- IRCOSC
- The external reference clock (input on the EXTAL pin)

The output clocks from PLL0 are:

- PHI: drives the peripheral clock and the system clock when PLL1 is not locked or active.
- PHI1: provides the input reference clock for PLL1.

27.5.1.2 PLL1: FMPLL

PLL1 is a Frequency Modulated PLL (FMPLL) that is used to drive the system clock.

The input clocks to PLL1 are:

- PHI1 output clock from PLL0
- The external reference clock.

The output clock from PLL1 is:

PHI: drives the system clock. The PHI output clock contains a fractional divider that can be applied to the loop divide of the PLL to achieve good granularity in the PLL1 PHI output clock frequency.

27.5.1.3 PLL register interface

This device has a single digital interface for the dual PLLs. The digital interface provides register control of all features of the PLLs. Details are provided in the “Dual PLL Digital Interface” chapter.

27.5.1.4 PLL register write protection

The PLL registers defined in *Table 392* have write protection with:

- Soft locking: can be unlocked by software after being previously locked.
- Hard locking: can only be unlocked by a reset once locked.

The REG_PROT module is used to implement the PLL register write protection (refer to *Section 7.10.5: Register protection (REG_PROT) configuration* in *Chapter 7: Device configuration* for details).

Table 392. PLL register write protection

Offset	Register
0x00	PLL0 Control Register (PLL0CR)
0x04	PLL0 Status Register (PLL0SR)
0x08	PLL0 Divider Register (PLL0DIVR)
0x20	PLL1 Control Register (PLL1CR)
0x24	PLL1 Status Register (PLL1SR)
0x28	PLL1 Divider Register (PLL1DIVR)
0x2C	PLL1 Frequency Modulation Register (PLL1FMR)
0x30	PLL1 Fractional Divider Register (PLL1FDR)
Note: Refer to Register Protection Configuration chapter for bit field details.	

27.5.1.5 PLL register reset values

All reset values for the PLLDIG registers that are device specific are shown in *Table 393*. All other register reset values are shown in the Dual PLL Digital Interface chapter.

Table 393. Dual PLL Digital Interface register reset values

Offset (hex)	Register	Reset value (hex)
0x0008	PLL0 Divider Register (PLL0DV)	0x0000_0000
0x0028	PLL1 Divider Register (PLL1DV)	0x0000_0000
0x0034	Reserved	—
0x0038	Reserved	—

27.5.1.6 Loss of clock detection

Loss of clock detection should be enabled at all times when the PLLs are enabled. There is a software option to generate a reset or interrupt in the event of a loss of clock condition.

27.5.1.7 PLLDIG initialization information

Configure PLL0 and related modules.

1. With PLL0 disabled, program PLL0 clock source in MC_CGM_AC3_SC[SELCTL]. Default source is 16 MHz IRCOSC. Write MC_CGM_AC3_SC[SELCTL] = 1 to select XOSC as source.
2. Program PLL0DV[PREDIV], PLL0DV[RFDPHI1], PLL0DV[MFD] and PLL0DV[RFDPHI].
3. If desired, modify the XOSC_CTL[EOCV] to adjust the external oscillator stabilization count, used when the external oscillator is turned on (by the MC_ME).
4. If necessary, modify the CMU frequency meter values (CMU_MDR and CMU_FDR) used to monitor the XOSC frequency. XOSC frequency is compared to the IRCOSC source when XOSC is turned on.

Turn on XOSC and PLL0.

5. Configure a mode configuration for turning on PLL0 and XOSC. In a mode configuration register (for example, MC_ME_DRUN_MC) set MC_ME_<mode>_MC[XOSCON] = 1 and MC_ME_<mode>_MC[PLL0ON] = 1. If desired, also set MC_ME_<mode>_MC[SYSCLK] = 2 for this new mode configuration to use PLL0(PLL0 PHI output) as the sysclk.
6. Enter that mode by two writes to MC_ME_CTRL register to enter that mode. This is required even if entering the same mode.

Wait for the mode transition to complete.

7. Wait for the mode transition to complete by polling MC_ME_GS[S_MTRANS] or enabling an interrupt for flag MC_ME_IS[I_MTC]. A timer, even if it is the watchdog, should be used to make sure the mode transition completes. Mode transition will NOT complete until:

   XOSC counter expires (if the new mode configuration changes MC_ME_<mode>_MC[XOSCON] = 1), and

   PLL is locked (if the new mode configuration changes MC_ME_<mode>_MC[PLL0ON] = 1)
8. Confirm the desired target mode was entered by checking the status of MC_ME_GS[S_CURRENT_MODE].

Configure PLL1 and related modules.

9. With PLL1 disabled, program PLL1 clock source in MC_CGM_AC4_SC[SELCTL]. Default is MC_CGM_AC4_SC[SELCTL] = 1 which selects XOSC.
10. Program PLL1DV[MFD] and PLL1DV[RFDPHI].
11. If required, program the PLL1FM register with the desired frequency modulation parameters and enable FM modulation, PLL1FM[MODEN] = 1. The PLL1FM register must not be written after PLL1 is enabled and operating in normal mode.

Turn on PLL1.

12. Configure a mode configuration for turning on PLL1, and keeping XOSC and PLL1 on. In a mode configuration register (for example, MC_ME_DRUN_MC), initialize MC_ME_<mode>_MC[XOSCON] = 1, MC_ME_<mode>_MC[PLL1ON] = 1 and MC_ME_<mode>_MC[PLL1ON] = 1. If desired, also set

MC_ME_<mode>_MC[SYSCLK] = 4 for this new mode configuration to use PLL1 as the sysclk.

13. Enter that mode by two writes to ME_CTRL register to enter that mode. This is required even if entering the same mode.

Wait for the mode transition to complete.

14. Wait for the mode transition to complete by polling MC_ME_GS[S_TRANS] or enabling an interrupt for flag MC_ME_IS[I_MTC]. A timer, even if it is the watchdog, should be used to make sure the mode transition completes. Mode transition will NOT complete until:

    XOSC counter expires (if the new mode configuration changes XOSCON to 1), and

    PLL1 is locked (if the new mode configuration changes MC_ME_<mode>_MC[PLL1ON] = 1).

15. Confirm the desired target mode was entered by checking the status of MC_ME_GS[S_CURRENT_MODE].

Note: Refer to the "Mode Entry Module (MC_ME)" chapter in this Reference Manual for more details.

27.5.2 External oscillator (XOSC)

The external oscillator (XOSC):

- Is a reference for the on-chip PLLs.
- Can be used as a clock source for the ADCs, timer, serial interfaces, RTI, LFAST and as a system clock source.
- Is available for observation on either of the SYSCLK0/1 pins.
- Is used as a reference to calibrate the IRCOSC frequency.

The external oscillator allows a crystal or external clock to be used as the reference clock for the microcontroller. The XOSC has the following features:

- Internal load capacitors for 20 MHz/40 MHz crystal oscillators
- Automatic Loop Control (ALC) to remove need for external series resistor
- Gain selection for <= 20 MHz and 40 MHz crystals to allow for optimal startup margin without overdrive issues
- Reference clock to PLL0
- Reference clock to CMU0 (IRCOSC trimming)
- Option to drive the CAN and FlexRay protocol clocks directly from the XOSC

The XOSC provides support for 4 MHz - 40 MHz crystal inputs, has integrated load capacitors and ALC to remove the need for external series resistor

27.5.2.1 XOSC Start-up

Enabling of the oscillator at startup is determined by a bit of Miscellaneous DCF in UTEST Flash memory (refer to *Table 129: DCF client list* in *Chapter 9: Device Configuration Format (DCF) Records*). The default value has the XOSC disabled:

UTEST Miscellaneous[24] = 0 (XOSC_EN).

If enabled, the oscillator is started in PHASE 3 of the reset sequence: after negation of the internal POR circuits but while the external PORST input is asserted. This is done to allow fast application start-up time while accounting for settling time of the power regulators.

The selection of internal or external load capacitors on the XTAL/EXTAL pins is also determined by a bit in the UTEST row of flash memory. The default value is external load capacitors:

UTEST Miscellaneous[25] = 1 (XOSC_EXT_LOAD).

Load capacitor values are determined by the crystal manufacturer data sheet requirements, while accounting for stray PCB and on-chip capacitance. Refer to the device data sheet for on-chip capacitance values.

When using the internal load capacitors for the oscillator, the startup value of the capacitors is stored in the UTEST row of the flash memory. During PHASE 3 of the device reset sequence, the oscillator UTEST values are read and driven to the oscillator. The enable/disable state of the oscillator is captured in the MC_ME module at this time. After reset, the oscillator can be enabled/disabled by software in the MC_ME module. Internal load capacitor selection is maintained by the oscillator if disabled after reset. The default reset value for the XOSC trimming capacitors does not trim the internal capacitor values:

UTEST Miscellaneous[26:30] = 00000b (XOSC_LOAD_CAP_SEL).

The internal load capacitor values stored in UTEST flash memory row have triple-voting flip-flop (TVF) implementation to prevent an incorrect value causing an undetectable error in the system.

The XOSC has the ability to be started with 4 MHz - 40 MHz crystal. The default value of this field is for an 35 MHz - 40 MHz:

UTEST Miscellaneous[19:21] = 111b (XOSC_FREQ_SEL).

Refer to the FLASH memory chapter for more information on the UTEST row programming.

27.5.2.2 XOSC register write protection

The XOSC registers defined in *Table 394* have write protection with:

- Soft locking: can be unlocked by software after being previously locked.
- Hard locking: can only be unlocked by a reset once locked.

The REG_PROT module is used to implement the XOSC register write protection.

Table 394. XOSC register write protection

Offset	Register	Protections
0x00	XOSC Control Register (XOSC_CTL)	Yes
Note: refer to “Register Protection Configuration” chapter for bit field details		

27.5.2.3 XOSC reset value

Table 395 shows the default reset value for the XOSC registers.

Table 395. XOSC register reset values

Offset	Register	Reset value
0x00	XOSC Control Register (XOSC_CTL)	0x8030_8000

27.5.3 16 MHz internal RC oscillator (IRCOSC)

The 16 MHz internal RC oscillator default clock is always enabled on reset and can be the clock source for the PLLs. The register interface is for user trimming of the oscillator and dividing output from 1 to 32.

27.5.3.1 IRCOSC register interface

A dedicated digital interface for the IRCOSC has a user register for fine tuning the IRCOSC frequency. Other read-only registers contain settings and values for the IRCOSC temperature sensor, voltage regulator and capacitor trimming.

27.5.3.2 IRCOSC reset value

Table 396 shows the default reset value for the IRCOSC registers.

Table 396. IRCOSC register reset values

Offset	Register	Reset value
0x04	Native Trimming Register (NT)	0x0000_0000
0x08	Temperature Trimming Register (TT)	0x0000_0000

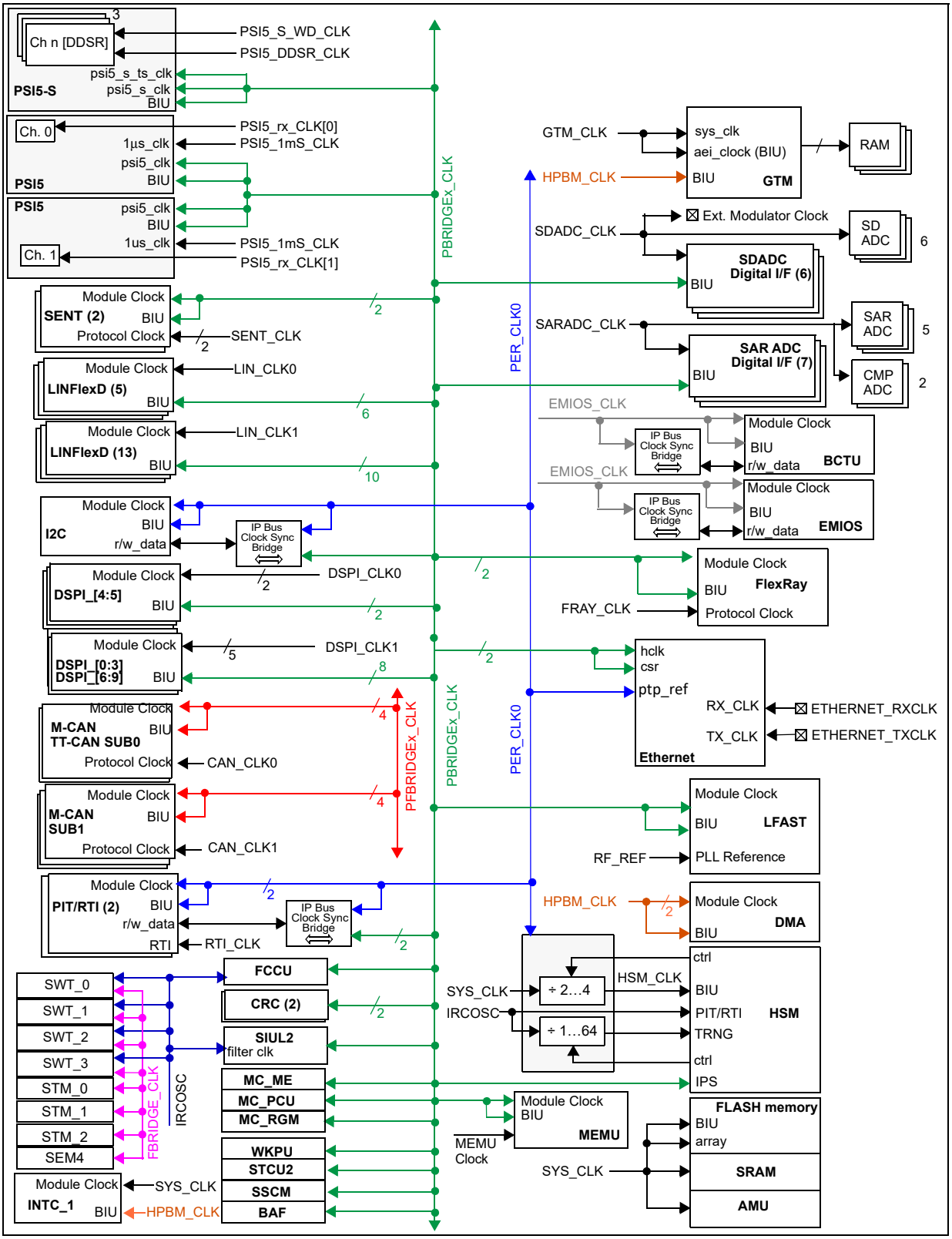
27.5.3.3 IRCOSC register write protection

The IRCOSC registers do not utilize write protection.

27.6 Peripheral clocks

Figure 283 shows the clock distribution to the cores and peripheral modules.

Figure 283. Clock distribution



Note: The frequency of the MEMU clock is equal to the frequency of the SYSTEM clock.

27.6.1 HSM clock divider

The clock dividers for the HSM core and random number generator (TRNG) are located outside the HSM, but not in the MC_CGM. These dividers are controlled by the HSM. The minimum possible divider after reset is ‘divide by 2’.

The HSM contains an internal PIT/RTI block, which is clocked by the IRCOSC.

The HSM has an internal clock monitor on its input clock, so there is no need for a Clock Monitoring Unit (CMU) for the HSM_CLK in *Figure 280*.

27.6.2 PSI5 clock dividers

The SPC58xEx/SPC58xGx microcontroller includes two PSI5 modules, all of which use the three common clocks shown in *Figure 280*. The required frequencies for these clocks are generated from PLL0 as follows:

- $F_{PSI5_f189_CLK} = F_{PLL0_PHI}$ / ((CGM_AC5_DC0[DIV] + 1) / 10CGM_AC5_DC0[FMT]) = 6.048 MHz ±0.1%
- $F_{PSI5_f125_CLK} = F_{PLL0_PHI}$ / (CGM_AC5_DC1[DIV] + 1) = 4 MHz
- $F_{PSI5_1us_CLK} = F_{PLL0_PHI}$ / (CGM_AC5_DC2[DIV] + 1) = 1 MHz

The nine PSI5-S clocks are generated by dividing the PSI5_1µs_CLK further via a second and a third level of 'cascaded' clock dividers. The individual frequencies are, therefore, derived from the PLL0 PHI frequency as follows:

- $F_{PSI5_S_TS_CLK} = F_{PLL0_PHI}$ / ((CGM_AC5_DC2[DIV] +1) * (CGM_AC5_CDC0[DIV]+1))
- $F_{PSI5_S_DDSR0_CLK} = F_{PLL0_PHI}$ / ((CGM_AC5_DC2[DIV] +1) * (CGM_AC5_CDC1[DIV]+1) * (CGM_AC5_CDC10[DIV]+1))
- $F_{PSI5_S_DDSR1_CLK} = F_{PLL0_PHI}$ / ((CGM_AC5_DC2[DIV] +1) * (CGM_AC5_CDC1[DIV]+1) * (CGM_AC5_CDC11[DIV]+1))
- $F_{PSI5_S_DDSR2_CLK} = F_{PLL0_PHI}$ / ((CGM_AC5_DC2[DIV] +1) * (CGM_AC5_CDC1[DIV]+1) * (CGM_AC5_CDC12[DIV]+1))
- $F_{PSI5_S_DDSR3_CLK} = F_{PLL0_PHI}$ / ((CGM_AC5_DC2[DIV] +1) * (CGM_AC5_CDC1[DIV]+1) * (CGM_AC5_CDC13[DIV]+1))
- $F_{PSI5_S_WD0_CLK} = F_{PLL0_PHI}$ / ((CGM_AC5_DC2[DIV] +1) * (CGM_AC5_CDC2[DIV]+1) * (CGM_AC5_CDC20[DIV]+1))
- $F_{PSI5_S_WD1_CLK} = F_{PLL0_PHI}$ / ((CGM_AC5_DC2[DIV] +1) * (CGM_AC5_CDC2[DIV]+1) * (CGM_AC5_CDC21[DIV]+1))
- $F_{PSI5_S_WD2_CLK} = F_{PLL0_PHI}$ / ((CGM_AC5_DC2[DIV] +1) * (CGM_AC5_CDC2[DIV]+1) * (CGM_AC5_CDC22[DIV]+1))
- $F_{PSI5_S_WD3_CLK} = F_{PLL0_PHI}$ / ((CGM_AC5_DC2[DIV] +1) * (CGM_AC5_CDC2[DIV]+1) * (CGM_AC5_CDC23[DIV]+1))

27.6.3 LFAST clocking

The SPC58xEx/SPC58xGx includes two LFAST modules:

- In conjunction with the SIPI module for interprocessor communications
- In conjunction with the JTAGM module for high speed debug

High-speed operation of both LFAST modules is supported by a single LFAST PLL requiring a 10–20 MHz reference. Low-speed LFAST operation uses the reference clock directly.

The LFAST PLL reference clock source may be:

- PLL0:PHI output
- External oscillator (XOSC)
- Input from the external LFAST device via the DRCLK_IO package pin

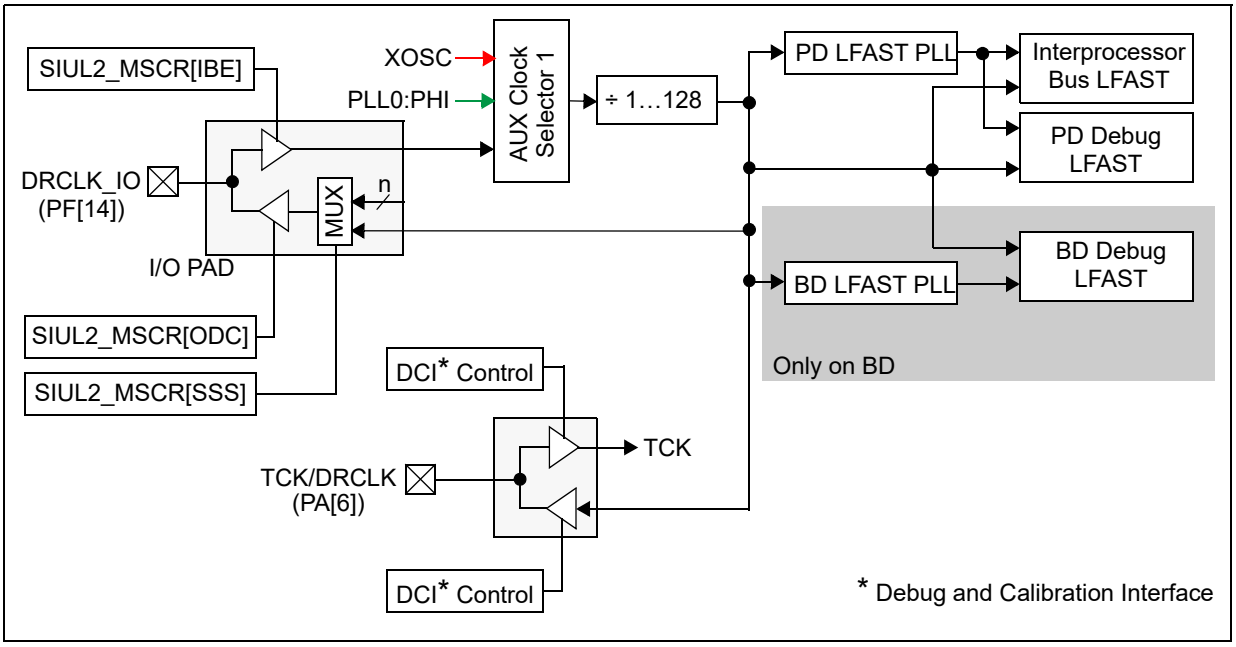
When the reference clock is generated internally, it can be output on the DRCLK_IO pin

The debug LFAST module only supports the operation mode where it outputs the reference clock to a connected LFAST capable tool as the DRCLK pin only supports output of the DRCLK reference.

On Emulation and Debug Devices (ED), the Buddy Device (BD) LFAST module connects to the package debug pins in place of the Production Device (PD) debug LFAST module. This module has a separate, dedicated LFAST PLL and the DRCLK pin only supports output of the reference clock to a connected LFAST capable tool (as with the Production Device).

Figure 284 shows the clock routing for the various LFAST modules on the device, including connections to the DRCLK_IO and DRCLK pins.

Figure 284. Device LFAST clocking

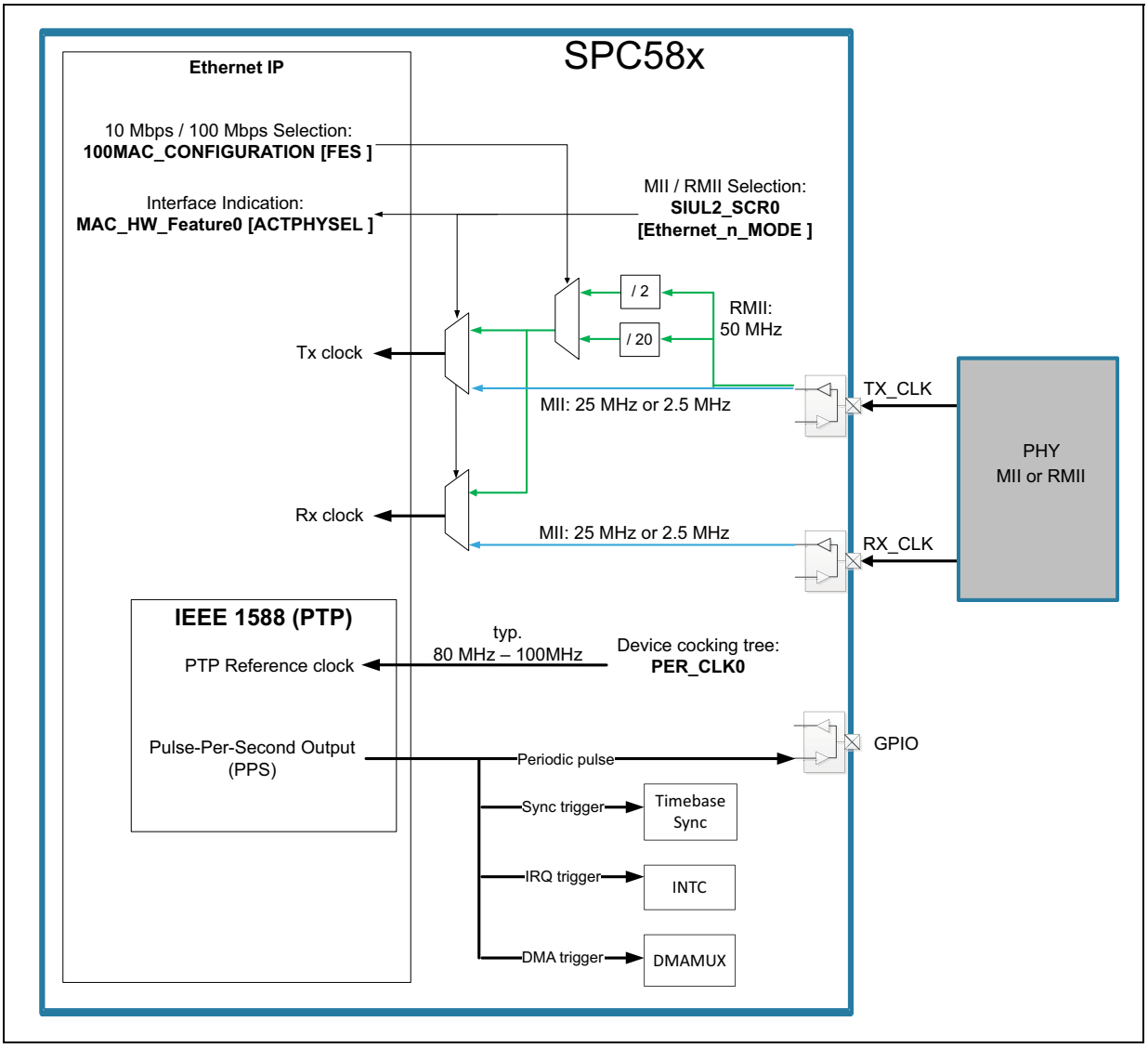


27.6.4 Ethernet clocking

The Ethernet module requires external protocol clock sources for MII and RMII interface modes. The MII clock is 25 MHz in 100 Mbps mode and 2.5 MHz in 10 Mbps mode. The RMII clock is 50 MHz in both modes.

The connections for the Ethernet clocks are as described below.

Figure 285. Ethernet clocking



The RX_CLK pin is an input used in MII mode only and ignored in RMII mode. The TX_CLK pin is an input used in MII mode and RMII mode.

27.6.4.1 Reference Clock for PTP Timestamp Logic

The Ethernet Module supports the Precision Time Protocol (PTP) as defined in IEEE 1588. The PTP Reference clock is the clock at which the PTP system time gets updated. It is connected to PER_CLK0 clock tree provided by the system clock generation module.

Based on the PTP a configurable Pulse-Per-Second signal can be generated as trigger source for several other modules on the device or as periodic pulse as alternate output on a GPIO (refer to *Figure 285*).

27.6.5 FlexRay clocking

The FlexRay protocol clock is sourced from the PLL0_PHI clock or the external oscillator (XOSC), but its source selection is different from other modules:

1. PLL0_PHI enters the MC_CGM where it is divided by a value configured in the CGM_AC2_DC0 register and output as FRAY_PLL_CLK.
2. FRAY_PLL_CLK passes through the Pseudo Random Clock Divider (PRCD) which:
   Divides FRAY_PLL_CLK by 16.
   Adds approximately 40% of random jitter to both edges of the clock.
3. An internal multiplexer selects the PRCD output or XOSC as the resultant FRAY_CLK.

PLL0_PHI is the source whenever the PRCD is enabled. PRCD is passive in normal operation and has no impact on the clock frequency.

When XOSC is the input clock to the FlexRay controller, a 40 MHz crystal is required for proper operation. Both edges of the clock are used by the module to achieve the required 80 MHz frequency.

Selection of PLL or XOSC as the FlexRay clock is shown in *Chapter 30: Clock Generation Module (MC_CGM)*.

27.6.6 M_TTCAN/M_CAN clocking

27.6.6.1 M_CAN/M_TTCAN subsystem 0

The CAN_CLK0 is configured in the CGM via the following registers:

- CGM_AC8_SC selects the XOSC or PLL0:PHI as the clock source.
- CGM_AC8_DC0 reduces the CAN_CLK by configuring the dividing factor from 1 to 64.

The CAN_0 clock passes through the Pseudo Random Clock Divider (PRCD) which:

- Divides CAN_CLK by 16.
- Adds approximately 40% of random jitter to both edges of the clock.

PLL0.PHI is the source whenever the PRCD is enabled. PRCD is passive in normal operation and has no impact on the clock frequency.

27.6.6.2 M_CAN subsystem 1

The CAN_CLK1 is configured in the CGM via the following registers:

- CGM_AC11_SC selects the XOSC or PLL0:PHI as the clock source.
- CGM_AC11_DC0 reduces the CAN_CLK by configuring the dividing factor from 1 to 64.

The CAN_1 clock passes through the Pseudo Random Clock Divider (PRCD) which:

- Divides CAN_CLK by 16.
- Adds approximately 40% of random jitter to both edges of the clock.

PLL0.PHI is the source whenever the PRCD is enabled. PRCD is passive in normal operation and has no impact on the clock frequency.

27.6.7 Sigma-Delta ADC clocking

The analog portion of the Sigma-Delta ADC (SDADC) works with either an internal or external modulator.

When the internal modulator in the SDADC is enabled the internal clock (SD_CLK) is always used, refer to *Figure 283*).

When the external modulator is enabled (by disabling the internal modulator), an SDADC internal mux selects either:

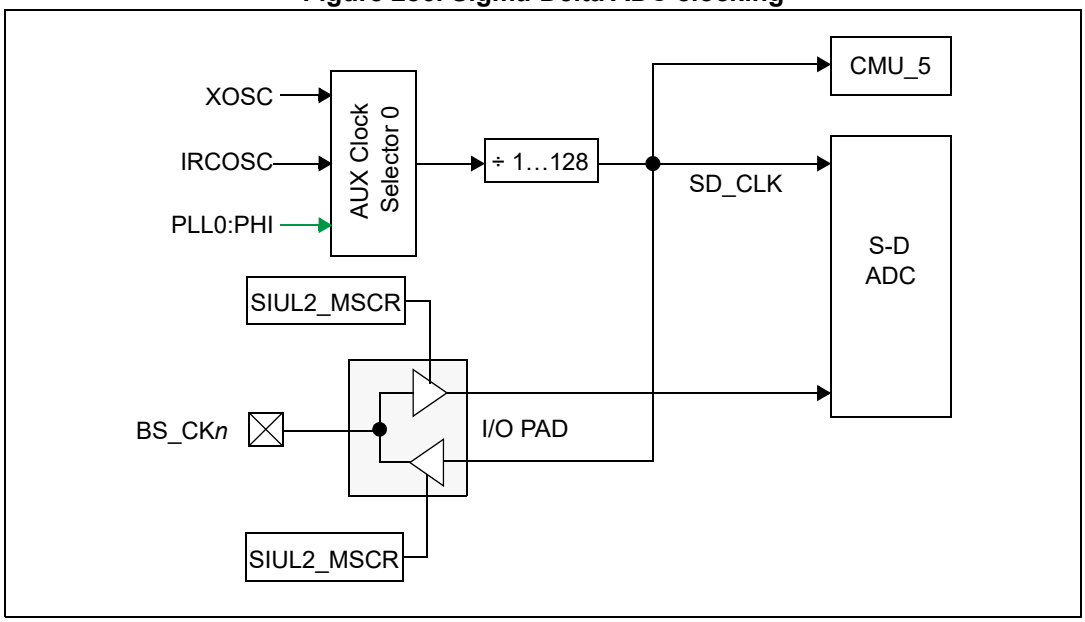
- An external input clock (BS_CK).
- An internally generated output clock (SD_CLK) connected to the external modulator clock package pin (BS_CK).

In order to provide an internal clock option when the internal modulator is enabled, the SD_CLK output from the MC_CGM must be connected to the BS_CK pin as shown in *Figure 286*.

The MSCR in the SIUL2 control of the input/output direction of the BS_CK pin.

Each SDADC is connected as shown in *Figure 286*, with the SD_CLK being common to all modules, but each SDADC having a dedicated BS_CK pin.

Figure 286. Sigma-Delta ADC clocking



27.6.8 System Clock/GTM output clock

The two system clocks (SYSCLK[1:0]) have an independent source clock selected from:

- PLL0_PHI and PLL1_PHI.
- Internal oscillator (IRCOSC)
- External oscillator (XOSC)
- RTC clock (only for SYSCLK0)

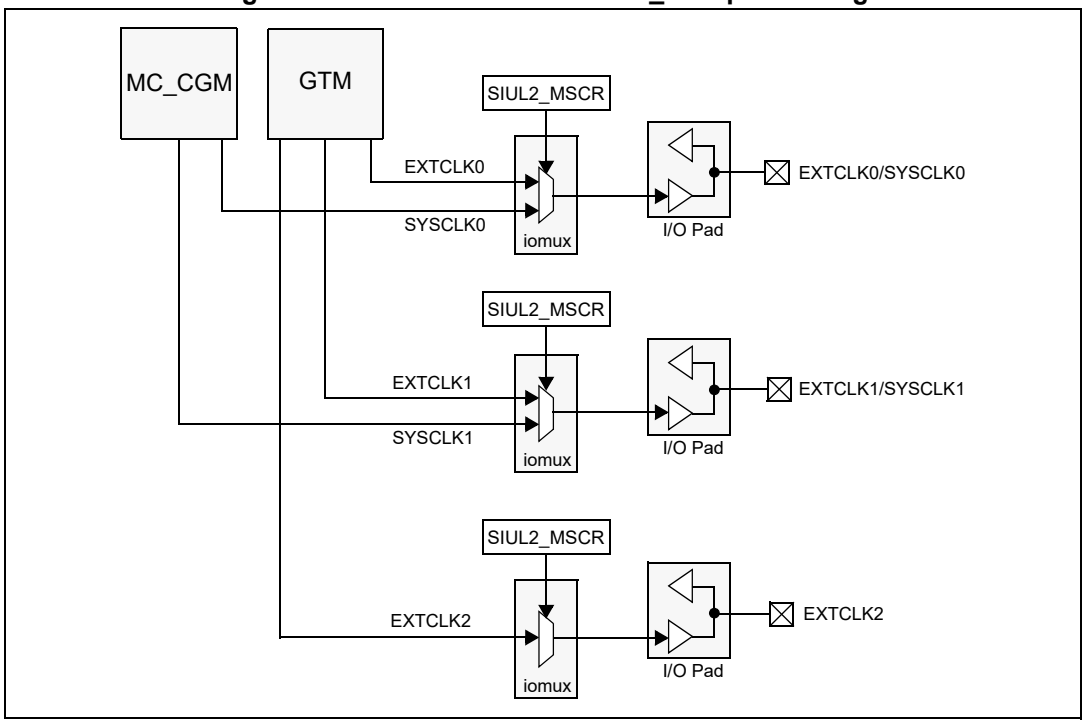
The registers CGM_AC6_SC and CGM_AC7_SC configure the clocks for SYSCLK0 and SYSCLK1 respectively. Refer to *Figure 280*.

There are three GTM output clocks available on package pins, EXTCLK[2:0]:

- EXTCLK[1:0] are muxed on the same pins as SYSCLK[1:0].

The selection of EXTCLK[*n*], SYSCLK[*n*] is handled in the SIUL2_MSCR registers. Refer to *Figure 287*.

Figure 287. SYSCLK/EXTCLK/REF_CLK pin muxing



27.7 Clock monitoring

For all safety relevant clocks the microcontroller uses clock monitoring units (CMUs) to detect a missing clock or incorrect frequency.

Each CMU is programmed independently (refer to *Figure 280* for CMU distribution) and uses the IRCOSC or XOSC as the clock monitor reference.

Detailed information on the CMUs can be found in the Clock Monitor Unit chapter.

27.7.1 CMU configuration

This section explains the CMU configuration.

Figure 288 shows the block diagram for CMU0 on the SPC58xEx/SPC58xGx.

Figure 288. CMU0 block diagram

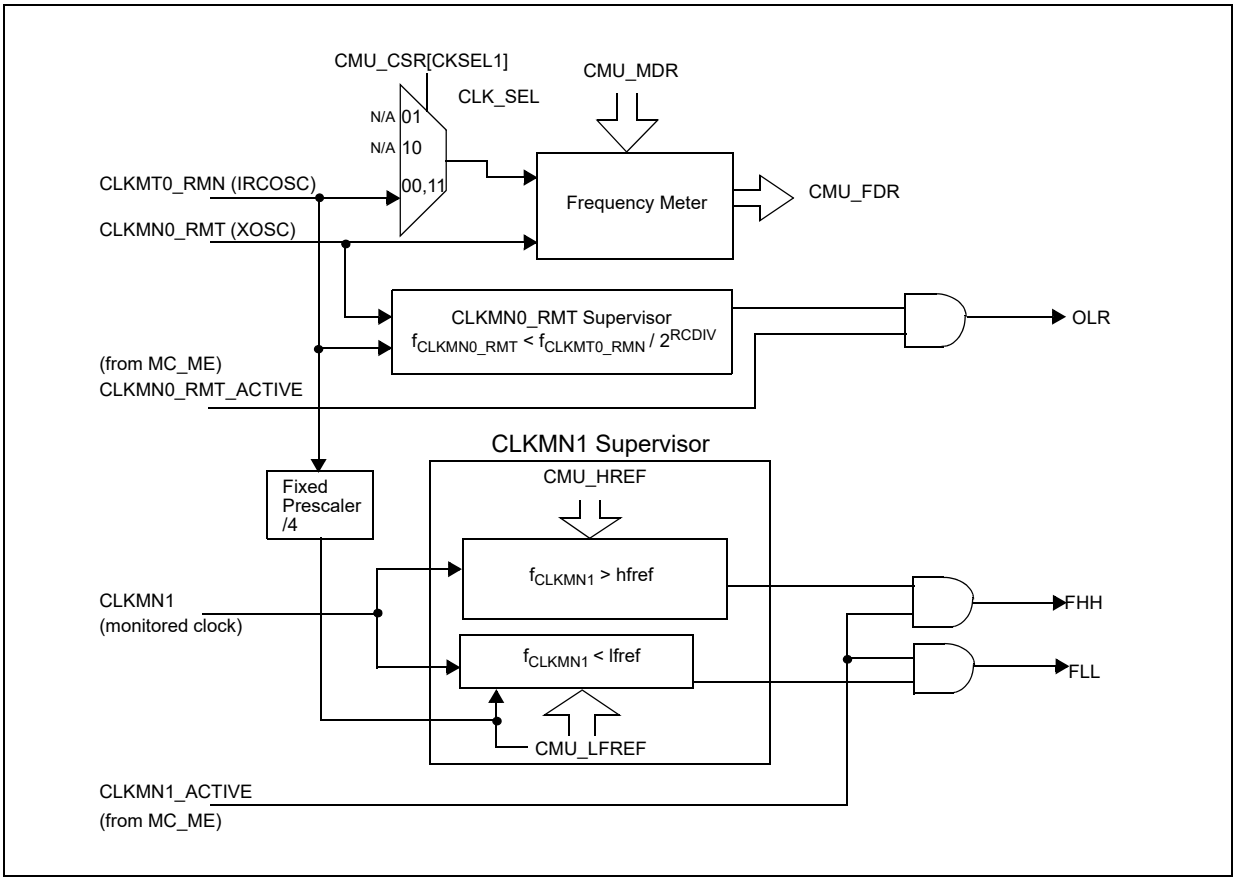
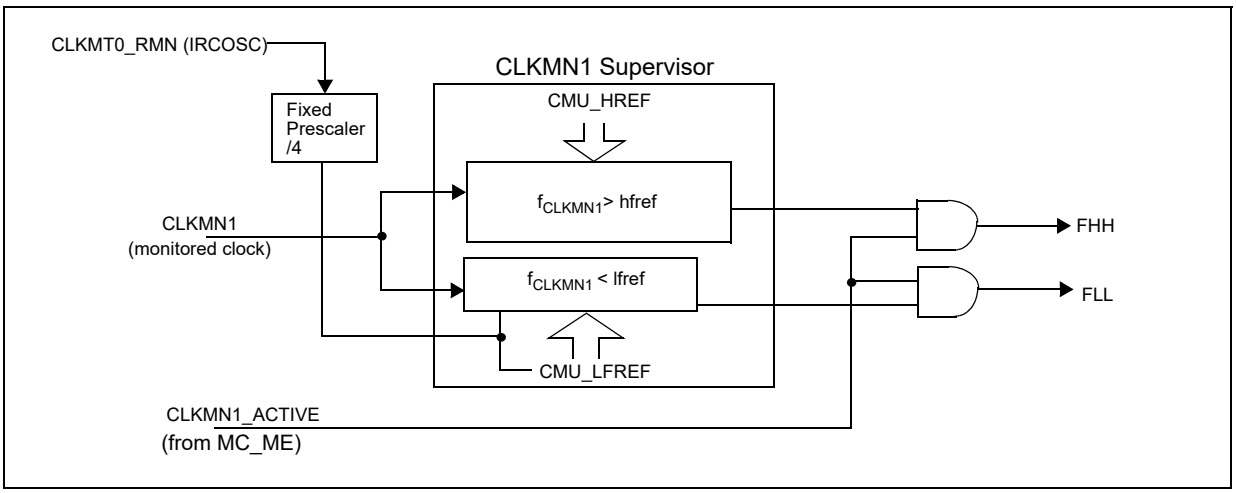


Figure 289 shows the block diagram for the other CMUs on the SPC58xEx/SPC58xGx.

Figure 289. Other CMUs block diagram



27.7.1.1 Clock input sources

Table 397 shows the clocks that are monitored by each CMU. These signals are connected internally on the chip, but are not accessible via pins on the boundary of the device. IRCOSC is the reference clock for all clock monitors. Only CMU0 implements the XOSC

monitor. CMU0 uses the IRCOSC clock to measure if the XOSC is too low. CMU0 can also be used to calibrate the IRCOSC frequency using the XOSC. All other CMUs are configured identically.

Table 397. Clock input sources

Clock module	Monitored clock	CMU group
CMU_0_PLL0_XOSC_IRCOSC	PLL0:PHI, XOSC, IRCOSC	—
CMU_1_CORE_XBAR_BD	CHKR_CLK, CORE_CLK, XBAR_CLK, BD_CLK	CMU_Platform
CMU_2_HPBM	HPBM_CLK	CMU_Platform
CMU_3_PBRIDGE	PBRIDGE_CLK	CMU_Platform
CMU_4_GTM	GTM_CLK	CMU_other
CMU_5_SDADC	SDADC_CLK	CMU_other
CMU_6_SARADC	SARADC_CLK	CMU_other
CMU_7_SENT	SENT_CLK	CMU_other
CMU_8_F189	PSI5_f189_CLK	CMU_other
CMU_9_F125	PSI5_f125_CLK	CMU_other
CMU_10_1us	PSI5_1us_CLK	CMU_other
CMU_11_FBRIDGE	FBRIDGE_CLK	CMU_Platform
CMU_12_EMIOS	EMIOS_CLK, BCTU_CLK	CMU_other
CMU_13_PFBRIDGE	PFBRIDGE_CLK	CMU_other

27.7.1.2 CMU registers and field availability

Table 398 specifies which registers and fields are available for a given CMU.

Table 398. CMU register availability

Address offset	Register	CMU	Note
0x0000	CMU_CSR	All	CMU_CSR[SFM] and CMU_CSR[CKSEL1] in CMU0 only.
0x0004	CMU_FDR	CMU0	—
0x0008	CMU_HFREFR	All	—
0x000C	CMU_LFREFR	All	—
0x0010	CMU_ISR	All	CMU_ISR[OLRI] in CMU0 only.
0x0014	Reserved	—	—
0x0018	CMU_MDR	CMU0	—

27.7.1.3 CMU register write protection

The CMU registers defined in *Table 399* have write protection with:

- Soft locking: can be unlocked by software after being previously locked.
- Hard locking: can only be unlocked by a reset once locked.

The REG_PROT module is used to implement the CMU register write protection (refer to *Section 7.10.5: Register protection (REG_PROT) configuration* in *Chapter 7: Device configuration* for details).

Table 399. CMU register write protection

Offset	Register(1)
0x0000	CMU Control Status Register (CMU_CSR)
0x0004	CMU Frequency Display Register (CMU_FDR)
0x0008	CMU High Frequency Reference Register CLKMN1 (CMU_HFREFR)
0x000C	CMU Low Frequency Reference Register CLKMN1 (CMU_LFREFR)
0x0010	CMU Interrupt Status Register (CMU_ISR)
0x0014	Reserved
0x0018	CMU Measurement Duration Register (CMU_MDR)

1. Refer to Register Protection chapter for bit field details.

27.7.2 PLL0 monitor

Software can program an upper and lower limit on the expected PLL0 PHI output clock frequency. If the monitor is enabled and the measured frequency is above or below the limits, a flag bit is set and an interrupt, if enabled, is generated. The default condition of the clock monitor is disabled.

27.7.3 External oscillator (XOSC) monitor

The XOSC frequency is compared to a minimum value limit. If the measured XOSC frequency is below the limit, a flag is set and an interrupt, if enabled, is generated.

27.7.4 Internal RC oscillator (IRCOSC) monitor

The period of the IRCOSC can be measured in CMU0, using the XOSC as a reference. This allows for application trimming of the IRCOSC frequency.

27.7.5 System clock monitors

A CMU is assigned to monitor the frequency of the cores, XBAR, peripheral bridges, peripheral, PSI5, SENT and ADC clocks (refer to *Figure 280*). The HSM does not have an assigned CMU since it has its own internal clock monitor.

27.8 Loss of system clock behavior

This device has built-in mechanisms for detecting loss of the oscillator or PLL clocks, and provides several options for reaction to a loss of clock in the application. *Figure 290* gives a high-level view of the loss of clock logic.

27.8.1 Loss of PLL/XOSC clock

As shown in *Figure 290*, each loss of lock signal from each PLL and the XOSC failure signals are monitored by the FCCU.

When a clock failure occurs, the FCCU can be programmed to generate:

- A short reset sequence: the PLL, XOSC, MC_ME, and MC_CGM are reset to their default states and the system clock is switched to the IRCOSC.
- A long reset sequence: the PLL, XOSC, MC_ME, and MC_CGM are reset to their default states and the system clock is switched to the IRCOSC.

There is no automatic system clock switch, and the user is required to program the switch through the Mode Entry module (MC_ME).

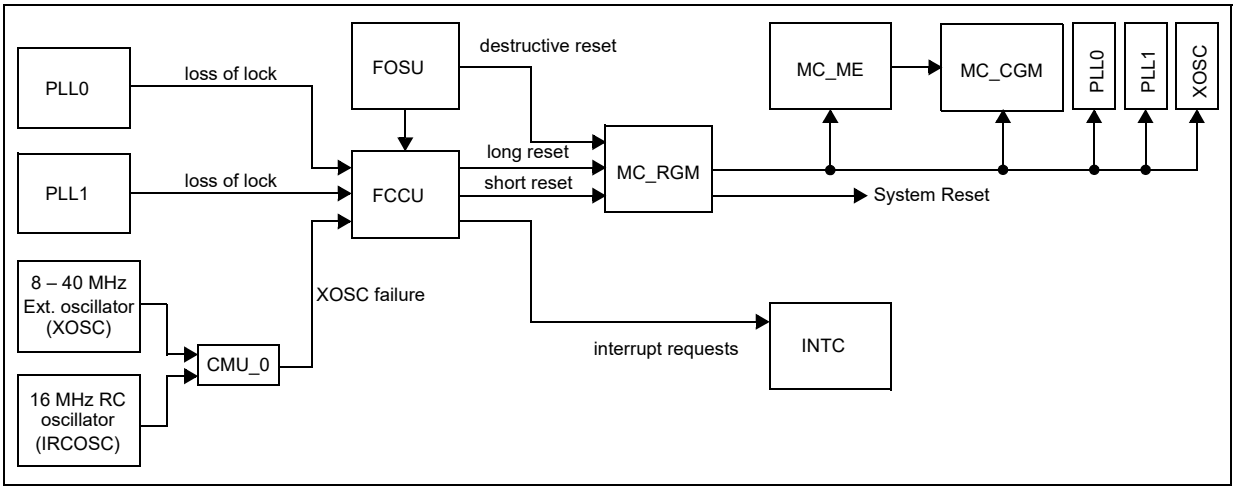
Note: In case of failures related to clocks, device functionality is no more granted till next reset. User should program properly all the safety mechanisms available in oder to reach a device safe-state.

27.8.2 Loss of IRCOSC clock

There is no automatic system clock switch, and the user is required to program the switch through the Mode Entry module (MC_ME).

The frequency of the IRCOSC clock is monitored by the frequency meter in CMU_0. There is no automated trigger of an FCCU error condition if the IRCOSC fails. Since the IRCOSC is the boot clock, a failure is a catastrophic failure.

Figure 290. Loss of clock logic



27.9 Progressive clock switching

The Progressive Clock Switching (PCS) blocks are shown in the system-level clock diagram in *Figure 280*. They are used for ramping the system. Refer to *Section 30.4.1.2: Progressive system clock switching* from *Chapter 30: Clock Generation Module (MC_CGM)* for more information on the PCS.

28 Dual PLL digital interface (PLLDIG)

28.1 Introduction

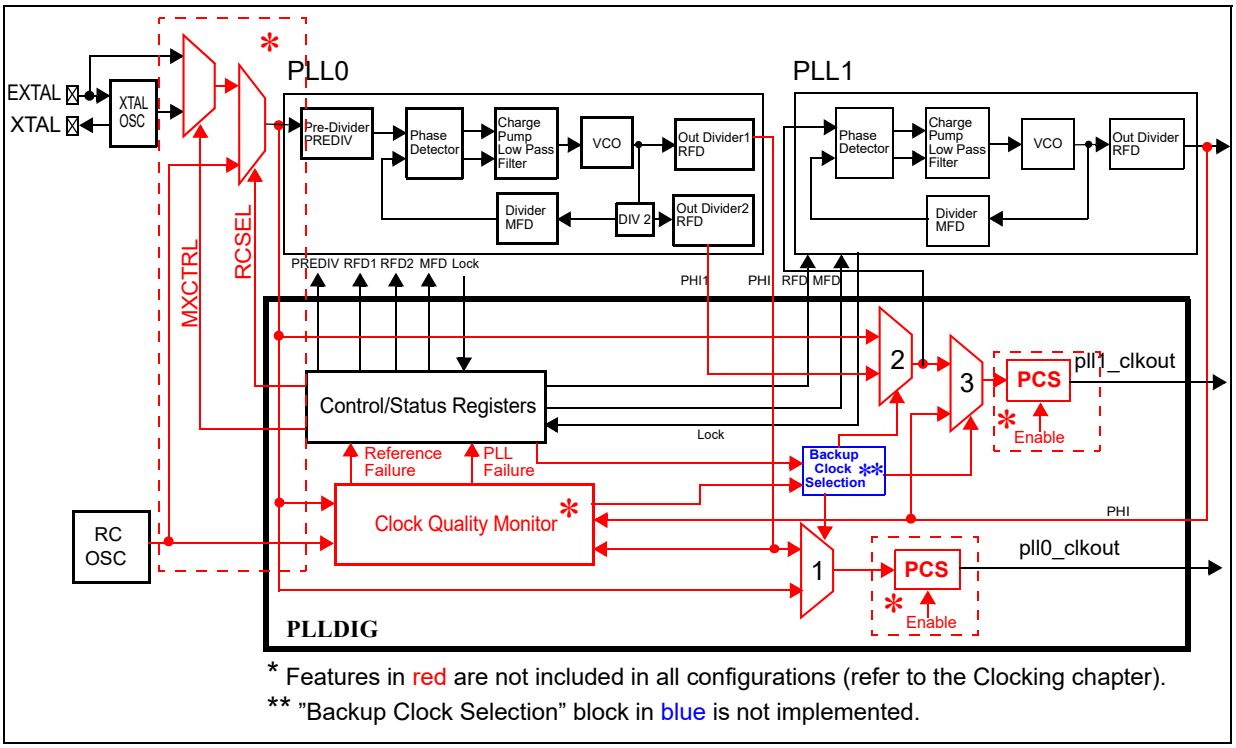
The MCU provides a user interface and control over the dual PLL system composed of PLL0 and PLL1 blocks (PLLDIG). The two analog PLL blocks are cascaded, with the PHI1 output of PLL0 feeding the clock input of PLL1 as shown in *Chapter 27: Clocking*. The key feature of the dual PLL architecture is the ability to drive peripherals from the PLL0 PHI output, which is non-modulated and independent of the core clock frequency. The core and platform clocks are driven by PLL1 PHI frequency modulated output.

Note: For chip-specific implementation details, refer to Chapter 27: Clocking.

28.2 Block diagram

Figure 291 shows the block diagram of the PLL Digital Interface.

Figure 291. Dual PLL digital interface block diagram



28.3 Features

The Dual PLLDIG has the following features:

- Supports dual PLL in cascaded mode with PLL0 clock out as reference clock to PLL1.
- Reference clock pre-divider for finer frequency synthesis resolution.
- Reduced frequency divider for reducing the PLL0/PLL1 output clock frequency without causing the PLLs to lose lock.
- Programmable frequency modulation on PLL1.
- The frequency range after the prescaler is required to be 8-20Mhz.
- Lock detect circuitry reports when the PLLs have achieved frequency lock, and continuously monitors lock status to report loss of lock conditions.
  - User-selectable ability to generate an interrupt request upon loss of lock.
  - The loss of lock indication is sent to the FCCU via the system glue logic.

Note: Refer to the device datasheet for information on the input and output clock frequency range.

28.4 Modes of operation

The operating mode of the PLLs is determined by the value of PLL*n*CR[CLKCFG]. The mode of operation for the PLL is defined below:

Normal mode with reference, and either PLL0 or both PLLs enabled.

Note: Mode changes are controlled by configuring the Mode Entry mode configuration registers (<ME_mode>_MC).

Normal mode is defined as the mode where the clocks are driven by the PLL. When using a crystal for the clock reference, reset should remain asserted until the oscillator has stabilized.

When PLL0 is powered down and working in bypass mode, PLL1 should be bypassed and powered down. PLL1 should be configured to work in normal mode only after PLL0 has achieved lock (for example, bypass with PLL0 running or PLL0 in normal mode, PLL*n*SR[LOCK] = 1).

28.4.1 Normal mode with reference, PLL0 or both PLLs enabled

In the normal mode, PLL0 receives an input clock from the reference and the pre-divider. PLL0 multiplies the frequency to create the PLL0 output clock. The user must supply a crystal that is within the appropriate frequency range, the crystal manufacturer recommended external support circuitry and short signal route from the MCU to the crystal.

PLL0 generates a non-modulated clock which drives PLL0_PHI1. PLL1 can generate a frequency modulated clock or a non-modulated clock (for example, locked on a single frequency). The modulation rate, modulation depth, output divider (Reduced Frequency Divider) and whether the PLL1 is modulated or not can be programmed by writing to the PLL1FM register, *Section 28.5.2.7: PLL1 frequency modulation register (PLL1FM)*.

Note: While powering down the PLLs, user must ensure that PLL1 is powered down first followed by powering down PLL0 for proper shut off.

Note: Only when the PLL0 achieves lock should PLL1 be configured to work in either mode by programming the appropriate <ME_mode>_MC register (see Section 28.7: Initialization information).

28.5 Memory map and register definition

This section provides the memory map and detailed descriptions of all registers for configuring the PLLs.

28.5.1 Memory map

Table 400 shows the memory map. Addresses are given as offsets from the module base address. All registers can be accessed using 8-bit, 16-bit or 32-bit addressing.

Table 400. Dual PLL digital interface memory map

Offset (hex)	Register	Section
0x0000	PLL0 Control Register (PLL0CR)	*Section 28.5.2.1*
0x0004	PLL0 Status Register (PLL0SR)	*Section 28.5.2.2*
0x0008	PLL0 Divider Register (PLL0DV)	*Section 28.5.2.3*
0x000C–0x001F	Reserved	
0x0020	PLL1 Control Register (PLL1CR)	*Section 28.5.2.4*
0x0024	PLL1 Status Register (PLL1SR)	*Section 28.5.2.5*
0x0028	PLL1 Divider Register (PLL1DV)	*Section 28.5.2.6*
0x002C	PLL1 Frequency Modulation Register (PLL1FM)	*Section 28.5.2.7*
0x0030	PLL1 Fractional Divide Register (PLL1FD)	*Section 28.5.2.8*
0x0034–0x003F	Reserved	

28.5.2 Register descriptions

Some of the register reset values are specifically configured for each unique device by external configuration signals or parameters.

28.5.2.1 PLL0 control register (PLL0CR)

Offset: 0x0000 Access: User read/write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	CLKCFG(3)		EXPDIE	0(1)	0(1)	0(1)	LOLIE(2)	0(1)	0(1)	0(1)
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. This field can be written at any time, but writes are ignored. Reads returns previously written value.
2. Refer to *Chapter 27: Clocking* for details on loss of lock management.
3. CLKCFG can be written, but writes have no effect. Mode changes are implemented by writing to the appropriate <ME_mode>_MC register.

Figure 292. PLL0 control register (PLL0CR)

Table 401. PLL0CR field descriptions

Field	Description
22:23 CLKCFG	Clock configuration This field indicates the operational state of PLL. 00 PLL off. 01 Reserved 10 Reserved. 11 Normal mode with PLL running When PLLs are externally powered down the CLKCFG field changes to 00b. When external power down is removed, these bits will read 11b. **Note:** In cascaded PLL configuration (PLL0 driving the reference clock for PLL1) the external power down signal for PLL1 must be removed only when PLL0 has locked, PLL0SR[LOCK] = 1.
24 EXPDIE	External power down cycle complete indication interrupt enable This bit enables generation of an interrupt when the external power down of the PLL is complete (for example, PLLs are powered down and powered back up). 0 Ignore interrupt. Interrupt not requested 1 Enable interrupt request on power down cycle completion and enable the FCCU failure "Loss of lock flag PLL0"
28 LOLIE	Loss of lock interrupt enable The LOLIE bit enables a loss of lock interrupt request when PLL0SR[LOLF] = 1. If PLL0SR[LOLF] = 0 or PLL0CR[LOLIE] = 0, the interrupt request is ignored. The interrupt request only occurs in normal mode. 0 Ignore loss of lock. Interrupt not requested 1 Enable interrupt request upon loss of lock

28.5.2.2 PLL0 status register (PLL0SR)

Offset: 0x0004 Access: User read/write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	EXTPDF	0	0	0	LOLF	LOCK	0	0
W									w1c		*(1)		w1c		*(1)	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. This field is reserved, read only.

Figure 293. PLL0 status register (PLL0SR)

Table 402. PLL0SR field descriptions

Field	Description
24 EXTPDF	External power down cycle complete indication interrupt flag This bit indicates that the external power down of the PLL is complete (for example, PLLs are powered down and then powered up). User must write 1 to clear this bit. 0 PLLs not power cycled. Interrupt not requested 1 PLLs power down cycle is complete. Interrupt is requested
28 LOLF	Loss of lock flag This bit provides the interrupt flag for the loss of lock condition. Software must write 1 to LOLF to clear it, writing 0 has no effect. The flag is set when the PLL loses lock or when the divider/modulation registers are changed on the fly. This flag bit is sticky in the sense that if lock is reacquired, the bit will remain set until cleared by either writing 1 or asserting reset. 0 No loss of lock detected. Interrupt service not requested 1 Loss of lock detected. Interrupt service requested
29 LOCK	Lock status bit Indicates whether PLL has acquired lock. 0 PLL is unlocked 1 PLL is locked

28.5.2.3 PLL0 divider register (PLL0DV)

The PLL0DV register provides the PHI/PHI1 output clock reduced frequency dividers, pre-divider, and loop divider.
The values of PREDIV and MFD should not be changed when the PLL is on and locked (working in Normal mode). If these fields are changed without powering down the PLL, the PLL will lose lock and generate either a reset or an interrupt, based on which is enabled.
The reduced frequency divider fields (RFDPHI, RFDPHI1) can be modified at anytime, but the changes only become effective after PLL0 is disabled, then re-enabled.

Offset: 0x0008 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	RFDPHI1				0	0	0	0	0	RFDPHI					
W																
Reset	0	_(1)	_(1)	_(1)	_(1)	0	0	0	0	0	_(1)	_(1)	_(1)	_(1)	_(1)	_(1)

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	PREDIV			0	0	0	0	0	MFD						
W																
Reset	0	_(1)	_(1)	_(1)	0	0	0	0	0	_(1)	_(1)	_(1)	_(1)	_(1)	_(1)	_(1)

1. Refer to *Chapter 27: Clocking* for default reset value information.

Figure 294. PLL0 divider register (PLL0DV)

Table 403. PLL0DV field descriptions

Field	Description
1:4 RFDPHI1	PHI1 reduced frequency divider This 4-bit field determines the VCO/2 clock post divider for driving the PHI1 output clock. 00xx Reserved 0100 Divide by 4 0101 Divide by 5 ... 1110 Divide by 14 1111 Divide by 15
10:15 RFDPHI	PHI reduced frequency divider This 6-bit field determines the VCO clock post divider for driving the PHI output clock. 00 Reserved 01 Divide by 1 02 Divide by 2 03 Divide by 3 ... 3E Divide by 62 3F Divide by 63

Table 403. PLL0DV field descriptions (continued)

Field	Description
17:19 PREDIV	Input clock pre-divider This 3-bit field controls the value of the divider on the input clock. The output of the pre-divider circuit generates the reference clock to the PLL analog loop. The PREDIV value 000b causes the input clock to be inhibited. 000 Clock inhibit 001 Divide by 1 010 Divide by 2 ... 110 Divide by 6 111 Divide by 7
25:31 MFD	Loop multiplication factor divider This 7-bit field controls the value of the divider in the feedback loop. The value specified by the MFD bits establishes the multiplication factor applied to the reference frequency. 00 Reserved ... 07 Reserved 08 Divide by 8 ... 7E Divide by 126 7F Divide by 127

28.5.2.4 PLL1 control register (PLL1CR)

Offset: 0x0020 Access: User read/write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	CLKCFG(3)		EXPDIE	0(1)	0(1)	0(1)	LOLIE(2)	0(1)	0(1)	0(1)
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. This field can be written at any time, but writes are ignored. Reads returns previously written value.
2. Refer to *Chapter 27: Clocking* for details on loss of lock management.
3. CLKCFG can be written, but writes have no effect. Mode changes are implemented by writing to the appropriate <ME_mode>_MC register.

Figure 295. PLL1 control register (PLL1CR)

Table 404. PLL1CR field descriptions

Field	Description
22:23 CLKCFG	Clock configuration This field indicates the operational state of PLL. 00 PLL off 01 Reserved 10 Reserved 11 Normal mode with PLL running When PLLs are externally powered down the CLKCFG field changes to 00b. When external power down is removed, these bits will read 11b. **Note:** In cascaded PLL configuration (PLL0 driving the reference clock for PLL1) the external power down signal for PLL1 must be removed only when PLL0 has locked, PLL0SR[LOCK] = 1.
24 EXPDIE	External power down cycle complete indication interrupt enable This bit enables generation of an interrupt when the external power down of the PLL is complete (for example, PLLs are powered down and powered back up). 0 Ignore interrupt. Interrupt not requested 1 Enable interrupt request on power down cycle completion
28 LOLIE	Loss of lock interrupt enable The LOLIE bit enables a loss of lock interrupt request when PLL1SR[LOLF] = 1. If PLL1SR[LOLF] = 0 or PLL1CR[LOLIE] = 0, the interrupt request is ignored. The interrupt request only occurs in normal mode. 0 Ignore loss of lock. Interrupt not requested 1 Enable interrupt request upon loss of lock and enable the FCCU failure” Loss of clock flag PLL1”

28.5.2.5 PLL1 status register (PLL1SR)

Offset: 0x0024 Access: User read/write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	EXTPDF	0	0	0	LOLF	LOCK	0	0
W									w1c		*(1)		w1c		*(1)	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. This field is reserved, read only.

Figure 296. PLL1 status register (PLL1SR)

Table 405. PLL1SR field descriptions

Field	Description
24 EXTPDF	External power down cycle complete indication interrupt flag This bit indicates that the external power down of the PLL is complete (for example, PLLs are powered down and then powered up). User must write 1 to clear this bit. 0 PLLs not power cycled. Interrupt not requested 1 PLLs power down cycle is complete. Interrupt is requested
28 LOLF	Loss of lock flag This bit provides the interrupt flag for the loss of lock condition. Software must write 1 to LOLF to clear it, writing 0 has no effect. The flag is set when the PLL loses lock or when the divider/modulation registers are changed on the fly. This flag bit is sticky in the sense that if lock is reacquired, the bit will remain set until cleared by either writing 1 or asserting reset. 0 No loss of lock detected. Interrupt service not requested 1 Loss of lock detected. Interrupt service requested
29 LOCK	Lock status bit Indicates whether PLL has acquired lock. 0 PLL is unlocked 1 PLL is locked

28.5.2.6 PLL1 divider register (PLL1DV)

The PLL1DV register provides the output clock reduced frequency dividers and loop divider values.
The value of MFD should not be changed when the PLL is on and locked (working in Normal mode). If the field is changed without powering down the PLL, the PLL will lose lock and generate either a reset or an interrupt, based on which is enabled.
The reduced frequency divider field (RFDPHI) can be modified at anytime, but the changes only become effective after PLL1 is disabled, then reenabled.

Offset: 0x0028 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	RFDPHI[1]					
W																
Reset	0	0	_(1)	_(1)	_(1)	_(1)	_(1)	_(1)	0	0	_(1)	_(1)	_(1)	_(1)	_(1)	_(1)

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	MFD[1]						
W																
Reset	0	0	0	0	0	0	0	0	0	_(1)	_(1)	_(1)	_(1)	_(1)	_(1)	_(1)

1. Refer to *Chapter 27: Clocking* for default reset value information.

Figure 297. PLL1 divider register (PLL1DV)

Table 406. PLL1DV field descriptions

Field	Description
10:15 RFDPHI	PHI reduced frequency divider This 6-bit field determines the VCO clock post divider for driving the PHI output clock. 0x00 Reserved 0x01 Divide by 1 0x02 Divide by 2 ... 0x3E Divide by 62 0x3F Divide by 63
25:31 MFD	Loop multiplication factor divider This 7-bit field controls the value of the divider in the feedback loop. The value specified by the MFD bits establishes the multiplication factor applied to the reference frequency. 0x10 Divide by 16 0x11 Divide by 17 .... 0x21 Divide by 33 0x22 Divide by 34 All other settings are reserved

28.5.2.7 PLL1 frequency modulation register (PLL1FM)

The PLL1FM register enables frequency modulation on PLL1 and provides controls for the FM spread spectrum, modulation depth and modulation rate. This register should be written when PLL1CR[CLKCFG] = 00b (PLL1 powered down).
Changing the values of PLL1FM[MODEN] and PLL1FM[MODSEL] fields once the PLL is running, and has locked, can result in the PLL losing its lock and the device being reset (if PLL1CR[LOLRE] = 1). The PLL would then require power cycling to regain normal functionality.
The modulation period (PLL1FM[MODPRD]) and increment step fields (PLL1FM[INCSTP]) can be changed without losing lock, however, the output clock from PLL may have an incorrect frequency for a few cycles after either of these updates (as defined in the datasheet).

Offset: 0x002C Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	MODEN	MODSEL	MODPRD												
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	INCSTP														
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 298. PLL1 frequency modulation register (PLL1FM)

Table 407. PLL1FM field descriptions

Field	Description
1 MODEN	Modulation enable This bit enables the frequency modulation. 0 Frequency modulation disabled 1 Frequency modulation enabled
2 MODSEL	Modulation selection This bit selects whether modulation is centered around the nominal frequency or spread below the nominal frequency. 0 Modulation centered around nominal frequency 1 Modulation spread below nominal frequency
3:15 MODPRD	Modulation period This 13-bit field is the binary equivalent of the modulation period variable derived from the formula: $MODPRD = round(\frac{f_{ref}}{4 \times f_{mod}})$ where f_{ref} represents the frequency of the feedback divider and f_{mod} represents the modulation frequency (refer to equation in *Section 28.6.3*).
17:31 INCSTP	Increment step This 15-bit field is the binary equivalent of the INCSTP variable derived from the formula: $INCSTEP = round(\frac{(2^{15} - 1) \times MD \times PLL1CR[MFD]}{100 \times 5 \times MODPRD})$ where *md* represents the peak modulation depth in percentage (+/- *MD* for centered modulation, -2**MD* for modulation below nominal frequency), and MFD represents the nominal value of the feedback loop divider (refer to equation in *Section 28.6.3*).

28.5.2.8 PLL1 fractional divider register (PLL1FD)

The PLL1FD register provides the enable and fractional divide factor for the loop divider. This register should be written when PLL1CR[CLKCFG] = 00b (PLL1 powered down). Changing the values of FDEN once the PLL is running, and has locked, can result in the PLL losing its lock and the device being reset (if PLL1CR[LOLRE] = 1). The PLL would then require power cycling to regain normal functionality.
The dither disable and fractional divider fields can be changed without losing lock, however, the output clock from PLL can have an incorrect frequency for a few cycles after either of these updates .

Offset: 0x0030 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	FDEN	0	0	0	0	0	0	0	0	0	0	0	0	DTHDIS	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	FRCDIV											
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 299. PLL1 fractional divider register (PLL1FD)

Table 408. PLL1FD field descriptions

Field	Description
1 FDEN	Fractional divide enable This bit enables the fractional divider in the loop divider for PLL1. 0 Fractional divide disabled 1 Fractional divide enabled
14:15 DTHDIS	Dither disable Bit 14 and bit 15 control noise of SSCG and Fractional controller. 00 Increase PLL multiplication factor by 1/2^13 and has maximum noise control of SSCG and Fractional controller. 01 Increase PLL multiplication factor by 1/2^13 and has some noise control of SSCG and Fractional controller. 10 No influence on PLL multiplication factor and has some noise control of SSCG and Fractional controller. 11 No influence on PLL multiplication factor and no noise control of SSCG and Fractional controller.
20:31 FRCDIV	Fractional divide input When the fractional divide is disabled, the VCO clock frequency is the product of the input clock and the loop divide factor (MFD). When fractional divide is enabled, the mean VCO clock frequency is given by: $\text{VCO Frequency} = \text{InputClock} \times (\text{PLL1DV[MFD]} + (\text{FRCDIV} \div 2^{12}))$ where FRCDIV is the decimal equivalent of the FRCDIV bits (refer to *Section 28.6.2*). **Note:** Depending on DTHDIS value the multiplication factor can change as described in DTHDIS field description.

28.5.3 Register classification for safety requirements

This module is classified as a safety Vital Module (ViMo). Hence, the registers of this module have been classified as shown in the SPC58xEx/SPC58xGx Reference Manual “Functional Safety” chapter.

28.6 Functional description

This section explains the operation and configuration of the Dual PLLDIG module.

28.6.1 Input clock frequency

PLL0 and PLL1 are designed to operate over an input clock frequency range. The operating ranges for each PLL are discussed in detail in the *SPC58xEx/SPC58xGx datasheet*.

28.6.2 Clock configuration

The relationship between input and output frequency is determined by programming the PLL0DV, PLL1DV and PLL1FD registers, and calculated according to the following equations:

Equation 4

$$f_{pll0_phi} = f_{pll0_ref} \times \frac{PLL0DV[MFD]}{PLL0DV[PREDIV] \times PLL0DV[RFDPHI]}$$

Equation 5

$$f_{pll0_phi1} = f_{pll0_ref} \times \frac{PLL0DV[MFD]}{PLL0DV[PREDIV] \times PLL0DV[RFDPHI1]}$$

Equation 6

$$f_{pll1_phi} = f_{pll1_ref} \times \left(\frac{PLL1DV[MFD] + \frac{PLL1FD[FRCDIV]}{2^{12}}}{2 \times PLL1DV[RFDPHI]} \right)$$

Note: Depending on DTHDIS value the multiplication factor of *Equation 6* can change as described in PLL1DV[DTHDIS] field description (refer to *Table 408*).

The relationship between the VCO frequency f_{VCO} and the output frequency of the PLLs is determined by the configuration of the PLL0DV, PLL1DV and PLL1FD registers, according to the following equations:

Equation 7

$$f_{pll0_VCO} = \frac{f_{pll0_ref} \times PLL0DV[MFD] \times 2}{PLL0DV[PREDIV]}$$

Equation 8

$$f_{pll1_VCO} = f_{pll1_ref} \times \left(PLL1DV[MFD] + \frac{PLL1FD[FRCDIV]}{2^{12}} \right)$$

Note: Depending on DTHDIS value the multiplication factor of Equation 8 can change as described in PLL1DV[DTHDIS] field description (refer to Table 408).

Note: f_{pll0_phi1} is the reference clock generated by PLL0 for PLL1.

When programming the PLLs, user software must not violate the maximum system clock frequency or max/min VCO frequency specification for PLL0 or PLL1 (refer to the device datasheet). Furthermore, the PLL0DV[PREDIV] value must not be set to any value that causes the input frequency to the phase detector of analog PLL blocks to go below the prescribed ranges.

The lock signal and PLL*n*SR[LOCK] flag are immediately negated if the fields of PLL*n*DV are changed without powering down the analog PLLs.

When any of these events occur, an internal timer is initialized to count 64 cycles of the PLL input clock. During this period (64 cycles and a few extra clock cycles for synchronization, for example, 64 to 72 cycles), the PLL*n*SR[LOCK] flag is held negated. After the timer expires, the PLL*n*SR[LOCK] flag reflects the value coming from the PLL lock detection circuitry. To prevent an immediate reset, the PLL*n*CR[LOLRE] bit of the respective PLLs must be cleared before doing any of the above operations.

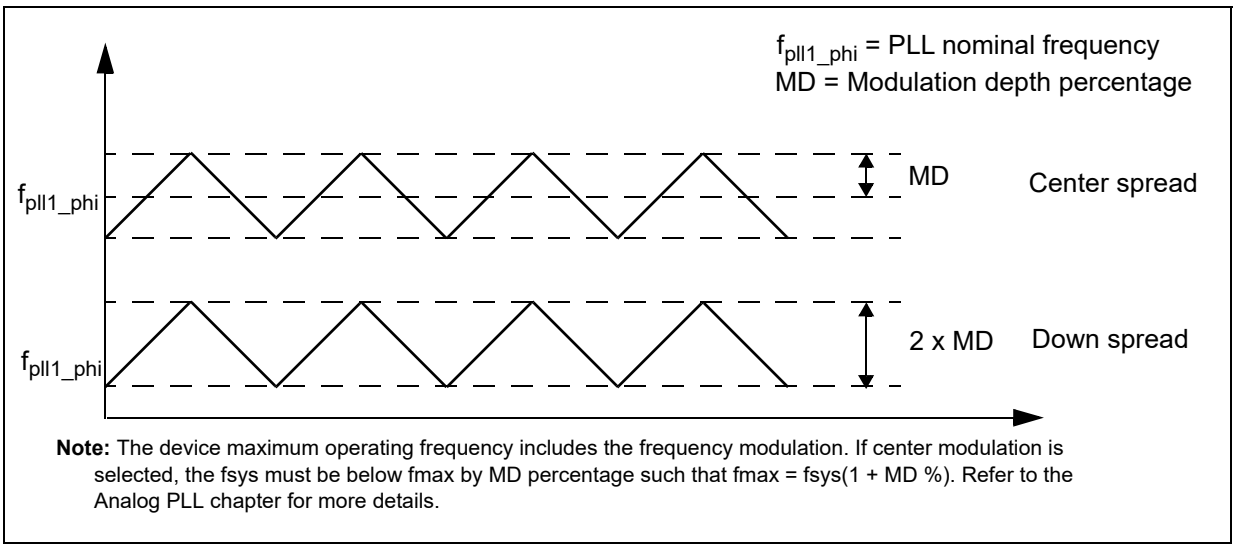
Note: The PLL must be powered down and powered up by configuring <ME_mode>_MC before PLL0CR[PREDIV], PLL0CR[MFD], PLL1CR[MFD] or the input reference clock values are modified.

The recommended procedure to program the PLLs and engage normal mode is shown in Section 28.7.

28.6.3 Frequency modulation

Frequency modulation uses a triangular profile as shown in Figure 300. The modulation frequency and depth are controlled using PLL1FM[MODPRD] and PLL1FM[INCSTP].

Figure 300. Triangular frequency modulation



Note: The device maximum operating frequency includes the frequency modulation. If center modulation is selected, the fsys must be below fmax by MD percentage such that fmax = fsys(1 + MD %). Refer to the Analog PLL chapter for more details.

The following equations define how to calculate PLL1FM[MODPRD] (Equation 9) and PLL1FM[INCSTP] (Equation 10) based on the frequency of the feedback divider (f_{pll1_ref}), the modulation frequency (f_{mod}) and the modulation depth percentage (MD):

Equation 9

$$\text{PLL1FM[MODPRD]} = \text{round}\left(\frac{f_{pll1_ref}}{4 \times f_{mod}}\right)$$

Equation 10

$$\text{PLL1FM[INCSTP]} = \text{round}\left(\frac{(2^{15} - 1) \times MD \times \text{PLL1DV[MFD]}}{100 \times 5 \times \text{PLL1FM[MODPRD]}}\right)$$

PLL1FM[MODPRD] and PLL1FM[INCSTP] are subject to the following restriction:

Equation 11

$$(\text{PLL1FM[MODPRD]} \times \text{PLL1FM[INCSTP]}) < 2^{15}$$

Because of the above rounding operations, the effective modulation depth applied to the FMPLL is given by the following formula:

Equation 12

$$\text{ModulationDepth} = \text{round}\left(\frac{\text{PLL1FM[MODPRD]} \times \text{PLL1FM[INCSTP]} \times 100 \times 5}{(2^{15} - 1) \times \text{PLL1DV[MFD]}}\right)$$

As an example, suppose the following configuration:

- Input frequency: 40 MHz
- Loop divider (PLL1DV[MFD]): 20
- Input divider: 1
- Output divider RFD: 5
- VCO frequency: 40 MHz × 20 = 800 MHz
- PLL output frequency: 800 MHz / 2*RFD = 800 MHz / 10 = 80 MHz
- Center spread (PLL1FM[MODSEL] = 0)
- Modulation frequency: 200 kHz
- Modulation depth: ± 1.9% (3.8% peak-to-peak)
- PLL1FM[MODPRD] = Round[(40 × 10^6) / (4 × 200 × 10^3)] = Round[50] = 50
- PLL1FM[INCSTP] = Round[((2^{15} – 1) × 1.9× 20) / (100 × 5 × 50)] = Round[49.8] = 50
- PLL1FM[MODPRD] × PLL1FM[INCSTP] = 50 × 50 = 2500 (which is less than 2^{15})
- MD (quantized) = ((50 × 50 × 100 × 5) / ((2^{15} – 1) × 20) = 1.90740 %

In this example, the modulation error is 0.00740 %.

The FM parameters do not get reset when PLL0 looses its lock. These parameters get reset on destructive reset. Whenever PLL0 gets relocked FM modulation starts with the previous values. The sequence for reprogramming the FM is:

1. Power down PLL1[12]
2. Program the PLL1FM while PLL1 is powered down
3. Power up the PLL1[13]

28.7 Initialization information

Coming out of reset the PLLs are disabled per the DRUN mode configuration register (<*ME*>_DRUN_MC). The recommended procedure[14] to program the PLLs and enter normal mode is:

1. Configure PLL0 and related modules.

   Program PLL0DV[PREDIV], PLL0DV[RFDPHI1], PLL0DV[MFD] and PLL0DV[RFDPHI].

   a) With PLL0 disabled, program PLL0 clock source in <CGM_ACn>_SC[SELCTL]. Default source is 16 MHz IRCOSC. Write <CGM_ACn>_SC[SELCTL] = 1 to select XOSC as source.

   b) Program PLL0DV[PREDIV], PLL0DV[RFDPHI1], PLL0DV[MFD] and PLL0DV[RFDPHI].

   c) If desired, modify the XOSC_CTL[EOCV] to adjust the external oscillator stabilization count, used when the external oscillator is turned on (by the Mode Entry module).

   d) If necessary, modify the CMU frequency meter values (CMU_MDR and CMU_FDR) used to monitor the XOSC frequency. XOSC frequency is compared to the IRCOSC source when XOSC is turned on.

2. Turn on XOSC and PLL0.

   a) Configure a mode configuration for turning on PLL0 and XOSC. In a mode configuration register (for example, <ME>_DRUN_MC) set <ME_mode>_MC[XOSCON] = 1 and <ME_mode>_MC[PLL0ON] = 1. If desired, also set <ME_mode>_MC[SYSCLK] = 2 for this new mode configuration to use PLL0 (PLL0 PHI output) as the sysclk.

   b) Enter that mode by two writes to <ME>_CTRL register to enter that mode. This is required even if re-entering the same mode.

3. Wait for the mode transition to complete.

   a) Wait for the mode transition to complete by polling <ME>_GS[S_MTRANS] or enabling an interrupt for flag <ME>_IS[I_MTC]. A timer, even if it is the watchdog,

12. By writing <ME_mode>_MC[PLL1ON] = 0b.
13. By writing <ME_mode>_MC[PLL1ON] = 1b.
14. The steps from 1 to 6 describe a recommended way to initialize the PLLs. But the mode_transition from DRUN to RUNx with the PLLs ON (with PLL1 as sys_clk and PLL0_PHI1 as source of PLL1 and all other clk sources enabled) can be done with a single mode transition. The only software requirement is to ensure that the reference clocks are also configured ON when the derived clock is configured ON in the target mode.

should be used to make sure the mode transition completes. Mode transition will NOT complete until:

- XOSC counter expires (if the new mode configuration changes <ME_mode>_MC[XOSCON] to 1), and

- PLL is locked (if the new mode configuration changes <ME_mode>_MC[PLL0ON] = 1).

b) Confirm the desired target mode was entered by checking the status of <ME>_GS[S_CURRENT_MODE].

4. Configure PLL1 and related modules
   a) With PLL1 disabled, program PLL1 clock source in <CGM_ACn>_SC[SELCTL]. Default is <CGM_ACn>_SC[SELCTL] = which selects XOSC.
   b) Program PLL1DV[MFD] and PLL1DV[RFDPHI].
   c) If required, program the PLL1FM register with the desired frequency modulation parameters and enable FM modulation, PLL1FM[MODEN] = 1. The PLL1FM register must not be written after PLL1 is enabled and operating in normal mode.
5. Turn on PLL1
   a) Configure a mode configuration for turning on PLL1, and keeping XOSC and PLL0 on. In a mode configuration register (for example, <ME>_DRUN_MC), initialize <ME_mode>_MC[XOSCON] = 1, <ME_mode>_MC[PLL0ON] = 1 and <ME_mode>_MC[PLL1ON] = 1. If desired, also set <ME_mode>_MC[SYSCLK] = 4 for this new mode configuration to use PLL1 as the sysclk.
   b) Enter that mode by two writes to <ME>_CTRL register to enter that mode. This is required even if re-entering the same mode.
6. Wait for the mode transition to complete.
   a) Wait for the mode transition to complete by polling <ME>_GS[S_TRANS] or enabling an interrupt for flag <ME>_IS[I_MTC]. A timer, even if it is the watchdog, should be used to make sure the mode transition completes. Mode transition will NOT complete until:

      XOSC counter expires (if the new mode configuration changes <ME_mode>_MC[XOSCON] = 1), and PLL1 is locked (if the new mode configuration changes <ME_mode>_MC[PLL1ON] = 1).

   b) Confirm the desired target mode was entered by checking the status of <ME>_GS[S_CURRENT_MODE].

Note: Refer to the Mode Entry module chapter for more details.

29 Clock Monitor Unit (CMU)

29.1 Introduction

The Clock Monitor Unit (CMU), also referred to as clock quality checker or clock fault detector, serves three purposes:

- Measures the frequency of clock sources CLKMT0_RMN with CLKMN0_RMT as the reference clock
- Monitors CLKMN0_RMT frequency with CLKMT0_RMN as reference clock
- Monitors CLKMN1 frequency with CLKMT0_RMN as reference clock and also detects if the monitored clock frequency leaves an upper or lower frequency boundary

One of the tasks is to supervise the integrity of the various clock sources on the chip, for example CLKMN0_RMT or CLKMN1. If the monitored clock frequency is less than the reference clock, or it violates an upper or lower frequency boundary, the CMU detects and reports this event. These events signal the FCCU, which can take necessary corrective actions as its configuration dictates.

The CMU can monitor CLKMN0_RMT, which must have a frequency higher than that of CLKMT0_RMN divided by the division factor shown in CMU_CSR[RCDIV], and reports this event. The CMU can also monitor CLKMN1 and generate events when the CLKMN1 frequency is less than the CLKMT0_RMN frequency divided by four or if CLKMN1 leaves an upper or lower frequency boundary. The upper and lower frequency boundaries are defined by the CMU High Frequency Reference Register (CMU_HFREFR) and CMU Low Frequency Reference Register (CMU_LFREFR).

The second task of the CMU is to provide a frequency meter, which allows measuring the frequency of one clock source against a reference clock. This is useful to allow the calibration of the metered clocks (such as CLKMT0_RMN), as well as to be able to correct/calculate the time deviation of a counter that is clocked by the metered clocks.

Note: Refer to the clocking chapter for the number of instances of the CMU in this device.

29.1.1 Main features

- CLKMT0_RMN frequency measurement with CLKMN0_RMT as the reference clock.
- CLKMN0_RMT monitoring with respect to CLKMT0_RMN/n clock.
- Upper or lower frequency boundary monitoring of CLKMN1 with respect to CLKMT0_RMN / 4.
- Event generation for various failures detected inside monitoring unit.

29.2 Block diagram

The block diagram of the CMU module(s) is shown in the clocking chapter of this reference manual.

29.3 Signals

Table 409 describes the signals on the boundary of the CMU (in alphabetical order).

Table 409. Signal description

Signal	I/O	Description
CLKMN0_RMT	I	Monitored Clock Signal 0/Metered Clock Signal Reference: Receives a clock signal that the CMU compares to a specified low-limit frequency to determine whether the frequency of the clock signal is greater than the specified limit. Also provides a reference clock signal for all metered clock signals.
CLKMN1	I	Monitored Clock Signal 1: Receives a clock signal that the CMU compares to specified low-limit and high-limit frequencies to determine whether the frequency of the clock signal is between the specified limits.
CLKMT0_RMN	I	Metered Clock Signal 0/Monitored Clock Signal Reference: Receives a clock signal that the CMU measures against a reference clock frequency. Also provides a reference clock signal for all monitored clock signals.

Note: Refer to the clocking chapter for device specific clock sources of each CMU.

29.4 Memory map and register definition

This section describes in address order all the CMU registers. Each description includes a standard register diagram with an associated figure number. The CMU memory map is listed in *Table 410*.

Table 410. CMU memory map

Address offset	Register	Location
0x0000	CMU Control Status Register (CMU_CSR)	*Section 29.4.1.1*
0x0004	CMU Frequency Display Register (CMU_FDR)	*Section 29.4.1.2*
0x0008	CMU High Frequency Reference Register CLKMN1 (CMU_HFREFR)	*Section 29.4.1.3*
0x000C	CMU Low Frequency Reference Register CLKMN1 (CMU_LFREFR)	*Section 29.4.1.4*
0x0010	CMU Interrupt Status Register (CMU_ISR)	*Section 29.4.1.5*
0x0014–0x0017	Reserved	
0x0018	CMU Measurement Duration Register (CMU_MDR)	*Section 29.4.1.6*

Note: See Clocking chapter for register and field availability details.

29.4.1 Register descriptions

29.4.1.1 CMU Control Status Register (CMU_CSR)

Offset: 0x0000 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	SFM(1)	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	CKSEL1(1)		0	0	0	0	0	RCDIV(1)		CME
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. Not all CMU blocks will utilize this feature. Refer to clocking chapter for device specific CMU implementation details.

Figure 301. CMU Control Status Register (CMU_CSR)

Table 411. CMU_CSR field descriptions

Field	Description
8 SFM	Start Frequency Measure The software can only set this bit to start a clock frequency measure. It is reset by hardware when the measure is ready in the CMU_FDR register. CMU_MDR[MD] must be written before enabling frequency measurement (CMU_CSR[SFM] = 1). Do not write CMU_MDR[MD] while CMU_CSR[SFM] = 1. Software should only read the value of CMU_FDR[FD] bits once the hardware has cleared the CMU_CSR[SFM] bit. 0 Frequency measurement is completed or not yet started. 1 Frequency measurement is not completed.
22:23 CKSEL1	Frequency measure clock selection bit CKSEL1 selects the clock to be measured by the frequency meter. This only effects CMU instances that utilizes clock metering. Not all CMU blocks will utilize this feature. See the “Clocking” chapter for device specific CMU implementation details. 00 CLKMT0_RMN is selected. 01 Reserved 10 Reserved. 11 CLKMT0_RMN is selected.

Table 411. CMU_CSR field descriptions (continued)

Field	Description
29:30 RCDIV	CLKMT0_RMN division factor These bits specify the CLKMT0_RMN division factor. The output clock frequency is f_{CLKMT0_RMN} divided by the factor 2^{RCDIV}. This output clock is used as reference clock to compare with CLKMN0_RMT for crystal clock monitor feature. Ensure clock source for CLKMN0_RMT is off before writing CMU_CSR[RCDIV] otherwise the operation can lead to a spurious OLR event. The clock division coding is as follows: 00 CLKMT0_RMN divided by 1 (No division). 01 CLKMT0_RMN divided by 2. 10 CLKMT0_RMN divided by 4. 11 CLKMT0_RMN divided by 8.
31 CME	CLKMN1 monitor enable CMU_HFREF and CMU_LFREF registers must be written before enabling the clock monitoring (CMU_CSR[CME] =1) 0 CLKMN1 monitor is disabled. 1 CLKMN1 monitor is enabled.

29.4.1.2 CMU Frequency Display Register (CMU_FDR)

The FDR is used to determine the measured frequency being monitored by the CMU.

Offset: 0x0004 Access: User read-only

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	FD			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	FD															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 302. CMU Frequency Display Register (CMU_FDR)

Table 412. CMU_FDR field descriptions

Field	Description
12:31 FD	Measured frequency bits This register displays the measured frequency (f_{sel}) with respect to the reference clock (f_{CLKMN0_RMT}). The measured value is given by the following formula: $f_{sel} = (f_{CLKMN0_RMT} \times CMU_MDR[MD]) / CMU_FDR[FD]$

29.4.1.3 CMU High Frequency Reference Register CLKMN1 (CMU_HFREFR)

The HFREFR is configured for the high frequency reference that the CMU will use for comparing against the monitored clock. *Figure 303* and *Table 413* show the CMU_HFREFR register.

Offset: 0x0008 Access: User read/write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	HFREF											
W																
Reset	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1

Figure 303. CMU High Frequency Reference Register CLKMN1 (CMU_HFREFR)

Table 413. CMU_HFREFR field descriptions

Field	Description
20:31 HFREF	High Frequency reference value These bits determine the high reference value for the CLKMN1 frequency. The reference value is given by: (HFREF / 16) × (f_{CLKMT0_RMN} / 4)

29.4.1.4 CMU Low Frequency Reference Register CLKMN1 (CMU_LFREFR)

The LFREFR is configured for the low frequency reference that the CMU will use for comparing against the monitored clock. *Figure 304* and *Table 414* show the CMU_LFREFR register.

Offset 0x000C Access: User read/write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	LFREF											
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 304. CMU Low Frequency Reference Register CLKMN1 (CMU_LFREFR)

Table 414. CMU_LFREFR field descriptions

Field	Description
20:31 LFREF	Low Frequency reference value These bits determine the low reference value for the CLKMN1 frequency. The reference value is given by: (LFREF / 16) × (f_{CLKMT0_RMN} / 4)

29.4.1.5 CMU Interrupt Status Register (CMU_ISR)

Offset: 0x0010 Access: User read/write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	FHHI	FLLI	OLRI(1)
W													—(2)	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. Not all CMU blocks will utilize this feature. See Clocking chapter for device specific CMU implementation details.
2. Field is read only.

Figure 305. CMU Interrupt Status Register (CMU_ISR)

Table 415. CMU_ISR field descriptions

Field	Description
29 FHHI	CLKMN1 frequency higher than high reference event status This bit is set by hardware when CLKMN1 frequency becomes higher than HFREF value and CLKMN1 is 'ON' as signaled by the MC_ME. It can be cleared by software by writing '1'. 0 No FHH event 1 FHH event occurred
30 FLLI	CLKMN1 frequency less than low reference event status This bit is set by hardware when CLKMN1 frequency becomes lower than LFREF value and CLKMN1 is 'ON' as signaled by the MC_ME. It can be cleared by software by writing '1'. 0 No FLL event 1 FLL event occurred
31 OLRI	Oscillator frequency less than f_{CLKMT0_RMN} / 2^{RCDIV} event status This bit is set by hardware when the f_{CLKMN0_RMT} is less than f_{CLKMT0_RMN} / 2^{RCDIV} frequency and CLKMN0_RMT is 'ON' as signaled by the MC_ME. It can be cleared by software by writing '1'. 0 No OLR event 1 OLR event occurred

Note: All the flags in CMU_ISR are set asynchronously. This register must be read only after a fault event is triggered by the CMU and mapped on FCCU.

Otherwise, the read access on this register may fetch an incorrect value.

Before entering low-power stop modes, all clock frequency measurements in CMU should be disabled. Failing to do so may result in spurious interrupts.

29.4.1.6 CMU Measurement Duration Register (CMU_MDR)

Offset: 0x0018 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	MD			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	MD															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 306. CMU Measurement Duration Register (CMU_MDR)

Table 416. CMU_MDR field descriptions

Field	Description
12:31 MD	Measurement duration bits This field displays the measurement duration in terms of selected clock (CLKMT0_RMN) cycles. This value is loaded in the frequency meter down-counter. The down-counter starts counting when CMU_CSR [SFM] = 1.

29.5 Functional description

This section describes the functionality of the CMU.

29.5.1 Frequency meter

The purpose of frequency meter is to evaluate the deviation from the nominal metered source (such as, CLKMT0_RMN) frequencies. This in turn allows either recalibration of these clocks or other timing corrections. Programming of the CMU_CSR[CKSEL1] field is used to select one of metered clocks from a multiplexer that drives a simple Frequency Meter (see the CMU Block Diagram in the Clocking chapter). The reference clock for the Frequency Meter is the CLKMN0_RMT signal. The measurement starts when CMU_CSR[SFM] = 1. The measurement duration is given by the contents of CMU_MDR[MD] in terms of number of clock cycles of the selected metered clock. The CMU_CSR[SFM] bit is cleared by hardware once the frequency measurement is complete and the count is loaded in CMU_FDR[FD]. The frequency of the selected clock (f_{sel}) can be derived from the value loaded in the CMU_FDR as shown in the following equation:

Equation 13 $f_{sel} = (f_{CLKMN0_RMT} \times MDR[MD]) / FDR[FD]$

29.5.2 CLKMN0_RMT supervisor

If frequency of CLKMN0_RMT is smaller than frequency of CLKMT0_RMN divided by $2^{CMU_CSR[RCDIV]}$ and CLKMN0_RMT is ‘ON’ as signaled by the MC_ME, then:

- The CMU writes 1 to CMU_ISR[OLRI].
- The CMU asserts the OLR signal.

29.5.3 CLKMN1 supervisor

The frequency of CLKMN1(f_{CLKMN1}) can be monitored by programming CMU_CSR[CME] = 1. CLKMN1 monitoring starts as soon as CMU_CSR[CME] = 1. This monitor can be disabled at any time by programming CMU_CSR[CME] = 0.

If fCLKMN1 is greater than the reference value determined by field CMU_HFREFR[HFREF] and CLKMN1 is ‘ON’ as signaled by the MC_ME, then:

- The CMU writes 1 to CMU_ISR[FHHI].
- The CMU asserts the FHH signal.

Note: An example of determining the HFREFActual value is as follows. Assume a fCLKMT0_RMN = 16 MHz with an accuracy of +/-5 %. In order to monitor fCLKMN1 = 200 MHz, the ideal HFREFIdeal = 800. The actual HFREF value will be 842 when the accuracy is taken into consideration (HFREFActual = (HFREFIdeal + 2) × 1.05).

If fCLKMN1 is lower than CMU_LFREFR[LFREF] and CLKMN1 is 'ON' as signaled by the MC_ME, then:

- The CMU writes 1 to CMU_ISR[FLLI].
- The CMU asserts the FLL signal.

Note: The low reference value must be programmed considering the following factors:

- *Two cycles of f_{CLKMN1} which is the built-in tolerance of the monitor implementation*
- *Frequency variation % of f_{CLKMT0_RMN} across PVT*

For example, assuming f_{CLKMT0_RMN} is 16 MHz with +/-5 % variation, in order to monitor the f_{CLKMN1} (200 MHz), the ideal $LFREF_{Ideal}$ is 800. The adjusted value considering the tolerances will be 758 ($LFREF_{Actual}$ = ($LFREF_{Ideal}$ - 2) x 0.95).

The minimum value that can be programmed in CMU_LFREFR[LFREF] is 3. The minimum frequency that can be monitored using the CLKMN1 supervisor must be >= f_{CLKMT0_RMN} / 16.

30 Clock Generation Module (MC_CGM)

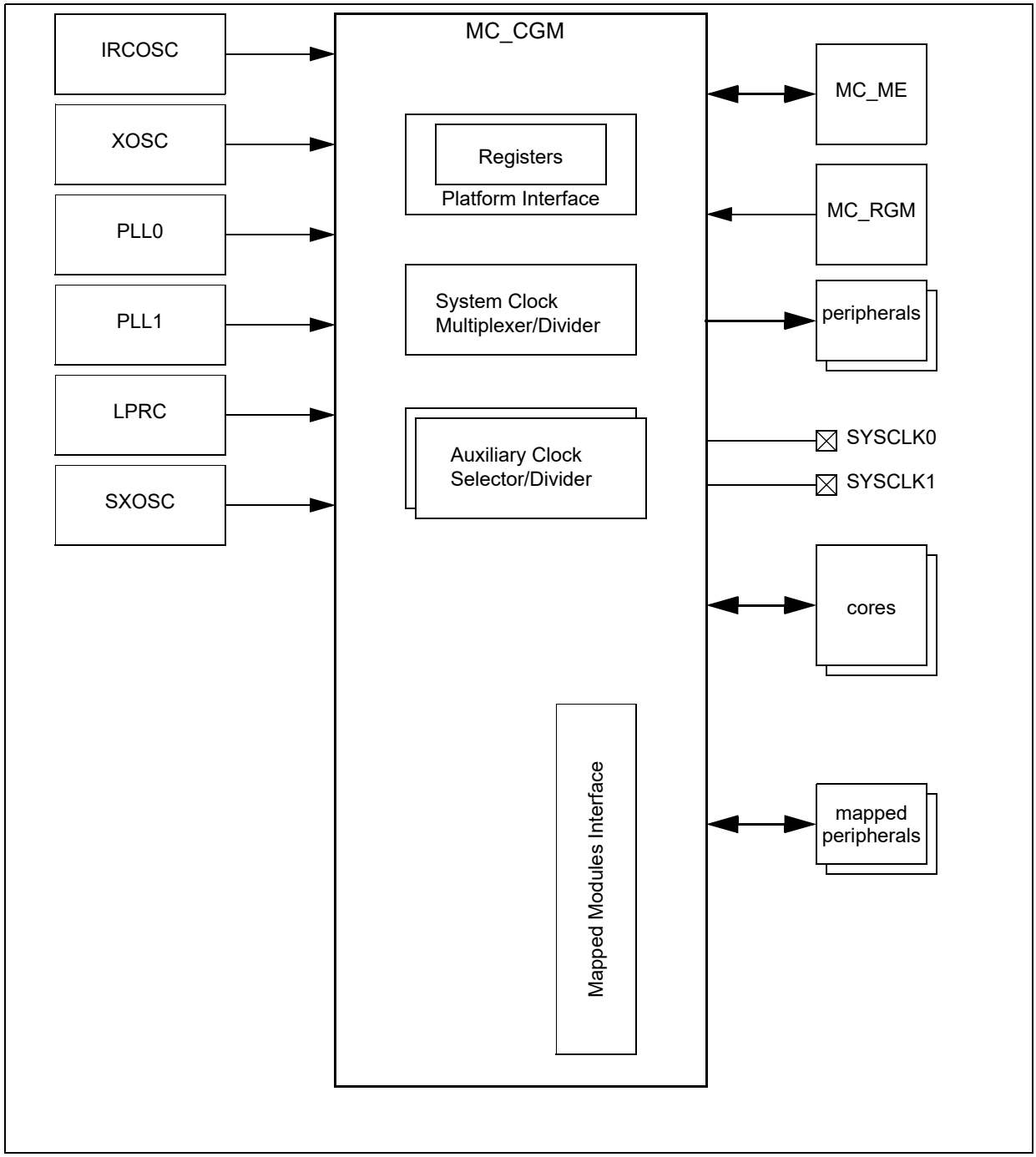
30.1 Introduction

30.1.1 Overview

The clock generation module (MC_CGM) generates reference clocks for all the modules on the chip. The MC_CGM selects one of the system clock sources to supply the system clock. The MC_ME controls the system clock selection (see the MC_ME chapter for more details). A set of MC_CGM registers controls the clock dividers which are used for divided system and peripheral clock generation. The memory spaces of system and peripheral clock sources which have addressable memory spaces are accessed through the MC_CGM memory space.

Figure 307 shows the MC_CGM block diagram.

Figure 307. MC_CGM block diagram



30.1.2 Features

The MC_CGM includes the following features:

- generates system and peripheral clocks
- selects and enables/disables the system clock supply from system clock sources according to MC_ME control
- performs progressive system clock frequency change depending on MC_ME mode configuration
- contains a set of registers to control clock dividers for divided clock generation
- supports multiple clock sources and maps their address spaces to its memory map
- guarantees glitch-less clock transitions when changing the system clock selection
- supports 8, 16, and 32-bit wide read/write accesses

30.2 External signal description

The MC_CGM delivers output clocks to the SYSCLK0 and SYSCLK1 pins for off-chip use or observation or both.

30.3 Memory map and register definition

Unless otherwise noted, all registers may be accessed as 32-bit words, 16-bit half-words, or 8-bit bytes. The bytes are ordered according to big endian. For example, the CGM_PCS_DIVC1 register RATE bits may be accessed as a word at address offset 0x0104, as a half-word at address offset 0x0106, or as a byte at address offset 0x0107.

Note: Any access to unused registers as well as write accesses to read-only registers will:

- *not change register content*
- *cause a transfer error*

Table 417. MC_CGM memory map

Address offset	Register	Location
0x0080	Auxiliary Clock 5 Cascaded Divider 0 Configuration Register (CGM_AC5_CDC0)	*Section 30.3.1.1*
0x0084	Auxiliary Clock 5 Cascaded Divider 1 Configuration Register (CGM_AC5_CDC1)	*Section 30.3.1.2*
0x0088	Auxiliary Clock 5 Cascaded Divider 2 Configuration Register (CGM_AC5_CDC2)	*Section 30.3.1.3*
0x0090	Auxiliary Clock 5 Cascaded Divider 10 Configuration Register (CGM_AC5_CDC10)	*Section 30.3.1.4*
0x0094	Auxiliary Clock 5 Cascaded Divider 11 Configuration Register (CGM_AC5_CDC11)	*Section 30.3.1.5*
0x0098	Auxiliary Clock 5 Cascaded Divider 12 Configuration (Register CGM_AC5_CDC12)	*Section 30.3.1.6*
0x009C	Auxiliary Clock 5 Cascaded Divider 13 Configuration Register (CGM_AC5_CDC13)	*Section 30.3.1.7*
0x00A0	Auxiliary Clock 5 Cascaded Divider 20 Configuration Register (CGM_AC5_CDC20)	*Section 30.3.1.8*
0x00A4	Auxiliary Clock 5 Cascaded Divider 21 Configuration Register (CGM_AC5_CDC21)	*Section 30.3.1.9*
0x00A8	Auxiliary Clock 5 Cascaded Divider 22 Configuration Register (CGM_AC5_CDC22)	*Section 30.3.1.10*
0x00AC	Auxiliary Clock 5 Cascaded Divider 23 Configuration Register (CGM_AC5_CDC23)	*Section 30.3.1.11*

Table 417. MC_CGM memory map (continued)

Address offset	Register	Location
0x0100	PCS Switch Duration Register (CGM_PCS_SDUR)	*Section 30.3.1.12*
0x0104	PCS Divider Change Register 1 (CGM_PCS_DIVC1)	*Section 30.3.1.13*
0x0108	PCS Divider End Register 1 (CGM_PCS_DIVE1)	*Section 30.3.1.14*
0x010C	PCS Divider Start Register 1 (CGM_PCS_DIVS1)	*Section 30.3.1.15*
0x0110	PCS Divider Change Register 2 (CGM_PCS_DIVC2)	*Section 30.3.1.16*
0x0114	PCS Divider End Register 2 (CGM_PCS_DIVE2)	*Section 30.3.1.17*
0x0118	PCS Divider Start Register 2 (CGM_PCS_DIVS2)	*Section 30.3.1.18*
0x0128	PCS Divider Change Register 4 (CGM_PCS_DIVC4)	*Section 30.3.1.19*
0x012C	PCS Divider End Register 4 (CGM_PCS_DIVE4)	*Section 30.3.1.20*
0x0130	PCS Divider Start Register 4 (CGM_PCS_DIVS4)	*Section 30.3.1.21*
0x01D0	System Clock Divider Ratio Change Register (CGM_SC_DIV_RC)	*Section 30.3.1.22*
0x01D4	Divider Update Type Register (CGM_DIV_UPD_TYPE)	*Section 30.3.1.23*
0x01D8	Divider Update Trigger Register (CGM_DIV_UPD_TRIG)	*Section 30.3.1.24*
0x01DC	Divider Update Status Register (CGM_DIV_UPD_STAT)	*Section 30.3.1.25*
0x01E4	System Clock Select Status Register (CGM_SC_SS)	*Section 30.3.1.26*
0x01E8	System Clock Divider 0 Configuration Register (CGM_SC_DC0)	*Section 30.3.1.27*
0x01EC	System Clock Divider 1 Configuration Register (CGM_SC_DC1)	*Section 30.3.1.28*
0x01F0	System Clock Divider 2 Configuration Register (CGM_SC_DC2)	*Section 30.3.1.29*
0x01F4	System Clock Divider 3 Configuration Register (CGM_SC_DC3)	*Section 30.3.1.30*
0x01F8	System Clock Divider 4 Configuration Register (CGM_SC_DC4)	*Section 30.3.1.31*
0x0200	Auxiliary Clock 0 Select Control Register (CGM_AC0_SC)	*Section 30.3.1.32*
0x0204	Auxiliary Clock 0 Select Status Register (CGM_AC0_SS)	*Section 30.3.1.33*
0x0208	Auxiliary Clock 0 Divider 0 Configuration Register (CGM_AC0_DC0)	*Section 30.3.1.34*
0x020C	Auxiliary Clock 0 Divider 1 Configuration Register (CGM_AC0_DC1)	*Section 30.3.1.35*
0x0210	Auxiliary Clock 0 Divider 2 Configuration Register (CGM_AC0_DC2)	*Section 30.3.1.36*
0x0214	Auxiliary Clock 0 Divider 3 Configuration Register (CGM_AC0_DC3)	*Section 30.3.1.37*
0x0220	Auxiliary Clock 1 Select Control Register (CGM_AC1_SC)	*Section 30.3.1.48*
0x0224	Auxiliary Clock 1 Select Status Register (CGM_AC1_SS)	*Section 30.3.1.39*
0x0228	Auxiliary Clock 1 Divider 0 Configuration Register (CGM_AC1_DC0)	*Section 30.3.1.40*
0x0248	Auxiliary Clock 2 Divider 0 Configuration Register (CGM_AC2_DC0)	*Section 30.3.1.41*
0x024C	Auxiliary Clock 2 Divider 1 Configuration Register (CGM_AC2_DC1)	*Section 30.3.1.42*
0x0260	Auxiliary Clock 3 Select Control Register (CGM_AC3_SC)	*Section 30.3.1.43*
0x0264	Auxiliary Clock 3 Select Status Register (CGM_AC3_SS)	*Section 30.3.1.44*
0x0280	Auxiliary Clock 4 Select Control Register (CGM_AC4_SC)	*Section 30.3.1.45*

Table 417. MC_CGM memory map (continued)

Address offset	Register	Location
0x0284	Auxiliary Clock 4 Select Status Register (CGM_AC4_SS)	*Section 30.3.1.46*
0x02A8	Auxiliary Clock 5 Divider 0 Configuration Register (CGM_AC5_DC0)	*Section 30.3.1.47*
0x02AC	Auxiliary Clock 5 Divider 1 Configuration Register (CGM_AC5_DC1)	*Section 30.3.1.48*
0x02B0	Auxiliary Clock 5 Divider 2 Configuration Register (CGM_AC5_DC2)	*Section 30.3.1.49*
0x02C0	Auxiliary Clock 6 Select Control Register (CGM_AC6_SC)	*Section 30.3.1.50*
0x02C4	Auxiliary Clock 6 Select Status Register (CGM_AC6_SS)	*Section 30.3.1.51*
0x02C8	Auxiliary Clock 6 Divider 0 Configuration Register (CGM_AC6_DC0)	*Section 30.3.1.52*
0x02E0	Auxiliary Clock 7 Select Control Register (CGM_AC7_SC)	*Section 30.3.1.53*
0x02E4	Auxiliary Clock 7 Select Status Register (CGM_AC7_SS)	*Section 30.3.1.54*
0x02E8	Auxiliary Clock 7 Divider 0 Configuration Register (CGM_AC7_DC0)	*Section 30.3.1.55*
0x0300	Auxiliary Clock 8 Select Control Register (CGM_AC8_SC)	*Section 30.3.1.56*
0x0304	Auxiliary Clock 8 Select Status Register (CGM_AC8_SS)	*Section 30.3.1.57*
0x0308	Auxiliary Clock 8 Divider 0 Configuration Register (CGM_AC8_DC0)	*Section 30.3.1.58*
0x0320	Auxiliary Clock 9 Select Control Register (CGM_AC9_SC)	*Section 30.3.1.59*
0x0324	Auxiliary Clock 9 Select Status Register (CGM_AC9_SS)	*Section 30.3.1.60*
0x0328	Auxiliary Clock 9 Divider 0 Configuration Register (CGM_AC9_DC0)	*Section 30.3.1.61*
0x0360	Auxiliary Clock 11 Select Control Register (CGM_AC11_SC)	*Section 30.3.1.62*
0x0364	Auxiliary Clock 11 Select Status Register (CGM_AC11_SS)	*Section 30.3.1.63*
0x0368	Auxiliary Clock 11 Divider 0 Configuration Register (CGM_AC11_DC0)	*Section 30.3.1.64*
0x0380	Auxiliary Clock 12 Select Control Register (CGM_AC12_SC)	*Section 30.3.1.65*
0x0384	Auxiliary Clock 12 Select Status Register (CGM_AC12_SS)	*Section 30.3.1.66*
0x0388	Auxiliary Clock 12 Divider 0 Configuration Register (CGM_AC12_DC0)	*Section 30.3.1.67*
0x038C	Auxiliary Clock 12 Divider 1 Configuration Register (CGM_AC12_DC1)	*Section 30.3.1.68*
0x0390	Auxiliary Clock 12 Divider 2 Configuration Register (CGM_AC12_DC2)	*Section 30.3.1.69*
0x0394	Auxiliary Clock 12 Divider 3 Configuration Register (CGM_AC12_DC3)	*Section 30.3.1.70*
0x0398	Auxiliary Clock 12 Divider 4 Configuration Register (CGM_AC12_DC4)	*Section 30.3.1.71*

30.3.1 Register descriptions

Unless otherwise noted, all registers may be accessed as 32-bit words, 16-bit half-words, or 8-bit bytes. The bytes are ordered according to big endian. For example, the CGM_PCS_DIVC1 register RATE bits may be accessed as a word at address offset 0x0104, as a half-word at address offset 0x0106, or as a byte at address offset 0x0107.

30.3.1.1 Auxiliary Clock 5 Cascaded Divider 0 Configuration Register (CGM_AC5_CDC0)

This register controls auxiliary clock 5 cascaded divider 0.

Note: Byte and half-word write accesses are not allowed for this register. Such an access will not result in an exception; however, the value will not be loaded with the new value.

Address: 0x0080 Access: User read/write, Supervisor read/write, Test read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	DE	0	0	0	0	0	DIV									
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

Figure 308. Auxiliary Clock 5 Cascaded Divider 0 Configuration Register (CGM_AC5_CDC0)

Table 418. CGM_AC5_CDC0 field descriptions

Field	Description
0 DE	Divider Enable. 0 Disable auxiliary clock 5 cascaded divider 0 1 Enable auxiliary clock 5 cascaded divider 0
6:15 DIV	Divider Division Value The resultant PSI5_S_TS_CLK will have a period '(DIV + 1) × (CGM_AC5_DC2[DIV] + 1)' times that of auxiliary clock 5. If DE is set to 0 (divider is disabled), any write access to the DIV field is ignored and the PSI5_S_TS_CLK remains disabled.

30.3.1.2 Auxiliary Clock 5 Cascaded Divider 1 Configuration Register (CGM_AC5_CDC1)

This register controls system clock divider 1.

Note: Byte and half-word write accesses are not allowed for this register. Such an access will not result in an exception; however, the value will not be loaded with the new value.

Address: 0x0084 Access: User read/write, Supervisor read/write, Test read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	DE	0	0	0	0	0	DIV									
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

Figure 309. Auxiliary Clock 5 Cascaded Divider 1 Configuration Register (CGM_AC5_CDC1)

Table 419. CGM_AC5_CDC1 field descriptions

Field	Description
0 DE	Divider Enable. 0 Disable auxiliary clock 5 cascaded divider 1 1 Enable auxiliary clock 5 cascaded divider 1
6:15 DIV	Divider Division Value The resultant PSI5_S_DDSR_CLK will have a period '(DIV + 1) × (CGM_AC5_DC2[DIV] + 1)' times that of auxiliary clock 5. If DE is set to 0 (divider is disabled), any write access to the DIV field is ignored and the PSI5_S_DDSR_CLK remains disabled.

30.3.1.3 Auxiliary Clock 5 Cascaded Divider 2 Configuration Register (CGM_AC5_CDC2)

This register controls system clock divider 2.

Note: Byte and half-word write accesses are not allowed for this register. Such an access will not result in an exception; however, the value will not be loaded with the new value.

Address: 0x0088 Access: User read/write, Supervisor read/write, Test read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	DE	0	0	0	0	0	DIV									
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

Figure 310. Auxiliary Clock 5 Cascaded Divider 2 Configuration Register (CGM_AC5_CDC2)

Table 420. CGM_AC5_CDC2 field descriptions

Field	Description
0 DE	Divider Enable. 0 Disable auxiliary clock 5 cascaded divider 2 1 Enable auxiliary clock 5 cascaded divider 2
6:15 DIV	Divider Division Value The resultant PSI5_S_WD_CLK will have a period '(DIV + 1) × (CGM_AC5_DC2[DIV] + 1)' times that of auxiliary clock 5. If DE is set to 0 (divider is disabled), any write access to the DIV field is ignored and the PSI5_S_WD_CLK remains disabled.

30.3.1.4 Auxiliary Clock 5 Cascaded Divider 10 Configuration Register (CGM_AC5_CDC10)

This register controls auxiliary clock 5 cascaded divider 10.

Note: Byte and half-word write accesses are not allowed for this register. Such an access will not result in an exception, however the value will not be loaded with the new value.

Address: 0x0090 Access: User read/write, Supervisor read/write, Test read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	DE	0	0	0	0	0	DIV									
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

Figure 311. Auxiliary Clock 5 Cascaded Divider 10 Configuration Register (CGM_AC5_CDC10)

Table 421. CGM_AC5_CDC10 field descriptions

Field	Description
0 DE	Divider Enable 0 Disable auxiliary clock 5 cascaded divider 10 1 Enable auxiliary clock 5 cascaded divider 10
6:15 DIV	Divider Division Value The resultant PSI5_S_DDSR0_CLK will have a period '(DIV + 1) × (CGM_AC5_CDC1[DIV] + 1) × (CGM_AC5_DC2[DIV] + 1)' times that of auxiliary clock 5. If DE is set to 0 (divider is disabled), any write access to the DIV field is ignored and the PSI5_S_DDSR0_CLK remains disabled.

30.3.1.5 Auxiliary Clock 5 Cascaded Divider 11 Configuration Register (CGM_AC5_CDC11)

This register controls auxiliary clock 5 cascaded divider 11.

Note: Byte and half-word write accesses are not allowed for this register. Such an access will not result in an exception, however the value will not be loaded with the new value.

Address: 0x0094 Access: User read/write, Supervisor read/write, Test read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	DE	0	0	0	0	0	DIV									
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

Figure 312. Auxiliary Clock 5 Cascaded Divider 11 Configuration Register (CGM_AC5_CDC11)

Table 422. CGM_AC5_CDC11 field descriptions

Field	Description
0 DE	Divider Enable 0 Disable auxiliary clock 5 cascaded divider 11 1 Enable auxiliary clock 5 cascaded divider 11
6:15 DIV	Divider Division Value The resultant PSI5_S_DDSR1_CLK will have a period '(DIV + 1) × (CGM_AC5_CDC1[DIV] + 1) × (CGM_AC5_DC2[DIV] + 1)' times that of auxiliary clock 5. If DE is set to 0 (divider is disabled), any write access to the DIV field is ignored and the PSI5_S_DDSR1_CLK remains disabled.

30.3.1.6 Auxiliary Clock 5 Cascaded Divider 12 Configuration Register (CGM_AC5_CDC12)

This register controls auxiliary clock 5 cascaded divider 12.

Note: Byte and half-word write accesses are not allowed for this register. Such an access will not result in an exception, however the value will not be loaded with the new value.

Address: 0x0098 Access: User read/write, Supervisor read/write, Test read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	DE	0	0	0	0	0	DIV									
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

Figure 313. Auxiliary Clock 5 Cascaded Divider 12 Configuration Register (CGM_AC5_CDC12)

Table 423. CGM_AC5_CDC12 field descriptions

Field	Description
0 DE	Divider Enable 0 Disable auxiliary clock 5 cascaded divider 12 1 Enable auxiliary clock 5 cascaded divider 12
6:15 DIV	Divider Division Value The resultant PSI5_S_DDSR2_CLK will have a period '(DIV + 1) × (CGM_AC5_CDC1[DIV] + 1) × (CGM_AC5_DC2[DIV] + 1)' times that of auxiliary clock 5. If DE is set to 0 (divider is disabled), any write access to the DIV field is ignored and the PSI5_S_DDSR2_CLK remains disabled.

30.3.1.7 Auxiliary Clock 5 Cascaded Divider 13 Configuration Register (CGM_AC5_CDC13)

This register controls auxiliary clock 5 cascaded divider 13.

Note: Byte and half-word write accesses are not allowed for this register. Such an access will not result in an exception, however the value will not be loaded with the new value.

Address: 0x009C Access: User read/write, Supervisor read/write, Test read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	DE	0	0	0	0	0	DIV									
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

Figure 314. Auxiliary Clock 5 Cascaded Divider 13 Configuration Register (CGM_AC5_CDC13)

Table 424. CGM_AC5_CDC13 field descriptions

Field	Description
0 DE	Divider Enable 0 Disable auxiliary clock 5 cascaded divider 13 1 Enable auxiliary clock 5 cascaded divider 13
6:15 DIV	Divider Division Value The resultant PSI5_S_DDSR3_CLK will have a period '(DIV + 1) × (CGM_AC5_CDC1[DIV] + 1) × (CGM_AC5_DC2[DIV] + 1)' times that of auxiliary clock 5. If DE is set to 0 (divider is disabled), any write access to the DIV field is ignored and the PSI5_S_DDSR3_CLK remains disabled.

30.3.1.8 Auxiliary Clock 5 Cascaded Divider 20 Configuration Register (CGM_AC5_CDC20)

This register controls auxiliary clock 5 cascaded divider 20.

Note: Byte and half-word write accesses are not allowed for this register. Such an access will not result in an exception, however the value will not be loaded with the new value.

Address: 0x00A0 Access: User read/write, Supervisor read/write, Test read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	DE	0	0	0	0	0	DIV									
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

Figure 315. Auxiliary Clock 5 Cascaded Divider 20 Configuration Register (CGM_AC5_CDC20)

Table 425. CGM_AC5_CDC20 field descriptions

Field	Description
0 DE	Divider Enable 0 Disable auxiliary clock 5 cascaded divider 20 1 Enable auxiliary clock 5 cascaded divider 20
6:15 DIV	Divider Division Value The resultant PSI5_S_WD0_CLK will have a period '(DIV + 1) × (CGM_AC5_CDC2[DIV] + 1) × (CGM_AC5_DC2[DIV] + 1)' times that of auxiliary clock 5. If DE is set to 0 (divider is disabled), any write access to the DIV field is ignored and the PSI5_S_WD0_CLK remains disabled.

30.3.1.9 Auxiliary Clock 5 Cascaded Divider 21 Configuration Register (CGM_AC5_CDC21)

This register controls auxiliary clock 5 cascaded divider 21.

Note: Byte and half-word write accesses are not allowed for this register. Such an access will not result in an exception, however the value will not be loaded with the new value.

Address: 0x00A4 Access: User read/write, Supervisor read/write, Test read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	DE	0	0	0	0	0	DIV									
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

Figure 316. Auxiliary Clock 5 Cascaded Divider 21 Configuration Register (CGM_AC5_CDC21)

Table 426. CGM_AC5_CDC21 field descriptions

Field	Description
0 DE	Divider Enable 0 Disable auxiliary clock 5 cascaded divider 21 1 Enable auxiliary clock 5 cascaded divider 21
6:15 DIV	Divider Division Value The resultant PSI5_S_WD1_CLK will have a period '(DIV + 1) × (CGM_AC5_CDC2[DIV] + 1) × (CGM_AC5_DC2[DIV] + 1)' times that of auxiliary clock 5. If DE is set to 0 (divider is disabled), any write access to the DIV field is ignored and the PSI5_S_WD1_CLK remains disabled.

30.3.1.10 Auxiliary Clock 5 Cascaded Divider 22 Configuration Register (CGM_AC5_CDC22)

This register controls auxiliary clock 5 cascaded divider 22.

Note: Byte and half-word write accesses are not allowed for this register. Such an access will not result in an exception, however the value will not be loaded with the new value.

Address: 0x00A8 Access: User read/write, Supervisor read/write, Test read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	DE	0	0	0	0	0	DIV									
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

Figure 317. Auxiliary Clock 5 Cascaded Divider 22 Configuration Register (CGM_AC5_CDC22)

Table 427. CGM_AC5_CDC22 field descriptions

Field	Description
0 DE	Divider Enable 0 Disable auxiliary clock 5 cascaded divider 22 1 Enable auxiliary clock 5 cascaded divider 2
6:15 DIV	Divider Division Value The resultant PSI5_S_WD2_CLK will have a period '(DIV + 1) × (CGM_AC5_CDC2[DIV] + 1) × (CGM_AC5_DC2[DIV] + 1)' times that of auxiliary clock 5. If DE is set to 0 (divider is disabled), any write access to the DIV field is ignored and the PSI5_S_WD2_CLK remains disabled.

30.3.1.11 Auxiliary Clock 5 Cascaded Divider 23 Configuration Register (CGM_AC5_CDC23)

This register controls auxiliary clock 5 cascaded divider 23.

Note: Byte and half-word write accesses are not allowed for this register. Such an access will not result in an exception, however the value will not be loaded with the new value.

Address: 0x00AC Access: User read/write, Supervisor read/write, Test read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	DE	0	0	0	0	0	DIV									
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

Figure 318. Auxiliary Clock 5 Cascaded Divider 23 Configuration Register (CGM_AC5_CDC23)

Table 428. CGM_AC5_CDC23 field descriptions

Field	Description
0 DE	Divider Enable 0 Disable auxiliary clock 5 cascaded divider 23 1 Enable auxiliary clock 5 cascaded divider 23
6:15 DIV	Divider Division Value The resultant PSI5_S_WD3_CLK will have a period '(DIV + 1) × (CGM_AC5_CDC2[DIV] + 1) × (CGM_AC5_DC2[DIV] + 1)' times that of auxiliary clock 5. If DE is set to 0 (divider is disabled), any write access to the DIV field is ignored and the PSI5_S_WD3_CLK remains disabled.

30.3.1.12 PCS Switch Duration Register (CGM_PCS_SDUR)

This register contains the progressive system clock switching duration for each step. See *Section 30.4.1.2: Progressive system clock switching*, for details on how to set this value. This register is reset only on a power-on reset.

Address: 0x0100 Access: User read/write, Supervisor read/write, Test read/write

	0	1	2	3	4	5	6	7
R W	SDUR							
POR	0	0	0	0	0	0	0	0

Figure 319. PCS Switch Duration Register (CGM_PCS_SDUR)

Table 429. CGM_PCS_SDUR field descriptions

Field	Description
0:7 SDUR	Switch Duration This value defines the duration of one PCS clock switch step in terms of IRCOSC cycles.

30.3.1.13 PCS Divider Change Register 1 (CGM_PCS_DIVC1)

This register defines the rate of frequency change and initial change value for the progressive system clock switching when switching the system clock source to or from the XOSC on ramp-up and ramp-down, respectively. See *Section 30.4.1.2: Progressive system clock switching*, for details on how to set these values. This register is reset only on a power-on reset.

Note: Byte write accesses are not allowed for the INIT field of this register. Such an access will not result in an exception; however, the value will not be loaded with the new value.

Address: 0x0104 Access: User read/write, Supervisor read/write, Test read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	INIT															
W																
POR	0	0	0	0	0	0	1	1	1	1	1	0	0	1	1	1

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	RATE							
W																
POR	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 320. PCS Divider Change Register 1 (CGM_PCS_DIVC1)

Table 430. CGM_PCS_DIVC1 field descriptions

Field	Description
0:15 INIT	Divider Change Initial Value This is initial change value of the clock divider for the clock ramp-up phase when switching to the XOSC.
24:31 RATE	Divider Change Rate This value controls the change value of the clock divider for the clock ramp-up and ramp-down phase when switching to/from the XOSC.

30.3.1.14 PCS Divider End Register 1 (CGM_PCS_DIVE1)

This register defines the final division value for the progressive system clock switching when switching the system clock source from the XOSC on ramp-down. See *Section 30.4.1.2: Progressive system clock switching*, for details on how to set this value. This register is reset only on a power-on reset.

Note: Byte and half-word write accesses are not allowed for this register. Such an access will not result in an exception; however, the value will not be loaded with the new value.

Address: 0x0108 Access: User read/write, Supervisor read/write, Test read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	DIVE			
W																
POR	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	DIVE															
W																
POR	0	0	0	0	0	0	1	1	1	1	1	0	0	1	1	1

Figure 321. PCS Divider End Register 1 (CGM_PCS_DIVE1)

Table 431. CGM_PCS_DIVE1 field descriptions

Field	Description
12:31 DIVE	Divider End Value This is the clock divider end value for the clock ramp-down phase when switching from the XOSC.

30.3.1.15 PCS Divider Start Register 1 (CGM_PCS_DIVS1)

This register defines the initial division value for the progressive system clock switching when switching the system clock source to the XOSC on ramp-up. Register *n* corresponds to system clock source *n*. See *Section 30.4.1.2: Progressive system clock switching*, for details on how to set these values. This register is reset only on a power-on reset.

Note: Byte and half-word write accesses are not allowed for this register. Such an access will not result in an exception; however, the value will not be loaded with the new value.

Address: 0x010C Access: User read/write, Supervisor read/write, Test read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	DIVS			
W																
POR	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	DIVS															
W																
POR	0	0	0	0	0	0	1	1	1	1	1	0	0	1	1	1

Figure 322. PCS Divider Start Register 1 (CGM_PCS_DIVS1)

Table 432. CGM_PCS_DIVS1 field descriptions

Field	Description
12:31 DIVS	Divider Start Value This is the start value of the clock divider for the clock ramp-up phase when switching to the XOSC.

30.3.1.16 PCS Divider Change Register 2 (CGM_PCS_DIVC2)

This register defines the rate of frequency change and initial change value for the progressive system clock switching when switching the system clock source to or from the PLL0 PHI on ramp-up and ramp-down, respectively. See *Section 30.4.1.2: Progressive system clock switching*, for details on how to set these values. This register is reset only on a power-on reset.

Note: Byte write accesses are not allowed for the INIT field of this register. Such an access will not result in an exception; however, the value will not be loaded with the new value.

Address: 0x0110 Access: User read/write, Supervisor read/write, Test read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	INIT															
W																
POR	0	0	0	0	0	0	1	1	1	1	1	0	0	1	1	1

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	RATE							
W																
POR	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 323. PCS Divider Change Register 2 (CGM_PCS_DIVC2)

Table 433. CGM_PCS_DIVC2 field descriptions

Field	Description
0:15 INIT	Divider Change Initial Value This is initial change value of the clock divider for the clock ramp-up phase when switching to the PLL0 PHI.
24:31 RATE	Divider Change Rate This value controls the change value of the clock divider for the clock ramp-up and ramp-down phase when switching to/from the PLL0 PHI.

30.3.1.17 PCS Divider End Register 2 (CGM_PCS_DIVE2)

This register defines the final division value for the progressive system clock switching when switching the system clock source from the PLL0 PHI on ramp-down. See *Section 30.4.1.2: Progressive system clock switching* for details on how to set this value. This register is reset only on a power-on reset.

Note: Byte and half-word write accesses are not allowed for this register. Such an access will not result in an exception; however, the value will not be loaded with the new value.

Address: 0x0114 Access: User read/write, Supervisor read/write, Test read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	DIVE			
W																
POR	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	DIVE															
W																
POR	0	0	0	0	0	0	1	1	1	1	1	0	0	1	1	1

Figure 324. PCS Divider End Register 2 (CGM_PCS_DIVE2)

Table 434. CGM_PCS_DIVE2 field descriptions

Field	Description
12:31 DIVE	Divider End Value This is the clock divider end value for the clock ramp-down phase when switching from the PLL0 PHI.

30.3.1.18 PCS Divider Start Register 2 (CGM_PCS_DIVS2)

This register defines the initial division value for the progressive system clock switching when switching the system clock source to the PLL0 PHI on ramp-up. See *Section 30.4.1.2: Progressive system clock switching*, for details on how to set this value. This register is reset only on a power-on reset.

Note: Byte and half-word write accesses are not allowed for this register. Such an access will not result in an exception; however, the value will not be loaded with the new value.

Address: 0x0118 Access: User read/write, Supervisor read/write, Test read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	DIVS			
W																
POR	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	DIVS															
W																
POR	0	0	0	0	0	0	1	1	1	1	1	0	0	1	1	1

Figure 325. PCS Divider Start Register 2 (CGM_PCS_DIVS2)

Table 435. CGM_PCS_DIVS2 field descriptions

Field	Description
12:31 DIVS	Divider Start Value This is the start value of the clock divider for the clock ramp-up phase when switching to the PLL0 PHI.

30.3.1.19 PCS Divider Change Register 4 (CGM_PCS_DIVC4)

This register defines the rate of frequency change and initial change value for the progressive system clock switching when switching the system clock source to or from the PLL1 PHI on ramp-up and ramp-down, respectively. See *Section 30.4.1.2: Progressive system clock switching*, for details on how to set these values. This register is reset only on a power-on reset.

Note: Byte write accesses are not allowed for the INIT field of this register. Such an access will not result in an exception; however, the value will not be loaded with the new value.

Address: 0x0128 Access: User read/write, Supervisor read/write, Test read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	INIT															
W																
POR	0	0	0	0	0	0	1	1	1	1	1	0	0	1	1	1

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	RATE							
W																
POR	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 326. PCS Divider Change Register 4 (CGM_PCS_DIVC4)

Table 436. CGM_PCS_DIVC4 field descriptions

Field	Description
0:15 INIT	Divider Change Initial Value This is initial change value of the clock divider for the clock ramp-up phase when switching to the PLL1 PHI.
24:31 RATE	Divider Change Rate This value controls the change value of the clock divider for the clock ramp-up and ramp-down phase when switching to/from the PLL1 PHI.

30.3.1.20 PCS Divider End Register 4 (CGM_PCS_DIVE4)

This register defines the final division value for the progressive system clock switching when switching the system clock source from the PLL1 PHI on ramp-down. See *Section 30.4.1.2: Progressive system clock switching*, for details on how to set this value. This register is reset only on a power-on reset.

Note: Byte and half-word write accesses are not allowed for this register. Such an access will not result in an exception; however, the value will not be loaded with the new value.

Address: 0x012C Access: User read/write, Supervisor read/write, Test read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	DIVE			
W																
POR	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	DIVE															
W																
POR	0	0	0	0	0	0	1	1	1	1	1	0	0	1	1	1

Figure 327. PCS Divider End Register 4 (CGM_PCS_DIVE4)

Table 437. CGM_PCS_DIVE4 field descriptions

Field	Description
12:31 DIVE	Divider End Value This is the clock divider end value for the clock ramp-down phase when switching from the PLL1 PHI.

30.3.1.21 PCS Divider Start Register 4 (CGM_PCS_DIVS4)

This register defines the initial division value for the progressive system clock switching when switching the system clock source to the PLL1 PHI on ramp-up. See *Section 30.4.1.2: Progressive system clock switching* for details on how to set this value. This register is reset only on a power-on reset.

Note: Byte and half-word write accesses are not allowed for this register. Such an access will not result in an exception; however, the value will not be loaded with the new value.

Address: 0x0130 Access: User read/write, Supervisor read/write, Test read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	DIVS			
W																
POR	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	DIVS															
W																
POR	0	0	0	0	0	0	1	1	1	1	1	0	0	1	1	1

Figure 328. PCS Divider Start Register 4 (CGM_PCS_DIVS4)

Table 438. CGM_PCS_DIVS4 field descriptions

Field	Description
12:31 DIVS	Divider Start Value This is the start value of the clock divider for the clock ramp-up phase when switching to the PLL1 PHI.

30.3.1.22 System Clock Divider Ratio Change Register (CGM_SC_DIV_RC)

This register selects whether the system clock divider ratios will change from the current configuration to the configuration in the next update. If the value of SYS_DIV_RATIO_CHNG is ‘1’ at the time of a write to the CGM_DIV_UPD_TRIG register and a pre-loaded system clock divider update is pending, the crossbar is halted while the system clock divider update takes place. If the value of SYS_DIV_RATIO_CHNG is ‘0’, the crossbar switch is not halted. This register has no effect if the system clock divider update is configured in the CGM_DIV_UPD_TYPE register to be immediate.

An example of a divider ratio not changing is divider 0 changes from divide by 1 to divide by 2, and divider 1 changes from divide by 2 to divide by 4. The ratio is 1:2 in both configurations. An example of a divider ratio changing is divider 0 changes from divide by 1 to divide by 2, and divider 1 changes from divide by 2 to divide by 6. The ratio changes from

1:2 to 1:3. The value of SYS_DIV_RATIO_CHNG is returned to ‘1’ automatically on each write to the CGM_DIV_UPD_TRIG register.

Caution: Software must never write a ‘0’ to the SYS_DIV_RATIO_CHNG bit if the system clock divider ratios change from the current configuration to the next configuration. This can potentially cause errors in the crossbar which lead to lost data.

Address: 0x01D0 Access: User read/write, Supervisor read/write, Test read

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	SYS_DIV_RATIO_CHNG
W																
POR	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 329. System Clock Divider Ratio Change Register (CGM_SC_DIV_RC)

Table 439. CGM_SC_DIV_RC field descriptions

Field	Description
31 SYS_DIV_RATIO_CHNG	System Clock Divider Ratio Change 0 The system clock divider ratios does not change with the next system clock divider configuration update 1 The system clock divider ratios changes with the next system clock divider configuration update

30.3.1.23 Divider Update Type Register (CGM_DIV_UPD_TYPE)

This register selects whether the dividers associated with a clock are updated immediately on writing to the corresponding divider configuration register or only on writing to the divider update trigger register CGM_DIV_UPD_TRIG, after which the pre-loaded configuration in the divider configuration registers will take effect.

Address: 0x01D4 Access: User read/write, Supervisor read/write, Test read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W	SYS_UPD_TYPE															
POR	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

Figure 330. Divider Update Type Register (CGM_DIV_UPD_TYPE)

Table 440. CGM_DIV_UPD_TYPE field descriptions

Field	Description
0 SYS_UPD_TYPE	System Clock Divider Update Type 0 The configuration for each system clock divider is updated immediately on writing to the corresponding CGM_SC_DCi register 1 The configuration for each system clock divider is pre-loaded on writing to the corresponding CGM_SC_DCi register, and the pre-loaded configurations of all the system clock dividers are updated on writing to the CGM_DIV_UPD_TRIG register

30.3.1.24 Divider Update Trigger Register (CGM_DIV_UPD_TRIG)

Writing any value to this register will cause all dividers that have corresponding bits set to ‘1’ in the CGM_DIV_UPD_TYPE register and have pre-loaded configurations to be updated immediately. Reading this register will always return all zeroes.

Address: 0x01D8 Access: User read/write, Supervisor read/write, Test read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0															
W	DIV_UPD_TRIGGER															
POR	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0															
W	DIV_UPD_TRIGGER															
POR	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 331. Divider Update Trigger Register (CGM_DIV_UPD_TRIG)

Table 441. CGM_DIV_UPD_TRIG field descriptions

Field	Description
0:31 DIV_UPD_TRIGGER	Writing any value to this register causes all dividers that have corresponding bits set to '1' in the CGM_DIV_UPD_TYPE register and have pre-loaded configurations to be updated immediately. Reading this register always returns all zeroes.

30.3.1.25 Divider Update Status Register (CGM_DIV_UPD_STAT)

This register indicates whether a divider for each clock is in the process of being updated. If the CGM_DIV_UPD_TYPE[i] is '0', CGM_DIV_UPD_STAT[i] is set to '1' immediately on writing to one of the corresponding divider configuration registers. Otherwise, CGM_DIV_UPD_STAT[i] is set to '1' immediately on writing to the CGM_DIV_UPD_TRIG register if one of the corresponding divider configurations has been pre-loaded. CGM_DIV_UPD_STAT[i] is cleared to '0' as soon as all the corresponding dividers have been updated and the dividers generate the desired frequency.

Address: 0x01DC Access: User read/write, Supervisor read/write, Test read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	SYS_UPD_STAT	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
POR	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

Figure 332. Divider Update Status Register (CGM_DIV_UPD_STAT)

Table 442. CGM_DIV_UPD_STAT field descriptions

Field	Description
0 SYS_UPD_STAT	System Clock Divider Update Status 0 The configuration for at least one system clock divider is not being updated 1 The configuration for at least one system clock divider is being updated

30.3.1.26 System Clock Select Status Register (CGM_SC_SS)

This register provides the current system clock source selection.

Address: 0x01E4 Access: User read, Supervisor read, Test read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	SELSTAT				0	0	0	0	SWTRG			SWIP
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

Figure 333. System Clock Select Status Register (CGM_SC_SS)

Table 443. CGM_SC_SS field descriptions

Field	Description
4:7 SELSTAT	System Clock Source Selection Status This value indicates the current source for the system clock. 0000 IRCOSC 0001 XOSC 0010 PLL0 PHI 0011 Reserved 0100 PLL1 PHI 0101 Reserved ... 1110 Reserved 1111 system clock is disabled
12:14 SWTRG	Switch Trigger cause This value indicates the cause for the latest clock source switch. 000 Reserved 001 switch after request from MC_ME succeeded 010 switch after request from MC_ME failed due inactive target clock 011 switch after request from MC_ME failed due inactive current clock 100 switch to IRCOSC due to SAFE mode request or reset succeeded 101 switch to IRCOSC due to SAFE mode request or reset, but current clock source was inactive 110 Reserved 111 Reserved
15 SWIP	Switch In Progress 0 clock source switching has completed 1 clock source switching is in progress

30.3.1.27 System Clock Divider 0 Configuration Register (CGM_SC_DC0)

This register controls system clock divider 0. See *Section 30.4.1.4.1: System clock divider synchronization*, for details on division value limitations.

Note: Byte and half-word write accesses are not allowed for this register. Such an access will not result in an exception; however, the value will not be loaded with the new value.

Address: 0x01E8 Access: User read/write, Supervisor read/write, Test read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	DE	0	0	0	0	0	0	0	0	0	DIV					
W																
Reset	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

Figure 334. System Clock Divider 0 Configuration Register (CGM_SC_DC0)

Table 444. CGM_SC_DC0 field descriptions

Field	Description
0 DE	Divider 1 Enable This divider is always enabled. Therefore, this bit is read-only and always returns a value of '1'.
10:15 DIV	Divider 1 Division Value The resultant SYS_CLK will have a period 'DIV + 1' times that of the system clock.

30.3.1.28 System Clock Divider 1 Configuration Register (CGM_SC_DC1)

This register controls system clock divider 1. See *Section 30.4.1.4.1: System clock divider synchronization*, for details on division value limitations.

Note: Byte and half-word write accesses are not allowed for this register. Such an access will not result in an exception; however, the value will not be loaded with the new value.

Address: 0x01EC Access: User read/write, Supervisor read/write, Test read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	DE	0	0	0	0	0	0	0	0	0	DIV					
W																
Reset	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

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

Figure 335. System Clock Divider 1 Configuration Register (CGM_SC_DC1)

Table 445. CGM_SC_DC1 field descriptions

Field	Description
0 DE	Divider 1 Enable This divider is always enabled. Therefore, this bit is read-only and always returns a value of '1'.
10:15 DIV	Divider 1 Division Value The resultant HPBM_CLK will have a period 'DIV + 1' times that of the system clock.

30.3.1.29 System Clock Divider 2 Configuration Register (CGM_SC_DC2)

This register controls system clock divider 2. See *Section 30.4.1.4.1: System clock divider synchronization*, for details on division value limitations.

Note: Byte and half-word write accesses are not allowed for this register. Such an access will not result in an exception; however, the value will not be loaded with the new value.

Address: 0x01F0 Access: User read/write, Supervisor read/write, Test read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	DE	0	0	0	0	0	0	0	0	0	DIV					
W																
Reset	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1

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

Figure 336. System Clock Divider 2 Configuration Register (CGM_SC_DC2)

Table 446. CGM_SC_DC2 field descriptions

Field	Description
0 DE	Divider 2 Enable This divider is always enabled. Therefore, this bit is read-only and always returns a value of '1'.
10:15 DIV	Divider 2 Division Value The resultant PBRIDGE_CLK will have a period 'DIV + 1' times that of the system clock.

30.3.1.30 System Clock Divider 3 Configuration Register (CGM_SC_DC3)

This register controls system clock divider 3. See *Section 30.4.1.4.1: System clock divider synchronization*, for details on division value limitations.

Note: Byte and half-word write accesses are not allowed for this register. Such an access will not result in an exception; however, the value will not be loaded with the new value.

Address: 0x01F4 Access: User read/write, Supervisor read/write, Test read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	DE	0	0	0	0	0	0	0	0	0	DIV					
W																
Reset	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

Figure 337. System Clock Divider 3 Configuration Register (CGM_SC_DC3)

Table 447. CGM_SC_DC3 field descriptions

Field	Description
0 DE	Divider 2 Enable This divider is always enabled. Therefore, this bit is read-only and always returns a value of '1'.
10:15 DIV	Divider 2 Division Value The resultant FBRIDGE_CLK will have a period 'DIV + 1' times that of the system clock.

30.3.1.31 System Clock Divider 4 Configuration Register (CGM_SC_DC4)

This register controls system clock divider 4. See *Section 30.4.1.4.1: System clock divider synchronization*, for details on division value limitations.

Note: Byte and half-word write accesses are not allowed for this register. Such an access will not result in an exception; however, the value will not be loaded with the new value.

Address: 0x01F8 Access: User read/write, Supervisor read/write, Test read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	DE	0	0	0	0	0	0	0	0	0	DIV					
W																
Reset	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

Figure 338. System Clock Divider 4 Configuration Register (CGM_SC_DC4)

Table 448. CGM_SC_DC4 field descriptions

Field	Description
0 DE	Divider 2 Enable This divider is always enabled. Therefore, this bit is read-only and always returns a value of '1'.
10:15 DIV	Divider 2 Division Value The resultant PFBRIDGE_CLK will have a period 'DIV + 1' times that of the system clock.

30.3.1.32 Auxiliary Clock 0 Select Control Register (CGM_AC0_SC)

This register is used to select the current clock source for the following clocks:

- undivided: (unused)
- divided by auxiliary clock 0 divider 0: GTM_CLK
- divided by auxiliary clock 0 divider 1: SDADC_CLK
- divided by auxiliary clock 0 divider 2: SARADC_CLK
- divided by auxiliary clock 0 divider 3: EMIOS_CLK

Address: 0x0200 Access: User read/write, Supervisor read/write, Test read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	SELCTL				0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

Figure 339. Auxiliary Clock 0 Select Control Register (CGM_AC0_SC)

Table 449. CGM_AC0_SC field descriptions

Field	Description
4:7 SELCTL	Auxiliary Clock 0 Source Selection Control Selects the source for auxiliary clock 0. **Note:** Status of respective clock source must be ensured before selection. 0000 IRCOSC 0001 XOSC 0010 PLL0 PHI 0011 Reserved ... 1111 Reserved

30.3.1.33 Auxiliary Clock 0 Select Status Register (CGM_AC0_SS)

This register provides the current auxiliary clock 0 source selection.

Address: 0x0204 Access: User read, Supervisor read, Test read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	SELSTAT				0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

Figure 340. Auxiliary Clock 0 Select Status Register (CGM_AC0_SS)

Table 450. CGM_AC0_SS field descriptions

Field	Description
4:7 SELSTAT	Auxiliary Clock 0 Source Selection Status This value indicates the current source for auxiliary clock 0. 0000 IRCOSC 0001 XOSC 0010 PLL0 PHI 0011 Reserved ... 1111 Reserved

30.3.1.34 Auxiliary Clock 0 Divider 0 Configuration Register (CGM_AC0_DC0)

This register controls auxiliary clock 0 divider 0.

Note: Byte and half-word write accesses are not allowed for this register. Such an access will not result in an exception; however, the value will not be loaded with the new value.

Address: 0x0208 Access: User read/write, Supervisor read/write, Test read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	DE	0	0	0	0	0	0	0	0	0	0	0	DIV			
W																
Reset	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

Figure 341. Auxiliary Clock 0 Divider 0 Configuration Register (CGM_AC0_DC0)

Table 451. CGM_AC0_DC0 field descriptions

Field	Description
0 DE	Divider Enable 0 Disable auxiliary clock 0 divider 0 1 Enable auxiliary clock 0 divider 0
12:15 DIV	Divider Division Value The resultant GTM_CLK will have a period 'DIV + 1' times that of auxiliary clock 0. If DE is set to 0 (divider 0 is disabled), any write access to the DIV field is ignored and the GTM_CLK remains disabled.

30.3.1.35 Auxiliary Clock 0 Divider 1 Configuration Register (CGM_AC0_DC1)

This register controls auxiliary clock 0 divider 1.

Note: Byte and half-word write accesses are not allowed for this register. Such an access will not result in an exception; however, the value will not be loaded with the new value.

Address: 0x020C Access: User read/write, Supervisor read/write, Test read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	DE	0	0	0	0	0	0	0	0	DIV						
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

Figure 342. Auxiliary Clock 0 Divider 1 Configuration Register (CGM_AC0_DC1)

Table 452. CGM_AC0_DC1 field descriptions

Field	Description
0 DE	Divider Enable 0 Disable auxiliary clock 0 divider 1 1 Enable auxiliary clock 0 divider 1
9:15 DIV	Divider Division Value The resultant SDADC_CLK will have a period 'DIV + 1' times that of auxiliary clock 0. If DE is set to 0 (divider 1 is disabled), any write access to the DIV field is ignored and the SDADC_CLK remains disabled.

30.3.1.36 Auxiliary Clock 0 Divider 2 Configuration Register (CGM_AC0_DC2)

This register controls auxiliary clock 0 divider 2.

Note: Byte and half-word write accesses are not allowed for this register. Such an access will not result in an exception; however, the value will not be loaded with the new value.

Address: 0x0210 Access: User read/write, Supervisor read/write, Test read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	DE	0	0	0	0	0	0	0	0	DIV						
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

Figure 343. Auxiliary Clock 0 Divider 2 Configuration Register (CGM_AC0_DC2)

Table 453. CGM_AC0_DC2 field descriptions

Field	Description
0 DE	Divider Enable 0 Disable auxiliary clock 0 divider 2 1 Enable auxiliary clock 0 divider 2
9:15 DIV	Divider Division Value The resultant SARADC_CLK will have a period 'DIV + 1' times that of auxiliary clock 0. If DE is set to 0 (divider 2 is disabled), any write access to the DIV field is ignored and the SARADC_CLK remains disabled.

30.3.1.37 Auxiliary Clock 0 Divider 3 Configuration Register (CGM_AC0_DC3)

This register controls auxiliary clock 0 divider 3.

Note: Byte and half-word write accesses are not allowed for this register. Such an access will not result in an exception, however the value will not be loaded with the new value.

Address: 0x0214 Access: User read/write, Supervisor read/write, Test read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	DE	0	0	0	0	0	0	0	0	0	0	0	DIV			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

Figure 344. Auxiliary Clock 0 Divider 3 Configuration Register (CGM_AC0_DC3)

Table 454. CGM_AC0_DC3 field descriptions

Field	Description
0 DE	Divider Enable 0 Disable auxiliary clock 0 divider 3 1 Enable auxiliary clock 0 divider 3
12:15 DIV	Divider Division Value The resultant EMIOS_CLK will have a period 'DIV + 1' times that of auxiliary clock 0. If DE is set to 0 (divider 0 is disabled), any write access to the DIV field is ignored and the EMIOS_CLK remains disabled.

30.3.1.38 Auxiliary Clock 1 Select Control Register (CGM_AC1_SC)

This register is used to select the current clock source for the following clocks:

- undivided: (unused)
- divided by auxiliary clock 1 divider 0: LF_REF_CLK

Address: 0x0220 Access: User read/write, Supervisor read/write, Test read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	SELCTL				0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0

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

Figure 345. Auxiliary Clock 1 Select Control Register (CGM_AC1_SC)

Table 455. CGM_AC1_SC field descriptions

Field	Description
4:7 SELCTL	Auxiliary Clock 1 Source Selection Control This value selects the current source for auxiliary clock 1. **Note:** Status of respective clock source must be ensured before selection. 0000 Reserved 0001 XOSC 0010 PLL0 PHI 0011 Reserved 0100 Reserved 0101 LFAST-SysClk pin 0110 Reserved ... 1111 Reserved

30.3.1.39 Auxiliary Clock 1 Select Status Register (CGM_AC1_SS)

This register provides the current auxiliary clock 0 source selection.

Address: 0x0224 Access: User read, Supervisor read, Test read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	SELSTAT				0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0

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

Figure 346. Auxiliary Clock 1 Select Status Register (CGM_AC1_SS)

Table 456. CGM_AC1_SS field descriptions

Field	Description
4:7 SELSTAT	Auxiliary Clock 1 Source Selection Status This value indicates the current source for auxiliary clock 1. 0000 Reserved 0001 XOSC 0010 PLL0 PHI 0011 Reserved 0100 Reserved 0101 LFAST-SysClk pin 0110 Reserved ... 1111 Reserved

30.3.1.40 Auxiliary Clock 1 Divider 0 Configuration Register (CGM_AC1_DC0)

This register controls auxiliary clock 0 divider 0.

Note: Byte and half-word write accesses are not allowed for this register. Such an access will not result in an exception; however, the value will not be loaded with the new value.

Address: 0x0228 Access: User read/write, Supervisor read/write, Test read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	DE	0	0	0	0	0	0	0	0	DIV						
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

Figure 347. Auxiliary Clock 1 Divider 0 Configuration Register (CGM_AC1_DC0)

Table 457. CGM_AC1_DC0 field descriptions

Field	Description
0 DE	Divider Enable 0 Disable auxiliary clock 1 divider 0 1 Enable auxiliary clock 1 divider 0
9:15 DIV	Divider Division Value The resultant LF_REF_CLK will have a period 'DIV + 1' times that of auxiliary clock 1. If DE is set to 0 (divider 0 is disabled), any write access to the DIV field is ignored and the LF_REF_CLK remains disabled.

30.3.1.41 Auxiliary Clock 2 Divider 0 Configuration Register (CGM_AC2_DC0)

This register controls auxiliary clock 2 divider 0 when the "CAN jitter" feature is not enabled (see *Section 30.4.3.1: Divider jitter injection*).

Note: Byte and half-word write accesses are not allowed for this register. Such an access will not result in an exception; however, the value will not be loaded with the new value.

Address: 0x0248 Access: User read/write, Supervisor read/write, Test read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	DE	0	0	0	0	0	0	0	0	0	DIV					
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

Figure 348. Auxiliary Clock 2 Divider 0 Configuration Register (CGM_AC2_DC0)

Table 458. CGM_AC2_DC0 field descriptions

Field	Description
0 DE	Divider Enable 0 Disable auxiliary clock 2 divider 0 1 Enable auxiliary clock 2 divider 0
10:15 DIV	Divider Division Value The resultant FRAY_CLK will have a period 'DIV + 1' times that of auxiliary clock 2. If DE is set to 0 (divider 0 is disabled), any write access to the DIV field is ignored and the FRAY_CLK remains disabled.

30.3.1.42 Auxiliary Clock 2 Divider 1 Configuration Register (CGM_AC2_DC1)

This register controls auxiliary clock 2 divider 1.

Note: Byte and half-word write accesses are not allowed for this register. Such an access will not result in an exception; however, the value will not be loaded with the new value.

Address: 0x024C Access: User read/write, Supervisor read/write, Test read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	DE	0	0	0	0	0	0	0	0	0	DIV					
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

Figure 349. Auxiliary Clock 2 Divider 1 Configuration Register (CGM_AC2_DC1)

Table 459. CGM_AC2_DC1 field descriptions

Field	Description
0 DE	Divider Enable 0 Disable auxiliary clock 2 divider 1 1 Enable auxiliary clock 2 divider 1
10:15 DIV	Divider Division Value The resultant SENT_CLK will have a period 'DIV + 1' times that of auxiliary clock 2. If DE is set to 0 (divider 1 is disabled), any write access to the DIV field is ignored and the SENT_CLK remains disabled.

30.3.1.43 Auxiliary Clock 3 Select Control Register (CGM_AC3_SC)

This register is used to select the current clock source for the PLL0 reference clock.

Address: 0x0260 Access: User read/write, Supervisor read/write, Test read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	SELCTL				0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

Figure 350. Auxiliary Clock 3 Select Control Register (CGM_AC3_SC)

Table 460. CGM_AC3_SC field descriptions

Field	Description
4:7 SELCTL	Auxiliary Clock 3 Source Selection Control This value selects the current source for auxiliary clock 3. **Note:** Status of respective clock source must be ensured before selection. 0000 IRCOSC 0001 XOSC 0010 Reserved ... 1111 Reserved

30.3.1.44 Auxiliary Clock 3 Select Status Register (CGM_AC3_SS)

This register provides the current auxiliary clock 3 source selection.

Address: 0x0264 Access: User read, Supervisor read, Test read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	SELSTAT				0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

Figure 351. Auxiliary Clock 3 Select Status Register (CGM_AC3_SS)

Table 461. CGM_AC3_SS field descriptions

Field	Description
4:7 SELSTAT	Auxiliary Clock 3 Source Selection Status This value indicates the current source for auxiliary clock 3. 0000 IRCOSC 0001 XOSC 0010 Reserved ... 1111 Reserved

30.3.1.45 Auxiliary Clock 4 Select Control Register (CGM_AC4_SC)

This register is used to select the current clock source for the PLL1 reference clock.

Address: 0x0280 Access: User read/write, Supervisor read/write, Test read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	SELCTL				0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0

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

Figure 352. Auxiliary Clock 4 Select Control Register (CGM_AC4_SC)

Table 462. CGM_AC4_SC field descriptions

Field	Description
4:7 SELCTL	Auxiliary Clock 4 Source Selection Control This value selects the current source for auxiliary clock 4. **Note:** Status of respective clock source must be ensured before selection. 0000 Reserved 0001 XOSC 0010 Reserved 0011 PLL0 PHI1 0100 Reserved ... 1111 Reserved

30.3.1.46 Auxiliary Clock 4 Select Status Register (CGM_AC4_SS)

This register provides the current auxiliary clock 4 source selection.

Address: 0x0284 Access: User read, Supervisor read, Test read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	SELSTAT				0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0

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

Figure 353. Auxiliary Clock 4 Select Status Register (CGM_AC4_SS)

Table 463. CGM_AC4_SS field descriptions

Field	Description
4:7 SELSTAT	Auxiliary Clock 4 Source Selection Status This value indicates the current source for auxiliary clock 4. 0000 Reserved 0001 XOSC 0010 Reserved 0011 PLL0 PHI1 0100 Reserved ... 1111 Reserved

30.3.1.47 Auxiliary Clock 5 Divider 0 Configuration Register (CGM_AC5_DC0)

This register controls auxiliary clock 5 divider 0. See *Section 30.4.3.2: Fractional divider details*, for divider characteristics.

Note: Byte and half-word write accesses are not allowed for this register. Such an access will not result in an exception; however, the value will not be loaded with the new value.

Address: 0x02A8 Access: User read/write, Supervisor read/write, Test read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	DE	0	0	DIV												
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	DIV_FMT	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 354. Auxiliary Clock 5 Divider 0 Configuration Register (CGM_AC5_DC0)

Table 464. CGM_AC5_DC0 field descriptions

Field	Description
0 DE	Divider Enable 0 Disable auxiliary clock 5 divider 0 1 Enable auxiliary clock 5 divider 0
3:15 DIV	Divider Division Value The resultant PSI5_f189_CLK will have a period ‘DIV + 1’ times that of auxiliary clock 1. If DE is set to 0 (divider 0 is disabled), any write access to the DIV field is ignored and the PSI5_f189_CLK remains disabled.
30:31 DIV_FMT	Divider Value Format This value indicates the format of the division value for calculating the division factor 00 Division factor = (DIV + 1) / 1 01 Division factor = (DIV + 1) / 10 10 Division factor = (DIV + 1) / 100 11 Division factor = (DIV + 1) / 1,000

30.3.1.48 Auxiliary Clock 5 Divider 1 Configuration Register (CGM_AC5_DC1)

This register controls auxiliary clock 5 divider 1.

Note: Byte and half-word write accesses are not allowed for this register. Such an access will not result in an exception; however, the value will not be loaded with the new value.

Address: 0x02AC Access: User read/write, Supervisor read/write, Test read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	DE	0	0	0	0	0	DIV									
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

Figure 355. Auxiliary Clock 5 Divider 1 Configuration Register (CGM_AC5_DC1)

Table 465. CGM_AC5_DC1 field descriptions

Field	Description
0 DE	Divider Enable 0 Disable auxiliary clock 5 divider 1 1 Enable auxiliary clock 5 divider 1
6:15 DIV	Divider Division Value The resultant PSI5_f125_CLK will have a period 'DIV + 1' times that of auxiliary clock 5. If DE is set to 0 (divider 1 is disabled), any write access to the DIV field is ignored and the PSI5_f125_CLK remains disabled.

30.3.1.49 Auxiliary Clock 5 Divider 2 Configuration Register (CGM_AC5_DC2)

This register controls auxiliary clock 5 divider 2.

Note: Byte and half-word write accesses are not allowed for this register. Such an access will not result in an exception; however, the value will not be loaded with the new value.

Address: 0x02B0 Access: User read/write, Supervisor read/write, Test read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	DE	0	0	0	0	0	DIV									
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

Figure 356. Auxiliary Clock 5 Divider 2 Configuration Register (CGM_AC5_DC2)

Table 466. CGM_AC5_DC2 field descriptions

Field	Description
0 DE	Divider Enable 0 Disable auxiliary clock 5 divider 2 1 Enable auxiliary clock 5 divider 2
6:15 DIV	Divider Division Value The resultant PSI5_1us_CLK will have a period 'DIV + 1' times that of auxiliary clock 5. If DE is set to 0 (divider 2 is disabled), any write access to the DIV field is ignored and the PSI5_1us_CLK remains disabled.

30.3.1.50 Auxiliary Clock 6 Select Control Register (CGM_AC6_SC)

This register is used to select the current clock source for the following clocks:

- undivided: (unused)
- divided by auxiliary clock 6 divider 0: SYSCLK0 pin clock

Address: 0x02C0 Access: User read/write, Supervisor read/write, Test read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	SELCTL				0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

Figure 357. Auxiliary Clock 6 Select Control Register (CGM_AC6_SC)

Table 467. CGM_AC6_SC field descriptions

Field	Description
4:7 SELCTL	Auxiliary Clock 6 Source Selection Control This value selects the current source for auxiliary clock 6. **Note:** Status of respective clock source must be ensured before selection. 0000 IRCOSC 0001 XOSC 0010 PLL0 PHI 0011 Reserved 0100 PLL1 PHI 0101 Reserved 0110 rtc_clk 0111 Reserved ... 1111 Reserved

30.3.1.51 Auxiliary Clock 6 Select Status Register (CGM_AC6_SS)

This register provides the current auxiliary clock 6 source selection.

Address: 0x02C4 Access: User read, Supervisor read, Test read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	SELSTAT				0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

Figure 358. Auxiliary Clock 6 Select Status Register (CGM_AC6_SS)

Table 468. CGM_AC6_SS field descriptions

Field	Description
4:7 SELSTAT	Auxiliary Clock 6 Source Selection Status This value indicates the current source for auxiliary clock 6. 0000 IRCOSC 0001 XOSC 0010 PLL0 PHI 0011 Reserved 0100 PLL1 PHI 0101 Reserved 0110 rtc_clk 0111 Reserved ... 1111 Reserved

30.3.1.52 Auxiliary Clock 6 Divider 0 Configuration Register (CGM_AC6_DC0)

This register controls auxiliary clock 6 divider 0.

Note: Byte and half-word write accesses are not allowed for this register. Such an access will not result in an exception; however, the value will not be loaded with the new value.

Address: 0x02C8 Access: User read/write, Supervisor read/write, Test read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	DE	0	0	0	0	0	0	DIV								
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

Figure 359. Auxiliary Clock 6 Divider 0 Configuration Register (CGM_AC6_DC0)

Table 469. CGM_AC6_DC0 field descriptions

Field	Description
0 DE	Divider Enable 0 Disable auxiliary clock 6 divider 0 1 Enable auxiliary clock 6 divider 0
7:15 DIV	Divider Division Value The resultant SYSCLK0 pin clock will have a period 'DIV + 1' times that of auxiliary clock 6. If DE is set to 0 (divider 0 is disabled), any write access to the DIV field is ignored and the SYSCLK0 pin clock remains disabled.

30.3.1.53 Auxiliary Clock 7 Select Control Register (CGM_AC7_SC)

This register is used to select the current clock source for the following clocks:

- undivided: (unused)
- divided by auxiliary clock 7 divider 0: SYSCLK1 pin clock

Address: 0x02E0 Access: User read/write, Supervisor read/write, Test read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	SELCTL				0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

Figure 360. Auxiliary Clock 7 Select Control Register (CGM_AC7_SC)

Table 470. CGM_AC7_SC field descriptions

Field	Description
4:7 SELCTL	Auxiliary Clock 7 Source Selection Control This value selects the current source for auxiliary clock 7. **Note:** Status of respective clock source must be ensured before selection. 0000 IRCOSC 0001 XOSC 0010 PLL0 PHI 0011 Reserved 0100 PLL1 PHI 0101 Reserved ... 1111 Reserved

30.3.1.54 Auxiliary Clock 7 Select Status Register (CGM_AC7_SS)

This register provides the current auxiliary clock 7 source selection.

Address: 0x02E4 Access: User read, Supervisor read, Test read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	SELSTAT				0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

Figure 361. Auxiliary Clock 7 Select Status Register (CGM_AC7_SS)

Table 471. CGM_AC7_SS field descriptions

Field	Description
4:7 SELSTAT	Auxiliary Clock 7 Source Selection Status This value indicates the current source for auxiliary clock 7. 0000 IRCOSC 0001 XOSC 0010 PLL0 PHI 0011 Reserved 0100 PLL1 PHI 0101 Reserved ... 1111 Reserved

30.3.1.55 Auxiliary Clock 7 Divider 0 Configuration Register (CGM_AC7_DC0)

This register controls auxiliary clock 7 divider 0.

Note: Byte and half-word write accesses are not allowed for this register. Such an access will not result in an exception; however, the value will not be loaded with the new value.

Address: 0x02E8 Access: User read/write, Supervisor read/write, Test read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	DE	0	0	0	0	0	0	DIV								
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

Figure 362. Auxiliary Clock 7 Divider 0 Configuration Register (CGM_AC7_DC0)

Table 472. CGM_AC7_DC0 field descriptions

Field	Description
0 DE	Divider Enable 0 Disable auxiliary clock 7 divider 0 1 Enable auxiliary clock 7 divider 0
7:15 DIV	Divider Division Value The resultant SYSCLK1 pin clock will have a period 'DIV + 1' times that of auxiliary clock 7. If DE is set to 0 (divider 0 is disabled), any write access to the DIV field is ignored and the SYSCLK1 pin clock remains disabled.

30.3.1.56 Auxiliary Clock 8 Select Control Register (CGM_AC8_SC)

This register is used to select the current clock source for the following clocks:

- undivided: (unused)
- divided by auxiliary clock 8 divider 0: CAN_CLK0

Address: 0x0300 Access: User read/write, Supervisor read/write, Test read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	SELCTL				0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0

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

Figure 363. Auxiliary Clock 8 Select Control Register (CGM_AC8_SC)

Table 473. CGM_AC8_SC field descriptions

Field	Description
4:7 SELCTL	Auxiliary Clock 8 Source Selection Control This value selects the current source for auxiliary clock 8. **Note:** Status of respective clock source must be ensured before selection. 0000 Reserved 0001 XOSC 0010 PLL0 PHI 0011 Reserved ... 1111 Reserved

30.3.1.57 Auxiliary Clock 8 Select Status Register (CGM_AC8_SS)

This register provides the current auxiliary clock 8 source selection.

Address: 0x0304 Access: User read, Supervisor read, Test read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	SELSTAT				0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0

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

Figure 364. Auxiliary Clock 8 Select Status Register (CGM_AC8_SS)

Table 474. CGM_AC8_SS field descriptions

Field	Description
4:7 SELSTAT	Auxiliary Clock 8 Source Selection Status This value indicates the current source for auxiliary clock 8. 0000 Reserved 0001 XOSC 0010 PLL0 PHI 0011 Reserved ... 1111 Reserved

30.3.1.58 Auxiliary Clock 8 Divider 0 Configuration Register (CGM_AC8_DC0)

This register controls auxiliary clock 8 divider 0 when the “CAN jitter” feature is not enabled (see *Section 30.4.3.1: Divider jitter injection*).

Note: Byte and half-word write accesses are not allowed for this register. Such an access will not result in an exception; however, the value will not be loaded with the new value.

Address: 0x0308 Access: User read/write, Supervisor read/write, Test read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	DE	0	0	0	0	0	0	0	0	0	DIV					
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

Figure 365. Auxiliary Clock 8 Divider 0 Configuration Register (CGM_AC8_DC0)

Table 475. CGM_AC8_DC0 field descriptions

Field	Description
0 DE	Divider Enable 0 Disable auxiliary clock 8 divider 0 1 Enable auxiliary clock 8 divider 0
10:15 DIV	Divider Division Value The resultant CAN_CLK0 will have a period 'DIV + 1' times that of auxiliary clock 8. If DE is set to 0 (divider 0 is disabled), any write access to the DIV field is ignored and the CAN_CLK0 remains disabled.

30.3.1.59 Auxiliary Clock 9 Select Control Register (CGM_AC9_SC)

This register is used to select the current clock source for the following clocks:

- undivided: (unused)
- divided by auxiliary clock 9 divider 0: RTI_CLK

Address: 0x0320 Access: User read/write, Supervisor read/write, Test read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	SELCTL				0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

Figure 366. Auxiliary Clock 9 Select Control Register (CGM_AC9_SC)

Table 476. CGM_AC9_SC field descriptions

Field	Description
4:7 SELCTL	Auxiliary Clock 9 Source Selection Control This value selects the current source for auxiliary clock 9. **Note:** Status of respective clock source must be ensured before selection. 0000 IRCOSC 0001 XOSC 0010 Reserved ... 1111 Reserved

30.3.1.60 Auxiliary Clock 9 Select Status Register (CGM_AC9_SS)

This register provides the current auxiliary clock 9 source selection.

Address: 0x0324 Access: User read, Supervisor read, Test read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	SELSTAT				0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

Figure 367. Auxiliary Clock 9 Select Status Register (CGM_AC9_SS)

Table 477. CGM_AC9_SS field descriptions

Field	Description
4:7 SELSTAT	Auxiliary Clock 9 Source Selection Status This value indicates the current source for auxiliary clock 9. 0000 IRCOSC 0001 XOSC 0010 Reserved ... 1111 Reserved

30.3.1.61 Auxiliary Clock 9 Divider 0 Configuration Register (CGM_AC9_DC0)

This register controls auxiliary clock 9 divider 0.

Note: Byte and half-word write accesses are not allowed for this register. Such an access will not result in an exception; however, the value will not be loaded with the new value.

Address: 0x0328 Access: User read/write, Supervisor read/write, Test read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	DE	0	0	0	0	0	0	0	0	0	DIV					
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

Figure 368. Auxiliary Clock 9 Divider 0 Configuration Register (CGM_AC9_DC0)

Table 478. CGM_AC9_DC0 field descriptions

Field	Description
0 DE	Divider Enable 0 Disable auxiliary clock 9 divider 0 1 Enable auxiliary clock 9 divider 0
10:15 DIV	Divider Division Value The resultant RTI_CLK will have a period 'DIV + 1' times that of auxiliary clock 9. If DE is set to 0 (divider 0 is disabled), any write access to the DIV field is ignored and the RTI_CLK remains disabled.

30.3.1.62 Auxiliary Clock 11 Select Control Register (CGM_AC11_SC)

This register is used to select the current clock source for the following clocks:

- undivided: (unused)
- divided by auxiliary clock 11 divider 0: CAN_CLK1

Address: 0x0360 Access: User read/write, Supervisor read/write, Test read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	SELCTL				0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0

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

Figure 369. Auxiliary Clock 11 Select Control Register (CGM_AC11_SC)

Table 479. CGM_AC11_SC field descriptions

Field	Description
4:7 SELCTL	Auxiliary Clock 1 Source Selection Control This value selects the current source for auxiliary clock 1. **Note:** Status of respective clock source must be ensured before selection. 0000 Reserved 0001 XOSC 0010 PLL0 PHI 0011 Reserved ... 1111 Reserved

30.3.1.63 Auxiliary Clock 11 Select Status Register (CGM_AC11_SS)

This register provides the current auxiliary clock 11 source selection.

Address: 0x0364 Access: User read, Supervisor read, Test read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	SELSTAT				0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0

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

Figure 370. Auxiliary Clock 11 Select Status Register (CGM_AC11_SS)

Table 480. CGM_AC11_SS field descriptions

Field	Description
4:7 SELSTAT	Auxiliary Clock 11 Source Selection Status This value indicates the current source for auxiliary clock 11. 0000 Reserved 0001 XOSC 0010 PLL0 PHI 0011 Reserved ... 1111 Reserved

30.3.1.64 Auxiliary Clock 11 Divider 0 Configuration Register (CGM_AC11_DC0)

This register controls auxiliary clock 11 divider 0 when the “CAN jitter” feature is not enabled (see *Section 30.4.3.1: Divider jitter injection*).

Note: Byte and half-word write accesses are not allowed for this register. Such an access will not result in an exception; however, the value will not be loaded with the new value.

Address: 0x0368 Access: User read/write, Supervisor read/write, Test read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	DE	0	0	0	0	0	0	0	0	0	DIV					
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

Figure 371. Auxiliary Clock 11 Divider 0 Configuration Register (CGM_AC11_DC0)

Table 481. CGM_AC11_DC0 field descriptions

Field	Description
0 DE	Divider Enable 0 Disable auxiliary clock 11 divider 0 1 Enable auxiliary clock 11 divider 0
10:15 DIV	Divider Division Value The resultant CAN_CLK1 will have a period 'DIV + 1' times that of auxiliary clock 11. If DE is set to 0 (divider 0 is disabled), any write access to the DIV field is ignored and the CAN_CLK1 remains disabled.

30.3.1.65 Auxiliary Clock 12 Select Control Register (CGM_AC12_SC)

This register is used to select the current clock source for the following clocks:

- undivided: (unused)
- divided by auxiliary clock 12 divider 0: DSPI_CLK0
- divided by auxiliary clock 12 divider 1: DSPI_CLK1
- divided by auxiliary clock 12 divider 2: LIN_CLK0
- divided by auxiliary clock 12 divider 3: LIN_CLK1
- divided by auxiliary clock 12 divider 4: PER_CLK0

Address: 0x0380 Access: User read/write, Supervisor read/write, Test read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	SELCTL				0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

Figure 372. Auxiliary Clock 12 Select Control Register (CGM_AC12_SC)

Table 482. CGM_AC12_SC field descriptions

Field	Description
4:7 SELCTL	Auxiliary Clock 12 Source Selection Control Selects the source for auxiliary clock 12. **Note:** Status of respective clock source must be ensured before selection. 0000 IRCOSC 0001 XOSC 0010 PLL0 PHI 0011 Reserved ... 1111 Reserved

30.3.1.66 Auxiliary Clock 12 Select Status Register (CGM_AC12_SS)

This register provides the current auxiliary clock 12 source selection.

Address: 0x0384 Access: User read, Supervisor read, Test read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	SELSTAT				0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

Figure 373. Auxiliary Clock 12 Select Status Register (CGM_AC12_SS)

Table 483. CGM_AC12_SS field descriptions

Field	Description
4:7 SELSTAT	Auxiliary Clock 12 Source Selection Status This value indicates the current source for auxiliary clock 12. 0000 IRCOSC 0001 XOSC 0010 PLL0 PHI 0011 Reserved ... 1111 Reserved

30.3.1.67 Auxiliary Clock 12 Divider 0 Configuration Register (CGM_AC12_DC0)

This register controls auxiliary clock 12 divider 0. See *Section 30.4.3.2: Fractional divider details*, for divider characteristics.

Note: Byte and half-word write accesses are not allowed for this register. Such an access will not result in an exception, however the value will not be loaded with the new value.

Address: 0x0388 Access: User read/write, Supervisor read/write, Test read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	DE	0	0	0	0	0	0	DIV								
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	DIV_FMT	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 374. Auxiliary Clock 12 Divider 0 Configuration Register (CGM_AC12_DC0)

Table 484. CGM_AC12_DC0 field descriptions

Field	Description
0 DE	Divider Enable 0 Disable auxiliary clock 12 divider 0 1 Enable auxiliary clock 12 divider 0
7:15 DIV	Divider Division Value The resultant DSPI_CLK0 will have a period 'DIV + 1' times that of auxiliary clock 0. If DE is set to 0 (divider 0 is disabled), any write access to the DIV field is ignored and the DSPI_CLK0 remains disabled.
30:31 DIV_FMT	Divider Value Format - This value indicates the format of the division value for calculating the division factor 00 Division factor = (DIV + 1) / 1 01 Division factor = (DIV + 1) / 10 10 Division factor = (DIV + 1) / 100 11 Division factor = (DIV + 1) / 1,000

30.3.1.68 Auxiliary Clock 12 Divider 1 Configuration Register (CGM_AC12_DC1)

This register controls auxiliary clock 12 divider 1.

Note: Byte and half-word write accesses are not allowed for this register. Such an access will not result in an exception; however, the value will not be loaded with the new value.

Address: 0x038C Access: User read/write, Supervisor read/write, Test read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	DE	0	0	0	0	0	0	0	0	0	0	0	DIV			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

Figure 375. Auxiliary Clock 12 Divider 1 Configuration Register (CGM_AC12_DC1)

Table 485. CGM_AC12_DC1 field descriptions

Field	Description
0 DE	Divider Enable 0 Disable auxiliary clock 12 divider 1 1 Enable auxiliary clock 12 divider 1
12:15 DIV	Divider Division Value The resultant DSPI_CLK1 will have a period 'DIV + 1' times that of auxiliary clock 0. If DE is set to 0 (divider 0 is disabled), any write access to the DIV field is ignored and the DSPI_CLK1 remains disabled.

30.3.1.69 Auxiliary Clock 12 Divider 2 Configuration Register (CGM_AC12_DC2)

This register controls auxiliary clock 12 divider 1.

Note: Byte and half-word write accesses are not allowed for this register. Such an access will not result in an exception; however, the value will not be loaded with the new value.

Address: 0x0390 Access: User read/write, Supervisor read/write, Test read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	DE	0	0	0	0	0	0	0	0	0	0	0	DIV			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

Figure 376. Auxiliary Clock 12 Divider 2 Configuration Register (CGM_AC12_DC2)

Table 486. CGM_AC12_DC2 field descriptions

Field	Description
0 DE	Divider Enable 0 Disable auxiliary clock 12 divider 2 1 Enable auxiliary clock 12 divider 2
12:15 DIV	Divider Division Value The resultant LIN_CLK0 will have a period ‘DIV + 1’ times that of auxiliary clock 0. If DE is set to 0 (divider 0 is disabled), any write access to the DIV field is ignored and the LIN_CLK0 remains disabled.

30.3.1.70 Auxiliary Clock 12 Divider 3 Configuration Register (CGM_AC12_DC3)

This register controls auxiliary clock 12 divider 3.

Note: Byte and half-word write accesses are not allowed for this register. Such an access will not result in an exception; however, the value will not be loaded with the new value.

Address: 0x0394 Access: User read/write, Supervisor read/write, Test read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	DE	0	0	0	0	0	0	0	0	0	0	0	DIV			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

Figure 377. Auxiliary Clock 12 Divider 3 Configuration Register (CGM_AC12_DC3)

Table 487. CGM_AC12_DC3 field descriptions

Field	Description
0 DE	Divider Enable 0 Disable auxiliary clock 12 divider 3 1 Enable auxiliary clock 12 divider 3
12:15 DIV	Divider Division Value The resultant LIN_CLK1 will have a period 'DIV + 1' times that of auxiliary clock 0. If DE is set to 0 (divider 0 is disabled), any write access to the DIV field is ignored and the LIN_CLK1 remains disabled.

30.3.1.71 Auxiliary Clock 12 Divider 4 Configuration Register (CGM_AC12_DC4)

This register controls auxiliary clock 12 divider 4.

Note: Byte and half-word write accesses are not allowed for this register. Such an access will not result in an exception; however, the value will not be loaded with the new value.

Address: 0x0398 Access: User read/write, Supervisor read/write, Test read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	DE	0	0	0	0	0	0	0	0	0	0	0	DIV			
W																
Reset	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

Figure 378. Auxiliary Clock 12 Divider 4 Configuration Register (CGM_AC12_DC4)

Table 488. CGM_AC12_DC4 field descriptions

Field	Description
0 DE	Divider Enable 0 Disable auxiliary clock 12 divider 4 1 Enable auxiliary clock 12 divider 4
12:15 DIV	Divider Division Value The resultant PER_CLK0 will have a period 'DIV + 1' times that of auxiliary clock 0. If DE is set to 0 (divider 0 is disabled), any write access to the DIV field is ignored and the PER_CLK0 remains disabled.

30.4 Functional description

30.4.1 System clock generation

Figure 280: Clock generation(1)(2) (part 1 of 2) shows the block diagram of the system clock generation logic. The MC_ME provides the system clock select and switch mask (see MC_ME chapter for more details), and the MC_RGM provides the safe clock request (see

MC_RGM chapter for more details). The safe clock request forces the selector to select the IRCOSC as the system clock and to ignore the system clock select.

30.4.1.1 System clock source selection

During normal operation, the system clock selection is controlled

- on a SAFE mode or reset event, by the MC_RGM
- otherwise, by the MC_ME

30.4.1.2 Progressive system clock switching

In order to prevent sudden voltage drops and overshoots due to frequency and load changes, the MC_ME requests the MC_CGM to ramp the system clock frequency down or up or both, based on the power level values of the current and target modes. During ramp-down, the rate of the frequency change is based on the CGM_PCS_SDUR, CGM_PCS_DIVC*n*, and CGM_PCS_DIVE*n* registers, where n corresponds to the current system clock source selection. During ramp-up, the rate of the frequency change is based on the CGM_PCS_SDUR, CGM_PCS_DIVC*n*, and CGM_PCS_DIVS*n* registers, where *n* corresponds to the target system clock source selection.

30.4.1.2.1 Generic clock change requirements

For a maximum allowed change of the frequency (f_{chg}) and for a given source clock frequency f_{src} (current or target clock source), the maximum allowed frequency change rate a_{max} is given in *Equation 14*.

Equation 14

$$a_{max} = \frac{f_{chg}}{f_{src}}$$

30.4.1.2.2 Configuration of CGM_PCS_SDUR

The switch duration field CGM_PCS_SDUR[SDUR] defines the duration of one system clock switch step in terms of IRCOSC. After the expiration of this time, the module changes the clock divider value which changes the frequency of the system clock.

30.4.1.2.3 Configuration of CGM_PCS_DIVC*n*[RATE]

For a given maximum allowed frequency change rate a_{max} the value d to be programmed into the PCS_DIVC*n*[RATE] register is given in *Table 489*.

Table 489. MC_CGM CGM_PCS_DIVC*n*[RATE] values

a_{max}	d	CGM_PCS_DIVC*n*[RATE]
0.05	0.012	12
0.10	0.048	48
0.15	0.112	112
0.20	0.184	184

30.4.1.2.4 Generic clock change properties

The number *k* of required steps to reach or leave the divided system clock with frequency f_{div} from the source clock with frequency f_{src} is given in *Equation 15*.

Equation 15

$$k = \left\lceil 0.5 + \sqrt{0.25 - \frac{2\left(1 - \frac{f_{src}}{16MHz}\right)}{d}} \right\rceil$$

30.4.1.2.5 Clock ramp-down

The clock ramp-down starts with the divider value 1 and with the given divider increment value PCD_DIVC*n*[RATE] and ends with the given divider value PCS_DIVE*n*[DIVE].

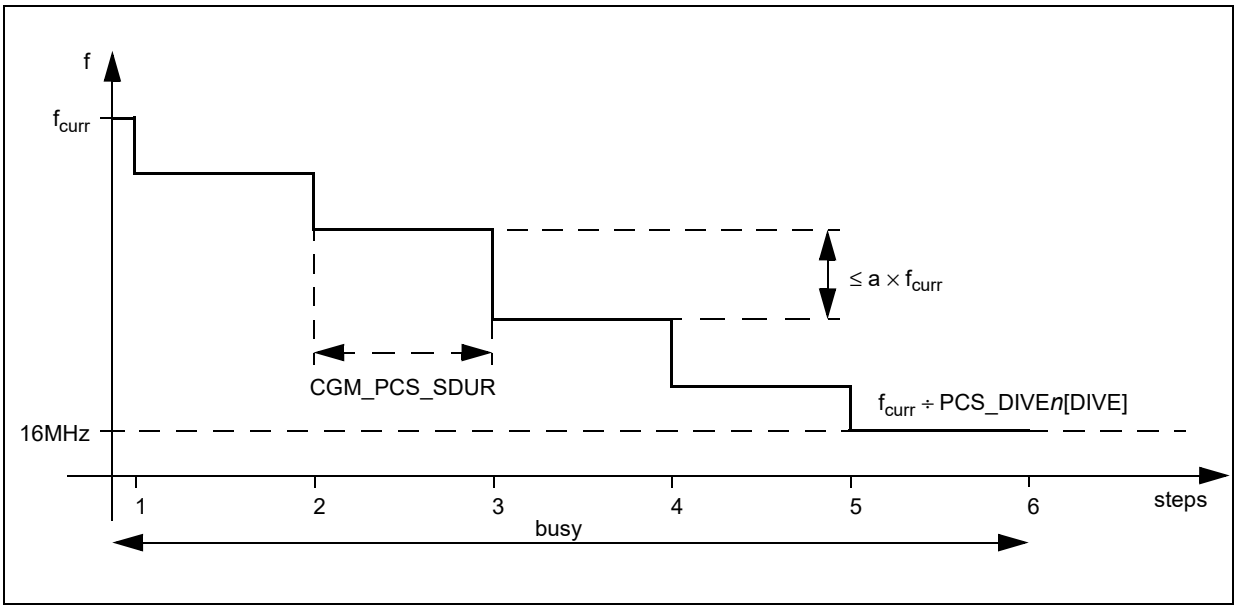
The divider end value for clock ramp-down is given in *Equation 16*.

Equation 16

$$\textbf{PCS_DIVEn[DIVE]} = \frac{f_{src}}{16MHz} \times 1000 - 1$$

Where f_{curr} is the frequency of the currently selected system clock source.

Figure 379. MC_CGM System Clock Ramp-Down Timing (k = 6 example)



30.4.1.2.6 Clock ramp-up

The clock ramp-up starts with the given divider value PCS_DIVS*n*[DIVS] and with the given divider decrement value PCD_DIVC*n*[INIT] and ends with the divider value 1.

The initial divider change start value PCS_DIVC*n*[INIT] for system clock ramp-up is given in *Equation 17*.

Equation 17

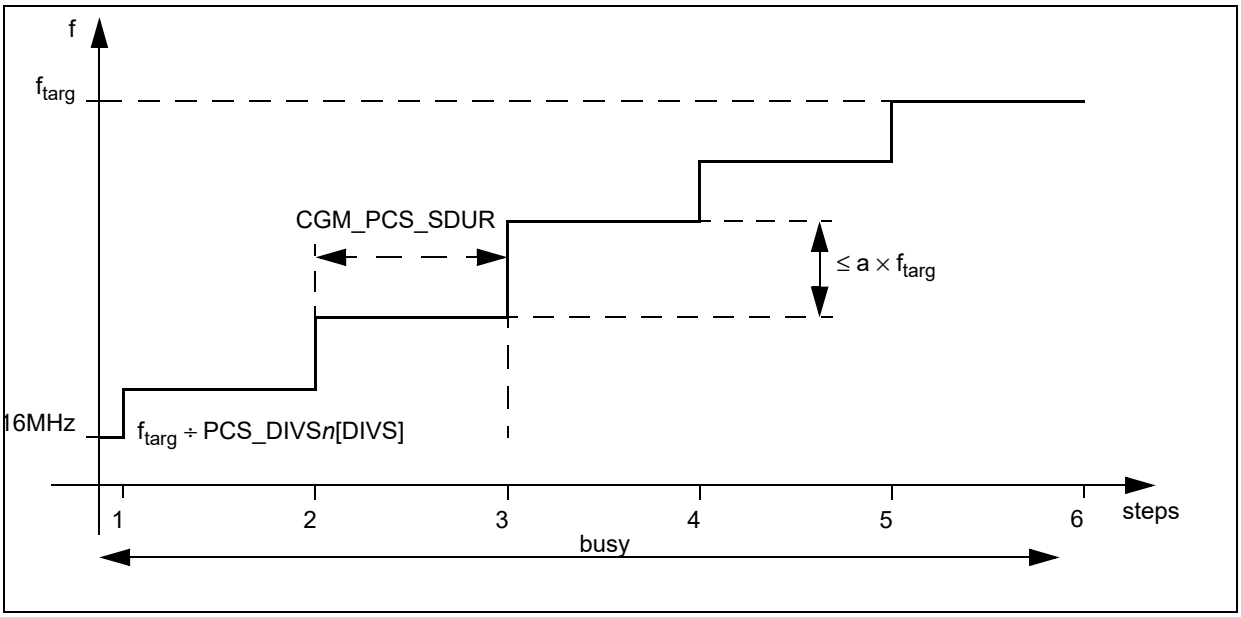
$$\text{PCS_DIVCn[INIT]} = d \times k \times 1000 - 1$$

Where k is calculated from *Equation 15* using the target system clock source frequency f_{targ}. The divider start value for clock ramp-up is given in *Equation 18*.

Equation 18

$$\text{PCS_DIVSn[DIVS]} = \left(1 + d\frac{k(k+1)}{2}\right) \times 1000 - 1$$

Figure 380. MC_CGM System Clock Ramp-Up Timing (k = 6 example)



30.4.1.3 System clock disable

During the TEST mode, the system clock can be disabled by the MC_ME.

30.4.1.4 System clock dividers

The MC_CGM generates the following derived clocks from the system clock. :

- SYS_CLK - controlled by the CGM_SC_DC0 register
- HPBM_CLK - controlled by the CGM_SC_DC1 register
- PBRIDGE_CLK - controlled by the CGM_SC_DC2 register
- FBRIDGE_CLK - controlled by the CGM_SC_DC3 register
- PFBRIDGE_CLK - controlled by the CGM_SC_DC4 register

30.4.1.4.1 System clock divider synchronization

The system clock dividers are synchronized to each other such that the rising edges of the lower frequency clocks are aligned with those of the higher frequency clocks. This, however, imposes limitations on the division factors which can be used. In order for the

synchronization to work properly, each division factor must be selected such that its value is an integer multiple of each division factor that has a lower value.

The allowed combinations of division factors is given in *Table 389: Maximum system level clock frequencies*.

30.4.1.4.2 System clock divider update triggering

As opposed to the auxiliary clock dividers, which update immediately on a configuration change, the system clock dividers are updated only with the next software triggered mode change. This is to ensure that the system clock dividers are updated simultaneously and that no intermediate inconsistent configurations can occur. It also ensures that the divider configurations are aligned with the target mode's system clock selection.

The system clock divider configuration update sequence is as follows:

1. Configure the target mode in the MC_ME to select the desired system clock source.
2. Configure the CGM_SC_DC0-4 registers as desired and aligned with the target system clock source frequency making sure that the dividers are configured consistently as described in *Section 30.4.1.4.1: System clock divider synchronization*.
3. Request a mode change in the MC_ME to the desired target mode.
4. When mode change has completed and the system clock switching has been successful, the new system clock divider configuration will be in effect.

See *Chapter 69: Mode Entry Module (MC_ME)*, for details on how to configure modes and make mode change requests.

30.4.2 Auxiliary clock generation

Figure 280: Clock generation(1)(2) (part 1 of 2) and *Figure 281: Clock generation (part 2 of 2) - PSI5 clock tree* show the block diagrams of the generation logic for the various auxiliary clocks.

See the following sections for auxiliary clock selection control:

- *Section 30.3.1.32: Auxiliary Clock 0 Select Control Register (CGM_AC0_SC)*
- *Section 30.3.1.38: Auxiliary Clock 1 Select Control Register (CGM_AC1_SC)*
- *Section 30.3.1.43: Auxiliary Clock 3 Select Control Register (CGM_AC3_SC)*
- *Section 30.3.1.45: Auxiliary Clock 4 Select Control Register (CGM_AC4_SC)*
- *Section 30.3.1.50: Auxiliary Clock 6 Select Control Register (CGM_AC6_SC)*
- *Section 30.3.1.53: Auxiliary Clock 7 Select Control Register (CGM_AC7_SC)*
- *Section 30.3.1.56: Auxiliary Clock 8 Select Control Register (CGM_AC8_SC)*
- *Section 30.3.1.59: Auxiliary Clock 9 Select Control Register (CGM_AC9_SC)*
- *Section 30.3.1.62: Auxiliary Clock 11 Select Control Register (CGM_AC11_SC)*
- *Section 30.3.1.65: Auxiliary Clock 12 Select Control Register (CGM_AC12_SC)*

30.4.2.1 Auxiliary clock dividers

The MC_CGM generates the following derived clocks.

Table 490. Derived clocks

Derived clock	Auxiliary Clock	Register
GTM_CLK	AC0	CGM_AC0_DC0
SDADC_CLK		CGM_AC0_DC1
SARADC_CLK		CGM_AC0_DC2
EMIOS_CLK		CGM_AC0_DC3
LF_REF_CLK	AC1	CGM_AC1_DC0
FRAY_CLK	AC2	CGM_AC2_DC0
SENT_CLK		CGM_AC2_DC1
PSI5_f189_CLK	AC5	CGM_AC5_DC0
PSI5_f125_CLK		CGM_AC5_DC1
PSI5_1us_CLK		CGM_AC5_DC2
PSI5_S_TS_CLK		CGM_AC5_DC2, CGM_AC5_CDC0
PSI5_S_DDSR0_CLK		CGM_AC5_DC2, CGM_AC5_CDC1, CGM_AC5_CDC10
PSI5_S_DDSR1_CLK		CGM_AC5_DC2, CGM_AC5_CDC1, CGM_AC5_CDC11
PSI5_S_DDSR2_CLK		CGM_AC5_DC2, CGM_AC5_CDC1, CGM_AC5_CDC12
PSI5_S_DDSR3_CLK		CGM_AC5_DC2, CGM_AC5_CDC1, CGM_AC5_CDC13
PSI5_S_WD0_CLK		CGM_AC5_DC2, CGM_AC5_CDC2, CGM_AC5_CDC20
PSI5_S_WD1_CLK		CGM_AC5_DC2, CGM_AC5_CDC2, CGM_AC5_CDC21
PSI5_S_WD2_CLK		CGM_AC5_DC2, CGM_AC5_CDC2, CGM_AC5_CDC22
PSI5_S_WD3_CLK		CGM_AC5_DC2, CGM_AC5_CDC2, CGM_AC5_CDC23
SYSCLK0 pin clock	AC6	CGM_AC6_DC0
SYSCLK1 pin clock	AC7	CGM_AC7_DC0
CAN_CLK0	AC8	CGM_AC8_DC0
RTI_CLK	AC9	CGM_AC9_DC0
CAN_CLK1	AC11	CGM_AC11_DC0
DSPI_CLK0	AC12	CGM_AC12_DC0
DSPI_CLK1		CGM_AC12_DC1
LIN_CLK0 (Fast LIN)		CGM_AC12_DC2
LIN_CLK1 (Slow LIN)		CGM_AC12_DC3
PER_CLK0		CGM_AC12_DC4

30.4.3 Dividers functional description

Dividers are used for the generation of divided system and peripheral clocks. The MC_CGM has the following control registers for built-in dividers:

- System clock divider configuration registers
  - CGM_SC_DC0-4 (*Section 30.3.1.27* to *Section 30.3.1.31*)
- Auxiliary clock divider configuration registers
  - CGM_AC0_DC0-3 (*Section 30.3.1.34* to *Section 30.3.1.37*)
  - CGM_AC1_DC0 (*Section 30.3.1.40*)
  - CGM_AC2_DC0-1(*Section 30.3.1.41* and *Section 30.3.1.42*)
  - CGM_AC5_DC0-2 (*Section 30.3.1.47* to *Section 30.3.1.49*), CGM_AC5_CDC0-2, CGM_AC5_CDC10-13, CGM_AC5_CDC20-23 (Auxiliary clock 5 cascaded divider configuration registers, see *Section 30.3.1.1* to *Section 30.3.1.11*)
  - CGM_AC6_DC0 (*Section 30.3.1.52*)
  - CGM_AC7_DC0 (*Section 30.3.1.55*)
  - CGM_AC8_DC0 (*Section 30.3.1.58*)
  - CGM_AC9_DC0 (*Section 30.3.1.61*)
  - CGM_AC11_DC0 (*Section 30.3.1.64*)
  - CGM_AC12_DC0-4 (*Section 30.3.1.67* to *Section 30.3.1.71*)

The reset value of all counters is ‘1’. If a divider has its DE bit in the respective configuration register set to ‘0’ (the divider is disabled), any value in its DIV field is ignored.

30.4.3.1 Divider jitter injection

The clock dividers for the FRAY_CLK and the CAN_CLK0 inject a jitter so that off-chip communication via the associated peripherals is not possible. For details on enabling this feature, see the “Production Disable” and “CAN Jitter Enable” sections of the PASS chapter.

In addition, PLL0 PHI is always selected to source the FRAY_CLK when jitter injection is enabled.

30.4.3.2 Fractional divider details

The clock divider used to generate the PSI5_f189_CLK and the DSPI_CLK0 divides the selected clock source frequency by a fractional value given by the DIV and DIV_FMT fields of the CGM_AC5_DC0 and the CGM_AC12_DC0 registers.

Equation 19

$$F_{\text{divided clock}} = F_{\text{source clock}} \div ((\mathbf{DIV} + 1) \div 10^{\mathbf{DIV_FMT}})$$

The divided clock frequency is actually an average frequency, since the divider uses only the available rising and falling edges of the source clock. Therefore, the actual frequency switches between two frequencies which are the two frequencies closest to the desired frequency (above and below) which can be extracted by a division factor *n*/2, where *n* is an integer greater than 1. The switching between the two frequencies is such that the resultant clock’s rising edge never deviates from the ideal clock’s rising edge by more than 25% of

the source clock period and that the resultant average frequency over DIV + 1 source clock periods (= 10^{DIV_FMT} divided clock periods) is the desired frequency.

Example 1 Fractional clock division

Desired division factor = 2.3

DIV = 22

DIV_FMT = 1

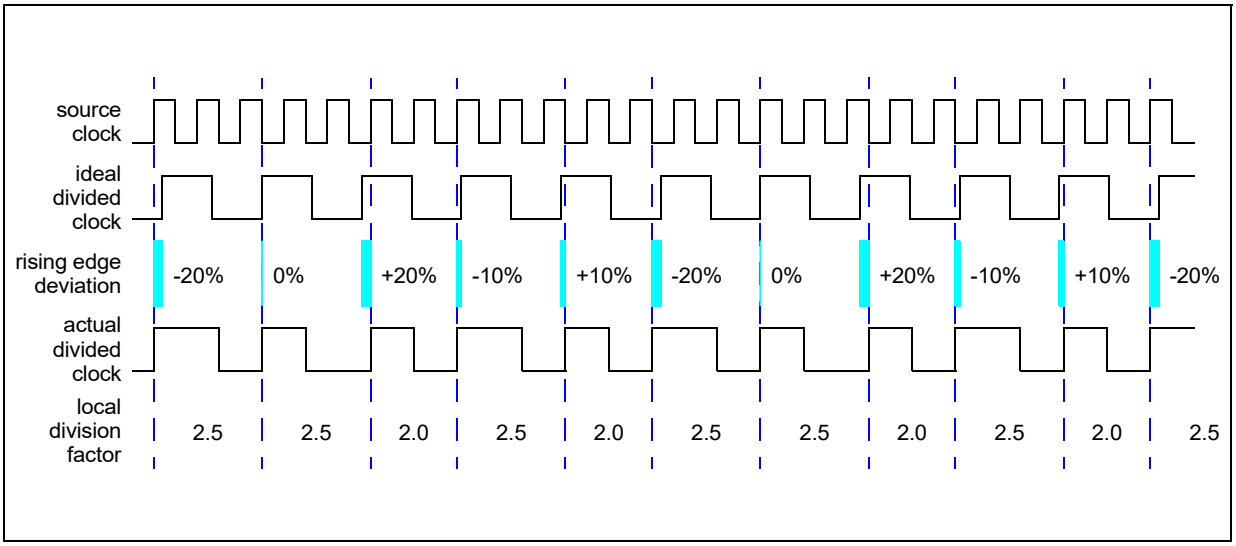


Figure 381. MC_CGM fractional division example waveform (divide by 2.3)

Equation 20

$$F_{\text{divided clock}}\big|_{max} = F_{\text{source clock}} \div 2.0$$

Equation 21

$$F_{\text{divided clock}}\big|_{min} = F_{\text{source clock}} \div 2.5$$

Equation 22

$$F_{\text{divided clock}}\big|_{ave} = F_{\text{source clock}} \div \frac{((2.5 \times 6) + (2.0 \times 4))}{10}$$

Therefore,

Equation 23

$$F_{\text{divided clock}}\big|_{ave} = F_{\text{source clock}} \div 2.3$$

As can be seen in *Figure 381*, the rising edge of the actual divided clock is 'snapped' to the rising or falling edge of the source clock closest to the rising edge of the ideal divided clock. This results in a theoretical maximum deviation, or jitter, of ±25% of the source clock period. In this particular case, the maximum deviation is ±20% of the source clock period.

30.4.3.3 Divider Update Triggering

There are two methods for updating the dividers associated with the system clock and each auxiliary clock: immediate and register triggered. Which method is used for which set of dividers is controlled by the CGM_DIV_UPD_TYPE register.

Regardless of which method is selected for a given group of dividers, once an update of one of the dividers of that group has been triggered, the associated bit in the CGM_UPD_STAT register is set. This bit is then cleared when all the dividers of that group have completed the update process and are running with the current configuration.

30.4.3.3.1 Immediate Divider Update

If a divider group has been configured to be updated immediately (the corresponding CGM_DIV_UPD_TYPE register bit = '0'), the configuration of a divider in that group will be applied immediately on the write to that divider's configuration register. The actual update will take some time to actually reach the divider output due to the clock domain crossing and edge alignment mechanisms.

30.4.3.3.2 Software-triggered divider update

If a divider group has been configured to be updated via a software trigger (the corresponding CGM_DIV_UPD_TYPE register bit = '1'), the configuration of a divider in that group will not be applied immediately on the write to that divider's configuration register. Instead, the configuration in the corresponding configuration register will be applied only when software writes any value to the CGM_DIV_UPD_TRIG register. The actual update will take some time to actually reach the divider output due to the clock domain crossing and edge alignment mechanisms.

In the case of the system clock dividers, if the SYS_DIV_RATIO_CHNG bit in the CGM_SC_DIV_RC register = '1', the crossbar switch is requested to halt its ongoing transactions, and only after the crossbar switch has acknowledged the request does the update of the system clock dividers take place. Once these updates have completed, the halt request to the crossbar is deasserted.

Note: If halting the crossbar can cause a bus master to starve and thus lead to lost data or other problems, it is recommended to vary the system clock divider configurations without changing the ratios and to set CGM_SC_DIV_RC[SYS_DIV_RATIO_CHNG] to '0'.

Note: In order to ensure a clean clock divider configuration update, software must check that all ongoing updates have already completed (that is all relevant bits in the CGM_DIV_UPD_STAT register are '0') before triggering a new update via the CGM_DIV_UPD_TRIG register.

31 OSC digital interface (XOSC)

31.1 Introduction

The XOSC digital interface is used to control the on-chip oscillator (XOSC) and provides the register interface for the programmable features. Selection of the XOSC as a source clock is controlled the Clock Generation module (MC_CGM). The XOSC is enabled and disabled by configuring the Mode Entry module (MC_ME).

The programmable features of the XOSC digital interface are as follows:

- Oscillator power-down control and status
- Oscillator startup delay
- Oscillator clock available interrupt
- Oscillator bypass mode control

31.2 Functional description

The XOSC digital interface provides control of the on-chip oscillator and the register interface for the mode and startup delay control. The oscillator provides an output clock that can used as an input to PLL(s) and can be selected as the reference clock for many of the peripherals on the device.

Table 491 shows the possible configurations of the XOSC.

Table 491. XOSC configurations

OSC EN	XOSC_CTL[OSCBYP]	XTAL	EXTAL	XOSC output	XOSC mode
0	0	No crystal, high-impedance	No crystal, high-impedance	0	Power down, IDDQ
0	1	x	external clock	Undefined	Reserved, OSC Disabled
1	1	x	external clock	XTALOUT	Bypass, OSC Enabled
1	0	crystal	crystal	XTALOUT	Normal, OSC Enabled

31.2.1 Oscillator power-down control and status

Enabling and disabling of the oscillator is done in the Mode Entry (MC_ME) module (see “Mode Entry Module (MC_ME)” chapter). The ME_<*mode*>_MC[XOSCON] controls the power-down of oscillator based on the current device mode (see “MC_ME Memory Map” table for available *modes*). The status of the XOSC is shown in the MC_ME_GS[S_XOSC].

31.2.2 Oscillator startup delay

A bit in the UTEST Flash memory determines whether the oscillator is enabled at power-up of the device (if the DCF client miscellaneous has been programmed, otherwise the valid configuration is the value present in *Chapter 9: Device Configuration Format (DCF) Records*, *Table 129: DCF client list*). If enabled, the oscillator is started after Phase3 of the

device reset sequence. In order to allow the oscillator to reach full amplitude before use, the internal counter (OSCCNT) is started when the device exits reset. The counter is driven by the oscillator output clock. When the counter reaches XOSC_CTL[EOCV] × 512, a signal is sent to the MC_ME module, allowing a mode switch to the oscillator (when used as a system clock or an input to PLL(s)).

The default value for this field after reset depends on implementation and can be found in the XOSC section of the Device Configuration chapter. After reset, software can clear or change the EOCV field in order to shorten or lengthen the delay until the oscillator is ready.

Note: It is advised that if the EOCV value is modified it should be done as early as possible in user code. This will ensure that the new value will be used before OSCCNT reaches the count value.

31.2.3 Oscillator clock available interrupt

An interrupt can be generated when the count value is reached (OSCCNT = XOSC_CTL[EOCV] × 512). The XOSC_CTL[I_OSC] will be set and an interrupt will be generated if the interrupt mask is set (XOSC_CTL[M_OSC] = 1).

31.2.4 Oscillator bypass mode

The oscillator circuit can be bypassed by setting XOSC_CTL[OSCBYP]. This field can only be set by software, and is only cleared by a system reset. In bypass mode, the oscillator is disabled, and the input clock on the EXTAL pins is level-shifted and driven to the logic on the device. The bypass configuration is independent of the power-down mode of the oscillator.

Note: The external oscillator must be running before the XOSC is turned off. If XOSC is being turned on, the status will be updated only after the clock starts toggling.

31.2.4.1 Automatic Level Control (ALC)

The function of the ALC block is to provide high drive current during oscillator start up, and reduce current after oscillation in order to reduce power, distortion, RFI; and to avoid overdriving the crystal. ALC block can be controlled via DCF by configuring the UTEST miscellaneous bit 17 [XOSC_ALC_DIS]. Comparing the configurations with ALC disabled respect the case with ALC enabled, there are the following attributes:

- Higher oscillation amplitudes at crystal
- Distorted oscillation with several harmonics
- Increased EMI
- Higher crystal drive level
- Less susceptible to noise

The ALC enable configuration is suitable for most applications. The amplitude of oscillations is lower, hence less distortion in oscillations, less EMI, and also less crystal drive level. Feedback resistor is integrated thus, it does not require external resistors by offering a lower cost solution and simpler PCB design.

Note: It is recommended to make ALC feature enabled (that is, XOSC_ALC_DIS = 0) for low dissipation in crystal (crystal drive level).

Consult with crystal supplier for determining the need of extra resistance in XTAL to avoid crystal overdrive.

31.3 Memory map and register definition

Table 492 shows the memory map of the XOSC registers.

Table 492. XOSC memory map

Offset	Register	Location
0x00	XOSC Control Register (XOSC_CTL)	*Section 31.3.1.1*

31.3.1 Register descriptions

Note: Module generates transfer error (ips_xfr_err) only for 32 bytes address space (00 : 1F). If the address space allocated for this module exceeds 32 bytes, then transfer error has to be managed at SoC level for the remaining space.

31.3.1.1 XOSC Control register (XOSC_CTL)

Offset: 0x00 Access: Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	OSCBYP(1)	0	0	0	0	0	0	0	EOCV(1)							
W																
Reset	–	0	0	0	0	0	0	0	–	–	–	–	–	–	–	–

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	M_OSC(1)	0	0	0	0	0	0	0	I_OSC	0	0	0	0	0	0	0
W									w1c							
Reset	–	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. See the Clocking chapter for reset value.

Figure 382. XOSC Control register (XOSC_CTL)

Table 493. XOSC_CTL field descriptions

Field	Description
0 OSCBYP	Crystal Oscillator bypass This bit specifies whether the oscillator should be bypassed or not. Software can only set this bit. System reset is needed to reset this bit. 0 Oscillator output is used as root clock. 1 EXTAL is used as root clock.
8:15 EOCV	End of Count Value These bits specify the end of count value to be used for comparison by the oscillator stabilization counter OSCCNT after reset or whenever it is switched on from the off state. This counting period ensures that external oscillator clock signal is stable before it can be selected by the system. When oscillator counter reaches the value EOCV×512, oscillator available interrupt request is generated. The OSCCNT counter is kept under reset if oscillator bypass mode is selected.

Table 493. XOSC_CTL field descriptions (continued)

Field	Description
16 M_OSC	Crystal oscillator clock interrupt mask This bit masks the I_OSC interrupt bit. 0 Crystal oscillator clock interrupt is masked. 1 Crystal oscillator clock interrupt is enabled.
24 I_OSC	Crystal oscillator clock interrupt This bit can only be set by hardware when OSCCNT counter reaches the count value EOCV $\times$ 512. It is cleared by software by writing 1. 0 No oscillator clock interrupt occurred. 1 Oscillator clock interrupt pending.

32 IRCOSC digital interface

32.1 Introduction

The Internal RC Oscillator digital interface (IRCOSC) controls the internal 16 MHz RC oscillator system. This oscillator system includes the main internal RC oscillator (MRC)

It provides access to trimming information used for safety implementations.

32.2 Functional description

The IRCOSC provides a high frequency (f_{MRC}) clock with a nominal frequency of 16 MHz. After deassertion of reset, the output clock is available after a short time (see Data Sheet for timing information) which is required for stabilization. The IRCOSC status indication is found in the MC_ME_GS[S_IRC] field (see the “Mode Entry Module MC_ME” chapter).

During a power on reset sequence the following are initialized from TEST Flash memory:

- Native trimming bits
- Fine trimming bits

For safety reasons, these bits can be accessed through the IRCOSC Native Trimming register (NT). The main IRCOSC output frequency can be trimmed by configuring IRCOSC_CTL[USER_TRIM[5:0]]. The USER_TRIM bits can be programmed to modify internal capacitor/resistor values to match output clock frequency with δf_{var_SW}, as defined in the Data Sheet.

32.3 Memory map and register definition

Table 494 shows the memory map of the IRCOSC registers.

Table 494. IRCOSC memory map

Offset (hex)	Register	Width (in bits)	Access	Reset Value (hex)	Location
0x00	IRCOSC Control Register (IRCOSC_CTL)	32	R/W	0x0000_0000	*Section 32.3.1.1*
0x04	IRCOSC Native Trimming Register (IRCOSC_NT)	32	R	0xXX00_0XXX(1)	*Section 32.3.1.2*

1. See register section for specific reset values.

32.3.1 Register descriptions

32.3.1.1 IRCOSC control register (IRCOSC_CTL)

Offset: 0x0000 Access: Read/Write(1)

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	USER_TRIM					
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

1. IRCOSC_CTL is writable only in supervisor mode.

Figure 383. IRCOSC control register (IRCOSC_CTL)

Table 495. IRCOSC_CTL field descriptions

Field	Description
10:15 USER_TRIM	User trimming bits TRIM_BIT_USER[5:0] with respect to nominal factory frequency The MSB of the TRIM_BIT_USER bits is used to determine whether the frequency will be increased or decreased: – If MSB = 0 then a change in TRIM_BIT_USER[4:0] will decrease the frequency, and likewise, – if MSB = 1 then a change in TRIM_BIT_USER[4:0] will increase the frequency. TRIM_BIT_USER frequency trimming calculation: 110000 (-16) F1 + 5.16% 110001 (-15) F1 + 4.83% 110010 (-14) F1 + 4.50% 110011 (-13) F1 + 4.17% 110100 (-12) F1 + 3.84% 110101 (-11) F1 + 3.51% 110110 (-10) F1 + 3.19% 110111 (-9) F1 + 2.86% 111000 (-8) F1 + 2.54% 111001 (-7) F1 + 2.22% 111010 (-6) F1 + 1.89% 111011 (-5) F1 + 1.57% 111100 (-4) F1 + 1.25% 111101 (-3) F1 + 0.94% 111110 (-2) F1 + 0.62% 111111 (-1) F1 + 0.31% 000000 (0) F1 = 16 MHz 000001 (1) F1 - 0.29% 000010 (2) F1 - 0.58% 000011 (3) F1 - 0.87% 000100 (4) F1 - 1.17% 000101 (5) F1 - 1.45% 000110 (6) F1 - 1.74% 000111 (7) F1 - 2.02% 001000 (8) F1 - 2.30% 001001 (9) F1 - 2.59% 001010 (10) F1 - 2.87% 001011 (11) F1 - 3.15% 001100 (12) F1 - 3.43% 001101 (13) F1 - 3.71% 001110 (14) F1 - 3.98% 001111 (15) F1 - 4.26% 010000 (16) F1 - 4.53% All other values are reserved.

32.3.1.2 IRCOSC Native Trimming register (IRCOSC_NT)

The NT register contains trim values loaded from TEST Flash memory for the IRCOSC. Its status is enabled for voltage regulator and temperature sensor. This register is read only.

Offset: 0x0004 Access: User Read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	FT_CODE						0	0	0	0	0	0	0	0
W																
Reset	0	0	—(1)	—(1)	—(1)	—(1)	—(1)	—(1)	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	REG_EN	0	0	0	RCTRIM							
W																
Reset	0	0	0	0	1	—(1)	0	0	—(1)	—(1)	—(1)	—(1)	—(1)	—(1)	—(1)	—(1)

1. Reset value is coming from factory calibration (TEST Flash).

Figure 384. IRCOSC Native Trimming register (IRCOSC_NT)

Table 496. IRCOSC_NT field descriptions

Field	Description
2:7 FT_CODE	This bitfield indicates the trimming code applied to RCOSC analog macro based on temperature measured by vtsense
20 REG_EN	Enables voltage regulator: trimming value for safety check 0 Voltage regulator is not enabled 1 Voltage regulator is enabled
24:31 RCTRIM	IRCOSC trimming value: this bitfield reflects the coarse trim and fine trim applied to RC analog macro read from DCF clients.

33 LPRC digital interface

33.1 Introduction

The LPRC digital interface controls the internal Low Power RC oscillator (LPRC). It holds control and status registers accessible for application.

33.2 Low Power RC oscillator

The Low Power RC oscillator provides a low frequency clock of 1.024 MHz with a good stability with respect to supply and temperature variation.

In all modes except STANDBY0, Low Power RC oscillator should be enabled by PMC digital interface (clearing MISC_CTRL_REG[RCOSC1M_ENB] bit). In STANDBY0 mode instead, Low Power RC oscillator is always enabled. After switch on, the output clock is available only after 8.5 us which is required for stabilization. The clock source status is updated in the S_LPRC bit of LPRC_CTL register.

The LPRC clock can be further divided by a configurable division factor in the range 1 to 32 to generate the divided clock to match system requirements. This division factor is specified by the LPRCDIV[4:0] bits of LPRC_CTL register.

The LPRC oscillator output frequency can be trimmed by LPRCTRIM[4:0] bits of LPRC_CTL register. These bits can be programmed to modify internal capacitor/resistor values so that the output clock frequency is within the +/- 1% of F_{LPRC} over the process. Once trimmed within this range using these bits, the variation over the voltage and temperature is the value given in SPC58xEx/SPC58xGx datasheet. After power on reset, the trimming bits are provided by the flash options. Only after first write access, the value specified by LPRCTRIM<5:0> bits will control the trimming.

33.3 Memory map and register definition

The LPRC registers are listed in *Table 497*.

Table 497. LPRC digital interface registers

Address offset (hex)	Register name	Location
0x0000	Low Power RC Control Register (CTL)	*Section 33.3.1.1*

33.3.1 Register description

33.3.1.1 Low Power RC Control register (CTL)

Offset: 0x0000 Access: Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	LPRCTRIM				
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	LPRCDIV					0	0	0	S_LPRC	0	0	0	0
W																
Reset	0	0	0	0	0	1	1	1	0	0	0	0	0	0	0	0

Figure 385. Low Power RC Control register (CTL)

Note: CTL register is writable only in supervisor mode.

Table 498. CTL register field descriptions

Field	Description
11:15 LPRCTRIM	Low Power RC trimming bits
19:23 LPRCDIV	Low Power RC clock division factor These bits specify the Low Power RC oscillator output clock division factor. The output clock is divided by the factor LPRCDIV+1.
27 S_LPRC	Low Power RC clock status 0 LPRC is not providing a stable clock 1 LPRC is providing a stable clock

34 OSC32K digital interface (OSC32K)

34.1 Introduction

The OSC32K digital interface controls the operation of the 32 KHz slow external crystal oscillator. It holds control and status registers accessible for application.

The programmable features of the OSC32K digital interface are as follows:

- Oscillator power-down control and status
- Oscillator clock available interrupt
- Oscillator bypass mode control
- Output clock division factors ranging from 1 to 32

34.2 Memory map and register definition

Table 499 shows the memory map of the OSC32K registers.

Table 499. XOSC32K memory map

Offset	Register	Location
0x00	OSC32K Control Register (SXOSC_CTL)	*Section 34.2.1.1*

34.2.1 Register descriptions

34.2.1.1 OSC32K Control register (SXOSC_CTL)

Offset: 0x00 Access: Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	OSCBYP(1)	0	0	0	0	0	0	0	EOCV							
W																
Reset	–	0	0	0	0	0	0	0	–	–	–	–	–	–	–	–

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	M_OSC	0	0	OSCDIV					I_OSC(2)	0	0	0	0	0	S_OSC	OSCON
W																
Reset	–	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. You can read this field, and you can write a value of “1” to it. Writing a “0” has no effect. A reset will also clear this bit.
2. You can write a value of "0" or "1" to this field. However, writing a "1" will clear this field, and writing "0" will have no effect on the field value.

Figure 386. OSC32K Control register (SXOSC_CTL)

Table 500. SXOSC_CTL register field descriptions

Field	Description
0 OSCBYP	Crystal Oscillator bypass This bit specifies whether the oscillator should be bypassed or not. Software can only set this bit. System reset is needed to reset this bit. 0 Oscillator output is used as root clock 1 EXTAL is used as root clock
8:15 EOCV	End of Count Value These bits specify the end of count value to be used for comparison by the oscillator stabilization counter OSCCNT after reset or whenever it is switched on from the off state. This counting period ensures that external oscillator clock signal is stable before it can be selected by the system. When oscillator counter reaches the value EOCV×512, oscillator available interrupt request is generated. The OSCCNT counter is kept under reset if oscillator bypass mode is selected.
16 M_OSC	Crystal oscillator clock interrupt mask This bit masks the I_OSC interrupt bit. 0 Crystal oscillator clock interrupt is masked 1 Crystal oscillator clock interrupt is enabled
19:23 OSCDIV	Crystal oscillator clock division factor These bits specify the crystal oscillator output clock division factor. The output clock is divided by the factor OSCDIV+1.

Table 500. SXOSC_CTL register field descriptions (continued)

Field	Description
24 I_OSC	Crystal oscillator clock interrupt This bit can only be set by hardware when OSCCNT counter reaches the count value EOCV × 512. It is cleared by software by writing 1. 0 No oscillator clock interrupt occurred 1 Oscillator clock interrupt pending
30 S_OSC	Crystal oscillator status 0 Crystal oscillator output clock is not stable 1 Crystal oscillator is providing a stable clock
31 OSCON	Crystal oscillator power-down control 0 Crystal oscillator is switched off 1 Crystal oscillator is switched on

34.3 Functional description

The OSC32K circuit includes an internal oscillator driver and an external crystal circuitry. It can be used as a reference clock to specific modules depending on system needs.

The OSC32K can be controlled via the SXOSC_CTL register. The OSCON bit controls the power-down while bit S_OSC provides the oscillator clock available status.

After system reset, the oscillator is put to power-down state and software has to switch on when required. Whenever the OSC32K is switched on from off state, the OSCCNT counter starts and when it reaches the value EOCV[7:0] × 512, the oscillator clock is made available to the system. Also, an interrupt pending SXOSC_CTL[I_OSC] bit is set. An interrupt will be generated if the interrupt mask bit M_OSC is set.

The oscillator circuit can be bypassed by writing SXOSC_CTL[OSCBYP] bit to ‘1’. This bit can only be set by software. A system reset is needed to reset this bit. In this bypass mode, the output clock has the same polarity as the external clock applied on the OSC32K_EXTAL pin and the oscillator status is forced to ‘1’. The bypass configuration is independent of the power-down mode of the oscillator.

Table 501 shows the truth table of different configurations of the oscillator.

Table 501. OSC32K configurations

SXOSC_CTL fields		OSC32K_XTAL	OSC32K_EXTAL	OSC32K	Oscillator mode
OSCON	OSCBYP				
0	0	No crystal, high Z	No crystal, high Z	0	Power-down, IDDQ
x	1	x	External clock	OSC32K_EXTAL	Bypass, OSC Disabled
1	0	Crystal	crystal	OSC32K_EXTAL	Normal, OSC Enabled
		Ground	External clock	OSC32K_EXTAL	Normal, OSC Enabled

The OSC32K clock can be further divided by a configurable factor in the range 1 to 32 to generate the divided clock to match system requirements. This division factor is specified by SXOSC_CTL[OSCDIV] field.

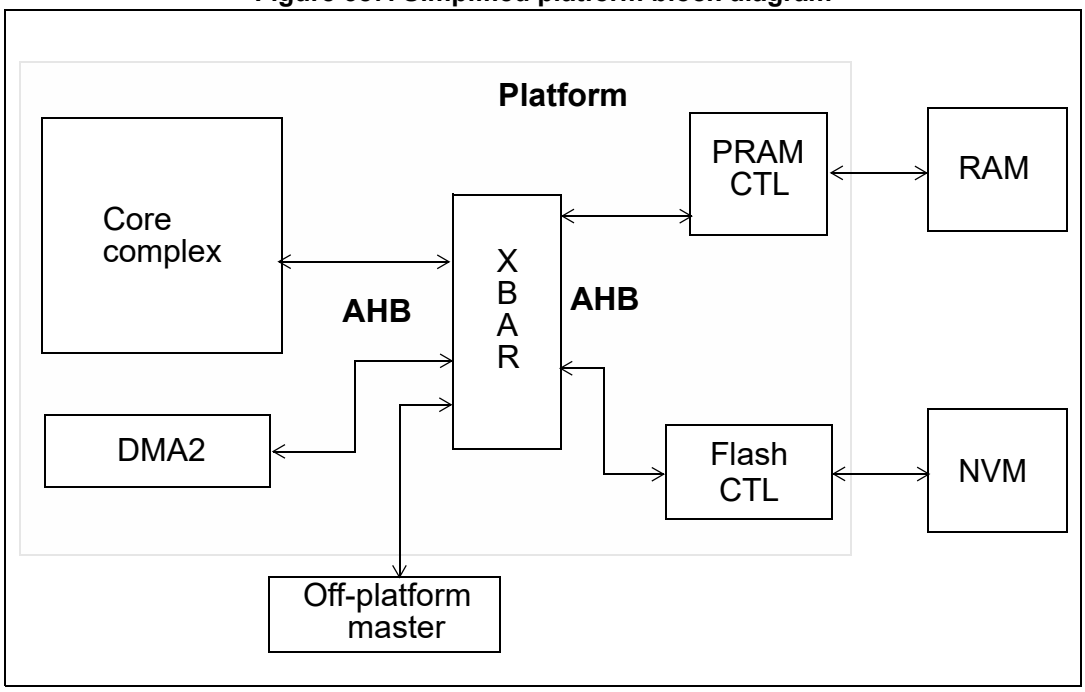
35 Platform RAM controller AHB (PRAMC_AHB)

35.1 Introduction

This section provides an overview of the Platform RAM Controller. The PRAM controller acts as an interface between the system bus (AHB-Lite 2.v6) and the integrated system RAM. It converts the protocols between the system bus and the dedicated RAM array interface.

The PRAM controller supports two 64-bit AHB and one 64-bit RAM array interfaces. The first AHB port provides a connection to the platform crossbar for direct RAM access from the various crossbar masters. The second AHB port provides a connection from the flash controller to facilitate calibration overlay access. *Figure 387* is a simplified block diagram depicting the position of the PRAM controller within a typical platform architecture:

Figure 387. Simplified platform block diagram



The following list summarizes the key features of the PRAM controller:

- System bus supports 64-bit data and 8-bit ECC AHB interfaces
- Array interface supports a 64-bit data and 8-bit ECC interface
- Late-write support via 64-bit data and 8-bit ECC storage buffer to support single-cycle write accesses
- Configurable read access timing (zero or one wait-state programmable) allowing use in large range of frequency targets
- Read-modify-write operation to support array write size less than 64-bit data

The address path of the PRAM controller contains a multiplexer that chooses among the addresses presented on the system bus during read/write requests to RAM. These addresses are transferred to RAM array either immediately, for read accesses, or, for write accesses, through a temporary write address buffer or through a late-write address buffer.

The temporary write address buffer contains the address which was presented during the AHB address phase of the write request. The request is stalled from being presented to the RAM by one cycle in order to simultaneously present the address and associated write data, because the write data is not valid until the AHB data phase. The late-write buffer holds write requests which are delayed to facilitate single-cycle responses on the system bus.

The read data is the result of a RAM access. Along with the read and write data, the corresponding ECC codewords traverse the entire data path of the PRAM controller, including the late-write buffer, to support end-to-end ECC (e2eECC) coverage.

The write data path contains the multiplexer that chooses the source of the write data as well as the control logic for generating read-modify-write operations on writes that are less than 64 bits in size. A write operation can be performed to empty the contents of the late-write buffer. In the case of back-to-back writes, the write may be forced to bypass the late-write buffer and stored directly in the RAM.

35.2 SRAM controller memory map and register definitions

The PRAM controller module provides an IPS programming model mapped to a standard 16 KB on-platform peripheral slot. The programming model can only be referenced using a 32-bit (word) access. Attempted references using different access sizes or to undefined (reserved) addresses will generate an IPS error.

Table 502. PRAMC_AHB memory map

Address offset (hex)	Register name	Reset value (hex)	Location
0x00	Platform RAM Configuration Register 1 (PRCR1)	0x0000_02X0(1)	*Section 35.2.1*

1. Refer to register section for specific reset values.

35.2.1 Platform RAM configuration register 1 (PRCR1)

The Platform RAM configuration register 1 (PRCR1) is used to specify the operation of the PRAM controller.

Note: This register is accessible only in supervisor mode. Accesses in user mode return a transfer error.

Offset: 0x00 Access: Read/Write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	PRI		P1_BO_DIS	P0_BO_DIS	P1_RB_DIS	P0_RB_DIS	0	0	0	FT_DIS
W																
Reset	0	0	0	0	0	0	1	0	0	0	X(1)	X(1)	0	0	0	0

1. Reset bit value depends on the device.

Figure 388. Platform RAM configuration register 1 (PRCR1)

Table 503. PRCR1 field descriptions

Field	Description
22:23 PRI	AHB port arbitration mode Use this register to select the AHB port arbitration mode. 00 Round robin 01 Port p0 has priority over p1 10 Port p1 has priority over p0 11 Reserved
24 P1_BO_DIS	Port p1 read burst optimization disable 0 64-bit WRP4 read bursts are optimized such that the controller returns a 2-1-1-1 response(1) when PRCR[FT_DIS]=1 1 64-bit WRP4 read bursts are not optimized; the controller returns a 2-2-2-2 response(1) when PRCR[FT_DIS]=1
25 P0_BO_DIS	Port p0 read burst optimization disable 0 64-bit WRP4 read bursts are optimized such that the controller returns a 2-1-1-1 response(1) when PRCR[FT_DIS]=1 1 64-bit WRP4 read bursts are not optimized; the controller returns a 2-2-2-2 response(1) when PRCR[FT_DIS]=1
26 P1_RB_DIS	Port p1 read buffer disable 0 Read buffer enabled and is used for servicing sequential, less-than 64-bit read requests 1 Read buffer disabled and invalidated, and all read requests generate array accesses

Table 503. PRCR1 field descriptions (continued)

Field	Description
27 P0_RB_DIS	Port p0 read buffer disable 0 Read buffer enabled and is used for servicing sequential, less-than 64-bit read requests 1 Read buffer disabled and invalidated, and all read requests generate array accesses
31 FT_DIS	Flow through disabled This field defines the AHB read response of the PRAM controller . The state of this field has no impact on the write response latency. This bit is cleared by hardware reset. **Note:** Do not change the FT_DIS bit value while accessing system RAM. Realocate code programming the FT_DIS bit to another area (for example, local core memory). 0 RAM read data is passed directly to the system bus, incurring no additional latency 1 RAM read data is registered prior to returning on the system bus, incurring 1 extra cycle of latency

1. The number of cycles taken for a RAM access can vary ±1 clock cycle depending on the RAM speed relative to the PRAM controller clock frequency. If system RAM is running at the same frequency as the PRAM Controller a random initial access takes 2 clock cycles. If system RAM is running at a slower frequency a random initial access may take 3 clock cycles. Subsequent burst beats take either 1 or 2 cycles depending on RAM speed relative to PRAM Controller clock frequency.

35.3 Functional description

35.3.1 Read/Write introduction

The PRAM controller generates chip-select and write-enable, the array address, and write data as inputs to the RAM array. The PRAM controller captures read data from the RAM array and drives it onto the AHB along with the associated dataphase termination signals. To facilitate operating at higher frequencies, the PRAM controller can optionally incur a single wait state on the AHB response.

35.3.1.1 Reads

Read transfers, of any size, can be configured to complete with a zero or one additional wait state response on the AHB.

35.3.1.2 Optional read wait-state

The PRAM controller can be optionally programed to register RAM read data prior to returning it on the system bus. Setting the PRCR1[FT_DIS] field inserts a register in the read data path for use when operating the controller at high frequencies. The state of PRCR1[FT_DIS] field has no effect on writes.

A random, initial access will take 2 clock cycles (2T) to complete if PRCR1[FT_DIS] = 0, and a WRAP4 burst will have an access time of 2-2-2-2. A random, initial access will take 3 clock cycles (3T) to complete if PRCR1[FT_DIS] = 1, and a WRAP4 burst will have an access time of 3-2-2-2

35.3.2 Writes

35.3.2.1 64-bit writes

Aligned **64**-bit writes execute in a single AHB data phase cycle, allowing for zero wait states on back-to-back writes of these sizes. If, during the data phase of a write, a read is requested on the AHB, the write is registered in the late-write buffer, allowing the read to take place without a wait state. The valid buffered or late-write data is stored on the next available array address phase.

Back-to-back 64-bit writes execute slightly differently whereby the first write bypasses the late-write buffer. Instead, the write data is stored directly to the array in the same cycle in which it is valid on the AHB.

35.3.2.2 Less than 64-bit writes

Writes of size less than 64 bits incur a read-modify-write action as a consequence of the ECC coding scheme's 64-bit granularity. In the case of a read-modify-write action, the PRAM controller performs single-error correction, double-error detection on the read data. The write data is merged into the appropriate byte lanes along with the potentially-corrected read data. A new codeword is generated based on the newly formed doubleword. The newly formed doubleword and its associated checkbits are subsequently written to the RAM. For details on the ECC coding scheme, refer to *Section 35.5: Safety considerations*.

On a read-modify-write operation, the array read data is registered after it is decoded and before it is merged with the AHB write data. Therefore, writes of less than 64 bits require the insertion of one wait state before the data phase can be completed.

35.3.2.3 Unaligned writes

The PRAM controller is compliant with the AMBA-AHB2.v6 extensions specification concerning byte strobes. The size of the transfer is sufficient to cover all bytes being written and covers more bytes in the case of an unaligned transfer. The address of the transfer is rounded down to the nearest boundary of the size of the transaction.

Note: Unaligned writes always generate read-modify-write operations in the PRAM controller in order to preserve the validity of the ECC codeword.

35.4 Initialization/application information

Prior to reading, the ECC must be initialized by writing a known 64-bit value to each memory address. This includes reads generated from the read-modify-write operation which occurs when a write transfer of less than 64 bits or an unaligned write is requested. Reading from an uninitialized address is likely to generate an uncorrectable ECC event.

The PRAM memory space can be initialized by storing a 64-bit word instruction such as Store Multiple Word (e_stmw) in Power Architecture VLE.

35.5 Safety considerations

35.5.1 Hsiao ECC algorithm

The e2eECC scheme implements a single-error correction, double-error detection (SECDED) code using the so-called Hsiao odd-weight column criteria. These codes are named after M.Y. Hsiao, an IBM researcher who published a paper in the early 1970s about SECDED codes better suited for implementation in protecting (mainframe) computer memories than traditional Hamming codes.

The Hsiao codes are Hamming distance 4 implementations which provide the SECDED capabilities. The minimum odd-weight constraints defined by Hsiao are relatively simple in the resulting implementation of the parity check H matrix which defines the association between the data (and address) bits and the checkbits. They are:

1. There are no all-zeroes columns
2. Every column is distinct
3. Every column contains an odd number of ones, and hence is “odd weight”

In defining the H-matrix for this family of devices, these requirements from Hsiao were applied. Additionally, there are a variety of ECC codeword requirements associated with specific functional requirements associated with the system RAM that further dictated the specific column definitions. In any case, the resulting ECC is organized based on 64 data bits plus 29 address bits (the upper bits of the 32-bit address field minus the 3 bits which select the byte within the 64-bit (8-byte) data field.

The basic H-matrix for this (101, 93) code (93 is the total number of data bits, 101 is the total number of data bits (93) plus 8 checkbits) is shown in *Table 504*. A ‘*’ in *Table 504* indicates the corresponding data or address bit is XOR’ed to form the final checkbit value on the left. For 64-bit data writes, the table sections corresponding to D[63:32], D[31:0], and A[31:3] are logically summed (output of each table section is XOR’ed) together to the final value driven on the hwchkbit[7:0] outputs. Note that this table uses *the AHB bit numbering convention where bit[0] is the least significant bit*.

Table 504. PRAM controller basic H-matrix definition

Checkbits [7:0]	Data Bit(1)																													
	Byte 7								Byte 6								Byte 5								Byte 4							
	63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
7	*				*					*			*		*	*		*	*	*	*					*	*			*	*	
6			*	*			*	*					*		*		*				*				*			*	*			*
5	*	*	*	*					*				*	*		*		*	*				*	*	*				*			
4	*		*			*		*					*		*		*			*			*	*			*				*	
3									*	*	*	*	*		*				*		*				*	*	*	*				
2					*		*					*		*			*	*				*	*							*		*
1		*		*		*		*	*	*	*	*						*		*		*		*						*		*
0		*			*	*	*				*			*	*	*		*				*				*		*	*		*	

Table 504. PRAM controller basic H-matrix definition (continued)

Checkbits [7:0]	Byte 3								Byte 2								Byte 1								Byte 0							
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
7					*		*				*		*				*	*			*	*			*				*	*		*
6	*	*		*	*				*				*				*	*		*	*					*	*				*	*
5			*	*			*	*	*			*	*	*					*		*	*		*							
4	*		*	*				*		*	*				*	*	*			*			*	*				*		*		
3	*	*	*			*						*	*		*	*			*				*		*	*	*			*		
2		*		*	*	*	*	*	*	*	*			*					*				*		*		*	*		*		
1		*						*		*			*	*	*			*				*				*			*	*	*	
0		*		*		*		*				*				*				*				*				*	*		*	*
Checkbits [7:0]	**Address Bit(1)**																															
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
7		*		*		*		*	*	*		*	*	*		*	*	*		*		*		*	*	*		*				
6		*		*	*	*	*			*	*			*	*	*	*		*		*	*		*	*	*		*				
5		*	*	*			*	*			*	*		*		*		*	*			*	*	*		*	*	*	*			
4	*	*	*		*	*		*		*	*	*	*		*					*	*		*	*	*		*	*				
3	*		*		*	*	*		*		*	*	*			*		*		*	*		*		*	*	*		*			
2	*	*			*		*	*	*	*				*	*	*	*	*	*	*		*	*		*		*		*			
1	*		*	*		*	*		*	*		*	*		*		*	*	*		*	*	*					*	*			
0	*		*	*	*			*	*		*		*	*	*		*		*	*	*			*		*	*		*			

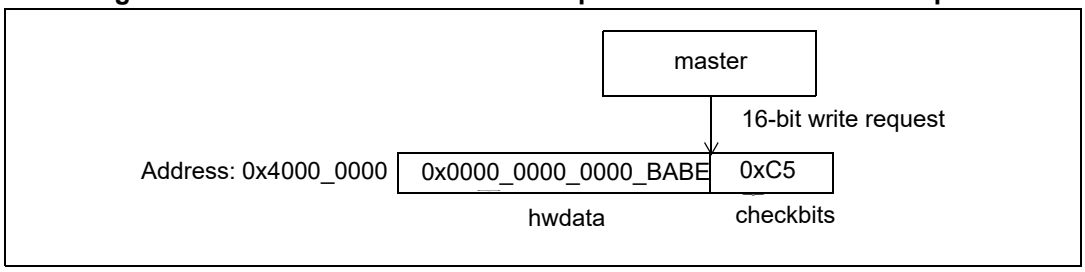
1. Bit numbering is AHB convention: bit 0 is LSB. D[7:0] corresponds to byte at address 0. D[63:56] corresponds to byte at address 7.

35.5.1.1 E2EECC considerations on less-than-64-bit write transactions

Recall that writes of less than 64 bits incur a read-modify-write action as a consequence of the ECC coding scheme 64 bits granularity. In the case of a read-modify-write action, the PRAM controller performs SEC/DED on the read data. The PRAM controller uses the following checkbit manipulation technique to ensure that any faults in the PRAM control logic, associated with performing the read-modify-write action, will ultimately be detectable by e2eECC:

- On an 8-bit, 16-bit or 32-bit write request, the write data is presented by the master on the AHB bus in the correct byte lanes, with the non pertinent byte lanes zeroed out, and the associated checkbit is based on this “zero-padded” write data.
  Consider the following example whereby the master presents a 16-bit write request at address 0x40000000:

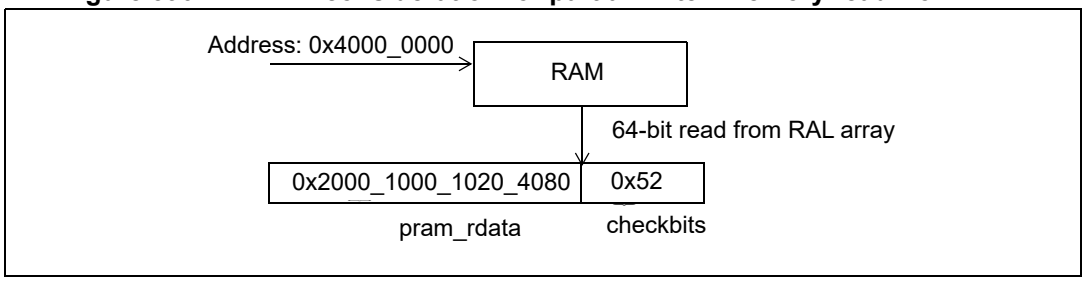
Figure 389. E2EECC consideration for partial write - 16-bit write request



- The PRAM controller detects that the request is a less-than-64-bit write request and initiates a read to address 0x40000000. An ECC check and potential single-bit correction is performed on the data returned from the RAM. In addition, the ECC event is reported to the Memory Error Management Unit (MEMU).

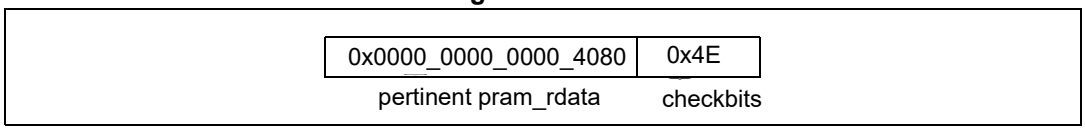
  If a non correctable ECC event is detected, the AHB request is terminated with error on the system bus. In addition, the ECC event is reported to the MEMU.

Figure 390. E2EECC consideration for partial write - memory read from RAM



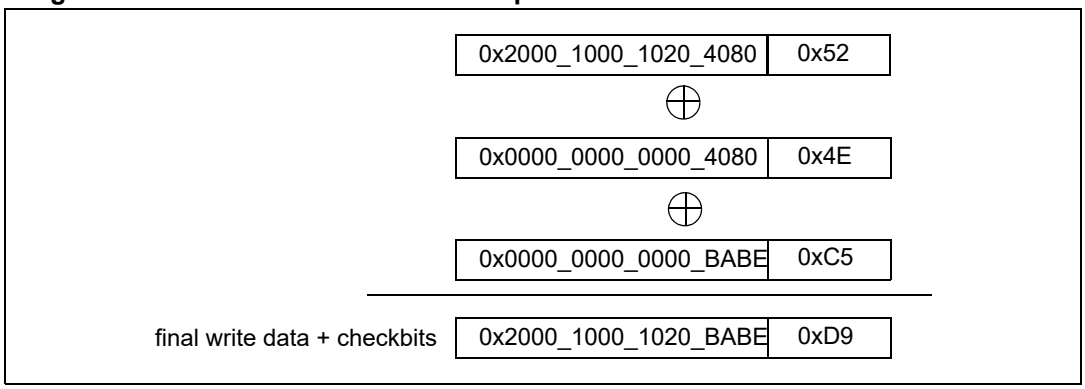
- The PRAM controller isolates the pertinent byte lanes of the read data by zero-padding the non pertinent byte lanes and calculating the checkbit contribution associated with the read data in the byte lanes to be updated as a result of the write request.

Figure 391. E2EECC consideration for partial write - Separation of check-bits of targeted bit-lines



- The PRAM controller merges the non-pertinent read data bytes with the write data in the appropriate byte lanes. Take the checkbit value that was stored in the RAM and remove the checkbit contribution associated with the read data in the byte lanes to be updated. Then introduce the checkbits associated with the write data. The result is ultimately a checkbit value associated with the complete 64-bit data to be written to the array.

Figure 392. E2EECC consideration for partial write - calculation of final check-bits



This technique calculates the ultimate checkbit value on a read-modify-write transaction through a series of data manipulations, taking care to avoid discarding the original checkbits provided by the requesting master, such that faults in the crossbar (XBAR) and PRAM controller datapath and control logic are covered by e2eECC.

Malfunction of the ECC logic described above may result in the corruption of the SEC/DED event reporting. The PRAM controller performs EDC after ECC check on all read-modify-write transactions. If a mismatch is detected, indicating a failure in the ECC logic, the event is reported to the FCCU module.

35.5.2 Transaction monitor

The integrity of the address and data on a system RAM transaction is covered by an e2eECC check performed by the requesting master. Functional safety coverage of the address path and control within the RAM controller is handled by a transaction monitor which verifies the integrity of the transactions between the PRAM controller and the RAM array. The transaction monitor verifies RAM transactions initiated to service the following types of transactions

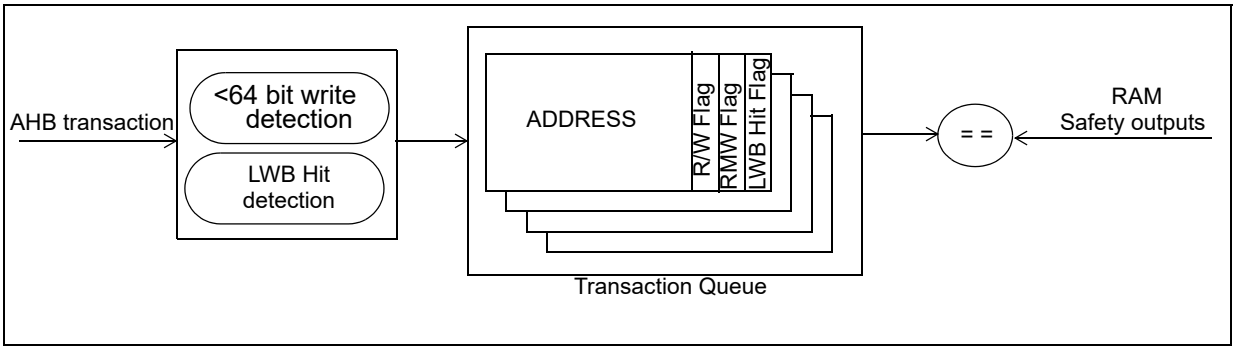
- Direct system bus read or write request
- Read followed by write to fulfill a read-modify write transaction
- Write transaction to empty the late-write buffer

The function of the transaction monitor relies on a feedback path between the RAM controller and RAM array, wherein the RAM provides latched address and control feedback outputs as an indication of received inputs when a RAM access is initiated.

This feedback information is used by the RAM controller transaction monitor to verify the expected transaction. If a mismatch is detected, indicating a failure in the address generation or control logic within the RAM controller or the transmission path between the RAM controller and RAM array, the event is forwarded to the Fault Collection and Control Unit (FCCU).

Since writes to the RAM can be buffered, the transaction monitor also tracks the contents of the late-write buffer. When the late-write buffer is emptied, the safety feature compare outputs from the RAM are evaluated against the expected contents of the late-write buffer as tracked by the transaction monitor.

Figure 393. RAM controller transaction monitor block diagram



36 Flash memory controller

36.1 Introduction

This device implements two Flash memory controllers and two Flash arrays. The Flash memory controllers act as an interface between the system bus (AHB-Lite 2.v6) and the Flash memory arrays.

- Serve as the interface to the on-chip overlay RAM.
- Serve as the interface to on-chip System RAM that can be used as overlay RAM.
- Serve as the interface to the off-chip buddy device.

The Flash memory controllers support two 64-bit AHB buses and a 256-bit read data interface from each Flash memory array. Each AHB port contains a 4-entry, 2-way set-associative mini-cache as well as an associated controller that prefetches sequential lines of data from the Flash arrays into the mini-cache. This buffer mechanism serves to deliver Flash read data with zero-wait state response on lines that reside in the cache. AHB requests that miss the cache generate the needed Flash array access and are forwarded to the AHB upon completion. Each mini-cache entry is 256 bits in size, matching the Flash array page size and providing 256 bytes of total buffered storage.

Along with the read and write data, the corresponding ECC codewords traverse the entire datapath of the Flash memory controller, including the mini-cache, to support end-to-end ECC (e2eECC) coverage.

36.2 Features

The following list summarizes the key features of the Flash memory controller:

- Two 64-bit AHB interface ports (p0, p1) allowing simultaneous access to dedicated prefetch mini-cache per slave port
- 256-bit read data bus + 64-bit write data bus
- Configurable read buffering and line prefetching support via 4-entry, 2-way set-associative mini-cache plus prefetch controller per AHB port to provide single-cycle “buffer hit” read response
- Configurable access control based on read/write and AHB master ID attributes
- Configurable access timing (wait-state programmable) allowing use in a wide range of frequency targets
- Optional address pipelining capability to maximize throughput
- Support for reporting of single-, double- and multi-bit Flash ECC events on a 64-bit doubleword boundary
- Configurable overlay remapping of logical Flash accesses to on-chip calibration RAM, extended off-chip calibration RAM, on-chip system RAM
- Nexus trace stream interface supports message dumping to the on-chip overlay RAM[15]

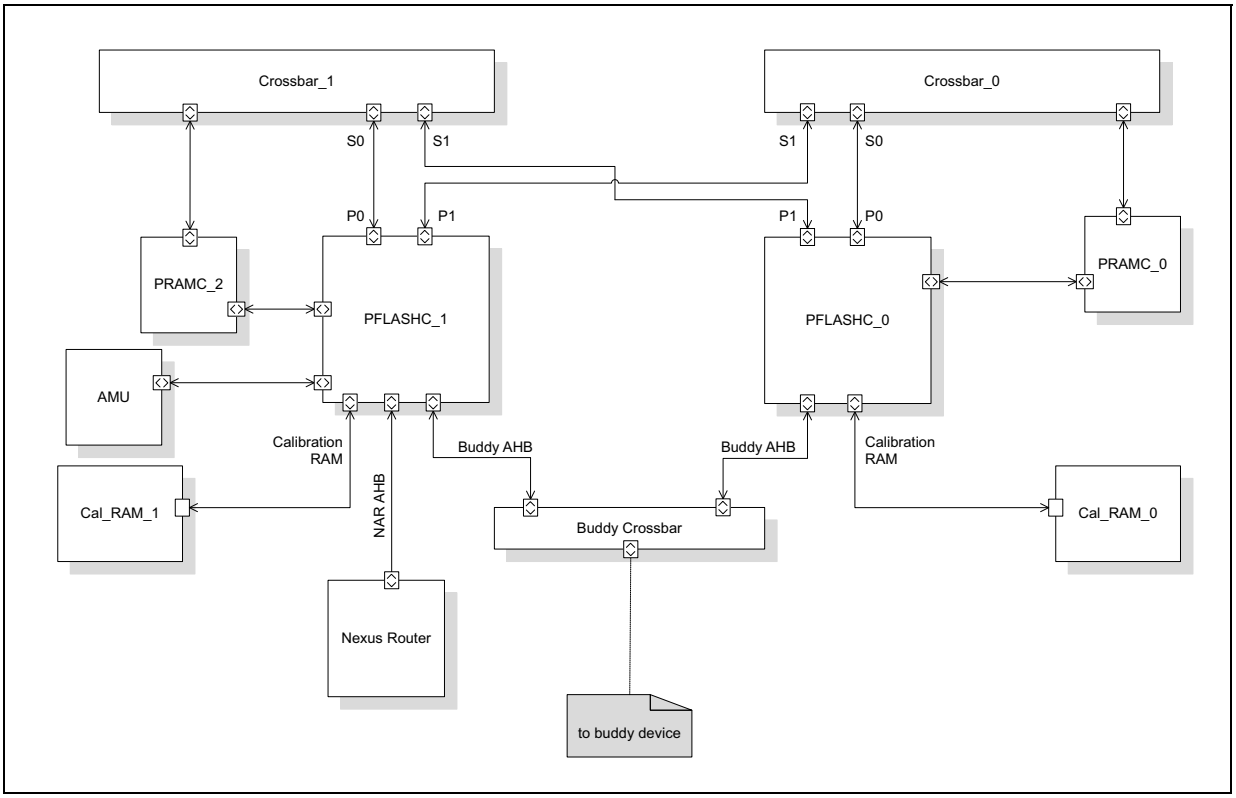
15. Available only on PFLASHC_1, as shown in *Figure 394*

36.3 Block diagrams

Figure 394 provides a block diagram showing the Flash memory controller and the attached Flash array.

Note: This module is also known as the Platform Flash controller, that is, PFlash.

Figure 394. Platform-centric simple block diagram with Flash memory controllers



36.4 Flash memory controller memory map

36.4.1 Overview

The Flash memory controller module provides an IPS programming model mapped to a standard 16 KB on-platform peripheral slot. The programming model consists of Flash memory access configuration and overlay remapping configuration.

The programming model can only be referenced using a 32-bit (word) access. Attempted references using different access sizes or to undefined (reserved) addresses or in user mode generates an IPS error termination. The Flash controller allows access to the programming model by all system bus masters.

The programming model can only be accessed in supervisor mode.

Attempted updates to the programming model while the Flash memory controller module is in the midst of an operation will result in non-deterministic behavior. Software must be architected to avoid this scenario.

There is no idle indicator for the Flash controller. The recommended flow for multi-core devices is to start only one core and execute initialization code to completion before starting the remaining cores. If the user needs to reconfigure the Flash, code execution must be temporarily moved to system RAM.

Table 505. Flash memory controller memory map

Address Offset	Register	Section
00x00	Platform Flash configuration register 1 (PFCR1)	*Section 36.4.1.1*
0x004	Platform Flash configuration register 2 (PFCR2)	*Section 36.4.1.2*
0x008	Platform Flash configuration register 3 (PFCR3)	*Section 36.4.1.3*
0x00c	Platform Flash access protection register (PFAPR)	*Section 36.4.1.4*
0x010	Platform Flash calibration remap control register (PFCRCR)	*Section 36.4.1.5*
0x014	Platform Flash calibration remap descriptor enable register (PFCRDE)	*Section 36.4.1.6*
0x100	Platform Flash calibration region descriptor 0 (PFCRD0)	*Section 36.4.1.7*
0x110	Platform Flash calibration region descriptor 1 (PFCRD1)	*Section 36.4.1.7*
0x120	Platform Flash calibration region descriptor 2 (PFCRD2)	*Section 36.4.1.7*
0x130	Platform Flash calibration region descriptor 3 (PFCRD3)	*Section 36.4.1.7*
0x140	Platform Flash calibration region descriptor 4 (PFCRD4)	*Section 36.4.1.7*
0x150	Platform Flash calibration region descriptor 5 (PFCRD5)	*Section 36.4.1.7*
0x160	Platform Flash calibration region descriptor 6 (PFCRD6)	*Section 36.4.1.7*
0x170	Platform Flash calibration region descriptor 7 (PFCRD7)	*Section 36.4.1.7*
0x180	Platform Flash calibration region descriptor 8 (PFCRD8)	*Section 36.4.1.7*
0x190	Platform Flash calibration region descriptor 9 (PFCRD9)	*Section 36.4.1.7*
0x1A0	Platform Flash calibration region descriptor 10 (PFCRD10)	*Section 36.4.1.7*
0x1B0	Platform Flash calibration region descriptor 11 (PFCRD11)	*Section 36.4.1.7*
0x1C0	Platform Flash calibration region descriptor 12 (PFCRD12)	*Section 36.4.1.7*
0x1D0	Platform Flash calibration region descriptor 13 (PFCRD13)	*Section 36.4.1.7*
0x1E0	Platform Flash calibration region descriptor 14 (PFCRD14)	*Section 36.4.1.7*
0x1F0	Platform Flash calibration region descriptor 15 (PFCRD15)	*Section 36.4.1.7*
0x200	Platform Flash calibration region descriptor 16 (PFCRD16)	*Section 36.4.1.7*
0x210	Platform Flash calibration region descriptor 17 (PFCRD17)	*Section 36.4.1.7*
0x220	Platform Flash calibration region descriptor 18 (PFCRD18)	*Section 36.4.1.7*
0x230	Platform Flash calibration region descriptor 19 (PFCRD19)	*Section 36.4.1.7*
0x240	Platform Flash calibration region descriptor 20 (PFCRD20)	*Section 36.4.1.7*
0x250	Platform Flash calibration region descriptor 21 (PFCRD21)	*Section 36.4.1.7*
0x260	Platform Flash calibration region descriptor 22 (PFCRD22)	*Section 36.4.1.7*
0x270	Platform Flash calibration region descriptor 23 (PFCRD23)	*Section 36.4.1.7*
0x280	Platform Flash calibration region descriptor 24 (PFCRD24)	*Section 36.4.1.7*

Table 505. Flash memory controller memory map (continued)

Address Offset	Register	Section
0x290	Platform Flash calibration region descriptor 25 (PFCRD25)	*Section 36.4.1.7*
0x2A0	Platform Flash calibration region descriptor 26 (PFCRD26)	*Section 36.4.1.7*
0x2B0	Platform Flash calibration region descriptor 27 (PFCRD27)	*Section 36.4.1.7*
0x2C0	Platform Flash calibration region descriptor 28 (PFCRD28)	*Section 36.4.1.7*
0x2D0	Platform Flash calibration region descriptor 29 (PFCRD29)	*Section 36.4.1.7*
0x2E0	Platform Flash calibration region descriptor 30 (PFCRD30)	*Section 36.4.1.7*
0x2F0	Platform Flash calibration region descriptor 31 (PFCRD31)	*Section 36.4.1.7*

36.4.1.1 Platform Flash configuration register 1 (PFCR1)

The PFlash configuration register 1 (PFCR1) controls operation Port p0 of the Flash memory controller.

Offset 0x000 Access: Supervisor Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	P0_M15PFE	P0_M14PFE	P0_M13PFE	P0_M12PFE	P0_M11PFE	P0_M10PFE	P0_M9PFE	P0_M8PFE	P0_M7PFE	P0_M6PFE	P0_M5PFE	P0_M4PFE	P0_M3PFE	P0_M2PFE	P0_M1PFE	P0_M0PFE
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	APC				RWSC				0	P0_DPFEN	0	P0_IPFEN	0	P0_PFLIM		P0_BFEN
W																
Reset	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	1

Figure 395. Platform Flash configuration register 1 (PFCR1)

Note: See the chip-specific Flash controller information for details about the actual masters available on the device.

Table 506. PFCR1 field descriptions

Field	Description
0 P0_M15PFE	Port0 Master 15 Prefetch enable This bit controls whether prefetching may be triggered by AHB master 15. This bit is cleared by hardware reset. 0 No prefetching may be triggered by this master 1 Prefetching may be triggered by this master
1 P0_M14PFE	Port0 Master 14 Prefetch enable This bit controls whether prefetching may be triggered by AHB master 14. This bit is cleared by hardware reset. 0 No prefetching may be triggered by this master 1 Prefetching may be triggered by this master
2 P0_M13PFE	Port0 Master 13 Prefetch enable This bit controls whether prefetching may be triggered by AHB master 13. This bit is cleared by hardware reset. 0 No prefetching may be triggered by this master 1 Prefetching may be triggered by this master
3 P0_M12PFE	Port0 Master 12 Prefetch enable This bit controls whether prefetching may be triggered by AHB master 12. This bit is cleared by hardware reset. 0 No prefetching may be triggered by this master 1 Prefetching may be triggered by this master
4 P0_M11PFE	Port0 Master 11 Prefetch enable This bit controls whether prefetching may be triggered by AHB master 11. This bit is cleared by hardware reset. 0 No prefetching may be triggered by this master 1 Prefetching may be triggered by this master
5 P0_M10PFE	Port0 Master 10 Prefetch enable This bit controls whether prefetching may be triggered by AHB master 10. This bit is cleared by hardware reset. 0 No prefetching may be triggered by this master 1 Prefetching may be triggered by this master
6 P0_M9PFE	Port0 Master 9 Prefetch enable This bit controls whether prefetching may be triggered by AHB master 9. This bit is cleared by hardware reset. 0 No prefetching may be triggered by this master 1 Prefetching may be triggered by this master
7 P0_M8PFE	Port0 Master 8 Prefetch enable This bit controls whether prefetching may be triggered by AHB master 8. This bit is cleared by hardware reset. 0 No prefetching may be triggered by this master 1 Prefetching may be triggered by this master
8 P0_M7PFE	Port0 Master 7 Prefetch enable This bit controls whether prefetching may be triggered by AHB master 7. This bit is cleared by hardware reset. 0 No prefetching may be triggered by this master 1 Prefetching may be triggered by this master

Table 506. PFCR1 field descriptions (continued)

Field	Description
9 P0_M6PFE	Port0 Master 6 Prefetch enable This bit controls whether prefetching may be triggered by AHB master 6. This bit is cleared by hardware reset. 0 No prefetching may be triggered by this master 1 Prefetching may be triggered by this master
10 P0_M5PFE	Port0 Master 5 Prefetch enable This bit controls whether prefetching may be triggered by AHB master 5. This bit is cleared by hardware reset. 0 No prefetching may be triggered by this master 1 Prefetching may be triggered by this master
11 P0_M4PFE	Port0 Master 4 Prefetch enable This bit controls whether prefetching may be triggered by AHB master 4. This bit is cleared by hardware reset. 0 No prefetching may be triggered by this master 1 Prefetching may be triggered by this master
12 P0_M3PFE	Port0 Master 3 Prefetch enable This bit controls whether prefetching may be triggered by AHB master 3. This bit is cleared by hardware reset. 0 No prefetching may be triggered by this master 1 Prefetching may be triggered by this master
13 P0_M2PFE	Port0 Master 2 Prefetch enable This bit controls whether prefetching may be triggered by AHB master 2. This bit is cleared by hardware reset. 0 No prefetching may be triggered by this master 1 Prefetching may be triggered by this master
14 P0_M1PFE	Port0 Master 1 Prefetch enable This bit controls whether prefetching may be triggered by AHB master 1. This bit is cleared by hardware reset. 0 No prefetching may be triggered by this master 1 Prefetching may be triggered by this master
15 P0_M0PFE	Port0 Master 0 Prefetch enable This bit controls whether prefetching may be triggered by AHB master 0. This bit is cleared by hardware reset. 0 No prefetching may be triggered by this master 1 Prefetching may be triggered by this master
16:18 APC	Address Pipeline Control This field controls the number of cycles that a subsequent Flash read from the opposite AHB port can be initiated prior to the previous read data being valid. This field applies to the configuration of Port0 and Port1. 000 Pipelined access to the Flash disabled. 001 A pipelined access can be initiated 1 cycle before the previous data is valid 010 Reserved 011 Reserved 1xx Pipelined access to the Flash is disabled and one wait state is inserted before a subsequent access can be initiated

Table 506. PFCR1 field descriptions (continued)

Field	Description
19:23 RWSC	Read Wait State Control This field controls the number of wait-states to be added to the best-case Flash array access time for reads. The best-case Flash array access time for reads is one cycle. This field must be set to a value corresponding to the operating frequency of the Flash memory controller and the actual read access time of the Flash memory controller. The required settings are documented in the device data sheet. Higher operating frequencies require non-zero settings for this field for proper Flash operation. This field applies to the configuration of Port0 and Port1. 00000 No additional wait-states are added 00001 One additional wait-state is added ... 11111 Thirty-one additional wait-states are added **Note:** Refer to the device data sheet for wait state configuration
25 P0_DPFEN	Port0 Data Prefetch Enable This field enables or disables prefetching initiated by a data read access. This field is cleared by hardware reset. 0 No prefetching is triggered by a data read access 1 Prefetching may be triggered by any data read access
27 P0_IPFEN	Port0 Instruction Prefetch Enable This bit enables or disables prefetching initiated by an instruction read access. This field is cleared by hardware reset. 0 No prefetching is triggered by an instruction read access 1 Prefetching may be triggered by any instruction read access
29:30 P0_PFLIM	Port0 PFlash Prefetch Limit This field controls the prefetch algorithm used by the prefetch controller. This field defines a limit on the maximum number of sequential prefetches which will be attempted between buffer misses. In all situations when enabled, only a single prefetch is initiated on each buffer miss or hit. This field is cleared by hardware reset. 00 No prefetching or buffering is performed. 01 The referenced line is prefetched on a buffer miss, that is, prefetch on miss. 1x The referenced line is prefetched on a buffer miss, or the next sequential line is prefetched on a buffer hit (if not already present), that is, prefetch on miss or hit.
31 P0_BFEN	Port0 PFlash Line Read Buffers Enable This bit enables or disables line read buffer hits. It is also used to invalidate the buffers. This bit can only be updated while PFCR3[BAF_DIS] = 0. After execution of the boot code, as indicated by PFCR3[BAF_DIS] = 1, this bit becomes read-only until the next hardware reset. 0 The line read buffers are disabled from satisfying read requests, and all buffer valid bits are cleared. 1 The line read buffers are enabled to satisfy read requests on hits. Buffer valid bits may be set when the buffers are successfully filled.

36.4.1.2 Platform Flash configuration register 2 (PFCR2)

The PFlash configuration register 2 (PFCR2) specifies operation Port1 of the Flash memory controller.

Offset 0x004 Access: Supervisor Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	P1_M15PFE	P1_M14PFE	P1_M13PFE	P1_M12PFE	P1_M11PFE	P1_M10PFE	P1_M9PFE	P1_M8PFE	P1_M7PFE	P1_M6PFE	P1_M5PFE	P1_M4PFE	P1_M3PFE	P1_M2PFE	P1_M1PFE	P1_M0PFE
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	P1_DPFEN	0	P1_IPFEN	0	P1_PFLIM		P1_BFEN
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

Figure 396. Platform Flash configuration register 2 (PFCR2)

Note: See the chip-specific Flash controller information for details about the actual masters available on the device.

Table 507. PFCR2 field descriptions

Name	Description
0 P1_M15PFE	Port1 Master 15 Prefetch enable This bit controls whether prefetching may be triggered by AHB master 15. This bit is cleared by hardware reset. 0 No prefetching may be triggered by this master 1 Prefetching may be triggered by this master
1 P1_M14PFE	Port1 Master 14 Prefetch enable This bit controls whether prefetching may be triggered by AHB master 14. This bit is cleared by hardware reset. 0 No prefetching may be triggered by this master 1 Prefetching may be triggered by this master
2 P1_M13PFE	Port1 Master 13 Prefetch enable This bit controls whether prefetching may be triggered by AHB master 13. This bit is cleared by hardware reset. 0 No prefetching may be triggered by this master 1 Prefetching may be triggered by this master
3 P1_M12PFE	Port1 Master 12 Prefetch enable This bit controls whether prefetching may be triggered by AHB master 12. This bit is cleared by hardware reset. 0 No prefetching may be triggered by this master 1 Prefetching may be triggered by this master
4 P1_M11PFE	Port1 Master 11 Prefetch enable This bit controls whether prefetching may be triggered by AHB master 11. This bit is cleared by hardware reset. 0 No prefetching may be triggered by this master 1 Prefetching may be triggered by this master

Table 507. PFCR2 field descriptions (continued)

Name	Description
5 P1_M10PFE	Port1 Master 10 Prefetch enable This bit controls whether prefetching may be triggered by AHB master 10. This bit is cleared by hardware reset. 0 No prefetching may be triggered by this master 1 Prefetching may be triggered by this master
6 P1_M9PFE	Port1 Master 9 Prefetch enable This bit controls whether prefetching may be triggered by AHB master 9. This bit is cleared by hardware reset. 0 No prefetching may be triggered by this master 1 Prefetching may be triggered by this master
7 P1_M8PFE	Port1 Master 8 Prefetch enable This bit controls whether prefetching may be triggered by AHB master 8. This bit is cleared by hardware reset. 0 No prefetching may be triggered by this master 1 Prefetching may be triggered by this master
8 P1_M7PFE	Port1 Master 7 Prefetch enable This bit controls whether prefetching may be triggered by AHB master 7. This bit is cleared by hardware reset. 0 No prefetching may be triggered by this master 1 Prefetching may be triggered by this master
9 P1_M6PFE	Port1 Master 6 Prefetch enable This bit controls whether prefetching may be triggered by AHB master 6. This bit is cleared by hardware reset. 0 No prefetching may be triggered by this master 1 Prefetching may be triggered by this master
10 P1_M5PFE	Port1 Master 5 Prefetch enable This bit controls whether prefetching may be triggered by AHB master 5. This bit is cleared by hardware reset. 0 No prefetching may be triggered by this master 1 Prefetching may be triggered by this master
11 P1_M4PFE	Port1 Master 4 Prefetch enable This bit controls whether prefetching may be triggered by AHB master 4. This bit is cleared by hardware reset. 0 No prefetching may be triggered by this master 1 Prefetching may be triggered by this master
12 P1_M3PFE	Port1 Master 3 Prefetch enable This bit controls whether prefetching may be triggered by AHB master 3. This bit is cleared by hardware reset. 0 No prefetching may be triggered by this master 1 Prefetching may be triggered by this master
13 P1_M2PFE	Port1 Master 2 Prefetch enable This bit controls whether prefetching may be triggered by AHB master 2. This bit is cleared by hardware reset. 0 No prefetching may be triggered by this master 1 Prefetching may be triggered by this master

Table 507. PFCR2 field descriptions (continued)

Name	Description
14 P1_M1PFE	Port1 Master 1 Prefetch enable This bit controls whether prefetching may be triggered by AHB master 1. This bit is cleared by hardware reset. 0 No prefetching may be triggered by this master 1 Prefetching may be triggered by this master
15 P1_M0PFE	Port1 Master 0 Prefetch enable This bit controls whether prefetching may be triggered by AHB master 0. This bit is cleared by hardware reset. 0 No prefetching may be triggered by this master 1 Prefetching may be triggered by this master
25 P1_DPFEN	Port1 Data Prefetch Enable This field enables or disables prefetching initiated by a data read access. This field is cleared by hardware reset. 0 No prefetching is triggered by a data read access 1 Prefetching may be triggered by any data read access
27 P1_IPFEN	Port1 Instruction Prefetch Enable This bit enables or disables prefetching initiated by an instruction read access. This field is cleared by hardware reset. 0 No prefetching is triggered by an instruction read access 1 Prefetching may be triggered by any instruction read access
29:30 P1_PFLIM	Port1 PFlash Prefetch Limit This field controls the prefetch algorithm used by the prefetch controller. This field defines a limit on the maximum number of sequential prefetches which will be attempted between buffer misses. In all situations when enabled, only a single prefetch is initiated on each buffer miss or hit. This field is cleared by hardware reset. 00 No prefetching or buffering is performed. 01 The referenced line is prefetched on a buffer miss, that is, prefetch on miss. 1x The referenced line is prefetched on a buffer miss, or the next sequential line is prefetched on a buffer hit (if not already present), that is, prefetch on miss or hit.
31 P1_BFEN	Port1 PFlash Line Read Buffers Enable This bit enables or disables line read buffer hits. It is also used to invalidate the buffers. This bit can only be updated while PFCR3[BAF_DIS] = 0. After execution of the boot code, as indicated by PFCR3[BAF_DIS] = 1, this bit becomes read-only until the next hardware reset. 0 The line read buffers are disabled from satisfying read requests, and all buffer valid bits are cleared. 1 The line read buffers are enabled to satisfy read requests on hits. Buffer valid bits may be set when the buffers are successfully filled.

36.4.1.3 Platform Flash configuration register 3 (PFCR3)

Offset 0x008 Access: Supervisor Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	P0_WCFG		P1_WCFG		0	0	0	0	0	0	0	BDRM	0	0	0	BAF_DIS
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	ARBM		0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 397. Platform Flash configuration register 3 (PFCR3)

Table 508. PFCR3 field descriptions

Name	Description
0:1 P0_WCFG	Port0 Way Configuration This field controls the configuration of the line buffers for a given set across two ways in the controller cache. The indexed set can be organized as a “pool” of available resources, or with a fixed partition between instruction and data buffers. In all cases, when a buffer miss occurs, the Flash page is assigned a location within the controller cache. Within the indexed set, the way is selected using a least-recently-used replacement policy, and the entry is then marked as most-recently-used for that set. If the Flash access is for the next-sequential line (prefetch), the buffer is not marked as most-recently-used until the given address produces a buffer hit. This field is initialized by hardware reset. 00 Both buffers in an indexed set are available for any Flash access, that is, there is no partitioning of the buffers based on the access type. 01 Reserved 10 The buffers in an indexed set are partitioned into two groups with way 0 allocated for instruction fetches and way 1 for data accesses. 11 Reserved.
2:3 P1_WCFG	Port1 Line Buffer Configuration This field controls the configuration of the line buffers for a given set across two ways in the controller cache. The indexed set can be organized as a “pool” of available resources, or with a fixed partition between instruction and data buffers. In all cases, when a buffer miss occurs, the Flash page is assigned a location within the controller cache. Within the indexed set, the way is selected using a least-recently-used replacement policy, and the entry is then marked as most-recently-used for that set. If the Flash access is for the next-sequential line (prefetch), the buffer is not marked as most-recently-used until the given address produces a buffer hit. This field is initialized by hardware reset. 00 Both buffers are available for any Flash access, that is, there is no partitioning of the buffers based on the access type. 01 Reserved 10 The buffers are partitioned into two groups with buffer 0 allocated for instruction fetches and buffer 1 for data accesses. 11 Reserved

Table 508. PFCR3 field descriptions (continued)

Name	Description
11 BDRM	Buddy Device Read Mode This field controls the Flash memory controller bus response when the master presents a read request targeting the extended overlay RAM on the Buddy Device (BD) using the direct, non-remapped address. This field is ignored on production ECUs where the Buddy Device is not present. On production ECUs, this memory map region is provided for the “OLDA” (on-line data acquisition) function. Read requests on production ECUs targeting the non-existent extended overlay RAM using the direct, non-remapped address are terminated with error on the system bus. This field is ignored on write requests targeting the extended overlay RAM using the direct, non-remapped address. Write requests on production ECUs targeting the non-existent extended overlay RAM using the direct, non-remapped address and terminate cleanly on the system bus. Write requests on engineering ECUs targeting the extended overlay RAM using the direct, non-remapped address are ignored are executed. This field is initialized by hardware reset. 0 Read requests targeting the extended overlay RAM on the Buddy Device (BD) using the direct, non-remapped address are aborted, and the request is terminated with error on the system bus. 1 Read requests targeting the extended overlay RAM on the Buddy Device (BD) using the direct, non-remapped address are allowed. The Flash memory controller returns the requested data from the extended overlay RAM, which resides on the BD. Any system bus errors returned by the BD are passed onto the master.
15 BAF_DIS	BAF Disable This field controls executable access to the BAF (Boot Assist Flash) region of the Flash. Once this field is set, attempted instruction accesses targeting the BAF region are aborted and terminated with a system bus error. The affect of this field applies to system bus transfers through both, Port0 and Port1. Data-type accesses to the BAF region are not affected by this field. Once this field is set, it becomes a read-only field and can only be cleared by hardware reset. Once this field is set, any subsequent write attempts to modify this field are ignored with an error-free data transfer termination. This field is initialized by hardware reset. 0 Executable access to the BAF Flash region is allowed. 1 Executable access to the BAF Flash region is prohibited.
16:17 ARBM	Arbitration Mode This 2-bit field controls the arbitration of concurrent Flash access requests from the two AHB ports of the Flash memory controller. In both fixed priority or round-robin modes, write requests are prioritized higher than read requests, and read requests are prioritized higher than speculative prefetch requests whenever both ports issue concurrent requests. This field is initialized by hardware reset. 00 Fixed priority arbitration with AHB p0>p1 01 Fixed priority arbitration with AHB p1>p0 1x Round-robin arbitration

36.4.1.4 Platform Flash access protection register (PFAPR)

The PFlash access protection register (PFAPR) is used to control read and write accesses to the Flash array.

Offset 0x00C Access: Supervisor Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	M0AP		M1AP		M2AP		M3AP		M4AP		M5AP		M6AP		M7AP	
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	M8AP		M9AP		M10AP		M11AP		M12AP		M13AP		M14AP		M15AP	
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Figure 398. Platform Flash access protection register (PFAPR)

Note: See the chip-specific Flash controller information for details about the actual masters available on the device.

Table 509. PFAPR field descriptions

Field	Description
0:1 M0AP	Master 0 Access Protection This field controls whether read and write accesses to the Flash are allowed based on the master ID of a requesting master. These fields are initialized by hardware reset. 00 No accesses may be performed by this master 01 Only read accesses may be performed by this master 10 Only write accesses may be performed by this master 11 Both read and write accesses may be performed by this master
2:3 M1AP	Master 1 Access Protection This field controls whether read and write accesses to the Flash are allowed based on the master ID of a requesting master. These fields are initialized by hardware reset. 00 No accesses may be performed by this master 01 Only read accesses may be performed by this master 10 Only write accesses may be performed by this master 11 Both read and write accesses may be performed by this master
4:5 M2AP	Master 2 Access Protection This field controls whether read and write accesses to the Flash are allowed based on the master ID of a requesting master. These fields are initialized by hardware reset. 00 No accesses may be performed by this master 01 Only read accesses may be performed by this master 10 Only write accesses may be performed by this master 11 Both read and write accesses may be performed by this master

Table 509. PFAPR field descriptions (continued)

Field	Description
6:7 M3AP	Master 3 Access Protection This field controls whether read and write accesses to the Flash are allowed based on the master ID of a requesting master. These fields are initialized by hardware reset. 00 No accesses may be performed by this master 01 Only read accesses may be performed by this master 10 Only write accesses may be performed by this master 11 Both read and write accesses may be performed by this master
8:9 M4AP	Master 4 Access Protection This field controls whether read and write accesses to the Flash are allowed based on the master ID of a requesting master. These fields are initialized by hardware reset. 00 No accesses may be performed by this master 01 Only read accesses may be performed by this master 10 Only write accesses may be performed by this master 11 Both read and write accesses may be performed by this master
10:11 M5AP	Master 5 Access Protection This field controls whether read and write accesses to the Flash are allowed based on the master ID of a requesting master. These fields are initialized by hardware reset. 00 No accesses may be performed by this master 01 Only read accesses may be performed by this master 10 Only write accesses may be performed by this master 11 Both read and write accesses may be performed by this master
12:13 M6AP	Master 6 Access Protection This field controls whether read and write accesses to the Flash are allowed based on the master ID of a requesting master. These fields are initialized by hardware reset. 00 No accesses may be performed by this master 01 Only read accesses may be performed by this master 10 Only write accesses may be performed by this master 11 Both read and write accesses may be performed by this master
14:15 M7AP	Master 7 Access Protection This field controls whether read and write accesses to the Flash are allowed based on the master ID of a requesting master. These fields are initialized by hardware reset. 00 No accesses may be performed by this master 01 Only read accesses may be performed by this master 10 Only write accesses may be performed by this master 11 Both read and write accesses may be performed by this master
16:17 M8AP	Master 8 Access Protection This field controls whether read and write accesses to the Flash are allowed based on the master ID of a requesting master. These fields are initialized by hardware reset. 00 No accesses may be performed by this master 01 Only read accesses may be performed by this master 10 Only write accesses may be performed by this master 11 Both read and write accesses may be performed by this master

Table 509. PFAPR field descriptions (continued)

Field	Description
18:19 M9AP	Master 9 Access Protection This field controls whether read and write accesses to the Flash are allowed based on the master ID of a requesting master. These fields are initialized by hardware reset. 00 No accesses may be performed by this master 01 Only read accesses may be performed by this master 10 Only write accesses may be performed by this master 11 Both read and write accesses may be performed by this master
20:21 M10AP	Master 10 Access Protection This field controls whether read and write accesses to the Flash are allowed based on the master ID of a requesting master. These fields are initialized by hardware reset. 00 No accesses may be performed by this master 01 Only read accesses may be performed by this master 10 Only write accesses may be performed by this master 11 Both read and write accesses may be performed by this master
22:23 M11AP	Master 11 Access Protection This field controls whether read and write accesses to the Flash are allowed based on the master ID of a requesting master. These fields are initialized by hardware reset. 00 No accesses may be performed by this master 01 Only read accesses may be performed by this master 10 Only write accesses may be performed by this master 11 Both read and write accesses may be performed by this master
24:25 M12AP	Master 12 Access Protection This field controls whether read and write accesses to the Flash are allowed based on the master ID of a requesting master. These fields are initialized by hardware reset. 00 No accesses may be performed by this master 01 Only read accesses may be performed by this master 10 Only write accesses may be performed by this master 11 Both read and write accesses may be performed by this master
26:27 M13AP	Master 13 Access Protection This field controls whether read and write accesses to the Flash are allowed based on the master ID of a requesting master. These fields are initialized by hardware reset. 00 No accesses may be performed by this master 01 Only read accesses may be performed by this master 10 Only write accesses may be performed by this master 11 Both read and write accesses may be performed by this master

Table 509. PFAPR field descriptions (continued)

Field	Description
28:29 M14AP	Master 14 Access Protection This field controls whether read and write accesses to the Flash are allowed based on the master ID of a requesting master. These fields are initialized by hardware reset. 00 No accesses may be performed by this master 01 Only read accesses may be performed by this master 10 Only write accesses may be performed by this master 11 Both read and write accesses may be performed by this master
30:31 M15AP	Master 15 Access Protection This field controls whether read and write accesses to the Flash are allowed based on the master ID of a requesting master. These fields are initialized by hardware reset. 00 No accesses may be performed by this master 01 Only read accesses may be performed by this master 10 Only write accesses may be performed by this master 11 Both read and write accesses may be performed by this master

36.4.1.5 PFlash calibration remap control register (PFCRCR)

The PFCRCR is used to globally enable/disable the calibration remap function.

Offset 0x010 Access: Supervisor Read/Write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	SAFE_CAL	0	0	0	IRMEN	0	0	0	GRMEN
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 399. PFlash calibration remap control register (PFCRCR)

Table 510. PFCRCR field descriptions

Field	Description
23 SAFE_CAL	Safe Calibration This bit facilitates the use of safety-critical calibration data. When this bit is enabled, the Flash memory controller is configured to perform redundancy checking on the calibration remap function and reduce the total number of potential calibration regions from 32 to 16. This field is initialized by hardware reset. 0 Established calibration overlay regions are not considered safety-critical. 1 Established calibration overlay regions are considered safety-critical.
27 IRMEN	Instruction Remap This bit enables calibration remapping evaluation on instruction fetches. When PFCRCR[GRMEN] is disabled, this bit is ignored. This field is initialized by hardware reset. 0 Calibration remap evaluation is performed on any incoming data fetch requests only. 1 Calibration remap evaluation is performed on all incoming Flash access requests - data and instruction fetches.
31 GRMEN	Global Remap Enable This bit globally enables or disables the calibration remap evaluation on system Flash accesses. Calibration remap evaluation is immediately available the next cycle following an update to enable PFCRCR[GRMEN] This field is initialized by hardware reset. 0 Calibration remap evaluation is not performed on any incoming system Flash access requests. 1 Calibration remap evaluation is performed on all incoming system Flash access requests.

36.4.1.6 PFlash calibration remap descriptor enable register (PFCRDE)

The PFlash calibration remap descriptor enable register (PFCRDE) is used to enable or disable up to 32 calibration remap descriptors. Note there is also a global remap enable (PFCRCR[GRMEN]) that also must be asserted in conjunction with the individual CRDnEN flags to enable a given remap descriptor.

Offset 0x014 Access: Supervisor Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	CRD0EN	CRD1EN	CRD2EN	CRD3EN	CRD4EN	CRD5EN	CRD6EN	CRD7EN	CRD8EN	CRD9EN	CRD10EN	CRD11EN	CRD12EN	CRD13EN	CRD14EN	CRD15EN
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	CRD16EN	CRD17EN	CRD18EN	CRD19EN	CRD20EN	CRD21EN	CRD22EN	CRD23EN	CRD24EN	CRD25EN	CRD26EN	CRD27EN	CRD28EN	CRD29EN	CRD30EN	CRD31EN
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 400. PFlash calibration remap descriptor enable register (PFCRDE)

Table 511. PFCRDE field descriptions

Field	Description
0 CRD0EN	Calibration Remap Descriptor 0 Enable This bit indicates whether the corresponding remap descriptor is valid. There is also a global calibration remap enable (PFCRCR[GRMEN]) that also must be asserted to enable any remap functionality. Any write to PFCRDn.Word{0,1,2} clears the corresponding PFCRDE[CRDnEN] bit, leaving the calibration remap descriptor invalid. This bit is cleared by power-on reset and unaffected by other types of system reset. 0 Calibration Remap Descriptor *n* invalid 1 Calibration Remap Descriptor *n* valid
1 CRD1EN	Calibration Remap Descriptor 1 Enable This bit indicates whether the corresponding remap descriptor is valid. There is also a global calibration remap enable (PFCRCR[GRMEN]) that also must be asserted to enable any remap functionality. Any write to PFCRDn.Word{0,1,2} clears the corresponding PFCRDE[CRDnEN] bit, leaving the calibration remap descriptor invalid. This bit is cleared by power-on reset and unaffected by other types of system reset. 0 Calibration Remap Descriptor *n* invalid 1 Calibration Remap Descriptor *n* valid
2 CRD2EN	Calibration Remap Descriptor 2 Enable This bit indicates whether the corresponding remap descriptor is valid. There is also a global calibration remap enable (PFCRCR[GRMEN]) that also must be asserted to enable any remap functionality. Any write to PFCRDn.Word{0,1,2} clears the corresponding PFCRDE[CRDnEN] bit, leaving the calibration remap descriptor invalid. This bit is cleared by power-on reset and unaffected by other types of system reset. 0 Calibration Remap Descriptor *n* invalid 1 Calibration Remap Descriptor *n* valid
3 CRD3EN	Calibration Remap Descriptor 3 Enable This bit indicates whether the corresponding remap descriptor is valid. There is also a global calibration remap enable (PFCRCR[GRMEN]) that also must be asserted to enable any remap functionality. Any write to PFCRDn.Word{0,1,2} clears the corresponding PFCRDE[CRDnEN] bit, leaving the calibration remap descriptor invalid. This bit is cleared by power-on reset and unaffected by other types of system reset. 0 Calibration Remap Descriptor *n* invalid 1 Calibration Remap Descriptor *n* valid
4 CRD4EN	Calibration Remap Descriptor 4 Enable This bit indicates whether the corresponding remap descriptor is valid. There is also a global calibration remap enable (PFCRCR[GRMEN]) that also must be asserted to enable any remap functionality. Any write to PFCRDn.Word{0,1,2} clears the corresponding PFCRDE[CRDnEN] bit, leaving the calibration remap descriptor invalid. This bit is cleared by power-on reset and unaffected by other types of system reset. 0 Calibration Remap Descriptor *n* invalid 1 Calibration Remap Descriptor *n* valid
5 CRD5EN	Calibration Remap Descriptor 5 Enable This bit indicates whether the corresponding remap descriptor is valid. There is also a global calibration remap enable (PFCRCR[GRMEN]) that also must be asserted to enable any remap functionality. Any write to PFCRDn.Word{0,1,2} clears the corresponding PFCRDE[CRDnEN] bit, leaving the calibration remap descriptor invalid. This bit is cleared by power-on reset and unaffected by other types of system reset. 0 Calibration Remap Descriptor *n* invalid 1 Calibration Remap Descriptor *n* valid

Table 511. PFCRDE field descriptions (continued)

Field	Description
6 CRD6EN	Calibration Remap Descriptor 6 Enable This bit indicates whether the corresponding remap descriptor is valid. There is also a global calibration remap enable (PFCRCR[GRMEN]) that also must be asserted to enable any remap functionality. Any write to PFCRDn.Word{0,1,2} clears the corresponding PFCRDE[CRDnEN] bit, leaving the calibration remap descriptor invalid. This bit is cleared by power-on reset and unaffected by other types of system reset. 0 Calibration Remap Descriptor *n* invalid 1 Calibration Remap Descriptor *n* valid
7 CRD7EN	Calibration Remap Descriptor 7 Enable This bit indicates whether the corresponding remap descriptor is valid. There is also a global calibration remap enable (PFCRCR[GRMEN]) that also must be asserted to enable any remap functionality. Any write to PFCRDn.Word{0,1,2} clears the corresponding PFCRDE[CRDnEN] bit, leaving the calibration remap descriptor invalid. This bit is cleared by power-on reset and unaffected by other types of system reset. 0 Calibration Remap Descriptor *n* invalid 1 Calibration Remap Descriptor *n* valid
8 CRD8EN	Calibration Remap Descriptor 8 Enable This bit indicates whether the corresponding remap descriptor is valid. There is also a global calibration remap enable (PFCRCR[GRMEN]) that also must be asserted to enable any remap functionality. Any write to PFCRDn.Word{0,1,2} clears the corresponding PFCRDE[CRDnEN] bit, leaving the calibration remap descriptor invalid. This bit is cleared by power-on reset and unaffected by other types of system reset. 0 Calibration Remap Descriptor *n* invalid 1 Calibration Remap Descriptor *n* valid
9 CRD9EN	Calibration Remap Descriptor 9 Enable This bit indicates whether the corresponding remap descriptor is valid. There is also a global calibration remap enable (PFCRCR[GRMEN]) that also must be asserted to enable any remap functionality. Any write to PFCRDn.Word{0,1,2} clears the corresponding PFCRDE[CRDnEN] bit, leaving the calibration remap descriptor invalid. This bit is cleared by power-on reset and unaffected by other types of system reset. 0 Calibration Remap Descriptor *n* invalid 1 Calibration Remap Descriptor *n* valid
10 CRD10EN	Calibration Remap Descriptor 10 Enable This bit indicates whether the corresponding remap descriptor is valid. There is also a global calibration remap enable (PFCRCR[GRMEN]) that also must be asserted to enable any remap functionality. Any write to PFCRDn.Word{0,1,2} clears the corresponding PFCRDE[CRDnEN] bit, leaving the calibration remap descriptor invalid. This bit is cleared by power-on reset and unaffected by other types of system reset. 0 Calibration Remap Descriptor *n* invalid 1 Calibration Remap Descriptor *n* valid
11 CRD11EN	Calibration Remap Descriptor 11 Enable This bit indicates whether the corresponding remap descriptor is valid. There is also a global calibration remap enable (PFCRCR[GRMEN]) that also must be asserted to enable any remap functionality. Any write to PFCRDn.Word{0,1,2} clears the corresponding PFCRDE[CRDnEN] bit, leaving the calibration remap descriptor invalid. This bit is cleared by power-on reset and unaffected by other types of system reset. 0 Calibration Remap Descriptor *n* invalid 1 Calibration Remap Descriptor *n* valid

Table 511. PFCRDE field descriptions (continued)

Field	Description
12 CRD12EN	Calibration Remap Descriptor 12 Enable This bit indicates whether the corresponding remap descriptor is valid. There is also a global calibration remap enable (PFCRCR[GRMEN]) that also must be asserted to enable any remap functionality. Any write to PFCRDn.Word{0,1,2} clears the corresponding PFCRDE[CRDnEN] bit, leaving the calibration remap descriptor invalid. This bit is cleared by power-on reset and unaffected by other types of system reset. 0 Calibration Remap Descriptor *n* invalid 1 Calibration Remap Descriptor *n* valid
13 CRD13EN	Calibration Remap Descriptor 13 Enable This bit indicates whether the corresponding remap descriptor is valid. There is also a global calibration remap enable (PFCRCR[GRMEN]) that also must be asserted to enable any remap functionality. Any write to PFCRDn.Word{0,1,2} clears the corresponding PFCRDE[CRDnEN] bit, leaving the calibration remap descriptor invalid. This bit is cleared by power-on reset and unaffected by other types of system reset. 0 Calibration Remap Descriptor *n* invalid 1 Calibration Remap Descriptor *n* valid
14 CRD14EN	Calibration Remap Descriptor 14 Enable This bit indicates whether the corresponding remap descriptor is valid. There is also a global calibration remap enable (PFCRCR[GRMEN]) that also must be asserted to enable any remap functionality. Any write to PFCRDn.Word{0,1,2} clears the corresponding PFCRDE[CRDnEN] bit, leaving the calibration remap descriptor invalid. This bit is cleared by power-on reset and unaffected by other types of system reset. 0 Calibration Remap Descriptor *n* invalid 1 Calibration Remap Descriptor *n* valid
15 CRD15EN	Calibration Remap Descriptor 15 Enable This bit indicates whether the corresponding remap descriptor is valid. There is also a global calibration remap enable (PFCRCR[GRMEN]) that also must be asserted to enable any remap functionality. Any write to PFCRDn.Word{0,1,2} clears the corresponding PFCRDE[CRDnEN] bit, leaving the calibration remap descriptor invalid. This bit is cleared by power-on reset and unaffected by other types of system reset. 0 Calibration Remap Descriptor *n* invalid 1 Calibration Remap Descriptor *n* valid
16 CRD16EN	Calibration Remap Descriptor 16 Enable This bit indicates whether the corresponding remap descriptor is valid. There is also a global calibration remap enable (PFCRCR[GRMEN]) that also must be asserted to enable any remap functionality. Any write to PFCRDn.Word{0,1,2} clears the corresponding PFCRDE[CRDnEN] bit, leaving the calibration remap descriptor invalid. This bit is cleared by power-on reset and unaffected by other types of system reset. 0 Calibration Remap Descriptor *n* invalid 1 Calibration Remap Descriptor *n* valid
17 CRD17EN	Calibration Remap Descriptor 17 Enable This bit indicates whether the corresponding remap descriptor is valid. There is also a global calibration remap enable (PFCRCR[GRMEN]) that also must be asserted to enable any remap functionality. Any write to PFCRDn.Word{0,1,2} clears the corresponding PFCRDE[CRDnEN] bit, leaving the calibration remap descriptor invalid. This bit is cleared by power-on reset and unaffected by other types of system reset. 0 Calibration Remap Descriptor *n* invalid 1 Calibration Remap Descriptor *n* valid

Table 511. PFCRDE field descriptions (continued)

Field	Description
18 CRD18EN	Calibration Remap Descriptor 18 Enable This bit indicates whether the corresponding remap descriptor is valid. There is also a global calibration remap enable (PFCRCR[GRMEN]) that also must be asserted to enable any remap functionality. Any write to PFCRDn.Word{0,1,2} clears the corresponding PFCRDE[CRDnEN] bit, leaving the calibration remap descriptor invalid. This bit is cleared by power-on reset and unaffected by other types of system reset. 0 Calibration Remap Descriptor *n* invalid 1 Calibration Remap Descriptor *n* valid
19 CRD19EN	Calibration Remap Descriptor 19 Enable This bit indicates whether the corresponding remap descriptor is valid. There is also a global calibration remap enable (PFCRCR[GRMEN]) that also must be asserted to enable any remap functionality. Any write to PFCRDn.Word{0,1,2} clears the corresponding PFCRDE[CRDnEN] bit, leaving the calibration remap descriptor invalid. This bit is cleared by power-on reset and unaffected by other types of system reset. 0 Calibration Remap Descriptor *n* invalid 1 Calibration Remap Descriptor *n* valid
20 CRD20EN	Calibration Remap Descriptor 20 Enable This bit indicates whether the corresponding remap descriptor is valid. There is also a global calibration remap enable (PFCRCR[GRMEN]) that also must be asserted to enable any remap functionality. Any write to PFCRDn.Word{0,1,2} clears the corresponding PFCRDE[CRDnEN] bit, leaving the calibration remap descriptor invalid. This bit is cleared by power-on reset and unaffected by other types of system reset. 0 Calibration Remap Descriptor *n* invalid 1 Calibration Remap Descriptor *n* valid
21 CRD21EN	Calibration Remap Descriptor 21 Enable This bit indicates whether the corresponding remap descriptor is valid. There is also a global calibration remap enable (PFCRCR[GRMEN]) that also must be asserted to enable any remap functionality. Any write to PFCRDn.Word{0,1,2} clears the corresponding PFCRDE[CRDnEN] bit, leaving the calibration remap descriptor invalid. This bit is cleared by power-on reset and unaffected by other types of system reset. 0 Calibration Remap Descriptor *n* invalid 1 Calibration Remap Descriptor *n* valid
22 CRD22EN	Calibration Remap Descriptor 22 Enable This bit indicates whether the corresponding remap descriptor is valid. There is also a global calibration remap enable (PFCRCR[GRMEN]) that also must be asserted to enable any remap functionality. Any write to PFCRDn.Word{0,1,2} clears the corresponding PFCRDE[CRDnEN] bit, leaving the calibration remap descriptor invalid. This bit is cleared by power-on reset and unaffected by other types of system reset. 0 Calibration Remap Descriptor *n* invalid 1 Calibration Remap Descriptor *n* valid
23 CRD23EN	Calibration Remap Descriptor 23 Enable This bit indicates whether the corresponding remap descriptor is valid. There is also a global calibration remap enable (PFCRCR[GRMEN]) that also must be asserted to enable any remap functionality. Any write to PFCRDn.Word{0,1,2} clears the corresponding PFCRDE[CRDnEN] bit, leaving the calibration remap descriptor invalid. This bit is cleared by power-on reset and unaffected by other types of system reset. 0 Calibration Remap Descriptor *n* invalid 1 Calibration Remap Descriptor *n* valid

Table 511. PFCRDE field descriptions (continued)

Field	Description
24 CRD24EN	Calibration Remap Descriptor 24 Enable This bit indicates whether the corresponding remap descriptor is valid. There is also a global calibration remap enable (PFCRCR[GRMEN]) that also must be asserted to enable any remap functionality. Any write to PFCRDn.Word{0,1,2} clears the corresponding PFCRDE[CRDnEN] bit, leaving the calibration remap descriptor invalid. This bit is cleared by power-on reset and unaffected by other types of system reset. 0 Calibration Remap Descriptor *n* invalid 1 Calibration Remap Descriptor *n* valid
25 CRD25EN	Calibration Remap Descriptor 25 Enable This bit indicates whether the corresponding remap descriptor is valid. There is also a global calibration remap enable (PFCRCR[GRMEN]) that also must be asserted to enable any remap functionality. Any write to PFCRDn.Word{0,1,2} clears the corresponding PFCRDE[CRDnEN] bit, leaving the calibration remap descriptor invalid. This bit is cleared by power-on reset and unaffected by other types of system reset. 0 Calibration Remap Descriptor *n* invalid 1 Calibration Remap Descriptor *n* valid
26 CRD26EN	Calibration Remap Descriptor 26 Enable This bit indicates whether the corresponding remap descriptor is valid. There is also a global calibration remap enable (PFCRCR[GRMEN]) that also must be asserted to enable any remap functionality. Any write to PFCRDn.Word{0,1,2} clears the corresponding PFCRDE[CRDnEN] bit, leaving the calibration remap descriptor invalid. This bit is cleared by power-on reset and unaffected by other types of system reset. 0 Calibration Remap Descriptor *n* invalid 1 Calibration Remap Descriptor *n* valid
27 CRD27EN	Calibration Remap Descriptor 27 Enable This bit indicates whether the corresponding remap descriptor is valid. There is also a global calibration remap enable (PFCRCR[GRMEN]) that also must be asserted to enable any remap functionality. Any write to PFCRDn.Word{0,1,2} clears the corresponding PFCRDE[CRDnEN] bit, leaving the calibration remap descriptor invalid. This bit is cleared by power-on reset and unaffected by other types of system reset. 0 Calibration Remap Descriptor *n* invalid 1 Calibration Remap Descriptor *n* valid
28 CRD28EN	Calibration Remap Descriptor 28 Enable This bit indicates whether the corresponding remap descriptor is valid. There is also a global calibration remap enable (PFCRCR[GRMEN]) that also must be asserted to enable any remap functionality. Any write to PFCRDn.Word{0,1,2} clears the corresponding PFCRDE[CRDnEN] bit, leaving the calibration remap descriptor invalid. This bit is cleared by power-on reset and unaffected by other types of system reset. 0 Calibration Remap Descriptor *n* invalid 1 Calibration Remap Descriptor *n* valid
29 CRD29EN	Calibration Remap Descriptor 29 Enable This bit indicates whether the corresponding remap descriptor is valid. There is also a global calibration remap enable (PFCRCR[GRMEN]) that also must be asserted to enable any remap functionality. Any write to PFCRDn.Word{0,1,2} clears the corresponding PFCRDE[CRDnEN] bit, leaving the calibration remap descriptor invalid. This bit is cleared by power-on reset and unaffected by other types of system reset. 0 Calibration Remap Descriptor *n* invalid 1 Calibration Remap Descriptor *n* valid

Table 511. PFCRDE field descriptions (continued)

Field	Description
30 CRD30EN	Calibration Remap Descriptor 30 Enable This bit indicates whether the corresponding remap descriptor is valid. There is also a global calibration remap enable (PFCRCR[GRMEN]) that also must be asserted to enable any remap functionality. Any write to PFCRDn.Word{0,1,2} clears the corresponding PFCRDE[CRDnEN] bit, leaving the calibration remap descriptor invalid. This bit is cleared by power-on reset and unaffected by other types of system reset. 0 Calibration Remap Descriptor *n* invalid 1 Calibration Remap Descriptor *n* valid
31 CRD31EN	Calibration Remap Descriptor 31 Enable This bit indicates whether the corresponding remap descriptor is valid. There is also a global calibration remap enable (PFCRCR[GRMEN]) that also must be asserted to enable any remap functionality. Any write to PFCRDn.Word{0,1,2} clears the corresponding PFCRDE[CRDnEN] bit, leaving the calibration remap descriptor invalid. This bit is cleared by power-on reset and unaffected by other types of system reset. 0 Calibration Remap Descriptor *n* invalid 1 Calibration Remap Descriptor *n* valid

36.4.1.7 PFlash calibration region descriptor *n* (PFCRD*n*)

Each 96-bit (12-byte) region descriptor specifies an overlay region where a Flash access can be remapped during calibration and debug. The calibration remap descriptors are organized sequentially as 128-bit (16-byte) structures in the Platform Flash Controller's programming model. Each of the three 32-bit words that define a single calibration region are detailed in the subsequent sections; the fourth word is unused.

36.4.1.7.1 PFlash calibration region descriptor *n*, Word0 (PFCRD*n*_Word0)

The first word of the Flash memory controller overlay region descriptor defines the 0-modulo-*size* logical start (byte) address of the calibration remap region. It is software's responsibility to guarantee the low-order bits of the address, as defined by the remap descriptor size, are zeroed to enable the remap descriptor hit logic to function correctly.

Successful writes to this word clear the calibration remap descriptor's valid bit.

Offset 0x100 + *n**0x10 (n=0 to 31) Access: Supervisor Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	LSTARTADDR															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	LSTARTADDR													0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 401. PFlash calibration region descriptor *n*, Word0 (PFCRD*n*_Word0)

Table 512. PFCRD*n*_Word0 field descriptions

Field	Description
0:28 LSTARTADDR	Logical Start Address This field defines the most significant bits of the 0-modulo-*size* logical start address of the overlay remap region. It corresponds to the logical system address which maps to the Flash memory space. Any write to PFCRDn.Word{0,1,2} clears the corresponding PFCRDE[CRDnEN] bit, leaving the calibration remap descriptor invalid.

36.4.1.7.2 PFlash calibration region descriptor *n*, Word1 (PFCRD*n*_Word1)

The second word of the Flash memory controller calibration region descriptor defines the 0-modulo-*size* byte physical start (byte) address of the calibration remap region. *The contents of this word define the targeted destination overlay memory*. It is software's responsibility to guarantee the low-order bits of the address, as defined by the remap descriptor size, are zeroed to enable the remap descriptor hit logic to function correctly.

Successful writes to this word clear the calibration remap descriptor's valid bit.

Offset 0x104 + *n**0x10 (n=0 to 31) Access: Supervisor Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	PSTARTADDR															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	PSTARTADDR													0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 402. PFlash calibration region descriptor *n*, Word1 (PFCRD*n*_Word1)

Table 513. PFCRD*n*_Word1 field descriptions

Field	Description
0:28 PSTARTADDR	Calibration Remap Descriptor *n* Physical Start Address This field defines the most significant bits of the 0-modulo-size physical start byte address of the calibration remap descriptor. This address corresponds to the physical address which maps to the destination calibration overlay memory. Any write to PFCRDn.Word{0,1,2} clears the corresponding PFCRDE[CRDnEN] bit, leaving the calibration remap descriptor invalid.

36.4.1.7.3 PFlash calibration region descriptor *n*, Word2 (PFCRD*n*_Word2)

The third word of the calibration region descriptor defines a per-master calibration remap enable and the remap region size. Writes to this word clear the calibration remap descriptor's valid bit.

Offset 0x108 + *n**0x10 (n=0 to 31) Access: Supervisor Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	M0EN	M1EN	M2EN	M3EN	M4EN	M5EN	M6EN	M7EN	M8EN	M9EN	M10EN	M11EN	M12EN	M13EN	M14EN	M15EN
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R												CRDSize				
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 403. PFlash calibration region descriptor *n*, Word2 (PFCRD*n*_Word2)

Table 514. PFCRD*n*_Word2 field descriptions

Field	Description
0:15 MxEN	Calibration Remap Descriptor n Master x Enable These bits determine whether calibration remapping is performed as defined by this descriptor based on the logical master ID of the requesting AHB master. If the current access hits in the remap region defined by this descriptor, and the MxEN field which corresponds to the requesting AHB master is asserted, then calibration remapping is performed and the content is fetched from the overlay RAM. If the current access hits in the remap region defined by this descriptor, but the MxEN field which corresponds to the requesting AHB master is deasserted, then calibration remapping is not performed and the content is fetched from the Flash. 0 Calibration remap evaluation is ignored on Flash access requests from Master X as defined by this region descriptor. 1 Calibration remap evaluation is enabled on Flash access requests from Master X as defined by this region descriptor.
27:31 CRDSize	Calibration Remap Descriptor *n* Size This field specifies the size of the calibration remap region. 00000 Reserved 00001 Reserved 00010 Reserved 00011 Region size is 8 bytes. Low-order 3 bits of LSTARTADDR and PSTARTADDR must be zero 00100 Region size is 16 bytes. Low-order 4 bits of LSTARTADDR and PSTARTADDR must be zero 00101 Region size is 32 bytes. Low-order 5 bits of LSTARTADDR and PSTARTADDR must be zero 00110 Region size is 64 bytes. Low-order 6 bits of LSTARTADDR and PSTARTADDR must be zero 00111 Region size is 128 bytes. Low-order 7 bits of LSTARTADDR and PSTARTADDR must be zero 01000 Region size is 256 bytes. Low-order 8 bits of LSTARTADDR and PSTARTADDR must be zero 01001 Region size is 512 bytes. Low-order 9 bits of LSTARTADDR and PSTARTADDR must be zero 01010 Region size is 1 KB. Low-order 10 bits of LSTARTADDR and PSTARTADDR must be zero 01011 Region size is 2 KB. Low-order 11 bits of LSTARTADDR and PSTARTADDR must be zero 01100 Region size is 4 KB. Low-order 12 bits of LSTARTADDR and PSTARTADDR must be zero 01101 Region size is 8 KB. Low-order 13 bits of LSTARTADDR and PSTARTADDR must be zero 01110 Region size is 16 KB. Low-order 14 bits of LSTARTADDR and PSTARTADDR must be zero 01111 Region size is 32 KB. Low-order 15 bits of LSTARTADDR and PSTARTADDR must be zero 10000 Region size is 64 KB. Low-order 16 bits of LSTARTADDR and PSTARTADDR must be zero 10001 Region size is 128 KB. Low-order 17 bits of LSTARTADDR and PSTARTADDR must be zero 10010 Region size is 256 KB. Low-order 18 bits of LSTARTADDR and PSTARTADDR must be zero 10011 Region size is 512 KB. Low-order 19 bits of LSTARTADDR and PSTARTADDR must be zero 10100 Region size is 1 MB. Low-order 20 bits of LSTARTADDR and PSTARTADDR must be zero 10101 Region size is 2 MB. Low-order 21 bits of LSTARTADDR and PSTARTADDR must be zero 10110 Region size is 4 MB. Low-order 22 bits of LSTARTADDR and PSTARTADDR must be zero 10111 Region size is 8 MB. Low-order 23 bits of LSTARTADDR and PSTARTADDR must be zero 11000 Reserved 11001 Reserved 11010 Reserved 11011 Reserved 11100 Reserved 11101 Reserved 11110 Reserved 11111 Reserved

36.5 Functional description

As shown in *Figure 394* the Flash memory controllers interface between:

- The AHB system bus ports
- Nexus trace
- Flash memory array
- Overlay RAM
- Off-chip Buddy Device
- System RAM

For accesses targeting the Flash array, the Flash memory controller generates read and write enables, block selects, array address, write size and write data as inputs to the Flash array. The Flash memory controller captures read data from the Flash array and drives it onto the AHB system bus. Up to four pages of data (256-bit page size) may be buffered in each of two ways of the Flash memory controller mini-cache. Lines may be prefetched in advance of being requested, allowing single-cycle (zero AHB wait-states) read data responses on buffer hits.

Several prefetch control algorithms are available for controlling line read buffer fills. Prefetch triggering may be restricted to instruction accesses only, data accesses only, or may be unrestricted. Prefetch triggering may also be controlled on a per-master basis.

Buffers may also be selectively enabled or disabled for allocation by instruction and data prefetch.

Access protections may be applied on a per-master basis for both reads and writes to support security and privilege mechanisms.

36.5.1 Basic interface protocol

Read accesses are terminated under control of the appropriate wait state settings. Access timing can be varied to account for the operating conditions of the SoC (frequency, voltage, temp) by appropriately setting the PFCR1[RWSC] field.

36.5.2 Access protections

36.5.2.1 PFAPR - Platform Flash access protection register (PFAPR)

The Flash memory controller provides programmable, configurable access protections for both read and write cycles on a per-master basis via the PFlash access protection register (PFAPR). It allows restriction of read and write requests on a per-master basis. This functionality is described in *Section 36.4.1.4*. Detection of a protection violation results in an error response from the Flash memory controller on the AHB transfer.

36.5.2.2 BAF execution disable

The Flash memory controller provides a mechanism to disable instruction execution of the BAF (Boot Assist Flash) code. Setting the BAF Disable field (BAF_DIS) in the PFlash configuration register 3 (PFCR3) prevents execution of the BAF instructions. When this field is set, all incoming requests from the system bus on either Port0 or Port1 are evaluated. If the Flash memory controller detects an attempted instruction fetch access to the BAF region of the Flash, the Flash memory controller aborts the access and returns a system bus error. Data accesses to the BAF region are not affected by the state of PFCR3[BAF_DIS].

36.5.3 Read cycles – buffer miss

On an incoming AHB read request, a mini-cache lookup and access privilege evaluation are performed during the AHB address phase. In the event of a buffer miss, a Flash access is initiated. If globally enabled, the calibration remap evaluation is also performed. If the access is to be remapped, the appropriate access is routed to the destination memory.

If the Flash access was the direct result of an AHB transaction, the corresponding page buffer is loaded and marked as most-recently-used. If the Flash access was the result of a speculative prefetch to the next sequential line, it is loaded into the least-recently-used buffer. The status of this buffer is not changed to most-recently-used until a subsequent buffer hit occurs as a result of an AHB read request.

36.5.4 Read cycles – buffer hit

Single cycle read responses to the AHB are possible with the Flash memory controller when the requested read access was previously loaded into one of the page buffers. In these “buffer hit” cases, read data is returned on the system bus with a zero wait-state response.

36.5.5 Error termination

The Flash memory controller may invoke a system bus error termination in the following scenarios:

- Attempted access by an AHB master whose corresponding Read access Control or Write access Control settings do not allow the access, thus causing a protection violation. See *Section 36.4.1.4* for more detail. In this case the Flash memory controller does not initiate a Flash array access.
- Attempted instruction fetch by an AHB master to the BAF Flash region when the PFCR3[BAF_DISABLE] field in Platform Flash configuration register 3 (PFCR3) is set. The assertion of PFCR3[BAF_DISABLE] indicates that code stored in the BAF region should not be executed. See *Section 36.4.1.3* for more detail. In this case, the Flash memory controller does not initiate a Flash array access.
- The Flash or overlay RAM returns an error response on an attempted access (see *Section 36.5.6*).
- Attempted access by an AHB master to a reserved region in the Flash memory map.

36.5.6 Flash error response operation

The Flash array may signal an error response to terminate a requested access due to improper sequencing during program/erase operations, or improper sequencing during array integrity testing. When an error response is received, the Flash memory controller does not update or validate a page read buffer. An error response may be signaled on read or interlock write operation. For more information on the specifics related to signaling of Flash errors, including Flash ECC events, array integrity testing and read-while-write events, refer to the Flash memory chapter.

36.5.7 Security module exclusive control

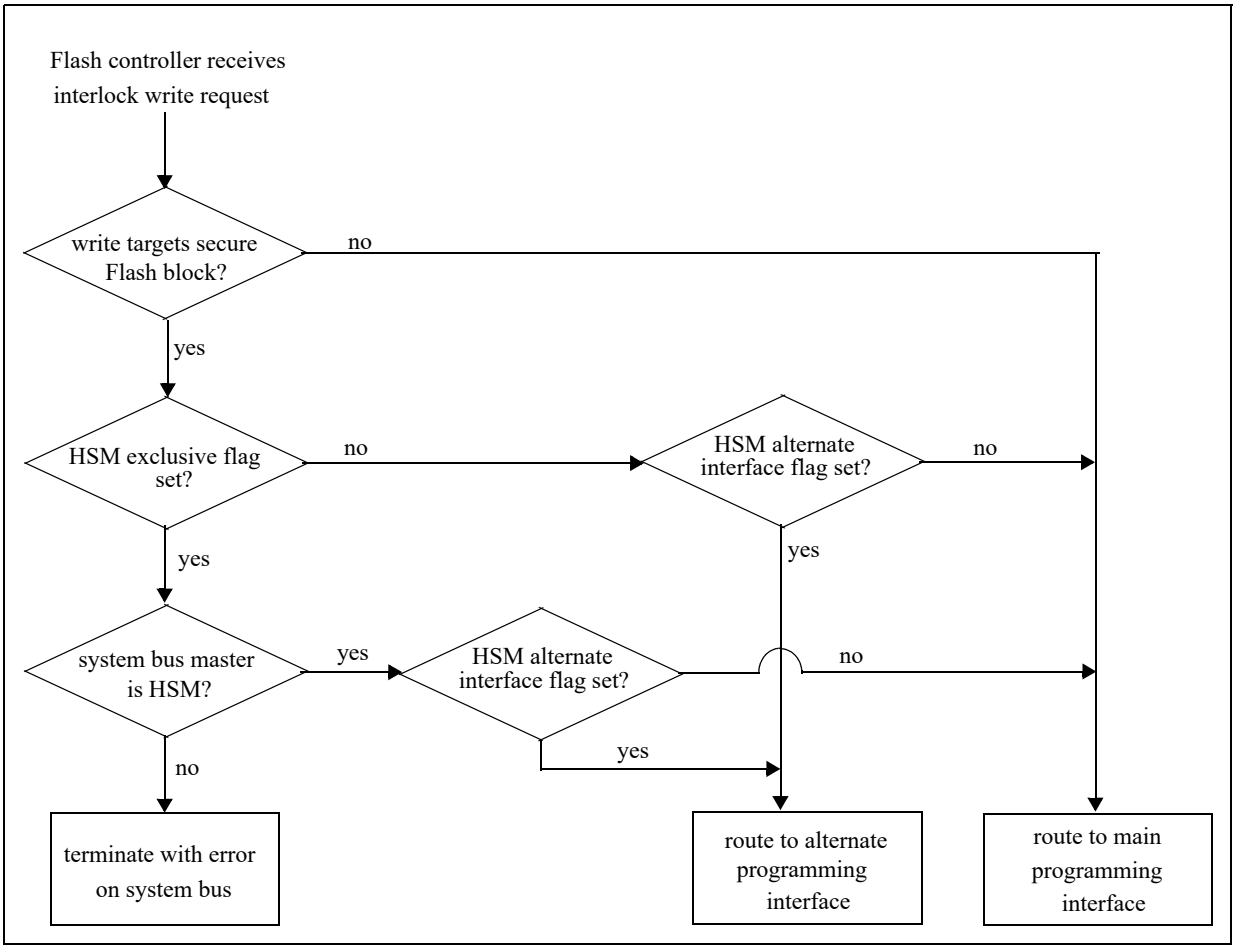
During “Production at OEM” lifecycle state and beyond, the HSM can establish the secure Flash blocks to be exclusive to the HSM and route program or erasing of secure Flash blocks to the alternate programming interface.

The secure Flash space is independently controlled for each of the following regions:

- 16 KB secure code Flash block at address range 0x0060_C0000 - 0x0060_FFFF
- 64 KB secure code Flash blockA at address range 0x0061_0000 - 0x0061_FFFF
- 64 KB secure code Flash blockB at address range 0x0062_0000 - 0x0062_FFFF
- Two 16 KB secure data Flash blocks at address range 0x0068_0000 - 0x0068_7FFF

The HSM provides flags to individually control the HSM-exclusivity for each of the four secure Flash regions. When a given secure Flash block is marked as exclusive to the HSM, the HSM is the only system bus master granted access to that block and access attempts by non-HSM masters are terminated with error. Once HSM-exclusivity is established for a given secure Flash block, the HSM is guaranteed access to that block, regardless of the state of the censorship controls. Secure Flash exclusivity applies to read and write accesses.

Figure 404. Interlock Write Alternate Programming Interface Routing Flowchart



36.5.8 Access pipelining

Accesses to the Flash array may be pipelined by driving a subsequent access address and control signals while waiting for the current access to complete. Pipelined access requests are always run to completion and are not aborted by the Flash memory controller. Flash access pipelining allows for improved performance by reducing the access latency seen by

the AHB master. Access pipelining may be applied only to read cycles targeting the Flash array. Address pipelining is enabled by setting the PFCR1[APC] field.

Access pipelining is only supported for normal Flash access and is not supported for calibration overlay RAM fetches. On calibration reads, the setting of PFCR1[APC] is ignored.

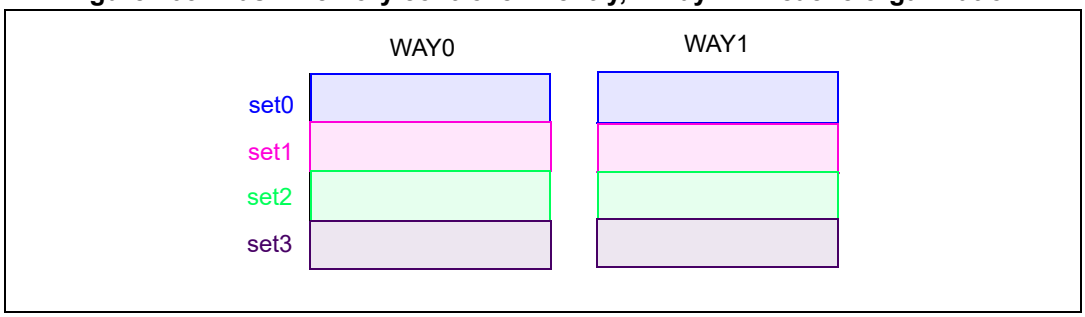
Note: Not all combinations of PFCR1[APC] (Address Pipeline Control) and PFCR1[RWSC] (Read Wait State Control) settings are recommended or supported. Please refer to the device data sheet for guidance.

36.5.9 Line read buffers and prefetch operation

The AHB ports of the Flash memory controller each contain a two-way set-associative mini cache, where each way contains four page buffers which are used to hold data read from the Flash arrays. Each 256-bit buffer operates independently, and is filled using a single array access. The buffers are used for both prefetch and normal demand fetches.

Prefetch triggering is controllable on a per-master and access-type basis (see *Section 36.4.1.1*). Bus masters may be enabled or disabled from triggering prefetches, and triggering may be further restricted based on whether a read access is for instruction or data. A read access by the Flash memory controller may trigger a prefetch to the next sequential line of array data on the cycle following the request. The access address is incremented by 32 bytes, and a subsequent Flash access is initiated. A Flash array prefetch is initiated if the data is not already resident in a line read buffer. Prefetched data is always loaded into the least-recently-used buffer.

Figure 405. Flash memory controller 4-entry, 2-way mini-cache organization



Once the candidate line buffer has been selected, the Flash array is accessed and read data loaded into the buffer. If the buffer load was in response to a miss, the buffer which was loaded is immediately marked as most-recently-used. If the buffer load was in response to a speculative fetch to the next-sequential line address after a buffer hit, *the recently-used status is not changed*. Rather, it is marked as most-recently-used only after a subsequent buffer hit.

This policy maximizes performance based on reference patterns of Flash accesses and allows for prefetched data to remain valid when non-prefetch enabled bus masters are granted Flash access.

Several algorithms are available for prefetch control which trade off performance for power. They are described in *Section 36.4.1.1*. More aggressive prefetching may increase power due to the number of potentially discarded prefetches, but may increase performance by lowering average read latency.

For prefetching to occur, all of the following must apply:

1. PFCR{1,2}[P{0,1}_BFEN] must be set to ‘1’,
2. PFCR{1,2}[P{0,1}_PFLIM] must be non-zero, and
3. either PFCR{1,2}[P{0,1}_IPFEN] or PFCR{1,2}[P{0,1}_DPFEN] must be asserted

Refer to *Section 36.4.1.1* for a description of these controls.

36.5.10 Instruction/Data prefetch triggering

Port0 prefetch triggering may be enabled for instruction reads via the PFCR1[IPFEN] control field, while Port0 prefetching for data reads is enabled via the PFCR1[DPFEN] control field. Additionally, the PFCR1[PFLIM] must also be set to enable prefetching on Port0. Refer to *Section 36.4.1.1* for a description of these controls. Prefetches are never triggered by write cycles.

Port1 prefetch triggering may be enabled for instruction reads via the PFCR2[IPFEN] control field, while Port1 prefetching for data reads is enabled via the PFCR2[DPFEN] control field. Additionally, the PFCR2[PFLIM] must also be set to enable prefetching on Port1. Refer to *Section 36.4.1.2* for a description of these controls. Prefetches are never triggered by write cycles.

36.5.11 Per-Master prefetch triggering

Prefetch triggering may be controlled for individual bus masters. Refer to *Section 36.4.1.1* for a description of these controls.

36.5.12 Buffer allocation

Allocation of the line read buffers is controlled via the PFCR3 control register, specifically the line buffer configuration ({P0,P1}_WCFG) field. Refer to *Section 36.4.1.3*. The buffers can be organized as a “pool” of available resources (with both ways within a given set) or with a fixed partition between ways allocated to instruction or data accesses. For the fixed partitions, way 0 is allocated for instruction fetches and way 1 for data accesses.

36.5.13 PFlash calibration remap support

The Flash memory controller supports calibration development by providing a remapping function to route Flash accesses to overlay RAM on-chip system RAM that can be used as overlay RAM. Recall this family of devices includes a comprehensive set of features including dedicated overlay RAM arrays to aid calibration and debug.

The overlay function supports the following features:

- Can be mapped over internal Flash memory
  - Allows calibration of constant data without requirement for additional external RAMs and calibration memory interfaces
- Can be used to hold device debug trace stream
  - Allows limited trace based debug without requirement for complex debug tool hardware and access to high speed trace port
- Protected with error detection and correction mechanism
- Accesses to all overlay RAMs can achieve the same timing as accesses to internal Flash
  - Timing for calibration accesses to a particular overlay RAM is only guaranteed if it is not being used as a destination for trace streaming at the same time
  - Timing for calibration accesses to overlay RAM is only guaranteed if RWSC is set sufficiently high to cover the intrinsic burst access incurred when fetching a full page of calibration data
- Support for up to 32 distinct calibration remap regions.
- Protection against accidental enable and reconfiguration of calibration remap function:
  - Two stage mechanism to enable remap, using both global and individual calibration region descriptor enable bits
  - Optional configuration of calibration region descriptors as redundant pairs, allowing detection of accidental region reconfiguration.
- Support for overlay remapping to unique calibration regions on a per-master basis.

The calibration remap function supports the following overlay RAM targets:

- An internal overlay RAM is included on all standard production devices.
- An extended overlay RAM is an additional memory that is only available on special ED devices.
- A portion of the system RAM can be used for overlay.

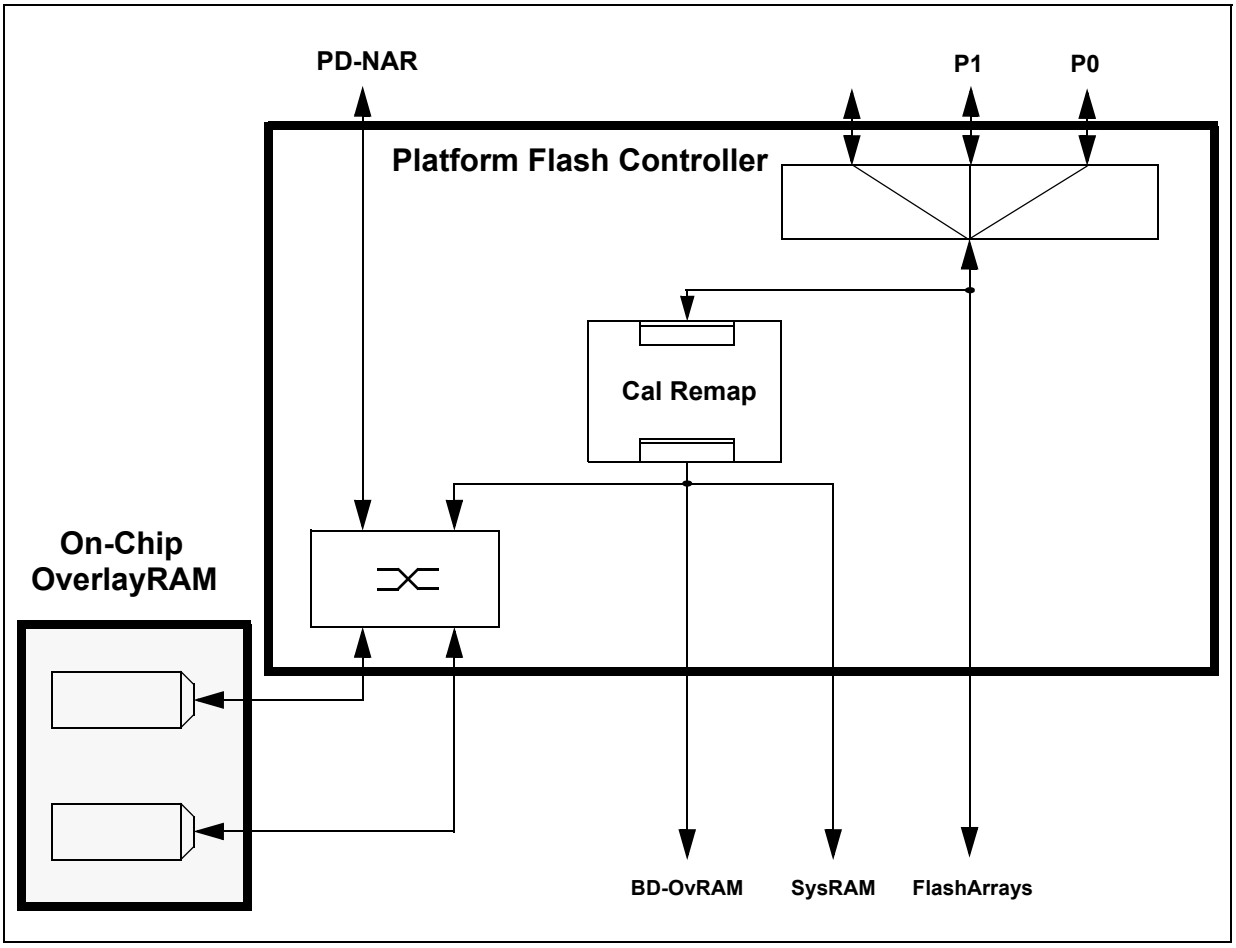
36.5.13.1 PFlash calibration remap to RAM

The *overlay RAM is selected based on the translated physical address*.

To better understand the functionality of the calibration remap, consider more detailed block diagrams, one showing the overall structure of the calibration remap function and another showing more detail on the actual remap logic. Recall the Flash memory controller implements a hardware interconnection matrix connecting the four input ports and four destination memories. See *Figure 406*.

The following figure illustrates the calibration remap connection matrix.

Figure 406. PFlash calibration remap connection matrix



Note the 64-bit on-chip overlay RAM is split into 2 equal-sized memories to allow concurrent accesses for calibration data from the P0, P1 (where these are the crossbar slave input ports) plus trace data writes from the Production Device Nexus Aurora Router (PD-NAR).

The translated physical address defines the destination memory. *Table 515* shows the base addresses for the calibration data memories as defined by the system memory map.

Table 515. Calibration data memory base addresses

Calibration memory	Base address(1)	Possible access sources
Flash Arrays	0x0{8,9,A}--_----	P0, P1
System RAM	0x40{0,1}-_----	P0, P1
On-chip Overlay RAM	0x0D00_----	P0, P1, PD-NAR
Buddy Device (Extended) Overlay RAM	0x0C{0,1}-_----	P0, P1

1. All memories are not implemented. Please refer to the *Table 505* for detailed information.

Note the trace data from the Nexus Aurora Router (PD-NAR) can only be routed to the on-chip overlay RAM; there is a different connection path in the Buddy Device from the BD-NAR into the extended overlay RAM.

The system RAM connection from the calibration remap logic is implemented via a private 64-bit connection to the platform RAM controller.

Accesses to the Flash arrays always use the logical, non-remapped system address. Calibration data store references to the system RAM, on-chip or buddy device (extended) overlay RAM always use the physical, non-remapped system address. Conversely, calibration data load references can be accessed using a translated physical address or, if provided by the requesting master, the physical, non-remapped system address. Byte-, halfword-, word- and doubleword-sized data store references to the on-chip overlay RAM are supported. Byte-, halfword-, word- and doubleword-sized store references to the buddy device (extended) overlay RAM are supported.

The read data interface of system RAM, on-chip overlay RAM and buddy (extended) device is 64 bits wide for each, whereas the Flash read data interface is 256 bits wide. As a result, a remapped calibration load access initiates a four-beat burst sequence to mimic the throughput of a single Flash access. There is a direct relationship between the operating frequency range and the required number of wait states to correctly sample Flash read data. By contrast, a remapped calibration load access has a fixed, intrinsic 4-beat latency, where the access time of each individual beat depends on the overlay target - system RAM, on-chip overlay RAM or buddy device. The access time when fetching data from the buddy device presents the longest latency. Therefore, in order for an initial, random calibration load access to mimic the Flash access time, PFCR1[RWSC] read wait state setting must be programmed to a value which sufficiently covers the intrinsic access time to complete a four-beat burst sequence from the slowest available overlay target at the specified device operating frequency. Expressed mathematically:

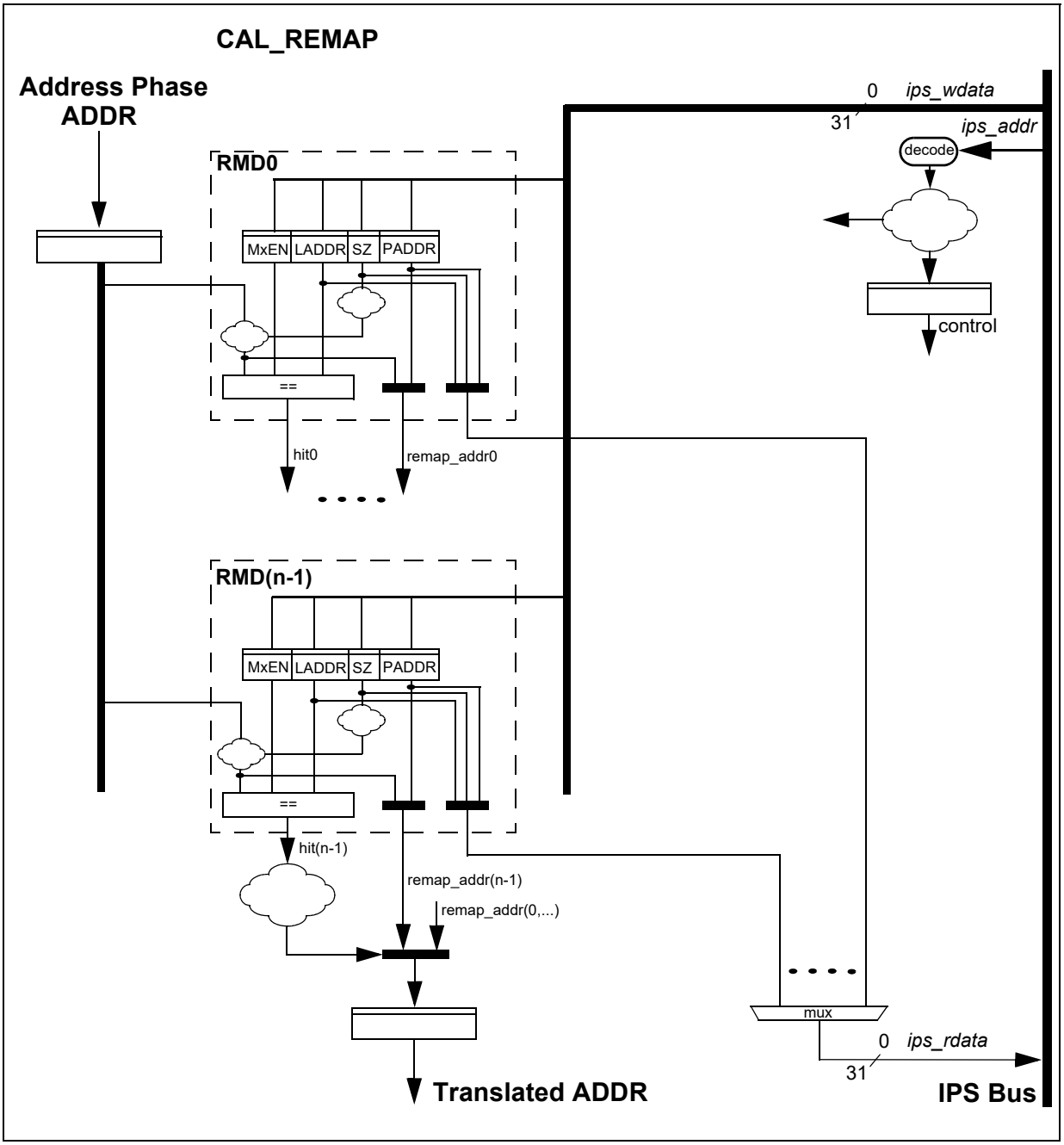
$$(\mathrm{RWSC}+2) \geq \left(\frac{\mathrm{IntrinsicFlashAccessTime}}{\mathrm{OperatingFrequencyClockPeriod}}+2\right) \geq (\mathrm{OverlayRAMTranferBurstLatency})$$

Notice the overall latency on a random, initial Flash access that does not hit in the prefetch mini-buffer incurs an additional cycle latency beyond the intrinsic Flash access time. This cycle accounts for pipeline stalls incurred by the Flash controller state machine. The formula above is based on specified intrinsic Flash access times of the array at 150C. Calibration data that is loaded in the Flash controller's mini-cache is returned with single-cycle latency.

The memory-mapped calibration remap (region) descriptor (CRD*n*) registers define the required address translation. Each calibration remap descriptor includes the logical base address, the translated physical base address and a region size (plus other control bits). The calibration remap evaluation is initiated when the Flash controller is presented with a Flash access request from any system bus master to a location in the mirrored Flash address space. Conversely, references using the associated non-mirrored Flash system address are not evaluated for calibration remapping. Generally, not all areas of the Flash memory map are available for calibration remapping. Secure Flash blocks, EEPROM, BAF and UTEST are usually not available for calibration remapping. Flash access requests which present an address targeting the secure Flash blocks, EEPROM, BAF or UTEST will bypass calibration remap evaluation and return Flash data.

The address translation mechanism requires the logical and physical base addresses of the region be aligned on 0-modulo-size boundaries. See *Figure 407*.

Figure 407. PFlash calibration remap logic detail



Each instantiation of the basic calibration remap function {RMD0,..., RMD(*n*-1)} in *Figure 407* includes the three descriptor registers containing the logical start address (LADDR in the figure), the physical start address (PADDR), region size (SZ), and per-master enable bits (MxEN). The region size is decoded to create the appropriate address bit enables which are then applied to the registered logical Flash address. An equality comparator then compares the adjusted logical Flash address against the logical address from the calibration remap descriptor to determine the region “hit”. The hit determination is further qualified based on the bus master number that initiated the Flash access. At the same time, the logical Flash address and the physical address defined in the calibration

remap descriptor are merged to form the translated address to be used in the event of a calibration region hit. Aside from the individual instantiations of the calibration remap descriptors, this logic examines all the descriptor hit indicators to select the appropriate translated address.

In the absence of a region “hit” data is returned from the Flash at the provided logical address.

It should be noted in the event of *overlapping remap regions*, the calibration remap descriptor with the largest number is used for the address translation, that is, if a logical Flash address hits in multiple descriptors, say CRDx and CRDy, the translated address from CRDy is used, given y > x. This definition is selected as it minimizes the gate cost and timing impact of the calibration remap descriptor “hit” logic.

In general, the calibration remap logic operates only on Flash data accesses. However, Flash instruction references can optionally remapped while in debug modes of operation to support the use of software breakpoints when PFCRC[IRMEN] is set.

36.5.13.2 PFlash calibration remap to Flash memory

Calibration remapping of a Flash access to another on-chip Flash address uses the same mechanism and controls as remapping to RAM addresses. The following should be noted:

- UTEST Flash, BAF, EEPROM, and secure Flash blocks cannot be remapped nor can accesses to other Flash blocks be rerouted to addresses in UTEST Flash, BAF, EEPROM, or secure Flash
- All other Flash blocks can be overlaid using the mirrored address range2.

36.5.14 Safety considerations

36.5.14.1 e2eECC (End-to-End ECC)

36.5.14.1.1 e2eECC and Flash accesses

There are multiple complications associated with the use of the standard e2eECC algorithm with Flash memory. The basic issues involve: the default state of a Flash block that is erased (all ones) and the fact that programming an erased Flash involves changing the state of memory bit from a logical 1 to a logical 0. These factors require that the system level e2eECC must be modified on references to the Flash memory in three ways.

First, the all ones codeword, since it reflects the state of an erased Flash location, is treated as a valid, error free encoding; in particular, it is required that the checkbits associated with the all ones data value and the all zeroes data value are required to be the same with a checkbit value of 0xFF. Second, since an erased Flash block generates an all ones data for all locations, the address field cannot be included in the ECC code used by the Flash. The Flash array implements special address re-encoding logic based on the decoded array address to protect against address faults. Third, there are additional ECC implications associated with the Flash memory and its support for EEPROM emulation. Lastly, there are ECC implications associated with the storage and retrieval of overlaid calibration data.

In summary, the basic operation of an NVM bit cell impacts the e2eECC algorithm used in the Flash memory in multiple ways. The required ECC adjustments are handled by the platform Flash controller before data is written to the Flash array(s) or applied to read data accessed from the array(s). These adjustments are two-fold. Consider a Flash write during a programming event:

1. Remove the address field from the ECC codeword by XOR'ing the address calculation of the H-matrix from the checkbits.
2. Invert the resulting 8-bit ECC checkbit vector. This step is required to support the all ones erased state.

On a Flash read operation, a similar but “reversed” set of steps are performed as the data is driven into the system bus interconnect:

1. Invert the 8-bit ECC checkbit vector read from the Flash.
2. Factor the address field into the ECC checkbits by XOR'ing the address calculation of the H-matrix.

Consider the following Flash ECC example. Let the Flash block containing address 0x00345678 be erased. The following is the sequence of operations:

1. Read address 0x00345678. Flash array returns fl_rdata = 0xFFFFFFFF_FFFFFFFF, fl_cdata = 0xFF. The Flash memory controller calculates the addr_chkbit = 0xC4 using the H-matrix; it inverts the fl_cdata vector to 0x00 and factors in the addr_chkbit = 0xC4 to produce the following valid e2eECC codeword:

```
addr = 0x00345678, rdata = 0xFFFFFFFF_FFFFFFFF, rchkbit = 0xC4 (0x00 ^ 0xC4)
```

2. Program address 0x00345678 with data = 0xBABEFACE_DEADBEEF. The initiating bus master calculates the e2eECC codeword as:

```
addr = 0x00345678, wdata = 0xBABEFACE_DEADBEEF, wchkbit = 0xE3
```

Before performing the interlock write to the Flash array, the Flash memory controller adjusts the chkbit value as fl_wchkbit = 0xD8 (0xE3 ^ 0xff ^ 0xC4) by toggling the original write checkbits and then factoring in the addr_ecc. Accordingly, the codeword stored in the Flash array is:

```
fl_data = 0xBABEFACE_DEADBEEF, fl_checkbit = 0xD8
```

3. Read address 0x00345678. The Flash array returns the stored codeword, fl_rdata = 0xBABEFACE_DEADBEEF, fl_rchkbit = 0xD8. The Flash memory controller inverts the checkbit vector and factors in the address checkbits (0xD8 ^ 0xFF ^ 0xC4) to produce:

```
addr = 0x00345678, rdata = 0xBABEFACE_DEADBEEF, rchkbit = 0xE3
```

This read codeword matches the original e2eECC codeword generated by the write.

Malfunction of the ECC logic described above may result in the corruption of the SEC/DED event reporting. The Flash memory controller performs EDC after ECC check on all Flash transactions. If a mismatch is detected, indicating a failure in the ECC logic, the event is reported to the device fault collection module.

36.5.14.2 Flash address generation check

Functional safety coverage of the address and data are handled by ECC performed within the Flash and e2eECC performed at the master. Functional safety coverage of the address path and control within the Flash memory controller rely on a feedback path between the Flash controller and Flash. Recall on a requested access to Flash, the Flash memory controller must decode the system AHB bus signals to generate the corresponding Flash interface signals to invoke a Flash lookup. In addition to providing the requested read data, the Flash also provides output sidebands reflecting the encoded address and block selects used to perform the actual row lookup.In the event of EDC after ECC errors detected on accesses to data and secure data flash, the event is not reported.

This sideband information is used by the Flash memory controller to verify the expected transaction. If a mismatch is detected, indicating a failure in the address generation or

control logic within the Flash memory controller or the transmission path between the Flash memory controller and Flash array, the event is forwarded to the device fault collection module and the corresponding buffer is invalidated.

36.5.14.2.1 On-Chip Overlay RAM feedback check

The integrity of the address and data on an overlay RAM transaction is covered by e2eECC check performed by the requesting master. Functional safety coverage of the address path and control within the Flash memory controller is handled by a transaction monitor which verifies the integrity of the transactions between the Flash memory controller and the on-chip overlay RAM. The function of the transaction monitor relies on a feedback path between the Flash memory controller and the on-chip overlay RAM, wherein the RAM provides latched address and control feedback outputs as an indication of received inputs when a RAM access is initiated. This feedback information is used by the Flash memory controller transaction monitor to verify the expected transaction. If a mismatch is detected, indicating a failure in the address generation or control logic within the Flash memory controller or the transmission path between the Flash memory controller and on-chip overlay RAM array, the event is forwarded to the Fault Collection and Control Unit (FCCU).

36.5.14.2.2 Safety considerations on overlay accesses

Because calibration write accesses are always presented with the physical, non-remapped system address, the write data checkbits presented by the master are calculated based on the physical overlay RAM system address. On a subsequent overlay read, the ECC checkbits must be manipulated to replace the physical overlay RAM address contribution with the logical Flash address contribution. This is required to satisfy the e2eECC check at the originating master.

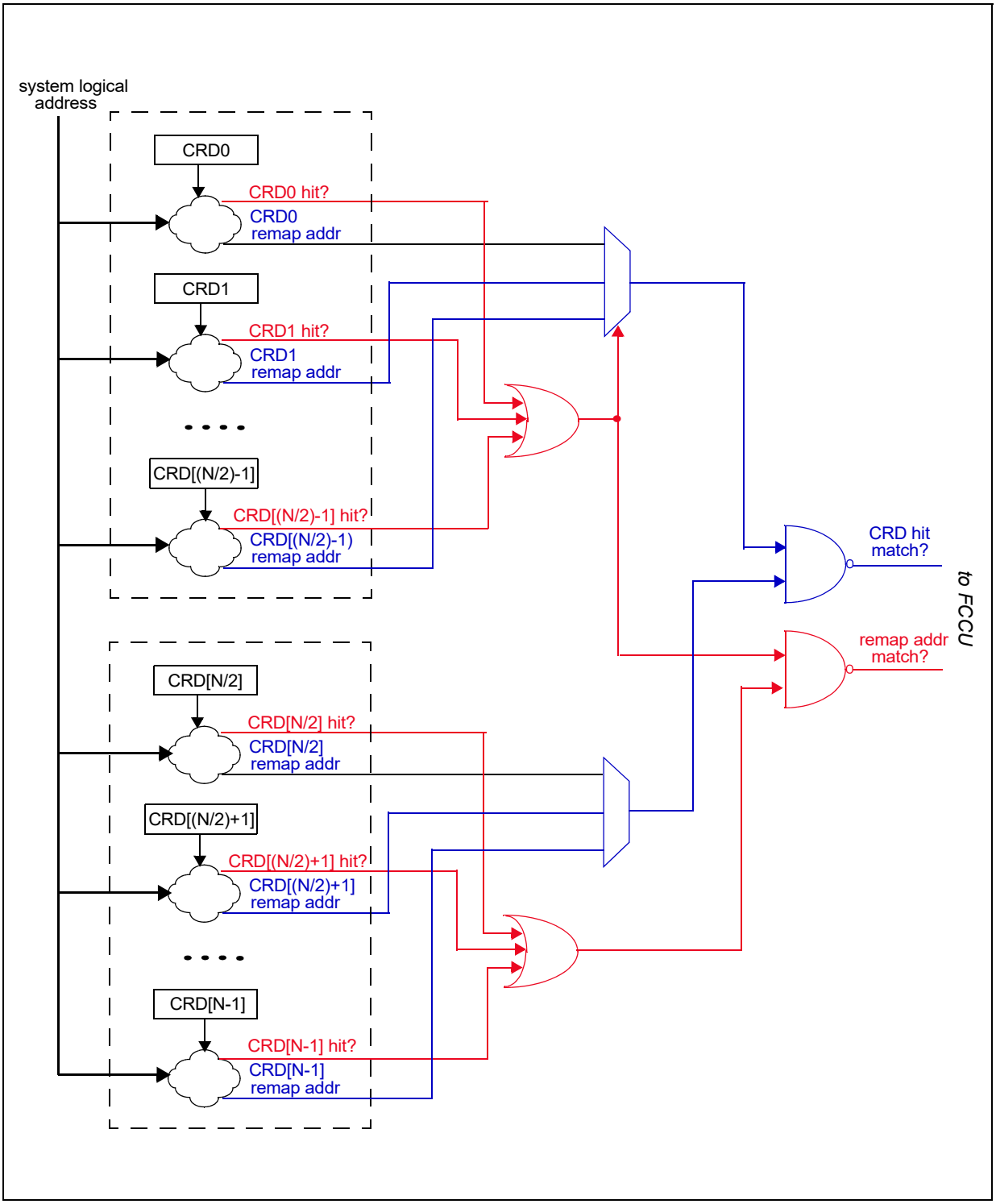
In addition, the Flash memory controller can be configured to treat calibration RAM space as a safety-critical component. By programming PFCRCR[SAFE_CAL]=1, the Flash memory controller treats the 32 total number of available calibration region descriptors (CRD0-CRD31) as a replicated group of 16 region descriptors that is half the size of the total number. More specifically, when N is the total number of region descriptors:

- The contents of CRD0 are expected to be redundantly defined in CRD[N/2]
- The contents of CRD1 are expected to be redundantly defined in CRD[(N/2)+1]
- The contents of CRD2 are expected to be redundantly defined in CRD[(N/2)+2]
- ...and so on
- The contents of CRD[(N/2)-1] are expected to be redundantly defined in CRD[N-1]

Note: It is the user's responsibility to establish redundant calibration region descriptors by programming the associated pairs of CRDs.

Incoming Flash read requests are compared simultaneously against the calibration regions defined in CRD0 to CRD[(N/2-1)] and the calibration regions defined in CRD[N/2] to CRD[N-1]. Due to the enabled redundancy, the parallel overlay remap hardware structures should evaluate to symmetric results. If a mismatch is detected, the event is reported to the FCCU and calibration remapping is not performed. Instead, data is returned from the Flash.

Figure 408. Safety-critical calibration remap datapath with redundancy



36.5.15 e2eECC on data Flash accesses

ECC events detected on accesses to the EEPROM Flash blocks are suppressed from being reported to the Memory Error Management Unit (MEMU).

In the event of a single-bit or double-bit correction, the corrected data and checkbits are returned to the requesting master, and the single-bit correction event is suppressed from being reported to the MEMU.

In the event of a non-correctable error detection, a fixed, illegal opcode value is returned to the requesting master along with the associated ECC checkbits as determined by the requesting address, and the non-correctable error event is suppressed from being reported to the MEMU. In the event of EDC after ECC errors detected on accesses to data and secure data flash, generation of the fault is suppressed. The related bit in MCR is not set and the event is not forwarded to the FCCU.

36.5.16 Array integrity considerations

During an array integrity sequence, the Flash memory array ignores any incoming read requests. When a Flash array integrity check is in progress, the Flash memory controller terminates all Flash access requests with an error.

37 Embedded Flash Memory

37.1 Introduction

The Flash Memory module serves as electrically programmable and erasable nonvolatile memory, for instructions and data storage.

The Flash Memory module is a nonvolatile, solid-state silicon memory device consisting of blocks of single transistor storage elements, an electrical means for selectively adding (programming) and removing (erasing) charge from these elements, and a means of selectively sensing (reading) the charge stored in these elements.

The Flash Memory module has two functional units:

- The Flash core: it is composed of a common high voltage HV block and a Flash array which can contain one or more Flash partitions. Each Flash partition is composed of arrayed nonvolatile storage elements, sense amplifiers, row decoders and column decoders. The arrayed storage elements in the Flash partitions are subdivided into physically separate units referred to as blocks.
- The Memory interface: it contains the registers and logic which control the operation of the Flash core and contains two main ports: the Array Interface and the Registers Interface. The memory interface is also the interface between the Flash Memory module and a Bus Interface Unit (BIU) and contains the ECC and redundancy logic.

37.1.1 Overview

The 6608 KB Flash Memory module contains six array partitions. Read-While-Write operations are only possible for separate partitions, meaning fetching/reading from one (or more) partition(s) while a program/erase operation is active on another partition. Each module counts as an RWW partition.

EEPROM emulation is managed as 4 × 64 KB blocks in two partitions, referred to as Data Flash.

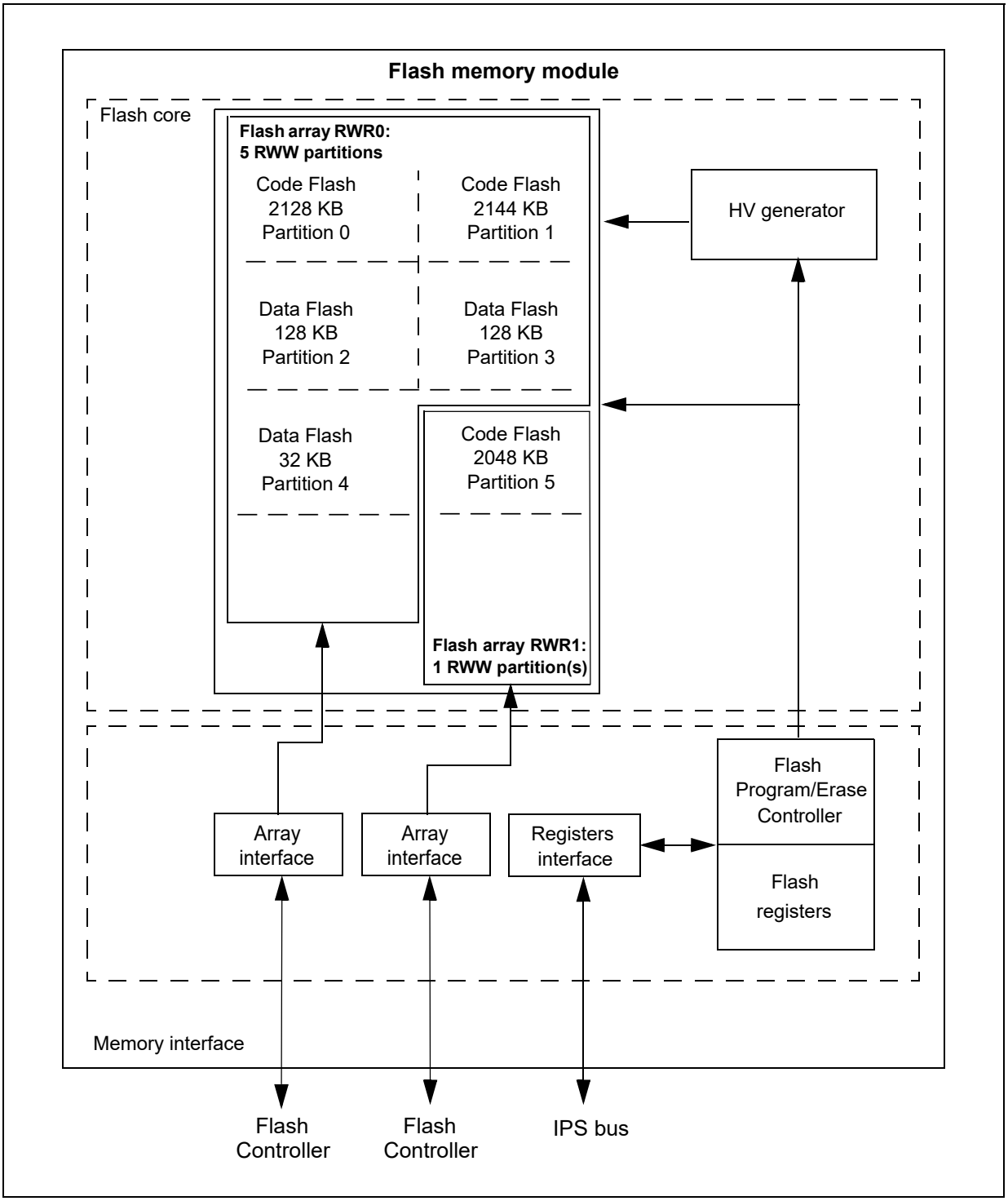
HSM EEPROM emulation is managed as 2 × 16 KB blocks in two partition, referred to as HSM Data Flash.

The Modify operations are managed by an embedded Flash Program/Erase Controller (FPEC). Commands to the FPEC are delivered through a User Registers interface.

The Read Data bus is 256-wide, while the Flash registers are on a separate, 32-bit bus.

The high voltages needed for Program and Erase operations are internally generated and are common for the six partitions.

Figure 409. Flash memory module structure



The Flash Memory module is designed for use in embedded applications which require high density non-volatile memories with high-speed Read access.

The Flash array is addressable:

- For programs in data partition: by word (32 bits) or doubleword (64 bits). ECC granularity is 64 bits.
- For programs in code partition: by word (32 bits) or doubleword (64 bits). ECC granularity is 128 bits.
- For reading code partitions – by page (256 bits).
- For reading data partitions – by half-page (128 bits).

Multiple-word or doubleword writes to the Flash memory may be performed to fill the program page buffer (256 bits), enabling page programming (256 bits – requiring 4 doubleword writes) and quad-page programming (1024 bits – requiring 16 doubleword writes). Code Flash memory reads always return 256, although Read buffering may be performed by the Bus Interface Unit (BIU). The address for each word retrieved within a page differs from the other addresses in the page by address bits (4:2).

Data Flash memory reads always return 128 bits, or two consecutive doublewords of information. The address for each word retrieved within a doubleword differs from the other addresses by address bits (3:2).

The Flash Memory module supports fault tolerance through Error Correction Code (ECC) and error detection. The ECC implemented within the Flash Memory module corrects double bit failures and detects triple bit failures.

The Flash Memory module uses an embedded hardware algorithm implemented in the Flash Program/Erase Controller to program and erase the Flash partitions.

Control logic that works with the software block-enables and software lock mechanisms are included in the embedded hardware algorithm to guard against accidental program/erase. The hardware algorithm performs the necessary steps to ensure that the storage elements are programmed and erased with sufficient margin to guarantee data integrity and reliability.

A programmed bit in the Flash array reads as logic level ‘0’ (or low). An erased bit in the Flash array reads as logic level ‘1’ (or high).

Programming and Erasing of the Flash array requires multiple system clock cycles to complete.

The Program and Erase sequences may be suspended or aborted.

37.1.2 Features

- 30 ns code/data array access time
- Code Partition High Read parallelism: 256 bits
- Data Partition Read parallelism: 128 bits
- Error Correction Code (Double Error Correction-Triple Error Detection) to enhance data integrity
- Data Partition Doubleword program granularity: 64 bits
- Erase with fast erase time at factory
- Blocks one time programmable (OTP) marking
- Read-While-Modify capability when the accesses are to different partitions
- Erase/Program Suspend
- Programming during Erase-Suspend
- Software programmable program/erase protection to avoid unwanted writings
- Temperature range (-40 °C to 165 junction)
- UTEST mode (user-accessible Test modes)
- Alternate interface provided for program and erase operations to dedicated master as private access to blocks
- Independent and Parallel Read access to specific Code Flash Partitions, namely Read-While-Read supported.
- Critical flops, which may affect Flash content, are triple-voted for safety application.

37.1.3 Modes of operation

The Flash Memory module supports the following modes of operation:

- Reset: the Flash Memory module initializes when the reset is deasserted.
- Power-down (Disable mode): the Flash Memory module removes all static consumption. The embedded Flash memory (array and registers) is not accessible to Read, Program or Erase once disabled. The wake-up time from Disable mode is longer than the wake-up time from Sleep mode.
- Low-power mode (Sleep mode): the Flash Memory module removes most of the static consumption. It is not accessible to Read, Program or Erase once disabled, however it does maintain a faster exit to Read mode with respect to Power-Down.
- Read mode: the Flash Memory module accepts Read accesses to the array.
- Modify mode: it is possible to read and write registers, Interlock Write the memory array, program the memory array, and erase the memory array. Program and Erase operations are initiated by performing array and register writes, and controlled by an internal state machine. Program/Erase operations or User Test modes (Margin Read or Array Integrity Check) are considered Main Modify modes.
- Read-While-Write: The Flash Memory module accepts Read accesses to array partitions that are not in Modify mode.
- Read-While-Read: The Flash Memory module accepts parallel Read accesses in separate Read-While-Read partitions.
- Alternate Modify mode: blocks in LOW and MID space could be assigned to Alternate interface, though which it is possible to read and write registers, Interlock Write the assigned memory array, program the assigned memory array, and erase the assigned memory array. Program and Erase operations are initiated by performing array and

alternate register writes, and are controlled by the same internal state machine as in Main. Alternate Program/Erase operations are considered Alternate Modify modes. Alternate Modify modes operates concurrently to Main ones and, although functionally independent, timing can be mutually influenced.

Such modes of operation are described in detail in *Section 37.4: Functional description*.

37.2 Flash memory map and description

The Flash Memory module block subdivision also facilitates independent Erase and Program protection. A software mechanism is provided to independently lock and unlock each block in Low, Mid, High and Data address spaces against program and erase.

37.2.1 Flash array memory map

37.2.1.1 Code Flash

Partition 0 accounts for 2128 KB of the total Flash Memory and is divided in 14 user blocks for code storage.

The sectorization of partition 0 of the Flash Memory is shown in *Table 516*.

Table 516. Flash memory partition 0 memory map

Block	Addresses	Size	Address space	LOCK	SEL
B0F0	0x00FC_4000 to 0x00FC_7FFF	16 KB	Low	LOWLOCK[1]	LOWSEL[1]
B0F1	0x00FC_C000 to 0x00FC_FFFF	16 KB	Low	LOWLOCK[2]	LOWSEL[2]
B0F2	0x00FD_0000 to 0x00FD_7FFF	32 KB	Low	LOWLOCK[6]	LOWSEL[6]
B0F3	0x00FE_0000 to 0x00FE_FFFF	64 KB	Low	LOWLOCK[8]	LOWSEL[8]
B0F4	0x00FF_0000 to 0x00FF_FFFF	64 KB	Low	LOWLOCK[9]	LOWSEL[9]
B0F5	0x0100_0000 to 0x0101_FFFF	128 KB	256K	A256KLOCK[0]	A256KSEL[0]
B0F6	0x0104_0000 to 0x0107_FFFF	256 KB	256K	A256KLOCK[2]	A256KSEL[2]
B0F7	0x0108_0000 to 0x010B_FFFF	256 KB	256K	A256KLOCK[3]	A256KSEL[3]
B0F8	0x010C_0000 to 0x010F_FFFF	256 KB	256K	A256KLOCK[4]	A256KSEL[4]
B0F9	0x0110_0000 to 0x0113_FFFF	256 KB	256K	A256KLOCK[5]	A256KSEL[5]
B0F10	0x0114_0000 to 0x0117_FFFF	256 KB	256K	A256KLOCK[6]	A256KSEL[6]
B0F11	0x0118_0000 to 0x011B_FFFF	256 KB	256K	A256KLOCK[7]	A256KSEL[7]
B0F12	0x011C_0000 to 0x011F_FFFF	256 KB	256K	A256KLOCK[8]	A256KSEL[8]
—	—	—	—	—	—
B0UT	0x0040_0000 to 0x0040_3FFF	16 KB	Low	TSLOCK	—
B0TF	0x0040_8000 to 0x0040_BFFF	16 KB	Reserved	—	—
—	—	—	—	—	—

The sectorization of partition 1 of the Flash Memory is shown in *Table 517*:

Table 517. Flash Memory Partition 1 memory map

Block	Addresses	Size	Address Space	LOCK	SEL
B1F0	0x00FC_0000 to 0x00FC_3FFF	16 KB	Low	LOWLOCK[3]	LOWSEL[3]
B1F1	0x00FC_8000 to 0x00FC_BFFF	16 KB	Low	LOWLOCK[4]	LOWSEL[4]
B1F2	0x00FD_8000 to 0x00FD_FFFF	32 KB	Low	LOWLOCK[7]	LOWSEL[7]
B1F3	0x0061_0000 to 0x0061_FFFF	64 KB	Low	LOWLOCK[10]	LOWSEL[10]
B1F4	0x0062_0000 to 0x0062_FFFF	64 KB	Low	LOWLOCK[11]	LOWSEL[11]
B1F5	0x0102_0000 to 0x0103_FFFF	128 KB	256K	A256KLOCK[1]	A256KSEL[1]
B1F6	0x0120_0000 to 0x0123_FFFF	256 KB	256K	A256KLOCK[9]	A256KSEL[9]
B1F7	0x0124_0000 to 0x0127_FFFF	256 KB	256K	A256KLOCK[10]	A256KSEL[10]
B1F8	0x0128_0000 to 0x012B_FFFF	256 KB	256K	A256KLOCK[11]	A256KSEL[11]
B1F9	0x012C_0000 to 0x012F_FFFF	256 KB	256K	A256KLOCK[12]	A256KSEL[12]
B1F10	0x0130_0000 to 0x0133_FFFF	256 KB	256K	A256KLOCK[13]	A256KSEL[13]
B1F11	0x0134_0000 to 0x0137_FFFF	256 KB	256K	A256KLOCK[14]	A256KSEL[14]
B1F12	0x0138_0000 to 0x013B_FFFF	256 KB	256K	A256KLOCK[15]	A256KSEL[15]
B1F13	—	—	—	—	—
B1BF	0x0040_4000 to 0x0040_7FFF	16 KB	LOW	LOWLOCK[0]	LOWSEL[0]
B1F15	0x0060_C000 to 0x0060_FFFF	16 KB	LOW	LOWLOCK[5]	LOWSEL[5]

The sectorization of partition 5 of the Flash Memory is listed below.

Table 518. Flash memory partition 5 memory map

Block	Addresses	Size	Address Space	LOCK	SEL
B2F0	0x013C_0000 to 0x013F_FFFF	256 KB	256K	A256KLOCK[16]	A256KSEL[16]
B2F1	0x0140_0000 to 0x0143_FFFF	256 KB	256K	A256KLOCK[17]	A256KSEL[17]
B2F2	0x0144_0000 to 0x0147_FFFF	256 KB	256K	A256KLOCK[18]	A256KSEL[18]
B2F3	0x0148_0000 to 0x014B_FFFF	256 KB	256K	A256KLOCK[19]	A256KSEL[19]
B2F4	0x014C_C000 to 0x014F_FFFF	256 KB	256K	A256KLOCK[20]	A256KSEL[20]
B2F5	0x0150_0000 to 0x0153_FFFF	256 KB	256K	A256KLOCK[21]	A256KSEL[21]
B2F6	0x0154_0000 to 0x0157_FFFF	256 KB	256K	A256KLOCK[22]	A256KSEL[22]
B2F7	0x0158_0000 to 0x015B_FFFF	256 KB	256K	A256KLOCK[23]	A256KSEL[23]

37.2.1.2 Data Flash

Data Flash is allocated in partition 2 and 3 of the Flash Memory as 4 × 64 KB blocks destined for data storage.

The Data Flash sectorization of the Flash Memory module is listed below.

Table 519. Flash Memory EEPROM DATA Partition 2 memory map

Block	Addresses	Size	Address Space	LOCK	SEL
D0_00	0x0080_0000 to 0x0080_FFFF	64 KB	High	HIGHLOCK[0]	HIGHSEL[0]
D0_01	0x0081_0000 to 0x0081_FFFF	64 KB	High	HIGHLOCK[1]	HIGHSEL[1]
—	—	—	—	—	—
—	—	—	—	—	—

Table 520. Flash Memory EEPROM DATA Partition 3 memory map

Block	Addresses	Size	Address Space	LOCK	SEL
D1_00	0x0082_0000 to 0x0082_FFFF	64 KB	High	HIGHLOCK[2]	HIGHSEL[2]
D1_01	0x0083_0000 to 0x0083_FFFF	64 KB	High	HIGHLOCK[3]	HIGHSEL[3]
—	—	—	—	—	—
—	—	—	—	—	—

HSM Data Flash is allocated in partition 4 and is composed by 2 x 16 KB blocks. The HSM Data Flash sectorization is listed below.

Table 521. Flash Memory HSM EEPROM Data Partition 4 memory map

Block	Addresses	Size	Address Space	LOCK	SEL
H0_00	0x0068_0000 to 0x0068_3FFF	16 KB	Mid	MIDLOCK[0]	MIDSEL[0]
H0_01	0x0068_4000 to 0x0068_7FFF	16 KB	Mid	MIDLOCK[1]	MIDSEL[1]

37.2.1.3 UTEST block memory map

The UTEST block is programmed and read independently of the other blocks. UTEST is included to support systems which require nonvolatile memory for security and to store system initialization information.

The usage of reserved UTEST block is detailed in *Table 522*.

Table 522. UTEST block memory map

Address Offset	Description	Size	Access
UTEST (Base Address 0x0040_0000 to 0x0040_3FFF)			
0x0000	Reserved	32 bytes	R
0x0020	Factory Erase Diary Location	16 bytes	R/P
0x0030	Test Mode Disable Override Passcode	16 bytes	R/P
0x0040	Test Mode Disable Seal	16 bytes	R/P
0x0050	Test Mode Disable Group A	16 bytes	R/P

Table 522. UTEST block memory map (continued)

Address Offset	Description	Size	Access
0x0060	Test Mode Disable Group B	16 bytes	R/P
0x00A0	UID	32 bytes	R
0x00C0	SW-DCF Start address	16 bytes	R
0x00D0	Reserved	48 bytes	R
0x0100	Password to enable the Flash Test Mode when in FA	32 bytes	R
0x0120	JTAG Password	32 bytes	R/P
0x0140(1)	PASS password - Group 0	32 bytes	R/P
0x0160	PASS password - Group 1	32 bytes	R/P
0x0180	PASS password - Group 2	32 bytes	R/P
0x01A0	PASS password - Group 3	32 bytes	R/P
0x01C0	Reserved	32 bytes	R/P
0x01E0	Life Cycle slot 0 - ST Production	32 bytes	R/P
0x0200	Life Cycle slot 1 - Customer Delivery	32 bytes	R/P
0x0220	Life Cycle slot 2 - OEM Production	32 bytes	R/P
0x0240	Life Cycle slot 3 - In Field	32 bytes	R/P
0x0260	Life Cycle slot 4 - Failure Analysis	32 bytes	R/P
0x0280	Customer Single Bit Correction area	32 bytes	R
0x02A0	Customer Double Bit Correction area	32 bytes	R
0x02C0	Customer Triple Bit Detection area	32 bytes	R
0x02E0	Customer Programmable Detection Area	32 bytes	R
0x0300	DCF Records	3328 bytes	R/P
0x1000	Customer OTP data	12288 bytes	R/P

1. The offset 0x140 contains the least significant 32-bit of the PASS password - Group 0. The offset 0x15C contains the most significant 32-bit of the PASS password - Group 0. The same endianess also applies to the other groups of the PASS password.

UTEST block supports RWW, and is grouped with the Code Flash in partition 0.

Programming of any section of the UTEST block is subject to similar restrictions as the array in terms of how ECC is calculated. Only one program is allowed per 128-bit ECC segment.

The UTEST Nonvolatile Memory section is OTP and erase is not allowed. User mode programming of the UTEST section is enabled only when MCR.PEAS is high. The UTEST section is divided into a *Flexible* area where content is managed at System level, and a *Reserved* area where addresses are pre-allocated for storing certain content. The UTEST section may be locked and unlocked against programming via LOCK0.TSLOCK in conjunction with the relative sideband (System specific). For locking purposes, UTEST is treated as a stand-alone block. UTEST access (both Read and Modify) is disabled in Test mode once the Test Mode Disable Seal password is written. Test Flash logical block gets also protected by the Test Mode Disable Seal.

The UTEST section is also set as OPP once the Test Mode Disable Seal password is written, thus programming is only allowed on virgin 128-bit locations.

37.2.1.4 RWR1_UTEST virtual block memory map

RWR1_UTEST virtual block is an area allocated for the purpose of injecting safety relevant error and to test the reaction path. This task is accomplished in the primary Read-While-Read partition 0 directly within UTEST memory map. RWR1_UTEST virtual block is assigned for the second, parallel, Read-While-Read partition 1 (see *Section 37.4.4.1: Read While Read*).

Warning: Customer Programmable Detection Area for RWR1_UTEST virtual block is qualified by UT0.CPA and UT0.CPE. UT0.CPR has no effect.

Table 523. RWR1_UTEST virtual block memory map

Address Offset	Description	Size	Access
RWR1_UTEST (Base Address 0x08FBC000)			
0x0000	Reserved	640 bytes	
0x0280	Customer Single Bit Correction area	32 bytes	R
0x02A0	Customer Double Bit Correction area	32 bytes	R
0x02C0	Customer Triple Bit Detection area	32 bytes	R
0x02E0	Customer Programmable Detection Area	32 bytes	R

37.3 Register memory maps and descriptions

In this section some nonvolatile registers are described as well as the usual volatile registers. Note that such entities are not Flip-Flops, but Test Flash block locations with a specific purpose.

During the Flash Initialization phase, the FPEC reads these nonvolatile registers and updates their corresponding volatile registers. When the FPEC detects ECC double errors in these special locations, it behaves in the following way:

- In the case of failing system locations (configurations, redundancy, EmbAlgo firmware), the initialization phase is interrupted and a Fatal Error is flagged.
- In the case of failing coherency checks, the volatile registers are filled with ‘1’s and the Flash Initialization ends setting MCR.RE.

37.3.1 Register memory maps

This section presents the memory mapping of all the Flash memory registers.

37.3.1.1 User register memory maps

The Flash User registers represent the communication interface between the host CPU and the FPEC.

Some register bits (command bits) are Read/Write for the CPU and read only for the FPEC. Other register bits (status bits) are Read/Write for the FPEC and read only for the CPU.

The Flash User Registers mapping is shown in *Table 524*.

Table 524. Flash user registers memory map

Address offset	Description	Location
0x0000	Module Configuration Register (MCR)	*Section 37.3.2.1*
0x0004	Alternate Module Configuration Register (MCRA)	*Section 37.3.2.2*
0x0008	Extended Module Configuration Register (MCRE)	*Section 37.3.2.3*
0x000C–0x000F	Reserved	
0x0010	Locking 0 register (LOCK0)	*Section 37.3.2.4*
0x0014	Locking 1 register (LOCK1)	*Section 37.3.2.5*
0x0018	Locking 2 register (LOCK2)	*Section 37.3.2.6*
0x001C	Locking 3 register (LOCK3)	*Section 37.3.2.7*
0x0020–0x0027	Reserved	
0x0028	Alternate Locking 0 register (LOCK0A)	*Section 37.3.2.8*
0x002C	Alternate Locking 1 register (LOCK1A)	*Section 37.3.2.9*
0x0038	Select 0 register (SEL0)	*Section 37.3.2.10*
0x003C	Select 1 register (SEL1)	*Section 37.3.2.11*
0x0040	Select 2 register (SEL2)	*Section 37.3.2.12*
0x0044	Select 3 register (SEL3)	*Section 37.3.2.13*
0x0048	Module Configuration Register 2 (MCR2)	*Section 37.3.2.14*
0x004C	Program Address Register (PAR)	*Section 37.3.2.15*
0x0050	Address Register (ADR)	*Section 37.3.2.16*
0x0054	User Test register 0 (UT0)	*Section 37.3.2.17*
0x0058	User Multiple Input Signature Register 0 (UM0)	*Section 37.3.2.18.1*
0x005C	User Multiple Input Signature Register 1 (UM1)	*Section 37.3.2.18.2*
0x0060	User Multiple Input Signature Register 2 (UM2)	*Section 37.3.2.18.3*
0x0064	User Multiple Input Signature Register 3 (UM3)	*Section 37.3.2.18.4*
0x0068	User Multiple Input Signature Register 4 (UM4)	*Section 37.3.2.18.5*
0x006C	User Multiple Input Signature Register 5 (UM5)	*Section 37.3.2.18.6*
0x0070	User Multiple Input Signature Register 6 (UM6)	*Section 37.3.2.18.7*
0x0074	User Multiple Input Signature Register 7 (UM7)	*Section 37.3.2.18.8*
0x0078	User Multiple Input Signature Register 8 (UM8)	*Section 37.3.2.18.9*
0x007C	User Multiple Input Signature Register 9 (UM9)	*Section 37.3.2.18.10*
0x0080	Over Program Protection 0 (OPP0)	*Section 37.3.2.19.1*
0x0084	Over Program Protection 1 (OPP1)	*Section 37.3.2.19.2*
0x0088	Over Program Protection 2 (OPP2)	*Section 37.3.2.19.3*

Table 524. Flash user registers memory map (continued)

Address offset	Description	Location
0x008C	Over Program Protection 3 (OPP3)	*Section 37.3.2.19.4*
0x0090	Test Mode Disable Password Check (TMD)	*Section 37.3.2.20*
0x0094	HSM ALT-SEL 0 register (ALTSEL0)	*Section 37.3.2.21.1*
0x0098	HSM ALT-SEL 1 register (ALTSEL1)	*Section 37.3.2.21.2*
0x0100	Erase Lock Protection 0 (ELOCK0)	*Section 37.3.2.22.1*
0x0104	Erase Lock Protection 1 (ELOCK1)	*Section 37.3.2.22.2*
0x0108	Erase Lock Protection 2 (ELOCK2)	*Section 37.3.2.22.3*
0x010C	Erase Lock Protection 3 (ELOCK3)	*Section 37.3.2.22.4*
0x0110	Program Lock Protection 0 (PLOCK0)	*Section 37.3.2.23.1*
0x0114	Program Lock Protection 1 (PLOCK1)	*Section 37.3.2.23.2*
0x0118	Program Lock Protection 2 (PLOCK2)	*Section 37.3.2.23.3*
0x011C	Program Lock Protection 3 (PLOCK3)	*Section 37.3.2.23.4*
0x0134	Parallel Address Register (ADR2)	*Section 37.3.2.24*

37.3.2 User register descriptions

This section consists of user register descriptions in address order.

37.3.2.1 Module Configuration Register (MCR)

The Module Configuration Register is used to enable and monitor all the Modify operations of the Flash Memory module and to monitor the Read operations of Read-While-Read Interface 0.

Address: 0x0000 Access: User Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	RRE	AEE	EEE	DWEE				0	0	0	SBC1	SAK	LSW	PEP	PES
W		w1c	w1c	w1c								w1c		w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	EER	RWE	SBC	RE	PEAS	DONE	PEG	PECIE	FERS	0	0	PGM	PSUS	ERS	ESUS	EHV
W	w1c	w1c	w1c													
Reset	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0

Figure 410. Module Configuration Register (MCR)

Table 525. MCR field descriptions

Field	Description
1 RRE	Read Reference Error RRE provides information on previous reads monitoring the analog Read references. If the current and voltage Read references are out of range, this bit is set to signify that previous reads requested may have been corrupted. This status flag is cleared to ‘0’ by writing ‘1’ to the register location. A Write of ‘0’ has no effect. 0 Reads are occurring normally. 1 A Read Reference Error occurred during a previous Read.
2 AEE	Address Encode Error On every Read request to the Flash, the incoming address is compared to an encoded address (row, column, and block) from the memory array using the read data sense amp timing. If these two values do not match (zero selected, multiple selected, wrong selected), or the timing is incorrect, an address encode error is recorded. This status flag is cleared to ‘0’ by writing ‘1’ to the register location. A Write of ‘0’ has no effect. 0 Reads are occurring normally. 1 An Address Encode Error occurred during a previous Read.
3 EEE	EDC after ECC Error EEE provides information on previous reads monitoring the EDC after ECC feature. On every Read request to the Flash, ECC parity is recomputed (encoding) just after ECC decoding, and if there is a mismatch between the ECC calculations (taking into account corrections or detections) a late error is reported. The EEE flag indicates this event when data is legal or containing up to 3 bit errors. This status flag is cleared to ‘0’ by writing ‘1’ to the register location. A Write of ‘0’ has no effect. 0 Reads are occurring normally. 1 A previous Read may have been decoded by an ECC logic corrupted.
4:7 DWEE	Double Word Uncorrectable ECC Error DWEE identifies which Double Word is involved in ECC Uncorrectable Error Event. In Code Flash 4 Double Words are read in parallel and ECC is evaluated on a couple of Double Words basis. Therefore DWEE value could be: 1111 if an ECC Uncorrectable Error is found in both couples of Double Words 1100 if an ECC Uncorrectable Error is found in the couple of Double Words at high address 0011 if an ECC Uncorrectable Error is found in the couple of Double Words at low address 0000 if no ECC Uncorrectable Error is found In Data Flash 2 Double Words are read in parallel and ECC is evaluated on a single Double Word basis. Therefore DWEE value could be: 1111 if an ECC Uncorrectable Error is found in both Double Words 1010 if an ECC Uncorrectable Error is found in Double Word at high address 0101 if an ECC Uncorrectable Error is found in Double Word at low address 0000 if no ECC Uncorrectable Error is found DWEE is a read only. Flags bits get cleared when EER is cleared.

Table 525. MCR field descriptions (continued)

Field	Description
11 SBC1	Single Bit Correction 1 SBC1 provides information on previous reads. This bit must then be cleared or a reset must occur before this bit returns to a ‘0’ state. This bit may not be set to ‘1’ by the user. In the event of an ECC Triple Error detection or ECC Double Error correction, this bit is not set. This status flag is cleared to ‘0’ by writing ‘1’ to the register location. A Write of ‘0’ has no effect. The function of this bit is SoC dependent and it can be disabled (through UT0.SBCE1). 0 Reads are occurring normally. 1 An ECC Single Error occurred and was corrected during a previous Read. In Code Flash, with 128b ECC coding, ECC correction is evaluated on couple of Double word basis. In Data Flash for EEE, ECC correction is evaluated on each Double word.
12 SAK	Suspend Acknowledge SAK provides information about the Acknowledge of a suspend request at the setting of DONE: SAK qualifies if the suspend has been accepted (initiated by PSUS or ESUS setting) or if the running command has been completed, since very close to natural end. The aim of this bit is for monitor. The sequence to resume normal functionality (PSUS or ESUS clearing) is the same both is suspend has been acknowledged or not. SAK is read Only 0 Operation has been completed prior that Suspend got processed 1 Suspend Acknowledged
13 LSW	Lock-Sel Write Error LSW provides information about Lock and Sel write denied during the program and erase sequence. If LSW is asserted, it must be cleared prior to attempting another high voltage operation. Since this bit is a status flag, it must be cleared to a 0 by writing a 1 to the register location. A write of 0 has no effect. 0 Lock and Sel write successful 1 Lock and Sel write not succeeded
14 PEP	Program and Erase Protection Error PEP provides information about program and erase operations with respect to protection errors. A protection error occurs if a program is attempted to a Locked block, or if an erase is selected to a locked block. This is evaluated prior to the operation beginning, and if an error is detected, high voltage operations will not be attempted for this request. If PEP is asserted, it must be cleared prior to attempting another high voltage operation. Since this bit is a status flag, it must be cleared to a 0 by writing a 1 to the register location. A write of 0 has no effect. 0 Program and Erase protection errors do not exist 1 Previous program or erase protection error encountered

Table 525. MCR field descriptions (continued)

Field	Description
15 PES	Program Erase Sequence Error PES provides information about program and erase operations with respect to sequence errors. A sequence error occurs when the program or erase sequence is not followed exactly. If an out-of-order event is detected, PES will assert, and the remainder of the sequence will not be accepted. Sequence errors are checked from the time the PGM or ERS are set (or remains set from previous operation) until EHV is set. Since this bit is a status flag, it must be cleared to a 0 by writing a 1 to the register location. A write of 0 has no effect. 0 Program or Erase phase accepted 1 Program or Erase command encountered an error
16 EER	ECC uncorrectable Event Error EER provides information on previous reads. This bit must then be cleared or a reset must occur before this bit returns to a ‘0’ state. EER may not be set to ‘1’ by the user. EER is not set for a ECC correctable Error detection and related correction event. This status flag is cleared to ‘0’ by writing ‘1’ to the register location. A Write of ‘0’ has no effect. 0 Reads are occurring normally. 1 An ECC uncorrectable Error occurred during a previous Read. In Code Flash, with 128b ECC coding, ECC error is evaluated on couple of Double word basis. In Data Flash for EEE, ECC error is evaluated on each Double word.
17 RWE	Read-while-Write event Error RWE provides information on previous Reads during a Modify operation. Read-While-Write Error means that a Read access to the Flash Array partition has occurred while the FPEC was performing a Program or Erase operation or an Array Integrity Check on the same partition. This bit must then be cleared or a reset must occur before this bit returns to a ‘0’ state. This bit may not be set to ‘1’ by the user. This status flag is cleared to ‘0’ by writing ‘1’ to the register location. A Write of ‘0’ has no effect. 0 Reads are occurring normally. 1 A RWW Error occurred during a previous Read.
18 SBC	double Bit Correction SBC provides information on previous reads. This bit must then be cleared or a reset must occur before this bit returns to a ‘0’ state. This bit may not be set to ‘1’ by the user. In the event of an ECC Triple Error detection, this bit is not set. This status flag is cleared to ‘0’ by writing ‘1’ to the register location. A Write of ‘0’ has no effect. The function of this bit is SoC dependent and it can be disabled (through UT0.SBCE). 0 Reads are occurring normally. 1 An ECC Double Error occurred and was corrected during a previous Read. In Code Flash, with 128b ECC coding, ECC correction is evaluated on couple of Double word basis. In Data Flash for EEE, ECC correction is evaluated on each Double word.

Table 525. MCR field descriptions (continued)

Field	Description
19 RE	Reset Error This bit provides information on previous resets. Checks are done within the Flash, and if a coherency issue or ECC error is detected during the reset reads used for trimming on the Flash, this status flag will assert. This bit is status only, and is not writable. 0 Flash Boot phase has occurred normally. 1 A reset error has been encountered.
20 PEAS	Program/Erase Access Space PEAS is used to indicate which space is valid for Program and Erase operations: main array space or UTEST section within Test Flash block. – PEAS is captured and held with the first Interlock Write done for Modify operations. – PEAS is retained between sampling events (that is subsequent to first interlock writes). – PEAS is changed during erase-suspended program (when program is in UTEST) and reverts back to its original state (related to erase suspended operation) once the erase-suspended program is completed (at MCR.PGM 1-0 transition). 0 UTEST address space is disabled for program and main address space enabled. 1 UTEST address space is enabled for program and main address space disabled.
21 DONE	Modify operation DONE DONE indicates if the Flash Memory module is performing a high voltage operation. DONE is set to ‘1’: – On termination of the Flash Memory module reset. – At the end of program and Erase High Voltage sequences. – After “1-0” transition of EHV, which aborts a high voltage Program/Erase operation (within t_{PABT} or t_{EABT}, equal to the P/E Abort Latency). – After “0-1” transition of ESUS/PSUS which suspends an Erase/Program operation (within t_{ESUS}/t_{PSUS} time equal to the Erase/Program Suspend Latency). DONE is cleared to ‘0’: – just after “0-1” transition of EHV, which initiates a high voltage operation, or after resuming a suspended operation. 0 Flash is executing a high voltage operation. 1 Flash is not executing a high voltage operation.

Table 525. MCR field descriptions (continued)

Field	Description
22 PEG	Program/Erase Good PEG is automatically updated to show the completion status of the last Flash Program or Erase sequence involving high voltage operations. PEG is evaluated on the completion status of only the interlocked doubleword during page program or quad-page programming where some doubleword addresses are not interlocked (already programmed or intended to remain virgin). PEG is set to ‘0’ after Aborting a Program/Erase high voltage operation, indicating sequence failure. PEG is set to ‘1’ when the Flash Memory module is reset, unless a Flash Initialization error has been detected. **Note:** PEG is valid only when PGM=1 or ERS=1, or both, and after DONE 0-1 transitions due to an Abort or the completion of a Program/Erase operation. PEG remains valid until PGM/ERS 1-0 transitions or EHV 0-1 transitions. PEG is not valid after DONE 0-1 transitions when ESUS/PSUS is set to logic ‘1’. If Program or Erase is attempted on blocks that are locked, PEG=1 indicating that the content of the block were properly protected from the Program or Erase operation. Blocks can be locked through LOCK0/1/2/3 registers OR with flh_plock0/1/2/3 sideband signals. If a Program operation tries to write ‘1’ to bits that are ‘0’, the program operation is correctly executed on the new bits to be programmed at ‘0’, but PEG is cleared indicating that the requested operation has failed. If Programming during an Erase-Suspend a location that was under suspended erase, program is not executed and PEG is set to ‘0’. With end2end ECC scheme, during an Interlock Write also ECC parity bits are input. After setting EHV, the FPEC checks the integrity of the data received in terms of ECC. If any doubleword in the doubleword program, in the page program or in the quad-page program, contains an inconsistency between data and parity, the whole operation is rejected and a PEG=0 is set. PEG behavior with blocks marked as OPP (though flh_otpen0/1/2/3 sideband signals) is the following: PEG=0 if the location to be programmed is not virgin, and thus would not be over programmed. In Array Integrity Check or Margin Read PEG is not altered when the operation is completed, regardless of the occurrence of any error, abort, break point or suspend. The presence of errors can only be detected by comparing the checksum value stored in UM0-9 and by checking the MCR flags. 0 Program, Erase operation failed or Program, Erase aborted. 1 Program or Erase operation successful.
23 PECIE	Program/Erase Complete Interrupt Enable PECIE provides a mechanism to trigger an interrupt request upon the assertion of the DONE flag. 0 An interrupt request is not generated when the DONE flag is set. 1 An interrupt request is generated when the DONE flag is set.

Table 525. MCR field descriptions (continued)

Field	Description
24 FERS	Factory Mode FERS is used for Erase and Program operations to enable a faster operation. Factory Mode must adhere to *Initial maximum (All Temperatures)* conditions, with respect to voltage conditions and Write/Erase cycling. The temperature can range from -40 °C to 150 °C. Factory Mode does not support Read and RWW operations. – FERS may be set to enable a Factory Erase or Program after setting ERS or PGM but prior to setting EHV. – FERS can be cleared only when EHV is low and DONE is high. – FERS is cleared on reset **Note:** Factory Mode may be permanently disabled by writing a location in the UTEST Nonvolatile Memory space. Offset 0x0020 in the UTEST Nonvolatile Memory space is checked at reset, and if programmed with data that contains at least one zero, a Factory Mode is not allowed, and FERS is locked. Writes attempted that violate the above cases result in FERS remaining cleared. 0 Factory Mode is disabled. 1 Factory Mode is enabled.
27 PGM	Program PGM is used to setup the Flash Memory module for a Program operation. – “0-1” transition of PGM initiates a program sequence. – “1-0” transition of PGM ends the program sequence. PGM can only be set in the following conditions: – User Mode Read (ERS is low and UT0.UTE is low); – Erase Suspend (ERS and ESUS high) with EHV low. PGM can only be cleared by the user when PSUS and EHV are low and DONE is high. PGM is cleared on reset. **Note:** In an erase-suspended program, programming Flash memory locations in blocks which were being operated on in the Erase is forbidden. Command is not accepted and PEG is set to ‘0’ 0 Flash is not executing a program sequence. 1 Flash is executing a program sequence.

Table 525. MCR field descriptions (continued)

Field	Description
28 PSUS	Program Suspend PSUS is used to indicate that the Flash Memory module is in Program Suspend or in the process of entering a Suspend state. The Flash Memory module is in Program Suspend when PSUS=1 and DONE=1. – PSUS can be set high only when PGM and EHV are high. – “0-1” transition of PSUS starts the sequence which sets DONE and places the Flash in Program Suspend. The Flash Memory module enters Suspend within t_{PSUS} of this transition. – PSUS can be cleared only when DONE and EHV are high. – “1-0” transition of PSUS with EHV=1 starts the sequence which clears DONE and returns the module to Program. The Flash Memory module cannot exit Program Suspend and clear DONE while EHV is low. In the case of Program Suspend while in Erase Suspend (PSUS=1, ESUS=1), PSUS has to be cleared first in order to resume currently higher level suspended operation that is program. PSUS is cleared on reset. 0 Program sequence is not suspended. 1 Program sequence is suspended.
29 ERS	Erase ERS is used to setup the Flash Memory module for an Erase operation. – “0-1” transition of ERS initiates an Erase sequence. – “1-0” transition of ERS ends the Erase sequence. ERS can be set only under User Mode Read (PGM is low and UT0.UTE is low). ERS can be cleared by the user only when ESUS and EHV are low and DONE is high. ERS is cleared on reset. 0 Flash is not executing an Erase sequence. 1 Flash is executing an Erase sequence.

Table 525. MCR field descriptions (continued)

Field	Description
30 ESUS	Erase Suspend ESUS is used to indicate that the Flash Memory module is in Erase Suspend or in the process of entering a Suspend state. The Flash Memory module is in Erase Suspend when ESUS = 1 and DONE=1. – ESUS can be set high only when ERS and EHV are high and PGM is low. – “0-1” transition of ESUS starts the sequence which sets DONE and places the Flash in Erase Suspend. The Flash Memory module enters Suspend within t_{ESUS} of this transition. – ESUS can be cleared only when DONE and EHV are high and PGM is low. – “1-0” transition of ESUS with EHV=1 starts the sequence which clears DONE and returns the module to Erase. The Flash Memory module cannot exit Erase Suspend and clear DONE while EHV is low. In case of Program Suspend while in Erase Suspend (PSUS=1, ESUS=1), ESUS cannot be cleared first. ESUS is cleared on reset. 0 Erase sequence is not suspended. 1 Erase sequence is suspended.

Table 525. MCR field descriptions (continued)

Field	Description
31 EHV	Enable High Voltage The EHV bit enables the Flash Memory module for a high voltage Program/Erase operation. EHV is cleared on reset. EHV must be set after an Interlock Write to start a Program/Erase sequence. EHV may be set under one of the following conditions: – Erase (ERS=1, ESUS=0, UT0.AIE=0) – Program (ERS=0, ESUS=0, PGM=1, UT0.AIE=0) – Erase-suspended program (ERS=1, ESUS=1, PGM=1, PSUS=0, UT0.AIE=0). For a program operation to initiate while an Erase is suspended, EHV must be cleared while in Erase-Suspend (after DONE transitions to 1) before setting PGM. In normal operation, a “1-0” transition of EHV with DONE high, PSUS low and ESUS low terminates the current Program/Erase high voltage operation. In an erase-suspended program operation, a “1-0” transition of EHV with DONE, ESUS, PGM and ERS high and PSUS low terminates the program that is the current high voltage operation. When an operation is aborted, there is a “1-0” transition of EHV with DONE low and the eventual Suspend bit for the current program/erase operation low. An Abort causes the value of PEG to be cleared, indicating failed Program/Erase. Address locations being operated on by the aborted operation contain indeterminate data after an Abort. EHV cannot be set to ‘1’ after an Abort request (EHV being cleared) until the Abort sequence is completed (DONE transitions to 1). Once the Abort sequence is completed (DONE=1) a new operation of the kind aborted can start provided that a new interlock is given and then EHV is set to ‘1’ again. A suspended operation cannot be aborted. Aborting a high voltage operation leaves the Flash Memory module addresses in an indeterminate data state. This may be recovered by executing an Erase on the affected blocks. EHV may be written during Suspend. EHV must be high to exit Suspend, resuming previously suspended command. EHV cannot not be written during transition into Suspend state (after Suspend bit (ESUS or PSUS) is set high) until DONE transitions to ‘1’. EHV may not be written after ESUS is set and before DONE transitions high. EHV may not be cleared after ESUS is cleared and before DONE transitions low. 0 Flash is not enabled to perform a high voltage operation. 1 Flash is enabled to perform a high voltage operation.

A number of MCR bits are protected against Write when another bit, or set of bits, are in a specific state. These Write locks are covered on a bit-by-bit basis in the preceding description, but those locks do not consider the effects of trying to write two or more bits simultaneously.

The Flash Memory module does not allow the user to write bits simultaneously (which puts the device into an illegal state). This is implemented through a priority mechanism among the bits shown in *Table 526*.

Table 526. MCR Bits Set/Clear Priority Levels

Priority level	MCR bits
1	ERS
2	PGM
3	EHV
4	ESUS, PSUS

If the user attempts to write two or more MCR bits simultaneously then only the bit with the lowest priority level is written.

37.3.2.2 Alternate Module Configuration Register (MCRA)

The Alternate Module Configuration Register is used to enable and monitor all the Modify operations of the Flash Memory module managed through Alternate interface.

Address: 0x0004 Access: User Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	SAKA	LSWA	PEPA	PESA
W														w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	PEASA	DONEA	PEGA	0	0	0	0	PGMA	PSUSA	ERSA	ESUSA	EHVA
W																
Reset	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0

Figure 411. Alternate Module Configuration Register (MCRA)

Table 527. MCRA field descriptions

Field	Description
12 SAKA	Alternate Suspend Acknowledge SAKA provides information about the Acknowledge of a suspend request at the setting of DONEa: SAKA qualifies if the suspend has been accepted (initiated by PSUSa or ESUSa setting) or if the running command has been completed, since very close to natural end. The aim of this bit is for monitor. The sequence to resume normal functionality (PSUSa or ESUSa clearing) is the same both is suspend has been acknowledged or not. SAKA is read Only 0 Operation has been completed prior that Suspend got processed. 1 Suspend Acknowledged
13 LSWA	Alternate Lock Write Error LSWA provides information about Lock write denied during the program and erase sequence. If LSWA is asserted, it must be cleared prior to attempting another high voltage operation. Since this bit is a status flag, it must be cleared to a 0 by writing a 1 to the register location. A write of 0 has no effect. 0 Lock and Sel write successful. 1 Lock and Sel write not succeeded.
14 PEPA	Alternate Program and Erase Protection Error PEPA provides information about program and erase operations with respect to protection errors. A protection error occurs if a program is attempted to a Locked block, or if an erase is selected to a locked block. This is evaluated prior to the operation beginning, and if an error is detected, high voltage operations will not be attempted for this request. If PEPA is asserted, it must be cleared prior to attempting another high voltage operation. Since this bit is a status flag, it must be cleared to a 0 by writing a 1 to the register location. A write of 0 has no effect. 0 Program and Erase protection errors do not exist. 1 Previous program or erase protection error encountered.
15 PESA	Alternate Program Erase Sequence Error PESA provides information about program and erase operations with respect to sequence errors. A sequence error occurs when the program or erase sequence is not followed exactly. If an out-of-order event is detected, PESA will assert, and the remainder of the sequence will not be accepted. Sequence errors are checked from the time the PGMA or ERSA are set (or remains set from previous operation) until EHVA is set. Since this bit is a status flag, it must be cleared to a 0 by writing a 1 to the register location. A write of 0 has no effect. 0 Program or Erase phase accepted 1 Program or Erase command encountered an error
20 PEASA	Program/Erase Access Space PEASA is used to indicate which space is valid for Program and Erase operations: main array space or UTEST block. It should be noted that erase is not allowed to the UTest NVM space, therefore If an interlock write is done to the UTest NVM space with ERSA set, PEASA will remain 0. See PEAS definition in the MCR register for more information(*Table 525*).
21 DONEA	Modify operation DONE DONEA indicates if the embedded Flash memory is performing a high-voltage operation for the Alternate program and erase interface. See DONE definition in the MCR register for more information (*Table 525*).

Table 527. MCRA field descriptions (continued)

Field	Description
22 PEGA	Program/Erase Good The PEGA bit indicates the completion status of the last Flash memory program or erase sequence for which high-voltage operations were initiated for the Alternate program and erase interface. See PEG definition in the MCR register for more information (*Table 525*). It should be noted that in the event of a error on the alternate interface PEGA will clear, but the ADR register will not be updated with the failing location. The ADR register is only valid for main interface program or erase fails, and other fails noted in the ADR register.
27 PGMA	Program PGMA is used to set up embedded Flash memory for a program operation for the Alternate program and erase interface. See PGM definition in the MCR register for more information (*Table 525*).
28 PSUSA	Program Suspend PSUSA is used to indicate the embedded Flash memory is in program suspend or in the process of entering a suspend state for the Alternate program and erase interface. See PSUS definition in the MCR register for more information (*Table 525*).
29 ERSA	Erase ERSA is used to set up embedded Flash memory for an erase operation for the Alternate program and erase interface. See ERS definition in the MCR register for more information (*Table 525*).
30 ESUSA	Erase Suspend ESUSA is used to indicate that the embedded Flash memory is in erase suspend or in the process of entering a suspend state for the Alternate program and erase interface. See ESUS definition in the MCR register for more information (*Table 525*).
31 EHVA	Enable High Voltage The EHVA bit enables the embedded Flash memory for a high-voltage program/erase operation for the Alternate program and erase interface. See EHV definition in the MCR register for more information (*Table 525*).

A number of MCR bits are protected against Write when another bit, or set of bits, are in a specific state. These Write locks are covered on a bit-by-bit basis in the preceding description, but those locks do not consider the effects of trying to write two or more bits simultaneously.

37.3.2.3 Extended Module Configuration Register (MCRE)

The Extended Module Configuration Register value is based on the size, blocks and address space configuration of the Flash Memory module.

Address: 0x0008 Access: User Read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	N128KL		N256K					N64KH			N32KH		N16KH		
W																
Reset	0	0	0	1	1	0	0	0	0	1	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	N64KM			N32KM		N16KM			N64KL			N32KL		N16KL		
W																
Reset	0	0	0	0	0	0	0	1	0	1	0	0	1	0	1	1

Figure 412. Extended Module Configuration Register (MCRE)

Table 528. MCRE field descriptions

Field	Description
1:2 N128KL	Number of 128 KB blocks in low space. 00 Zero 128 KB blocks. 01 One 128 KB blocks. 10 Two 128 KB blocks. 11 Three 128 KB blocks.
3:7 N256K	Number of 128 KB and 256 KB blocks in the 256K space The value of the N256K field is dependent upon the size of the embedded Flash memory. N256K is read only. N256K value represents in the number of blocks effectively available. 00000 Zero blocks. 00001 One block. 00010 Two blocks. 00011 Three blocks. 00100 Four blocks. 00101 Five blocks. 00110 Six blocks. 00111 Seven blocks 01000 Eight blocks. 01001 Nine blocks. 01010 Ten blocks. … 11111 thirty-one blocks.
8:10 N64KH	Number of 64 KB blocks in high space N64KH is read only. 000 Zero 64 KB blocks. 001 Two 64 KB blocks. 010 Four 64 KB blocks. 011 Six 64 KB blocks. 100 Eight 64 KB blocks. 101 Reserved. 110 Reserved. 111 Reserved.

Table 528. MCRE field descriptions (continued)

Field	Description
11:12 N32KH	Number of 32 KB blocks in high space N32K is read only. 00 Zero 32 KB blocks. 01 Two 32 KB blocks. 10 Four 32 KB blocks. 11 Reserved.
13:15 N16KH	Number of 16 KB blocks in high space N16K is read only. 000 Zero 16 KB blocks. 001 Two 16 KB blocks. 010 Four 16 KB blocks. 011 Six 16 KB blocks. 100 Eight 16 KB blocks. 101 Reserved. 110 Reserved. 111 Reserved.
16:18 N64KM	Number of 64 KB blocks in mid space N64KM is read only. 000 Zero 64 KB blocks. 001 Two 64 KB blocks. 010 Four 64 KB blocks. 011 Six 64 KB blocks. 100 Eight 64 KB blocks. 101 Reserved. 110 Reserved. 111 Reserved.
19:20 N32KM	Number of 32 KB blocks in mid space N32KM is read only. 00 Zero 32 KB blocks. 01 Two 32 KB blocks. 10 Four 32 KB blocks. 11 Reserved.
21:23 N16KM	Number of 16 KB blocks in mid space N16KM is read only. 000 Zero 16 KB blocks. 001 Two 16 KB blocks. 010 Four 16 KB blocks. 011 Six 16 KB blocks. 100 Eight 16 KB blocks. 101 Reserved. 110 Reserved. 111 Reserved.

Table 528. MCRE field descriptions (continued)

Field	Description
24:26 N64KL	Number of 64 KB blocks in low space N64KL is read only. 000 Zero 64 KB blocks. 001 Two 64 KB blocks. 010 Four 64 KB blocks. 011 Six 64 KB blocks. 100 Eight 64 KB blocks. 101 Three 64 KB blocks + One 96KB block. 110 Reserved. 111 Reserved.
27:28 N32KL	Number of 32 KB blocks in low space N32KL is read only. 00 Zero 32 KB blocks. 01 Two 32 KB blocks. 10 Four 32 KB blocks. 11 Three 32 KB blocks.
29:31 N16KL	Number of 16 KB blocks in low space N16KL is read only. 000 Zero 16 KB blocks. 001 Two 16 KB blocks. 010 Four 16 KB blocks. 011 Six 16 KB blocks. 100 Eight 16 KB blocks. 101 Three 16 KB blocks. 110 Five 16 KB blocks. 111 Reserved.

37.3.2.4 Lock 0 register (LOCK0)

The LOCK0 (Low/Mid Address Space Block Locking) register provides a means of protecting blocks from being modified. LOWLOCK and MIDLOCK bits corresponding to non existent blocks are read as ‘1’.

To determine the effective status of the Low/Mid Address Space, combine (Or) LOCK0 with sidebands:

- flh_plock0 – in order to evaluate protection against programming.
- flh_elock0 – in order to evaluate protection against erase.

The effect of programming or erasing a block protected with LOCK0 is that no modification occurs and PEG is set to ‘1’.

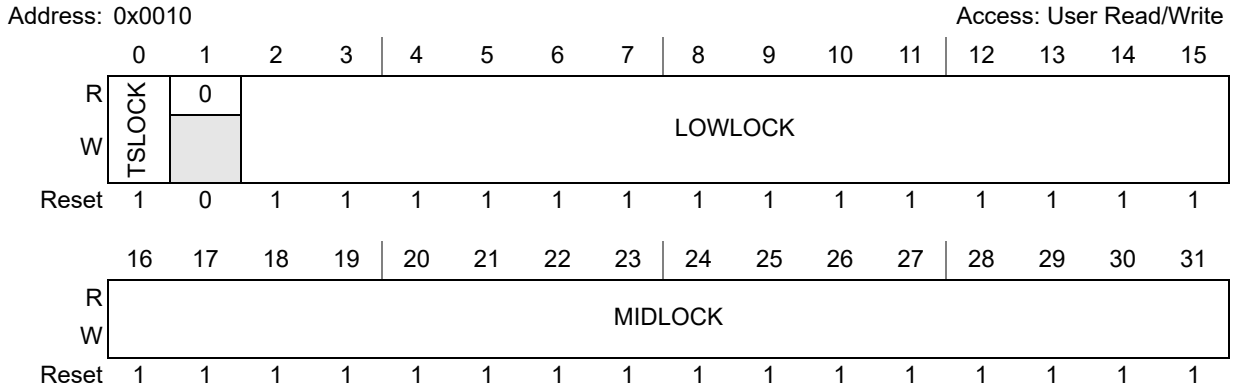


Figure 413. Lock 0 register (LOCK0)

Table 529. LOCK0 field descriptions

Field	Description
0 TSLOCK	UTEST NVM block lock This bit is used to lock the UTEST NVM from programming (UTEST NVM is not erasable). The LOCK field is not writable: – Once an Interlock Write is completed until MCR.DONE is set and MCR.EHV gets cleared. – If a high voltage operation is suspended. The default value of the TSLOCK bit is locked. The effective status of the UTEST protection is done combining (ORed) LOCK0.TSLOCK with sidebands *flh_plock0.31* in order to evaluate protection against program. 0 Test Address Space Block is unlocked and can be modified. 1 Test Address Space Block is locked and cannot be modified.

Table 529. LOCK0 field descriptions (continued)

Field	Description
2:15 LOWLOCK	Low address space block lock These bits are used to lock the existing blocks of Low Address Space from Program and Erase. The block numbering for the Low Blocks starts at LOWLOCK[0] with 16 KB block region (if available), then the 32 KB block region (if available), and lastly the 64 KB block region (if available). This continues until all blocks are accounted for. The LOCK field is not writable: – Once an Interlock Write is completed until MCR.DONE is set and MCR.EHV gets cleared. – If a high voltage operation is suspended. The default value of the LOWLOCK bits is locked. The reset state of the lock registers is affected by the alternate program and erase block select information. By default, all blocks are in the main program and erase space, and the main lock registers will have full functionality, and no blocks are in the alternative program and erase space (and will be forced to locked). Once selected for the alternate interface, the blocks selected in the alternative lock register will be available for locking or unlocking. The corresponding lock register in the main interface will be forced to locked, and are not writable. This enables blocks to be present in only one program and erase space (main or alternate). In the event that blocks are not present (due to configuration or total memory size), the LOWLOCK bits default to locked and are not writable. The reset value is always ‘1’ and register writes have no effect. 0 Low Address Space Block is unlocked and can be modified. 1 Low Address Space Block is locked and cannot be modified.
16:31 MIDLOCK	Mid address space block lock These bits are used to lock the blocks of Mid Address Space from Program and Erase. The block numbering for the Mid Blocks starts at MIDLOCK[0] with16 KB block region (if available), then the 32 KB block region (if available), and lastly the 64 KB block region (if available). This continues until all blocks are accounted for. The LOCK field is not writable: – Once an Interlock Write is completed until MCR.DONE is set and MCR.EHV gets cleared. – If a high voltage operation is suspended. The default value of the MIDLOCK bits is locked. The reset state of the lock registers is affected by the alternate program and erase block select information. By default, all blocks are in the main program and erase space, and the main lock registers will have full functionality, and no blocks are in the alternative program and erase space (and will be forced to locked). Once selected for the alternate interface, the blocks selected in the alternative lock register will be available for locking or unlocking. The corresponding lock register in the main interface will be forced to locked, and are not writable. This enables blocks to be present in only one program and erase space (main or alternate). In the event that blocks are not present (due to configuration or total memory size), the MIDLOCK bits default to locked, and are not writable. The reset value is always ‘1’ and register writes have no effect. 0 Mid Address Space Block is unlocked and can be modified. 1 Mid Address Space Block is locked and cannot be modified.

37.3.2.5 Lock 1 register (LOCK1)

The LOCK1 (High/Data Address Space Block Locking) register provides a means of protecting blocks from being modified. HIGHLOCK bits corresponding to nonexistent blocks on are read as ‘1’.

- To determine the effective status of the High Address Space, combine (Or) LOCK1 with sidebandsflh_plock1 – in order to evaluate protection against program
- flh_elock1 – in order to evaluate protection against erase.

The effect of programming/erasing a block protected with LOCK1 is that no modification occurs and PEG is set to ‘1’.

Address: 0x0014 Access: User Read/Write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	HIGHLOCK															
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Figure 414. Lock 1 register (LOCK1)

Table 530. LOCK1 field descriptions

Field	Description
16:31 HIGHLOCK	High address space block lock These bits are used to lock the blocks of High Address Space from Program and Erase. The block numbering for the High Blocks starts at HIGHLOCK[0] with16 KB block region (if available), then the 32 KB block region (if available), and lastly the 64 KB block region (if available). This continues until all blocks are accounted for. The LOCK field is not writable: – Once an Interlock Write is completed until MCR.DONE is set and MCR.EHV gets cleared. – If a high voltage operation is suspended. The default value of the HIGHLOCK bits is locked. The reset state of the lock registers is affected by the alternate program and erase block select information. By default, all blocks are in the main program and erase space, and the main lock registers will have full functionality, and no blocks are in the alternative program and erase space (and will be forced to locked). Once selected for the alternate interface, the blocks selected in the alternative lock register will be available for locking or unlocking. The corresponding lock register in the main interface will be forced to locked, and are not writable. This enables blocks to be present in only one program and erase space (main or alternate). When blocks are not present (due to configuration or total memory size), the HIGHLOCK bits default to locked, and are not writable. The reset value is always ‘1’ and register writes have no effect. 0 High Address Space Block is unlocked and can be modified. 1 High Address Space Block is locked and cannot be modified.

37.3.2.6 Lock 2 Register (LOCK2)

The LOCK2 (256K Address Space Block Locking) register provides a means of protecting blocks from being modified. The A256KLOCK bits corresponding to nonexistent blocks are read as ‘1’.

To determine the effective status of the A256KLOCK Address Space, combine (Or) LOCK2 with sidebands:

- flh_plock2 – in order to evaluate protection against program
- flh_elock2 – in order to evaluate protection against erase

The effect of programming/erasing a block protected with LOCK2 is that no modification occurs and PEG is set to ‘1’.

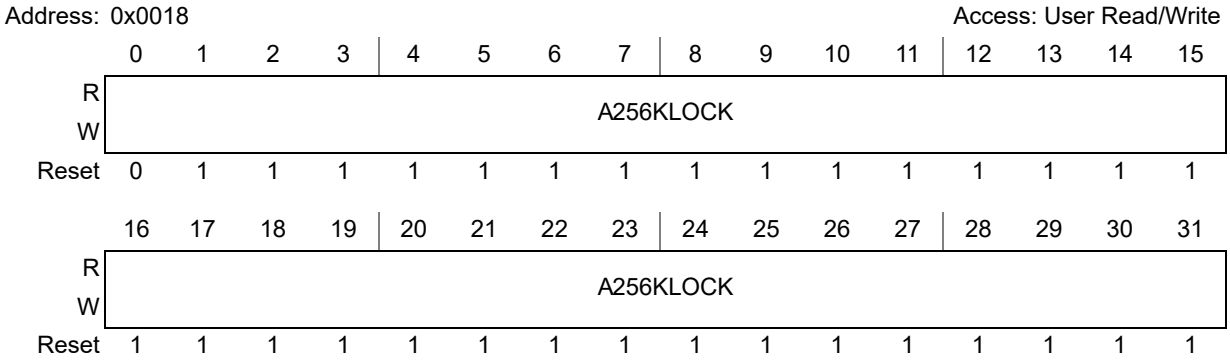


Figure 415. Lock 2 Register (LOCK2)

Table 531. LOCK2 field descriptions

Field	Description
0:31 A256KLOCK	256K address space block lock A256KLOCK[31:0] These bits are used to lock the existing blocks of Low Address Space from Program and Erase. The block numbering for the 256 KB blocks starts with A256KLOCK[0] and continues until all blocks are accounted for. The LOCK field is not writable: – Once an Interlock Write is completed until MCR.DONE is set and MCR.EHV gets cleared. – If a high voltage operation is suspended. The default value of the A256KLOCK bits is locked. When blocks are not present (due to configuration or total memory size), the A256KLOCK bits default to locked and are not writable. The reset value is always ‘1’ and register writes have no effect. 0 A256K Address Space Block is unlocked and can be modified. 1 A256K Address Space Block is locked and cannot be modified.

37.3.2.7 Lock 3 Register (LOCK3)

The LOCK3 (256K Address Space Block Locking) register provides a means of protecting blocks from being modified. The A256KLOCK bits corresponding to nonexistent blocks are read as ‘1’.

To determine the effective status of the A256KLOCK Address Space, combine (Or) LOCK3 with sidebands:

- flh_plock3 – in order to evaluate protection against program
- flh_elock3 – in order to evaluate protection against erase.

The effect of programming/erasing a block protected with LOCK3 is that no modification occurs and PEG is set to ‘1’.

Address: 0x001C Access: User Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	A256KLOCK															
Reset	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	A256KLOCK															
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Figure 416. Lock 3 Register (LOCK3)

Table 532. LOCK3 field descriptions

Field	Description
0:31 A256KLOCK	256K address space block lock A256KLOCK[63:32] These bits are used to lock the existing blocks of Low Address Space from Program and Erase. The block numbering for the 256 KB blocks starts with A256KLOCK[32] and continues until all blocks are accounted for. The LOCK field is not writable: – Once an Interlock Write is completed until MCR.DONE is set and MCR.EHV gets cleared. – If a high voltage operation is suspended. The default value of the A256KLOCK bits is locked. When blocks are not present (due to configuration or total memory size), the A256KLOCK bits default to locked and are not writable. The reset value is always ‘1’ and register writes have no effect. 0 A256K Address Space Block is unlocked and can be modified. 1 A256K Address Space Block is locked and cannot be modified.

37.3.2.8 Alternate Lock 0 register (LOCK0A)

The Alternate LOCK0A (Low/Mid Address Space Block Locking) register provides a means of protecting blocks from being modified on the alternate program and erase interface. LOWLOCKA and MIDLOCKA bits corresponding to non existent blocks are read as ‘1’.

The effective status of the Alternate Low Address Space is not affected by sidebands as in the Main Interface

The effect of programming/erasing a block protected with LOCK1A is that no modification occurs and PEGA is set to ‘1’.

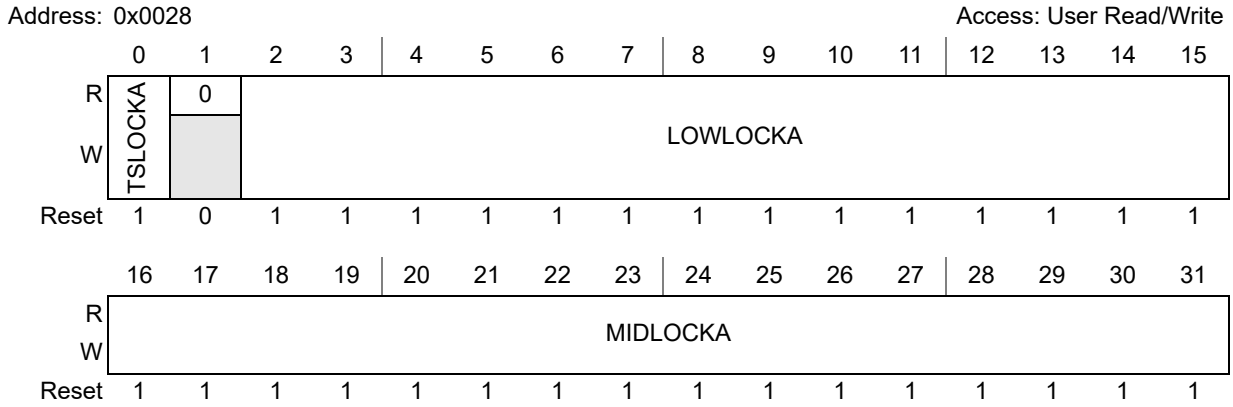


Figure 417. Alternate Lock 0 register (LOCK0A)

Table 533. LOCK0A field descriptions

Field	Description
0 TSLOCKA	Alternate UTEST NVM block lock This bit is used to lock the UTEST NVM from programming (UTEST NVM is not erasable). The LOCK field is not writable: – Once an Interlock Write is completed until MCR.DONE is set and MCR.EHV gets cleared. – If a high voltage operation is suspended. The default value of the TSLOCK bit is locked. The effective status of the UTEST protection is done combining (ORed) LOCK0A.TSLOCKA with sidebands *flh_aplock_utest* in order to evaluate protection against program. 0 The UTEST Block is unlocked and can be modified on the alternate interface. 1 The UTEST Block is locked and cannot be modified on the alternate interface.

Table 533. LOCK0A field descriptions (continued)

Field	Description
2:15 LOWLOCKA	Alternate Low address space block lock These bits are used to lock the existing blocks of Low Address Space from Program and Erase. The block numbering for the Low Blocks starts at LOWLOCKA[0] with16 KB block region (if available), then the 32 KB block region (if available), and lastly the 64 KB block region (if available). This continues until all blocks are accounted for. The LOCK field is not writable: – Once an Interlock Write is completed until DONEA is set and MCR.EHVA gets cleared. – If a high voltage operation is suspended. The default value of the LOWLOCKA bits is locked. By default, all blocks are in the main program and erase space, and the main lock registers will have full functionality, and no blocks are in the alternative program and erase space (and will be forced to locked). Once selected for the alternate interface, the blocks selected in the alternative lock register will be available for locking or unlocking. The corresponding lock register in the main interface will be forced to locked, and are not writable. This enables blocks to be present in only one program and erase space (main or alternate). In the event that blocks are not present (due to configuration or total memory size), the LOWLOCKA bits default to locked and are not writable. The reset value is always ‘1’ and register writes have no effect. 0 Alternate Low Address Space Block is unlocked and can be modified. 1 Alternate Low Address Space Block is locked and cannot be modified.
16:31 MIDLOCKA	Alternate Mid address space block lock These bits are used to lock the blocks of Mid Address Space from Program and Erase. The block numbering for the Mid Blocks starts at MIDLOCKA[0] with16 KB block region (if available), then the 32 KB block region (if available), and lastly the 64 KB block region (if available). This continues until all blocks are accounted for. The LOCK field is not writable: – Once an Interlock Write is completed until DONEA is set and MCR.EHVA gets cleared. – If a high voltage operation is suspended. The default value of the MIDLOCKA bits is locked. By default, all blocks are in the main program and erase space, and the main lock registers will have full functionality, and no blocks are in the alternative program and erase space (and will be forced to locked). Once selected for the alternate interface, the blocks selected in the alternative lock register will be available for locking or unlocking. The corresponding lock register in the main interface will be forced to locked, and are not writable. This enables blocks to be present in only one program and erase space (main or alternate). In the event that blocks are not present (due to configuration or total memory size), the MIDLOCKA bits default to locked, and are not writable. The reset value is always ‘1’ and register writes have no effect. 0 Alternate Mid Address Space Block is unlocked and can be modified. 1 Alternate Mid Address Space Block is locked and cannot be modified.

37.3.2.9 Alternate Lock 1 register (LOCK1A)

The Alternate LOCK1A (Alternate High Address Space Block Locking) register provides a means of protecting blocks from being modified on the alternate program and erase interface. HIGHLOCK bits corresponding to nonexistent blocks on are read as ‘1’.

The effect of programming/erasing a block protected with LOCK1a is that no modification occurs and PEGa is set to ‘1’.

Address: 0x002C Access: User Read/Write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	HIGHLOCKA															
W																
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Figure 418. Alternate Lock 1 register (LOCK1A)

Table 534. Alternate LOCK1A field descriptions

Field	Description
16:31 HIGHLOCKA	Alternate High address space block lock These bits are used to lock the blocks of High Address Space from Program and Erase. The block numbering for the High Blocks starts at HIGHLOCKA[0] with16 KB block region (if available), then the 32 KB block region (if available), and lastly the 64 KB block region (if available). This continues until all blocks are accounted for. The LOCK field is not writable: – Once an Interlock Write is completed until DONEA is set and MCR.EHVA gets cleared. – If a high voltage operation is suspended. The default value of the HIGHLOCKA bits is locked. By default, all blocks are in the main program and erase space, and the main lock registers will have full functionality, and no blocks are in the alternative program and erase space (and will be forced to locked). Once selected for the alternate interface, the blocks selected in the alternative lock register will be available for locking or unlocking. The corresponding lock register in the main interface will be forced to locked, and are not writable. This enables blocks to be present in only one program and erase space (main or alternate). When blocks are not present (due to configuration or total memory size), the HIGHLOCK bits default to locked, and are not writable. The reset value is always ‘1’ and register writes have no effect. 0 High Address Space Block is unlocked and can be modified. 1 High Address Space Block is locked and cannot be modified.

37.3.2.10 Select 0 register (SEL0)

The SEL0 (Low/Mid Address Space Block Select 0) register provides a means to select blocks to be operated on during Erase or Array Integrity Check – Margin Read.

Starting an Erase operation, SEL0 bits are latched and become non-alterable from the first interlock cycle that follows the rising transition of MCR.ERS (see step 4 in *Section 37.4.5.2: Erase*) until the completion of the Erase high voltage operation (1–0 transition of MCR.EHV with MCR.ERS=1 and with MCR.ESUS=0). Reading returns latched data (image of the SEL0 at the start of the Erase operation) while writing has no effect.

Address: 0x0038 Access: User Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	LOWSEL													
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	MIDSEL															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 419. Select 0 register (SEL0)

Table 535. SEL0 field descriptions

Field	Description
2:15 LOWSEL	Low address space block Select The block numbering for the Low Blocks starts at LOWSEL[0] with16 KB block region (if available), then the 32 KB block region (if available), and lastly the 64 KB block region (if available). This continues until all blocks are accounted. The blocks must be selected (or unselected) before doing an Erase Interlock Write as part of the Erase sequence. The select field is not writable: – once an Interlock Write is completed until DONE is set and MCR.EHV gets cleared. – a high voltage operation is suspended. – when UT0.AIE is high. When blocks are not present (due to configuration or total memory size) the corresponding LOWSEL bits default to unselected and are not writable. The reset value is always ‘0’, and register writes have no effect. 0 Low Address Space Block is unselected for Erase. 1 Low Address Space Block is selected for Erase.

Table 535. SEL0 field descriptions (continued)

Field	Description
16:31 MIDSEL	Mid address space block Select The block numbering for the Mid Blocks starts at MIDLSEL[0] with16 KB block region (if available), then the 32 KB block region (if available), and lastly the 64 KB block region (if available). This continues until all blocks are accounted for. The blocks must be selected (or unselected) before doing an Erase Interlock Write as part of the Erase sequence. The select field is not writable: – once an Interlock Write is completed until DONE is set and MCR.EHV gets cleared. – a high voltage operation is suspended. – when UT0.AIE is high. When blocks are not present (due to configuration or total memory size), the corresponding MIDSEL bits default to unselected and are not writable.The reset value is always ‘0’, and register writes have no effect. 0 Mid Address Space Block is unselected for Erase. 1 Mid Address Space Block is selected for Erase.

37.3.2.11 Select 1 register (SEL1)

The SEL1 (High Address Space Block Select Register 1) provides a means of selecting blocks to be operated on during Erase or Array Integrity Check – Margin Read.

Starting an Erase operation, SEL1 bits are latched and become non-alterable from the first interlock cycle that follows the rising transition of MCR.ERS (see step 4 in *Section 37.4.5.2: Erase*) until the completion of the Erase high voltage operation (1–0 transition of MCR.EHV with MCR.ERS=1 and with MCR.ESUS=0). Reading returns latched data (image of the SEL1 at the start of the Erase operation) while writing has no effect.

Address: 0x003C Access: User Read/Write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	HIGHSEL															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 420. Select 1 register (SEL1)

Table 536. SEL1 field descriptions

Field	Description
16:31 HIGHSEL	High/Data address space block Select 1 The block numbering for the High Blocks starts at HIGHSEL[0] with16 KB block region (if available), then the 32 KB block region (if available), and lastly the 64 KB block region (if available). This continues until all blocks are accounted for. The blocks must be selected (or unselected) before doing an Erase Interlock Write as part of the Erase sequence. The select field is not writable: – once an Interlock Write is completed until DONE is set and MCR.EHV gets cleared. – a high voltage operation is suspended. – when UT0.AIE is high. When blocks are not present (due to configuration or total memory size) the corresponding HIGHSEL bits default to unselected and are not writable. The reset value is always '0' and register writes have no effect. 0 High Address Space Block is unselected for Erase. 1 High Address Space Block is selected for Erase.

37.3.2.12 Select 2 register (SEL2)

The SEL2 (256K Address Space Block Select 2) register provides a means of selecting blocks to be operated on during Erase or Array Integrity Check – Margin Read.

Starting an Erase operation, SEL2 bits are latched and become non-alterable from the first interlock cycle that follows the rising transition of MCR.ERS (see step 4 in *Section 37.4.5.2: Erase*) until the completion of the Erase high voltage operation (1–0 transition of MCR.EHV with MCR.ERS=1 and with MCR.ESUS=0). Reading returns latched data (image of the SEL2 at the start of the Erase operation) while writing has no effect.

Address: 0x0040 Access: User Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	A256KSEL															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	A256KSEL															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 421. Select 2 register (SEL2)

Table 537. SEL2 field descriptions

Field	Description
0:31 A256KSEL	256K address space block select A256KSEL[31:0] The block numbering for the 256 KB blocks starts with A256KSEL[0] and continues until all blocks are accounted for. The blocks must be selected (or unselected) before doing an Erase Interlock Write as part of the Erase sequence. The select field is not writable: – once an Interlock Write is completed until DONE is set and MCR.EHV gets cleared. – a high voltage operation is suspended. – when UT0.AIE is high. When blocks are not present (due to configuration or total memory size) the corresponding A256KSEL bits default to unselected and are not writable. The reset value is always '0', and register writes have no effect. 0 A256K Address Space Block is unselected for Erase. 1 A256K Address Space Block is selected for Erase.

37.3.2.13 Select 3 register (SEL3)

The SEL3 (256K Address Space Block Select 3) register provides a means of selecting blocks to be operated on during Erase or Array Integrity Check – Margin Read.

Starting an Erase operation, SEL3 bits are latched and become non-alterable from the first interlock cycle that follows the rising transition of MCR.ERS (see step 4 in *Section 37.4.5.2: Erase*) until the completion of the Erase high voltage operation (1–0 transition of MCR.EHV with MCR.ERS=1 and with MCR.ESUS=0). Reading returns latched data (image of the SEL3 at the start of the Erase operation) while writing has no effect.

Address: 0x0044 Access: User Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	A256KSEL															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	A256KSEL															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 422. Select 3 register (SEL3)

Table 538. SEL3 field descriptions

Field	Description
0:31 A256KSEL	256K address space block select A256KSEL[63:32] The block numbering for the 256 KB blocks starts with A256KSEL[32] and continues until all blocks are accounted for. The blocks must be selected (or unselected) before doing an Erase Interlock Write as part of the Erase sequence. The select field is not writable: – once an Interlock Write is completed until DONE is set and MCR.EHV gets cleared. – a high voltage operation is suspended. – when UT0.AIE is high. When blocks are not present (due to configuration or total memory size) the corresponding A256KSEL bits default to unselected and are not writable. The reset value is always '0', and register writes have no effect. 0 A256K Address Space Block is unselected for Erase. 1 A256K Address Space Block is selected for Erase.

37.3.2.14 Module Configuration Register 2 (MCR2)

The Parallel Module Configuration Register is used to monitor all the Read operations of the Flash Memory module on Read-While-Read Partition 1.

Address: 0x0048 Access: User Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	PAEE	PEEE	PDWEE				0	0	0	PSBC1	0	0	0	0
W			w1c	w1c								w1c				
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	PEER	PRWE	PSBC	0	0	0	0	0	0	0	0	0	0	0	0	0
W	w1c	w1c	w1c													
Reset	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0

Figure 423. Module Configuration Register 2 (MCR2)

Table 539. MCR2 field descriptions

Field	Description
2 PAEE	Parallel Address Encode Error On every Read request to the Flash, the incoming address is compared to an encoded address (row, column, and block) from the memory array using the read data sense amp timing. If these two values do not match (zero selected, multiple selected, wrong selected), or the timing is incorrect, an address encode error is recorded. This status flag is cleared to ‘0’ by writing ‘1’ to the register location. A Write of ‘0’ has no effect. 0 Reads are occurring normally 1 An Address Encode Error occurred during a previous Read.
3 PEEE	Parallel EDC after ECC Error PEEE provides information on previous reads monitoring the EDC after ECC feature. On every Read request to the Flash, ECC parity is recomputed (encoding) just after ECC decoding, and if there is a mismatch between the ECC calculations (taking into account corrections or detections) a late error is reported. This status flag is cleared to ‘0’ by writing ‘1’ to the register location. A Write of ‘0’ has no effect. 0 Reads are occurring normally 1 A previous Read may have been corrupted.
4:7 PDWEE	Parallel Double Word Uncorrectable ECC Error PDWEE identifies which Double Word is involved in ECC Uncorrectable Error Event. In Code Flash 4 Double Words are read in parallel and ECC is evaluated on a couple of Double Words basis. Therefore PDWEE value could be: 1111 if an ECC Uncorrectable Error is found in both couples of Double Words 1100 if an ECC Uncorrectable Error is found in the couple of Double Words at high address 0011 if an ECC Uncorrectable Error is found in the couple of Double Words at low address 0000 if no ECC Uncorrectable Error is found PDWEE is a read only. Flags bits get cleared when EER is cleared.
11 PSBC1	Parallel Single Bit Correction 1 PSBC1 provides information on previous reads. This bit must then be cleared or a reset must occur before this bit returns to a ‘0’ state. This bit may not be set to ‘1’ by the user. In the event of an ECC Triple Error detection or ECC Double Error correction, this bit is not set. This status flag is cleared to ‘0’ by writing ‘1’ to the register location. A Write of ‘0’ has no effect. The function of this bit is SoC dependent and it can be disabled (through UT0.SBCE1). 0 Reads are occurring normally. 1 An ECC Single Error occurred and was corrected during a previous Read. In Code Flash, with 128b ECC coding, ECC correction is evaluated on couple of Double word basis. In Data Flash for EEE, ECC correction is evaluated on each Double word.

Table 539. MCR2 field descriptions (continued)

Field	Description
16 PEER	Parallel ECC Event Error PEER provides information on previous reads. This bit must then be cleared or a reset must occur before this bit returns to a ‘0’ state. PEER may not be set to ‘1’ by the user. PEER is not set for a ECC correctable Error detection and related correction event. This status flag is cleared to ‘0’ by writing ‘1’ to the register location. A Write of ‘0’ has no effect. 0 Reads are occurring normally. 1 An ECC uncorrectable Error occurred during a previous Read. In Code Flash, with 128b ECC coding, ECC error is evaluated on couple of Double word basis. In Data Flash for EEE, ECC error is evaluated on each Double word.
17 PRWE	Parallel Read-while-Write event Error PRWE provides information on previous Reads during a Modify operation. Read-While-Write Error means that a Read access to the Flash Array partition has occurred while the FPEC was performing a Program or Erase operation or an Array Integrity Check on the same partition. This bit must then be cleared or a reset must occur before this bit returns to a ‘0’ state. This bit may not be set to ‘1’ by the user. This status flag is cleared to ‘0’ by writing ‘1’ to the register location. A Write of ‘0’ has no effect. 0 Reads are occurring normally. 1 A RWW Error occurred during a previous Read.
18 PSBC	Parallel double Bit Correction PSBC provides information on previous reads. This bit must then be cleared or a reset must occur before this bit returns to a ‘0’ state. This bit may not be set to ‘1’ by the user. In the event of an ECC Triple Error detection, this bit is not set. This status flag is cleared to ‘0’ by writing ‘1’ to the register location. A Write of ‘0’ has no effect. The function of this bit is SoC dependent and it can be disabled (through UT0.SBCE). 0 Reads are occurring normally. 1 An ECC Double Error occurred and was corrected during a previous Read. In Code Flash, with 128b ECC coding, ECC correction is evaluated on couple of Double word basis. In Data Flash for EEE, ECC correction is evaluated on each Double word.

37.3.2.15 Program Address register (PAR)

The Program Address register provides the last successful address programmed. A successful program is defined as a program operation of data that is not all 1s, to an

unlocked block and the operation results in MCR.PEG returning a ‘1’. This value is held until the next successful address program event.

Address: 0x004C Access: User Read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	PAR							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	PAR													0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 424. Program Address register (PAR)

Table 540. PAR field descriptions

Field	Description
8:28 PAR	Address The Address Register provides the last successful address programmed through the Main Interface. Address given is in Flash format and has to be intended as in the Flash memory Map: offset respect to absolute address. Mapping can be found in *Section 37.3.2.16: Address register (ADR)*

37.3.2.16 Address register (ADR)

The Address register provides the first failing address in the event of module failures (ECC, RWW or FPEC). ADR captures the first failing address for the modes listed in *Table 542* according to the priority rules in *Table 543*.

In case of Array Integrity Check or Margin Read with Break Point enabled (UT0.AIBPE=1), the ADR value matches the address where the Break Point was triggered only when the current run (first start or re-start after a breakpoint (NAIBP cleared)) starts with MCR.ERR and MCR.SBC flags cleared, otherwise the ADR priority rule overrides Break Point.

Address: 0x0050 Access: User Read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	SAD	0	0	0	0	0	0	0	ADDR							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	ADDR													0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 425. Address register (ADR)

Table 541. ADR field descriptions

Field	Description
0 SAD	Test Address When this bit is high, the address indicated by ADDR[23:3] belongs to the Test Block.
8:28 ADDR	Address The Address Register provides the first failing address in the event of ECC Uncorrectable error (MCR.EER set) or the first failing address in the event of RWW error (MCR.RWE set), or the address of a failure that may have occurred in a FPEC operation (MCR.PEG cleared). The Address Register also provides the first address at which a ECC correction occurs (MCR.SBC or MCR.SBC1, or both are set) if the SoC is configured to show this feature through UT0.SBCE or UT0.SBCE1, or both. The ECC uncorrectable error detection takes the highest priority, followed by the RWW error, the FPEC error and the ECC error corrections. When accessed, ADR provides the address related to the first event occurred with the highest priority. In case of a simultaneous ECC Events Detection, bits ADDR[4] and ADDR[3] output ‘0’. In User mode the Address Register is read only. In Modify Mode ADDR can be updated on FPEC Error on the whole Flash Memory address space. In User Test Mode ADDR can be updated on Error events on the whole Flash Memory address space. In Read and RWW Modes ADDR can be updated on Error belonging to the address space of the primary Read-While-Read Partition 0. Address given is in Flash format and has to be intended as in the Flash memory Map.

Table 542. ADR update mode list

Mode	Update on error	Note
Read	ECC Triple Error Detection, ECC Double Error Correction (if UT0.SBCE=1) ECC Single Error Correction (if UT0.SBCE1=1)	—
Modify	FPEC error	—
RWW	Read-While-Write error	In RWW all Read and Modify cause of update on error may also occur
User Test	ECC Triple Error Detection, ECC Double Error Correction (if UT0.SBCE=1) ECC Single Error Correction (if UT0.SBCE1=1)	With Break Point enabled (UT0.AIBPE=1), ADR requires MCR.ERR and MCR.SBC to be cleared

Table 543. ADR content: priority list

Priority level	Error flag	ADR content
1	MCR.EER = 1	Address of first ECC Uncorrectable Error
2	MCR.RWE = 1	Address of first RWW Error

Table 543. ADR content: priority list (continued)

Priority level	Error flag	ADR content
3	MCR.PEG = 0	Address of first FPEC Error
4	MCR.SBC = 1	Address of first ECC Double Error Correction
5	MCR.SBC1 = 1	Address of first ECC Single Error Correction

Table 544. Flash memory partition 0 addresses mapping

Block	Flash format addresses	Absolute addresses
B0F0	0x0000_0000 to 0x0000_3FFF	0x00FC_4000 to 0x00FC_7FFF
B0F1	0x0000_4000 to 0x0000_7FFF	0x00FC_C000 to 0x00FC_FFFF
B0F2	0x0001_0000 to 0x0001_7FFF	0x00FD_0000 to 0x00FD_7FFF
B0F3	0x0002_0000 to 0x0002_FFFF	0x00FE_0000 to 0x00FE_FFFF
B0F4	0x0003_0000 to 0x0003_FFFF	0x00FF_0000 to 0x00FF_FFFF
B0F5	0x0088_0000 to 0x0089_FFFF	0x0100_0000 to 0x0101_FFFF
B0F6	0x0004_0000 to 0x0007_FFFF	0x0104_0000 to 0x0107_FFFF
B0F7	0x0008_0000 to 0x000B_FFFF	0x0108_0000 to 0x010B_FFFF
B0F8	0x000C_0000 to 0x000F_FFFF	0x010C_0000 to 0x010F_FFFF
B0F9	0x0010_0000 to 0x0013_FFFF	0x0110_0000 to 0x0113_FFFF
B0F10	0x0014_0000 to 0x0017_FFFF	0x0114_0000 to 0x0117_FFFF
B0F11	0x0018_0000 to 0x001B_FFFF	0x0118_0000 to 0x011B_FFFF
B0F12	0x001C_0000 to 0x001F_FFFF	0x011C_0000 to 0x011F_FFFF
—	—	—
B0UT	0x0080_0000 to 0x0080_3FFF	0x0040_0000 to 0x0040_3FFF
B0TF	0x0080_8000 to 0x0080_BFFF	0x0040_8000 to 0x0040_BFFF
—	—	—

Table 545. Flash memory partition 1 addresses mapping

Block	Flash format addresses	Absolute addresses
B1F0	0x0020_0000 to 0x0020_3FFF	0x00FC_0000 to 0x00FC_3FFF
B1F1	0x0020_4000 to 0x0020_7FFF	0x00FC_8000 to 0x00FC_BFFF
B1F2	0x0021_8000 to 0x0021_FFFF	0x00FD_8000 to 0x00FD_FFFF
B1F3	0x0022_0000 to 0x0022_FFFF	0x0061_0000 to 0x0061_FFFF
B1F4	0x0023_0000 to 0x0023_FFFF	0x0062_0000 to 0x0062_FFFF
B1F5	0x008A_0000 to 0x008B_FFFF	0x0102_0000 to 0x0103_FFFF
B1F6	0x0024_0000 to 0x0027_FFFF	0x0120_0000 to 0x0123_FFFF
B1F7	0x0028_0000 to 0x002B_FFFF	0x0124_0000 to 0x0127_FFFF

Table 545. Flash memory partition 1 addresses mapping (continued)

Block	Flash format addresses	Absolute addresses
B1F8	0x002C_0000 to 0x002F_FFFF	0x0128_0000 to 0x012B_FFFF
B1F9	0x0030_0000 to 0x0033_FFFF	0x012C_0000 to 0x012F_FFFF
B1F10	0x0034_0000 to 0x0037_FFFF	0x0130_0000 to 0x0133_FFFF
B1F11	0x0038_0000 to 0x003B_FFFF	0x0134_0000 to 0x0137_FFFF
B1F12	0x003C_0000 to 0x003F_FFFF	0x0138_0000 to 0x013B_FFFF
B1F13	—	—
B1BF	0x0080_4000 to 0x0080_7FFF	0x0040_4000 to 0x0040_7FFF
B1F15	0x0080_C000 to 0x0080_FFFF	0x0060_C000 to 0x0060_FFFF

Table 546. Flash memory partition 5 addresses mapping

Block	Flash format addresses	Absolute addresses
B2F0	0x0040_0000 to 0x0043_FFFF	0x013C_0000 to 0x013F_FFFF
B2F1	0x0044_0000 to 0x0047_FFFF	0x0140_0000 to 0x0143_FFFF
B2F2	0x0048_0000 to 0x004B_FFFF	0x0144_0000 to 0x0147_FFFF
B2F3	0x004C_0000 to 0x004F_FFFF	0x0148_0000 to 0x014B_FFFF
B2F4	0x0050_C000 to 0x0053_FFFF	0x014C_C000 to 0x014F_FFFF
B2F5	0x0054_0000 to 0x0057_FFFF	0x0150_0000 to 0x0153_FFFF
B2F6	0x0058_0000 to 0x005B_FFFF	0x0154_0000 to 0x0157_FFFF
B2F7	0x005C_0000 to 0x005F_FFFF	0x0158_0000 to 0x015B_FFFF

Table 547. Flash Memory EEPROM data partition 2 addresses mapping

Block	Flash format addresses	Absolute addresses
D0_00	0x0090_0000 to 0x0090_FFFF	0x0080_0000 to 0x0080_FFFF
D0_01	0x0091_0000 to 0x0091_FFFF	0x0081_0000 to 0x0081_FFFF
—	—	—
—	—	—

Table 548. Flash Memory EEPROM data partition 3 addresses mapping

Block	Flash format addresses	Absolute addresses
D1_00	0x0092_0000 to 0x0092_FFFF	0x0082_0000 to 0x0082_FFFF
D1_01	0x0093_0000 to 0x0093_FFFF	0x0083_0000 to 0x0083_FFFF
—	—	—
—	—	—

Table 549. Flash Memory HSM EEPROM Data Partition 4 addresses mapping

Block	Flash format addresses	Absolute addresses
H0_00	0x0098_0000 to 0x0098_3FFF	0x0068_0000 to 0x0068_3FFF
H0_01	0x0098_4000 to 0x0098_7FFF	0x0068_4000 to 0x0068_7FFF

37.3.2.17 User Test 0 register (UT0)

The User Test 0 register feature gives the user of the Flash Memory module the ability to perform test features on the Flash.

The User Test 0 register allows control of the way the Flash content check is performed. Bits MRE, MRV and AIS of the User Test 0 register are not accessible when MCR.DONE or UT0.AID are low. Reading returns indeterminate data while writing has no effect.

Address: 0x0054 Access: User Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	UTE	SBCE	0	0	0	0	0	0	SBCE1	0	0	0	0	CPR	CPA	CPE
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	VTD	0	0	NAIBP	AIBPE	0	AISUS	MRE	MRV	0	AIS	AIE	AID
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

Figure 426. User Test 0 register (UT0)

Table 550. UT0 field descriptions

Field	Description
0 UTE	User Test Enable This status bit gives indication when User Test is enabled. All bits in UT0 and UM0–UM9 are locked when this bit is ‘0’. This bit is not writable to a ‘1’, but may be cleared. The reset value is ‘0’. To set UTE, write the password 0xF9F99999 to the UT0 register. The UTE password is only accepted if PGM=0 and ERS=0 (Program and Erase are not being requested). UTE can only be cleared if AID=1, MRE=0 and AIE=0. While clearing UTE, writes to set AIE or MRE are ignored.
1 SBCE	Double Bit Correction Enable This bit, when high, enables to flag the information about any eventual ECC Double Bit Correction in the Flash array (MCR.SBC and ADR). This is true in read mode and in Array Integrity Check. SBCE is also used as an Enable for interrupt signals created by the embedded Flash memory. ECC corrections that occur when SBCE is cleared are not logged.

Table 550. UT0 field descriptions (continued)

Field	Description
8 SBCE1	Single Bit Correction Enable This bit, when high, enables to flag the information about any eventual ECC Single Bit Correction in the Flash array (MCR.SBC1 and ADR). This is true in read mode and in Array Integrity Check. SBCE1 is also used as an Enable for interrupt signals created by the embedded Flash memory. ECC corrections that occur when SBCE1 is cleared are not logged.
13 CPR	Customer Programmable Read Voltage and Reference Detection This bit is used to control the error generation for the Customer Programmable Detection area in the UTEST block when reading Read-While-Read partition 0. This bit is used to control the error generation for the Customer Programmable Detection area in the RWR1_UTEST block when reading Read-While-Read partition 1. If this bit is set and the Customer Programmable Detection location in UTEST is read, a Read Voltage and Reference Error detection is set. 0 Customer Programmable Read Voltage and Reference Detection Error is not enabled 1 Customer Programmable Read Voltage and Reference Detection Error is enabled.
14 CPA	Customer Programmable Address Encode Detection This bit is used to control the error generation for the Customer Programmable Detection area in the UTEST block when reading Read-While-Read partition 0. This bit is used to control the error generation for the Customer Programmable Detection area in the RWR1_UTEST block when reading Read-While-Read partition 1. If this bit is set, and the Customer Programmable Detection location in UTEST is read an Address Encode Error detection is set. 0 Customer Programmable Address Encode Detection Error is not enabled 1 Customer Programmable Address Encode Detection Error is enabled
15 CPE	Customer Programmable EDC after ECC Detection This bit is used to control the error generation for the Customer Programmable Detection area in the UTEST block when reading Read-While-Read partition 0. This bit is used to control the error generation for the Customer Programmable Detection area in the RWR1_UTEST block when reading Read-While-Read partition 1. If this bit is set, and the Customer Programmable Detection location in UTEST is read a EDC after ECC Error detection is set. 0 Customer Programmable EDC after ECC Detection Error is not enabled 1 Customer Programmable EDC after ECC Detection Error is enabled
19 VTD	Vth Distribution To enable Vth Distribution, VTD may be set. Effective setting of this bit is allowed when SoC has been brought into Failure Analysis Mode. When Vth Distribution is enabled, multiple run are needed to assess the Vth status of the cells belonging to selected blocks. At the end of each run NAIPB has to be checked and if equals to ‘1’ it has to be cleared to start an increased voltage step run. See NAIBP description for more information. 0 Vth Distribution is not enabled. 1 Vth Distribution is enabled.

Table 550. UT0 field descriptions (continued)

Field	Description
22 NAIBP	Next Array Integrity Break Point If AIBPE is set, NAIBP is set once a single bit correction (if enabled) or double bit detection is noted during the Array Integrity test. NAIBP is not writable to ‘1’ but may be cleared to ‘0’. Clearing NAIBP to ‘0’ enables the Array Integrity operation to be re-started after a breakpoint is encountered. If the Array Integrity operation completes without encountering another correction or detection, AID is set with NAIBP remaining ‘0’. If VTD is set, NAIBP is set once a voltage step run has been completed during the voltage sweep in Vth Distribution. NAIBP is not writable to ‘1’ but may be cleared to ‘0’. Clearing NAIBP to ‘0’ enables the Vth Distribution run yo be re-started with an increased voltage step. If the Vth Distribution completes the last voltage step run, AID is set with NAIBP remaining ‘0’. 0 Array Integrity-Vth Distribution is not currently at a break point. 1 Array Integrity-Vth Distribution is at a break point.
23 AIBPE	Array Integrity Break Point Enable To enable breakpoints during an Array Integrity test, AIBPE may be set. When breakpoints are enabled, if the run has been started with MCR.ERR (and MCR.SBC when UT0.SBCE=1) flag cleared, on the eventual occurrence of a breakpoint condition, MCR flags and ADR value may be updated. See NAIBP description for more information about Array Integrity breakpoints. 0 Array Integrity breakpoints are not enabled. 1 Array Integrity breakpoints are enabled during Array Integrity Checks.
25 AISUS	Array Integrity Suspend AISUS toggles Suspend of an Array Integrity - Margin Read Operation. AISUS can be written to ‘1’ only when AID is low AISUS can be written to ‘0’ only when AID is high. Attempting to Write AISUS and AIE on the same clock cycle results in only AIE being written. 0 Array Integrity sequence not suspended. 1 Array Integrity sequence is suspended.
26 MRE	Margin Read Enable MRE enables margin reads to be done. MRE combined with MRV enables User Margin mode reads to be done. Margin reads run on a block-by-block basis in linear address sequence selected by SEL0, SEL1 and SEL2. This bit is not accessible when MCR.DONE or UT0.AID are low or if AISUS or NAIBP are high: reading returns indeterminate data while writing has no effect. 0 Margin Read is not enabled. 1 Margin Read is enabled.
27 MRV	Margin Read Value If MRE is high, MRV selects the margin level that is being checked. Margin can be checked to an erased level (MRV=1) or to a programmed level (MRV=0). This bit is not accessible whenever MCR.DONE or UT0.AID are low or if AISUS or NAIBP are high: reading returns indeterminate data while writing has no effect. 0 Zero’s (programmed) margin reads are requested (if MRE=1). 1 One’s (erased) margin reads are requested (if MRE=1).

Table 550. UT0 field descriptions (continued)

Field	Description
29 AIS	Array Integrity Sequence AIS determines the address sequence to be used during Array Integrity checks. The default AIS=0 checks the read propagation paths along proprietary sequences followed by user code. The sequential run sequence time is significantly shorter than the proprietary sequence. AIS is not writable if AIE is high. This bit is not accessible whenever MCR.DONE or UT0.AID are low: reading returns indeterminate data while writing has no effect. 0 Array Integrity sequence is proprietary sequence. 1 Array Integrity sequence is sequential.
30 AIE	Array Integrity Enable Setting AIE to '1' starts the Array Integrity Check and the Margin Read on all selected blocks. The sequence is selected by AIS and the MISR (UM0–UM9) is checked on completion to determine if a correct signature is obtained. AIE can be set only if MCR.ERS, MCR.PGM and MCR.EHV are all low. AIE is not simultaneously writable to '1' while UTE is being cleared to '0'. Clearing AIE terminates the check when an Array Integrity operation finishes (AID=1) and aborts the check when and Array Integrity operation is ongoing (AID=0). AIE remains '1' when the Array Integrity Check - Margin read are suspended or are in breakpoint and cannot be cleared. (AIE can be cleared if NAIBP=0 and AISU=0) 0 Array Integrity Check and Margin Read and are not enabled. 1 Array Integrity Check and Margin Read are enabled.
31 AID	Array Integrity Done AID is set to indicate that the Array Integrity Check - Margin Read is complete and the MISR (UM0–9) can be checked. AID may also assert if breakpoints are enabled (AIBPE is set) and the breakpoint condition is met. AID may also assert if Suspend is set (AISUS 0–1). AID can not be written, and is status only. 0 Array Integrity Check is on-going. 1 Array Integrity Check is done.

37.3.2.18 User Multiple Input Signature registers

The Multiple Input Signature registers allow evaluation of the Array Integrity or Margin Read run on a block-by-block basis by accumulating signatures from data bit fields.

These registers are not accessible whenever MCR.DONE or UT0.AID are low. Reading returns indeterminate data while writing has no effect.

The registers can be written in order to seed signatures before starting an AIC or MR operation.

Note: Do not write to the MISR registers during an Array Integrity operation (including Suspend or Breakpoint) as it may alter the final signature in an unpredictable way.

If reads extend into an invalid address location during address sequencing (that is greater than the maximum address for a given array size), reads are still executed to the array but

the results are not deterministic: MISR registers are not recalculated and the previous value is retained.

37.3.2.18.1 User Multiple Input Signature 0 register (UM0)

UM0 represents bits 31–0 of the whole 288-bit word (4 doublewords including ECC).

Address: 0x0058 Access: User Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	MISR31	MISR30	MISR29	MISR28	MISR27	MISR26	MISR25	MISR24	MISR23	MISR22	MISR21	MISR20	MISR19	MISR18	MISR17	MISR16
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	MISR15	MISR14	MISR13	MISR12	MISR11	MISR10	MISR9	MISR8	MISR7	MISR6	MISR5	MISR4	MISR3	MISR2	MISR1	MISR0
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 427. User Multiple Input Signature 0 register (UM0)

Table 551. UM0 field descriptions

Field	Description
0:31 MISR[31:0]	Multiple input signature 31–0 MISR registers accumulate a signature from an array integrity event. The MISR captures all data fields, as well as ECC fields and the ECC error signal. Data (read and ECC) in the MISR represents corrected data (if required) captured after ECC logic is applied.

37.3.2.18.2 User Multiple Input Signature 1 register (UM1)

UM1 represents bits 63–32 of the whole 288-bit word (4 doublewords including ECC).

Address: 0x005C Access: User Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	MISR63	MISR62	MISR61	MISR60	MISR59	MISR58	MISR57	MISR56	MISR55	MISR54	MISR53	MISR52	MISR51	MISR50	MISR49	MISR48
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	MISR47	MISR46	MISR45	MISR44	MISR43	MISR42	MISR41	MISR40	MISR39	MISR38	MISR37	MISR36	MISR35	MISR34	MISR33	MISR32
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 428. User Multiple Input Signature 1 register (UM1)

Table 552. UM1 field descriptions

Field	Description
0:31 MISR[63:32]	Multiple input signature 63–32 MISR registers accumulate a signature from an array integrity event. The MISR captures all data fields, as well as ECC fields and the ECC error signal. Data (read and ECC) in the MISR represents corrected data (if required) captured after ECC logic is applied.

37.3.2.18.3 User Multiple Input Signature 2 register (UM2)

UM2 represents bits 95–64 of the whole 288-bit word (4 doublewords including ECC).

Address: 0x0060 Access: User Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	MISR95	MISR94	MISR93	MISR92	MISR91	MISR90	MISR89	MISR88	MISR87	MISR86	MISR85	MISR84	MISR83	MISR82	MISR81	MISR80
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	MISR79	MISR78	MISR77	MISR76	MISR75	MISR74	MISR73	MISR72	MISR71	MISR70	MISR69	MISR68	MISR67	MISR66	MISR65	MISR64
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 429. User Multiple Input Signature 2 register (UM2)

Table 553. UM2 field descriptions

Field	Description
0:31 MISR[95:64]	Multiple input signature 95–64 MISR registers accumulate a signature from an array integrity event. The MISR captures all data fields, as well as ECC fields and the ECC error signal. Data (read and ECC) in the MISR represents corrected data (if required) captured after ECC logic is applied.

37.3.2.18.4 User Multiple Input Signature 3 register (UM3)

UM3 represents bits 127–96 of the whole 288-bit word (4 doublewords including ECC).

Address: 0x0064 Access: User Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	MISR127	MISR126	MISR125	MISR124	MISR123	MISR122	MISR121	MISR120	MISR119	MISR118	MISR117	MISR116	MISR115	MISR114	MISR113	MISR112
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	MISR111	MISR110	MISR109	MISR108	MISR107	MISR106	MISR105	MISR104	MISR103	MISR102	MISR101	MISR100	MISR99	MISR98	MISR97	MISR96
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 430. User Multiple Input Signature 3 register (UM3)

Table 554. UM3 field descriptions

Field	Description
0:31 MISR[127:96]	Multiple input signature 127–96 MISR registers accumulate a signature from an array integrity event. The MISR captures all data fields, as well as ECC fields and the ECC error signal. Data (read and ECC) in the MISR represents corrected data (if required) captured after ECC logic is applied.

37.3.2.18.5 User Multiple Input Signature 4 register (UM4)

UM4 represents bits 159–128 of the whole 288-bit word (4 doublewords including ECC).

Address: 0x0068 Access: User Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	MISR159	MISR158	MISR157	MISR156	MISR155	MISR154	MISR153	MISR152	MISR151	MISR150	MISR149	MISR148	MISR147	MISR146	MISR145	MISR144
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	MISR143	MISR142	MISR141	MISR140	MISR139	MISR138	MISR137	MISR136	MISR135	MISR134	MISR133	MISR132	MISR131	MISR130	MISR129	MISR128
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 431. User Multiple Input Signature 4 register (UM4)

Table 555. UM4 field descriptions

Field	Description
0:31 MISR[159:128]	Multiple input signature 159–128 MISR registers accumulate a signature from an array integrity event. The MISR captures all data fields, as well as ECC fields and the ECC error signal. Data (read and ECC) in the MISR represents corrected data (if required) captured after ECC logic is applied.

37.3.2.18.6 User Multiple Input Signature 5 register (UM5)

UM5 represents bits 191–160 of the whole 288-bit word (4 doublewords including ECC).

Address: 0x006C Access: User Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	MISR191	MISR190	MISR189	MISR188	MISR187	MISR186	MISR185	MISR184	MISR183	MISR182	MISR181	MISR180	MISR179	MISR178	MISR177	MISR176
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	MISR175	MISR174	MISR173	MISR172	MISR171	MISR170	MISR169	MISR168	MISR167	MISR166	MISR165	MISR164	MISR163	MISR162	MISR161	MISR160
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 432. User Multiple Input Signature 5 register (UM5)

Table 556. UM5 field descriptions

Field	Description
0:31 MISR[191:160]	Multiple input signature 191–160 MISR registers accumulate a signature from an array integrity event. The MISR captures all data fields, as well as ECC fields and the ECC error signal. Data (read and ECC) in the MISR represents corrected data (if required) captured after ECC logic is applied.

37.3.2.18.7 User Multiple Input Signature 6 register (UM6)

UM6 represents bits 223–192 of the whole 288 bits word (4 doublewords including ECC).

Address: 0x0070 Access: User Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	MISR223	MISR222	MISR221	MISR220	MISR219	MISR218	MISR217	MISR216	MISR215	MISR214	MISR213	MISR212	MISR211	MISR210	MISR209	MISR208
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	MISR207	MISR206	MISR205	MISR204	MISR203	MISR202	MISR201	MISR200	MISR199	MISR198	MISR197	MISR196	MISR195	MISR194	MISR193	MISR192
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 433. User Multiple Input Signature 6 register (UM6)

Table 557. UM6 field descriptions

Field	Description
0:31 MISR[223:192]	Multiple input signature 223–192 MISR registers accumulate a signature from an array integrity event. The MISR captures all data fields, as well as ECC fields and the ECC error signal. Data (read and ECC) in the MISR represents corrected data (if required) captured after ECC logic is applied.

37.3.2.18.8 User Multiple Input Signature 7 register (UM7)

UM7 represents bits 255–224 of the whole 288 bits word (4 doublewords including ECC).

Address: 0x0074 Access: User Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	MISR255	MISR254	MISR253	MISR252	MISR251	MISR250	MISR249	MISR248	MISR247	MISR246	MISR245	MISR244	MISR243	MISR242	MISR241	MISR240
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	MISR239	MISR238	MISR237	MISR236	MISR235	MISR234	MISR233	MISR232	MISR231	MISR230	MISR229	MISR228	MISR227	MISR226	MISR225	MISR224
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 434. User Multiple Input Signature 7 register (UM7)

Table 558. UM7 field descriptions

Field	Description
0:31 MISR[255:224]	Multiple input signature 255–224 MISR registers accumulate a signature from an array integrity event. The MISR captures all data fields, as well as ECC fields and the ECC error signal. Data (read and ECC) in the MISR represents corrected data (if required) captured after ECC logic is applied.

37.3.2.18.9 User Multiple Input Signature 8 register (UM8)

UM8 represents the ECC parity bits of the whole 288-bits word (8 parity bits for each of the 4 doublewords).

Address: 0x0078 Access: User Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	MISR287	MISR286	MISR285	MISR284	MISR283	MISR282	MISR281	MISR280	MISR279	MISR278	MISR277	MISR276	MISR275	MISR274	MISR273	MISR272
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	MISR271	MISR270	MISR269	MISR268	MISR267	MISR266	MISR265	MISR264	MISR263	MISR262	MISR261	MISR260	MISR259	MISR258	MISR257	MISR256
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 435. User Multiple Input Signature 8 register (UM8)

Table 559. UM8 field descriptions

Field	Description
0:31 MISR[287:256]	Multiple input signature 287–256 MISR registers accumulate a signature from an array integrity event. The MISR captures all data fields, as well as ECC fields and the ECC error signal. Data (read and ECC) in the MISR represents corrected data (if required) captured after ECC logic is applied.

37.3.2.18.10 User Multiple Input Signature 9 register (UM9)

UM9 represents the ERR bits.

Address: 0x007C Access: User Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	MISR319	MISR318	MISR317	MISR316	MISR315	MISR314	MISR313	MISR312	MISR311	MISR310	MISR309	MISR308	MISR307	MISR306	MISR305	MISR304
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	MISR303	MISR302	MISR301	MISR300	MISR299	MISR298	MISR297	MISR296	MISR295	MISR294	MISR293	MISR292	MISR291	MISR290	MISR289	MISR288
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 436. User Multiple Input Signature 9 register (UM9)

Table 560. UM9 field descriptions

Field	Description
0:31 MISR[319:288]	Multiple input signature 319–288 MISR registers accumulate a signature from an array integrity event. The MISR captures all data fields, as well as ECC fields and the ECC error signal. Data (read and ECC) in the MISR represents corrected data (if required) captured after ECC logic is applied.

37.3.2.19 Over-program protection registers

Over-program protection prevents blocks from being over-programmed. These registers show the over-program protection status. See *Section 37.4.5.1.2: Over-programming protection (OPP) enable*.

37.3.2.19.1 Over-program protection 0 register (OPP0)

The Low/Mid Address Space is covered by the Over Program Protection 0 (OPP0) register.

Address: 0x0080 Access: User Read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	LOWOPP													
W																
Reset(1)	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	MIDOPP															
W																
Reset(1)	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—

1. A reset value of "—" indicates that the value is set at the factory and can vary among different applications

Figure 437. Over-program protection 0 register (OPP0)

Table 561. OPP0 field descriptions

Field	Description
2:15 LOWOPP	Low address space Over Program Protection The block numbering for the Low Blocks starts at LOWOPP[0] with 16 KB block region (if available), then the 32 KB block region (if available), and lastly the 64 KB block region (if available). This continues until all blocks are accounted for. The LOWOPP field is not writable, and is status only. The default value of the LOWOPP bits is not over-programming protected. When blocks are not present (due to configuration or total memory size), the LOWOPP bits default to not over-programming protected and will remain 0. 0 Low Address Space Block is unprotected from over-programming. 1 Low Address Space Block is protected from over-programming.
16:31 MIDOPP	Mid address space Over Program Protection The block numbering for the Mid Blocks starts at MIDOPP[0] with 16 KB block region (if available), then the 32 KB block region (if available), and lastly the 64 KB block region (if available). This continues until all blocks are accounted for. The MIDOPP field is not writable, and is status only. The default value of the MIDOPP bits is not over-programming protected. When blocks are not present (due to configuration or total memory size), the MIDOPP bits default to not over-programming protected, and will remain 0. 0 Mid Address Space Block is unprotected from over-programming. 1 Mid Address Space Block is protected from over-programming.

37.3.2.19.2 Over-program protection 1 register (OPP1)

The High Address Space is covered by the Over Program Protection 1 (OPP1) register.

Address: 0x0084 Access: User Read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset(1)	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	HIGHOPP															
W																
Reset(1)	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—

1. A reset value of "—" indicates that the value is set at the factory and can vary among different applications

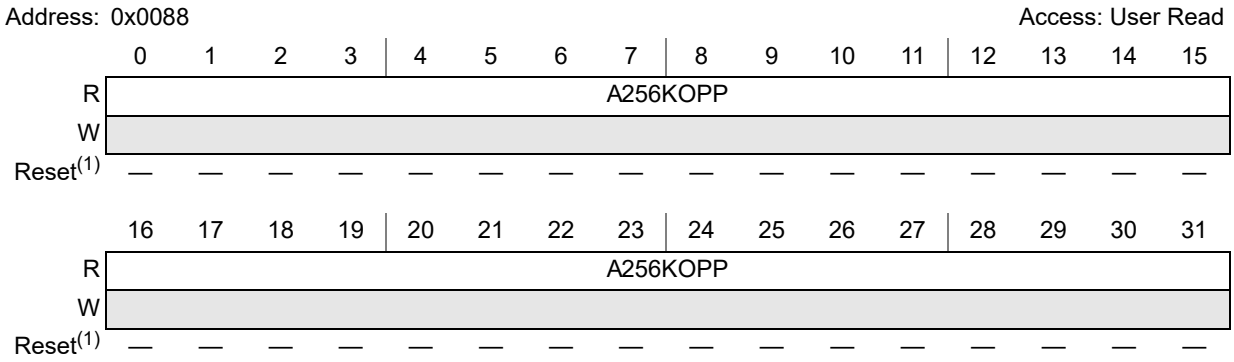
Figure 438. Over-program protection 1 register (OPP1)

Table 562. OPP1 field descriptions

Field	Description
16:31 HIGHOPP	High address space Over Program Protection The block numbering for the High Blocks starts at HIGHOPP[0] with 16 KB block region (if available), then the 32 KB block region (if available), and lastly the 64 KB block region (if available). This continues until all blocks are accounted for. The HIGHOPP field is not writable, and is status only. The default value of the HIGHOPP bits is not over-programming protected. When blocks are not present (due to configuration or total memory size), the HIGHOPP bits default to not over-programming protected and will remain 0. 0 High Address Space Block is unprotected from over-programming. 1 High Address Space Block is protected from over-programming.

37.3.2.19.3 Over-program protection 2 register (OPP2)

The A256K Address Space Block is covered by the Over Program Protection 2 (OPP2) register.



1. A reset value of "—" indicates that the value is set at the factory and can vary among different applications

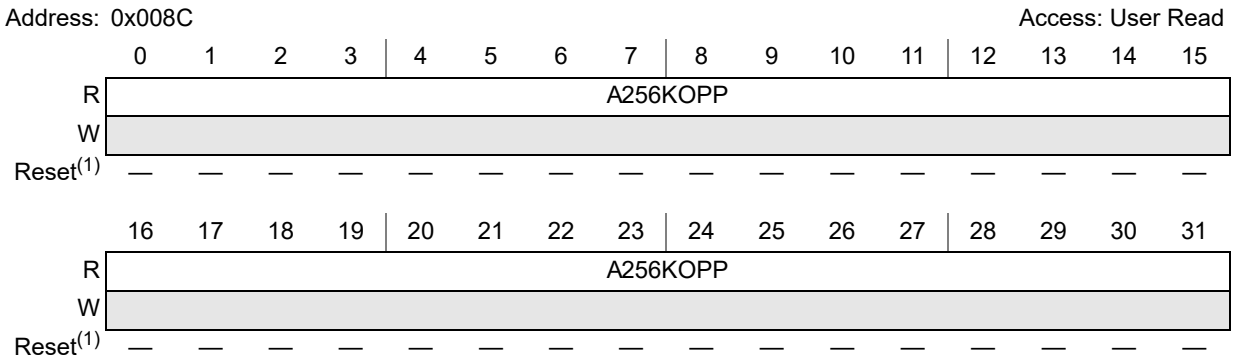
Figure 439. Over-program protection 2 register (OPP2)

Table 563. OPP2 field descriptions

Field	Description
0:31 A256KOPP	256K Address Space Block Over Program Protection A256KOPP[31:0] The block numbering for the 256 KB Blocks starts at A256KOPP[0] and continues until all blocks are accounted for. The A256KOPP field is not writable, and is status only. The default value of the A256KOPP bits is not over-programming protected. When blocks are not present (due to configuration or total memory size), the A256KOPP bits default to not over-programming protected and will remain 0. 0 A256K Address Space Block is unprotected from over-programming. 1 A256K Address Space Block is protected from over-programming.

37.3.2.19.4 Over-program protection 3 register (OPP3)

The 256K high Address Space is covered by the Over Program Protection 3 (OPP3) register.



1. A reset value of "—" indicates that the value is set at the factory and can vary among different applications

Figure 440. Over-program protection 3 register (OPP3)

Table 564. OPP3 field descriptions

Field	Description
0:31 A256KOPP	256K Address space Over Program Protection A256KOPP[63:32] The block numbering for the 256 KB Blocks starts at A256KOPP[32] and continues until all blocks are accounted for. The A256KOPP field is not writable, and is status only. The default value of the A256KOPP bits is not over-programming protected. When blocks are not present (due to configuration or total memory size), the A256KOPP bits default to not over-programming protected and will remain 0. 0 A256K Address Space Block is unprotected from over-programming. 1 A256K Address Space Block is protected from over-programming.

37.3.2.20 Test Mode Disable Password Check register (TMD)

The Test Mode Disable Password Check (TMD) register provides a means of entering the password to be matched with the one stored in UTEST location in order to allow manufacturer access to protected Test Mode blocks by writing 0x5A4B_3C2D in the Test Mode Disable Seal location.

Address: 0x0090 Access: User Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0															
W	PWD															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0															
W	PWD															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 441. Test Mode Disable Password Check register (TMD)

Table 565. TMD field descriptions

Field	Description
0:31 PWD	The Test Mode Disable Check register is used as a location to provide a password challenge to unlock blocks selected to be sealed as part of the Test Mode Disable seal mechanism. Please see *Section 37.4.8.2: Test Mode Disable* for more information. Writes to the register have no effect except for a password challenge. Reads to this register will always return 0. The register's reset value is 0. **Note:** PWD bit field is byte-reversed compared to UTEST TMD password: for instance, if UTEST TMD password is written 0x01020304, the PWD bit field must be written 0x04030201

37.3.2.21 HSM Alternate selector registers

HSM Alternate selectors assign blocks from default to ALTERNATE Program & Erase interface. These registers show the HSM Alternate assignment status. See *Section 37.4.6: Alternate program and erase interface*.

37.3.2.21.1 HSM ALT-SEL register 0 (ALTSEL0)

The Low/Mid Address Space is assigned between MAIN and ALT Interface. Through the HSM ALT-SEL register it is possible to read the actual status of the assignment.

Address: 0x0094 Access: User Read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	LOWALTSEL													
W																
Reset(1)	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	MIDALTSEL															
W																
Reset(1)	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—

1. A reset value of "—" indicates that the value is set at the factory and can vary among different applications

Figure 442. HSM ALT-SEL register 0 (ALTSEL0)

Table 566. ALTSEL0 field descriptions

Field	Description
2:15 LOWALTSEL	Low address space HSM Alternate Select The block numbering for the Low Blocks starts at LOWALTSEL[0] with 16 KB block region (if available), then the 32 KB block region (if available), and lastly the 64 KB block region (if available). This continues until all blocks are accounted for. The LOWALTSEL field is not writable, and is status only. When blocks are not present (due to configuration or total memory size), the LOWALTSEL bits will remain 0. 0 Low Address Space Block is assigned to Main interface. 1 Low Address Space Block is assigned to Alternate interface.
16:31 MIDALTSEL	Mid address space HSM Alternate Select 0 The block numbering for the Mid Blocks starts at MIDALTSEL[0] with 16 KB block region (if available), then the 32 KB block region (if available), and lastly the 64 KB block region (if available). This continues until all blocks are accounted for. The MIDALTSEL field is not writable, and is status only. When blocks are not present (due to configuration or total memory size), the MIDALTSEL bits will remain 0. 0 Mid Address Space Block is assigned to Main interface. 1 Mid Address Space Block is assigned to Alternate interface.

37.3.2.21.2 HSM ALT-SEL register 1 (ALTSEL1)

The High Address Space is assigned between MAIN and ALT Interface. Through the HSM ALT-SEL register it is possible to read the actual status of the assignment.

Address: 0x0098 Access: User Read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset(1)	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	HIGHALTSEL															
W																
Reset(1)	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—

1. A reset value of "—" indicates that the value is set at the factory and can vary among different applications

Figure 443. HSM ALT-SEL register 1 (ALTSEL1)

Table 567. ALTSEL1 field descriptions

Field	Description
16:31 HIGHALTSEL	High address space HSM Alternate Select The block numbering for the High Blocks starts at HIGHALTSEL[0] with 16 KB block region (if available), then the 32 KB block region (if available), and lastly the 64 KB block region (if available). This continues until all blocks are accounted for. The HIGHALTSEL field is not writable, and is status only. When blocks are not present (due to configuration or total memory size), the HIGHALTSEL bits will remain 0. 0 High Address Space Block is assigned to Main interface. 1 High Address Space Block is assigned to Alternate interface.

37.3.2.22 Erase Lock protection registers

Erase Lock protection prevents blocks from being erase. These registers show the Erase Lock protection status. See *Section 37.4.8.1: Modify Protection*.

37.3.2.22.1 Erase Lock Protection 0 register (ELOCK0)

The Low/Mid Address Space is covered by the Erase Lock Protection 0 (ELOCK0) register.

Address: 0x0100 Access: User Read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	LOWELOCK															
W																
Reset(1)	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	MIDELOCK															
W																
Reset(1)	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—

1. A reset value of "—" indicates that the value is set at the factory and can vary among different applications

Figure 444. Erase Lock Protection 0 register (ELOCK0)

Table 568. ELOCK0 field descriptions

Field	Description
0:15 LOWELOCK	Low address space Erase Lock Protection The block numbering for the Low Blocks starts at LOWELOCK[0] with 16 KB block region (if available), then the 32 KB block region (if available), and lastly the 64 KB block region (if available). This continues until all blocks are accounted for. The LOWELOCK field is not writable, and is status only. When blocks are not present (due to configuration or total memory size), the LOWELOCK bits default is Erase Lock protected and will remain 1. 0 Low Address Space Block is Erase unprotected. 1 Low Address Space Block is Erase protected.
16:31 MIDELOCK	Mid address space Erase Lock Protection The block numbering for the Mid Blocks starts at MIDELOCK[0] with 16 KB block region (if available), then the 32 KB block region (if available), and lastly the 64 KB block region (if available). This continues until all blocks are accounted for. The MIDELOCK field is not writable, and is status only. When blocks are not present (due to configuration or total memory size), the MIDELOCK bits default is Erase Lock protected and will remain 1. 0 Mid Address Space Block is Erase unprotected. 1 Mid Address Space Block is Erase protected.

37.3.2.22.2 Erase Lock Protection 1 register (ELOCK1)

The High Address Space is covered by the Erase Lock Protection 1 (ELOCK1) register.

Address: 0x0104 Access: User Read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset(1)	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	HIGHELOCK															
W																
Reset(1)	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—

1. A reset value of "—" indicates that the value is set at the factory and can vary among different applications

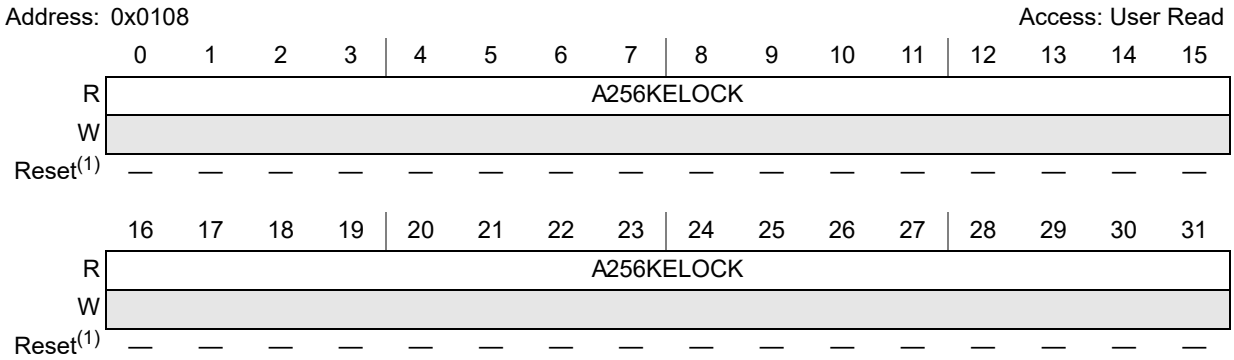
Figure 445. Erase Lock Protection 1 register (ELOCK1)

Table 569. ELOCK1 field descriptions

Field	Description
16:31 HIGHELOCK	High address space Erase Lock Protection The block numbering for the High Blocks starts at HIGHELOCK[0] with 16 KB block region (if available), then the 32 KB block region (if available), and lastly the 64 KB block region (if available). This continues until all blocks are accounted for. The HIGHELOCK field is not writable, and is status only. When blocks are not present (due to configuration or total memory size), the HIGHELOCK bits default is Erase Lock protected and will remain 1. 0 High Address Space Block is Erase unprotected. 1 High Address Space Block is Erase protected.

37.3.2.22.3 Erase Lock Protection 2 register (ELOCK2)

The 256K Address Space is covered by the Erase Lock Protection 2 (ELOCK2) register.



1. A reset value of "—" indicates that the value is set at the factory and can vary among different applications

Figure 446. Erase Lock Protection 2 register (ELOCK2)

Table 570. ELOCK2 field descriptions

Field	Description
0:31 A256KELOCK	256K High address space Erase Lock Protection A256KELOCK[31:0] The block numbering for the 256 KB Blocks starts at A256KELOCK[0] and continues until all blocks are accounted for. The A256KELOCK field is not writable, and is status only. When blocks are not present (due to configuration or total memory size), the A256KELOCK bits default is Erase Lock protected and will remain 1. 0 A256K Address Space Block is Erase unprotected. 1 A256K Address Space Block is Erase protected.

37.3.2.22.4 Erase Lock protection 3 register (ELOCK3)

The 256K Address Space is covered by the Erase Lock Protection 3 (ELOCK3) register.

Address: 0x010C Access: User Read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	A256KELOCK															
W																
Reset(1)	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	A256KELOCK															
W																
Reset(1)	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—

1. A reset value of "—" indicates that the value is set at the factory and can vary among different applications

Figure 447. Erase Lock protection 3 register (ELOCK3)

Table 571. ELOCK3 field descriptions

Field	Description
0:31 A256KELOCK	256K High address space Erase Lock Protection A256KELOCK[63:32] The block numbering for the 256 KB Blocks starts at A256KELOCK[32] and continues until all blocks are accounted for. The A256KELOCK field is not writable, and is status only. When blocks are not present (due to configuration or total memory size), the A256KELOCK bits default is Erase Lock protected and will remain 1. 0 A256K Address Space Block is erase unprotected. 1 A256K Address Space Block is Erase protected.

37.3.2.23 Program Lock protection registers

Program Lock protection prevents blocks from being programmed. These registers show the Program Lock protection status. See *Section 37.4.8.1: Modify Protection*.

37.3.2.23.1 Program Lock Protection 0 register (PLOCK0)

The Low/Mid Address Space is covered by the Program Lock Protection 0 (PLOCK0) register.

Address: 0x0110 Access: User Read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	LOWPLOCK															
W																
Reset(1)	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	MIDPLOCK															
W																
Reset(1)	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—

1. A reset value of "—" indicates that the value is set at the factory and can vary among different applications

Figure 448. Program Lock Protection 0 register (PLOCK0)

Table 572. PLOCK0 field descriptions

Field	Description
0:15 LOWPLOCK	Low address space Erase Lock Protection The block numbering for the Low Blocks starts at LOWPLOCK[0] with 16 KB block region (if available), then the 32 KB block region (if available), and lastly the 64 KB block region (if available). This continues until all blocks are accounted for. The LOWPLOCK field is not writable, and is status only. When blocks are not present (due to configuration or total memory size), the LOWPLOCK bits default is Program Lock protected and will remain 1. 0 Low Address Space Block is Program unprotected. 1 Low Address Space Block is Program protected.
16:31 MIDPLOCK	Mid address space Erase Lock Protection The block numbering for the Mid Blocks starts at MIDPLOCK[0] with 16 KB block region (if available), then the 32 KB block region (if available), and lastly the 64 KB block region (if available). This continues until all blocks are accounted for. The MIDPLOCK field is not writable, and is status only. When blocks are not present (due to configuration or total memory size), the MIDPLOCK bits default is Program Lock protected and will remain 1. 0 Mid Address Space Block is Program unprotected. 1 Mid Address Space Block is Program protected.

37.3.2.23.2 Program Lock Protection 1 register (PLOCK1)

The High Address Space is covered by the Program Lock Protection 1 (PLOCK1) register.

Address: 0x0114 Access: User Read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset(1)	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	HIGHPLOCK															
W																
Reset(1)	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—

1. A reset value of "—" indicates that the value is set at the factory and can vary among different applications

Figure 449. Program Lock Protection 1 register (PLOCK1)

Table 573. PLOCK1 field descriptions

Field	Description
16:31 HIGHPLOCK	High address space Program Lock Protection The block numbering for the High Blocks starts at HIGHPLOCK[0] with 16 KB block region (if available), then the 32 KB block region (if available), and lastly the 64 KB block region (if available). This continues until all blocks are accounted for. The HIGHPLOCK field is not writable, and is status only. When blocks are not present (due to configuration or total memory size), the HIGHPLOCK bits default is Program Lock protected and will remain 1. 0 High Address Space Block is Program unprotected. 1 High Address Space Block is Program protected.

37.3.2.23.3 Program Lock Protection 2 register (PLOCK2)

The 256K Address Space is covered by the Program Lock Protection 2 (PLOCK2) register.

Address: 0x0118 Access: User Read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	A256KPLOCK															
W																
Reset(1)	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	A256KPLOCK															
W																
Reset(1)	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—

1. A reset value of "—" indicates that the value is set at the factory and can vary among different applications

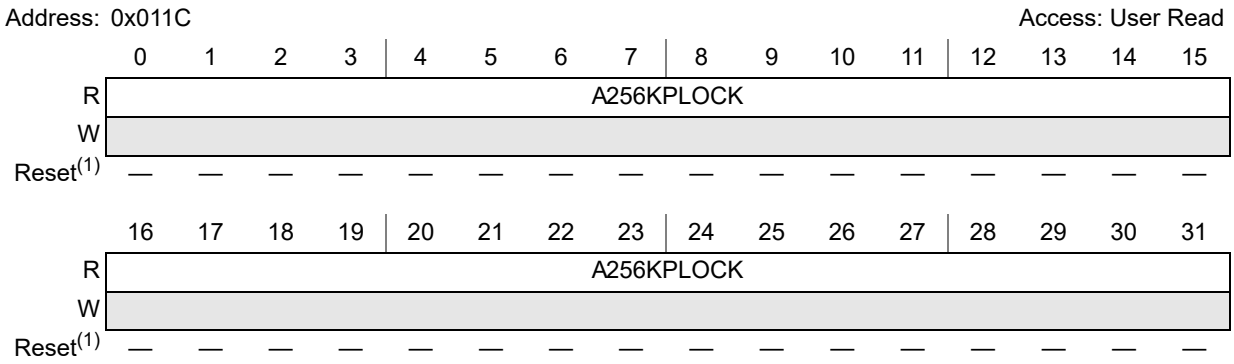
Figure 450. Program Lock Protection 2 register (PLOCK2)

Table 574. PLOCK2 field descriptions

Field	Description
0:31 A256KPLOCK	256K High address space Program Lock Protection A256KPLOCK[31:0] The block numbering for the 256 KB Blocks starts at A256KPLOCK[0] and continues until all blocks are accounted for. The A256KPLOCK field is not writable, and is status only. When blocks are not present (due to configuration or total memory size), the A256KPLOCK bits default is Program Lock protected and will remain 1. 0 A256K Address Space Block is Program unprotected. 1 A256K Address Space Block is Program protected.

37.3.2.23.4 Program Lock protection 3 register (PLOCK3)

The 256K Address Space is covered by the Program Lock Protection 3 (PLOCK3) register.



1. A reset value of "—" indicates that the value is set at the factory and can vary among different applications

Figure 451. Program Lock protection 3 register (PLOCK3)

Table 575. PLOCK3 field descriptions

Field	Description
0:31 A256KPLOCK	256K High address space Program Lock Protection A256KPLOCK[63:32] The block numbering for the 256 KB Blocks starts at A256KPLOCK[32] and continues until all blocks are accounted for. The A256KPLOCK field is not writable, and is status only. When blocks are not present (due to configuration or total memory size), the A256KPLOCK bits default is Program Lock protected and will remain 1. 0 A256K Address Space Block is Program unprotected. 1 A256K Address Space Block is Program protected.

37.3.2.24 Parallel Address register (ADR2)

The Parallel Address register provides the first failing address in the event of module failures (ECC) or the first address at which ECC single error correction occurs, when Reads are performed from Read While Read 1 interface. ADR2 captures the first failing address for the modes listed in *Table 542* according to the priority rules in *Table 543*.

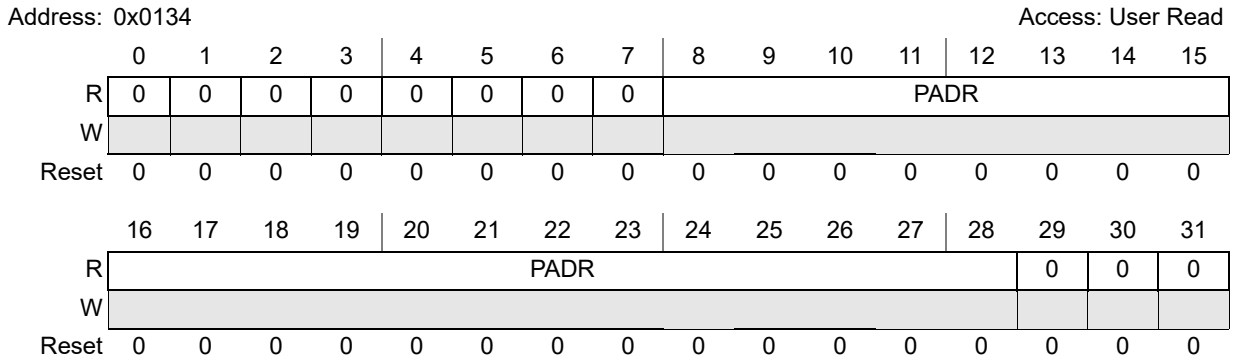


Figure 452. Parallel Address register (ADR2)

Table 576. ADR2 field descriptions

Field	Description
8:28 PADR	Parallel Address The Address Register provides the first failing address in the event of ECC Uncorrectable error (MCR2.PEER set) or the first failing address in the event of RWW error (MCR2.PRWE set). The Address Register also provides the first address at which a ECC correction occurs (MCR2.PSBC or MCR2.PSBC1, or both are set) if the SoC is configured to show this feature through UT0.SBCE or UT0.SBCE1, or both. The ECC uncorrectable error detection takes the highest priority, followed by the RWW error, and the ECC error corrections. When accessed, PADR provides the address related to the first event occurred with the highest priority. In case of a simultaneous ECC Events Detection, bits PADR[4] and PADR[3] output '0'. In Read and RWW Modes PADR can be updated on Error belonging to the address space of the primary Read-While-Read Partition 1. Address given is in Flash format and has to be intended as in the Flash memory Map.

Table 577. ADR2 update mode list

Mode	Update on error	Note
Read	ECC Triple Error Detection, ECC Double Error Correction (if UT0.SBCE=1) ECC Single Error Correction (if UT0.SBCE1=1)	—
RWW	Read-While-Write error	In RWW all Read and Modify cause of update on error may also occur
User Test	ECC Triple Error Detection, ECC Double Error Correction (if UT0.SBCE=1) ECC Single Error Correction (if UT0.SBCE1=1)	With Break Point enabled (UT0.AIBPE=1), ADR requires MCR.ERR and MCR.SBC to be cleared

Table 578. ADR2 content: priority list

Priority level	Error flag	ADR2 content
1	MCR2.PEER = 1	Address of first ECC Uncorrectable Error
2	MCR2.PRWE = 1	Address of first RWW Error
3	MCR2.PSBC = 1	Address of first ECC Double Error Correction
4	MCR2.PSBC1 = 1	Address of first ECC Single Error Correction

For the address mapping, refer to *Section 37.3.2.16: Address register (ADR)* description.

37.4 Functional description

This chapter describes the details of all the modes of operation available to the Flash Memory module.

37.4.1 Reset

A reset is the highest priority operation for the Flash Memory module and terminates all other operations.

The Flash Memory module uses reset to initialize register and status bits to their default reset values. If the Flash Memory module is executing a Program or Erase operation (MCR.PGM=1 or MCR.ERS=1) and a reset is issued, the operation is immediately terminated and the module disables the high voltage circuits.Reset terminates all operations and forces the Flash Memory module into Read mode ready to receive Read/Write accesses.

Warning: Reset and Power-Off must not be used as a systematic way to terminate a Program or Erase operation.

After reset is negated Read register access may be performed, although registers that require updating from shadow information or other inputs may not read updated values until MCR.DONE transforms. MCR.DONE may be polled to determine if the Flash Memory module has transitioned out of reset. Note that the registers cannot be written until MCR.DONE is high.

37.4.2 Power-down mode (Disable mode)

The Power-down (or Disable) mode allows the turning-off of all Flash DC current sources. All power dissipation in this mode is due solely to leakage.

In Power-down mode no Read from or Write to the Flash array is possible. Every Flash array Read access executed when the Power-down mode is active outputs invalid data,.

The user may not read some registers (UM0–UM9 and part of UT0) until the Power-down mode is exited.On the contrary, Write access is locked on all the registers in Power-Down mode.

When enabled the Flash Memory module returns to its predisable state in all cases unless in the process of executing an Erase or a Program High Voltage operation at the time of disable. If the Flash Memory module is disabled during an Erase operation, the operation is suspended and MCR.ESUS bits are set to ‘1’. The user may resume the Erase operation at the time the module is enabled by clearing MCR.ESUS bit. MCR.EHV must be high to resume the Erase operation.

If the Flash Memory module is disabled during a program operation, the operation is suspended and MCR.PSUS bit is set to ‘1’. The user may resume the program operation at the time the module is enabled by clearing MCR.PSUS bit. MCR.EHV must be high to resume the Erase operation.

If the Flash Memory module is disabled during an erase-suspended program operation, the program operation is suspended and both ESUS and PSUS are set to a ‘1’. When the module is enabled PSUS must be cleared first to resume the program. Erase may be resumed after the program ends. EHV must be high to resume both operations.

If the Flash Memory module is disabled during AIC or MR, the operation is aborted and the MISR value is not deterministic.

If the Flash Memory module is put in Power-down mode and the Vector Table remains mapped in the Flash Address space, the user must be mindful that the Flash Memory module significantly increases the interrupt response time by adding several Wait States.

Note: Do not enter Low-power mode when the Power-down mode is active.

37.4.3 Low-power mode (Sleep mode)

The Low-power (or Sleep) mode turns-off most of the DC current sources within the Flash Memory module. Wake-up time from Low-power mode is faster than wake-up time from Power-down mode.

In Low-power mode no Read from or Write to the Flash array is possible. Every Read access executed when the Low-power mode is active outputs invalid data.

The user may not read some registers (UM0–UM9 and part of UT0) until the Low-power mode is exited. Write access is locked on all the registers in Low-power mode.

When exiting from Low-power mode the Flash Memory module returns to its presleep state in all cases unless in the process of executing an Erase High Voltage operation at the time of entering sleep mode. If the Flash Memory module is put in sleep mode during an Erase operation, MCR.ESUS is set to ‘1’. The user may resume the Erase operation at the time the module exits from sleep by clearing MCR.ESUS. MCR.EHV must be high to resume the Erase operation.

If the Flash Memory module is put in Low-power mode during a program operation, the operation is suspended and MCR.PSUS is set to ‘1’. The user may resume the program operation at the time the module is enabled by clearing MCR.PSUS. MCR.EHV must be high to resume the Erase operation.

If the Flash Memory module is put in Low-power mode during an erase-suspended program operation, the program operation is suspended and both ESUS and PSUS are set to a ‘1’. When the module is enabled, PSUS must first be cleared for the program to resume. The Erase may be resumed after the program ends. EHV must be high to resume both operations.

If the Flash Memory module is disabled during AIC or MR, the operation is aborted and the MISR value is not deterministic.

Note: Do not enter Power-down mode when the Low-power mode is active.

37.4.4 Read mode

Read mode is the active mode of operation of the Flash Memory module in which the Flash array can be read or ‘written’ (the so-called Interlock Write), or the Flash registers can be read or written.

The Flash Memory module enters Read mode when exiting from reset. The default state of the Flash Memory module is Read mode.

The main spaces (Low, Mid, High, and 256K) and UTEST address space of Flash array can only be read in Read mode.

Flash registers can even be read when the module is in Power-down or Low-power mode (except a few documented registers) as well as in Read mode. However, Flash registers can only be written while in Read mode.

Flash memory module is in Read mode, granting the same access time, under the following conditions:

- The Read state is active when the module is enabled (User Mode Read)
- The Read state is active when MCR.ERS and MCR.ESUS are high and MCR.PGM is low (Erase Suspend)
- The Read state is active when MCR.PGM and MCR.PSUS are high (Program Suspend)
- The Read state is active when MCR.PGM or MCR.ERS are high and High Voltage operation is ongoing on one partition, thus MCR.EHV is high (condition referred as Read-While-Write). In this case reads done on the other partitions are allowed, while reads done to the partition being operated on (either Erase or Program) result in an error with MCR.RWE set and the data read back possibly being random including the possibility of ECC errors.
- The Read state is active when MCRA.ERSA and MCRA.ESUSA are high and MCRA.PGMA is low (Alternate Erase Suspend)
- The Read state is active when MCRA.PGMA and MCRA.PSUSA are high (Alternate Program Suspend)
- The Read state is active when MCRA.PGMA or MCRA.ERSA are high and High Voltage operation is ongoing on one partition, thus MCRA.EHVa is high (condition referred as Alternate Read-While-Write). In this case reads done on the other partitions are allowed, while reads done to the partition being operated on (either Alternate Erase or Alternate Program) result in an error with MCRA.RWEA set and the data read back possibly being random including the possibility of ECC errors.

37.4.4.1 Read While Read

Flash Memory Module is organized in Two independent Read Interfaces. Each Read interface is associated exclusively to a specific group of Read While Write Partitions. Flash Memory Module supports independent and Parallel Read access through the Two interfaces, namely Read-While-Read. Read interface assignment to Read While Write Partition is described in the following table:

Table 579. Read Interface Assignment

Read-While-Read Interface 0	Read-While Read Interface 1
RWW P0	—
RWW P1	—
RWW P2	—
RWW P3	—
RWW P4	—
—	RWW P5

Code Flash partitions reads return 256 bits (1 page = 4 doublewords). Data Flash partition reads return 128 bits. Register reads return 32 bits (1 word).

Flash array reads are performed through the Bus Interface Unit. In many cases the BIU applies “Read Page Buffering” to allow sequential reads to be executed with higher performance. This can provide data coherency issues that must be handled by software.

Data coherency may be an issue after a program or an Erase operation, as well as Test block operations.

Flash array reads to invalid locations result in indeterminate data. Invalid locations occur when addressing points to blocks that are not included in the Flash memory Map of any partition. Interlock writes to invalid locations result in an interlock occurring, but attempts to program these blocks do not eventuate since they are forced into a locked state. Erase occurs to selected and unlocked blocks even if the Interlock Write is to an invalid location.

Register reads to unmapped register address spaces return all 0's. Registers writes to unmapped register address spaces have no effect.

Simultaneous Read cycles on the Flash array and Read/Write cycles on the registers are possible, but not simultaneous register Read/Write accesses with Flash Array Interlock Writes are forbidden.

37.4.4.2 Error correction code

The Flash Memory module implements a strategy for improving the reliability of the data stored in Flash via the use of Error Correction Code.
In Code Memory the ECC size is 128 bits with 17 ECC bits associated. In Data Memory the ECC size is 64 bits with 15 ECC bits associated.
ECC function is implemented in such a way to guarantee Double Error Correction and Triple Error Detection (DEC-TED).
ECC coding has all 0's (intended as data and parity) as an invalid word and all 1's as a valid word.

In addition to providing the requested Read data, the Flash also provides output sidebands reflecting the status of the single and double error corrections and triple error detections performed at the end of each Read.

For functional safety coverage, Flash performs an EDC post ECC operation at every Read to detect mismatches in the ECC logic detection and correction. The result is output as sideband. Also active is a monitor circuit that checks the reference current in the sense amplifiers, including control of the internally regulated Read voltage, to prevent anomalous Read operation.

ECC circuitry provides single-bit fault correction used to achieve automotive reliability targets. Some units experience single-bit corrections throughout the life of the product with no impact on product reliability.

37.4.4.2.1 Multi-bits error management (more than 3 bits error >=4)

In this case, we can't predict the behavior without knowing the original data and the position of the bit flips. The hardware can react in multiple ways:

1. do nothing (that is, not signaling of any flag)
2. assert either SBC, SBC1, or EER without EEE
3. assert either SBC, SBC1, or EER with EEE
4. assert EEE only.

Note: SBC and SBC1 reporting have to be activated in UT0.

It is worth highlighting that by design the purpose of the ECC logic is not to detect multi-bit errors but cope with errors in the data and parity containing maximum 3 bit flips. In this condition, the EDC after ECC monitors the integrity of the ECC logic.

Other mechanisms must detect multi-bit errors or avoid their occurrence.

For example running the Array Integrity Check once after the boot before the safety application starts to protect the Data with an application level checksum.

An example of a mechanism that decreases the possibility of multi-bit errors in the array is the multiplexing. The bits of the same word are not one next to each other, but they are spaced of a fixed number of bits.

37.4.4.2.2 EEPROM emulation

The chosen ECC allows some bit manipulation so that a doubleword can be re-written several times without needing an erase of the block. This allows the use of a doubleword to store flags useful for the EEPROM emulation.

The sequence provided in *Table 580: Bits Manipulation: doublewords preserving ECC integrity* is related to Data Memory (therefore would not apply on Code Memory). The ECC coding allows, starting from an all 1's doubleword value, to rewrite subsequent predefined code values changing only 0s into 1s and preserving the ECC integrity.

Caution: Predefined code values have to be applied following the exact Byte order, as reported in the table.

Table 580. Bits Manipulation: doublewords preserving ECC integrity

Doubleword	ECC parity bits	Sequence
B7-B6-B5-B4-B3-B2_B1-B0		
0xFFFF_FFFF_FFFF_FFFF	0x7FFF	erase
0xFFFF_FFFF_FFFF_FFFE	0x3C88	write1
0xFFFF_FFFF_FEFE_FC80	0x3C88	write2
0xFFFF_FFFC_FEFC_FC80	0x1880	write3
0xFCFC_FFFC_FCFC_F880	0x1880	write4
0xFCF8_F8F0_FCE0_E080	0x1800	write5
0xF8F8_E080_F0E0_E000	0x1000	write6
0xE0E0_E080_E0E0_C000	0x1000	write7
0xE000_8000_00C0_0000	0x1000	write8

37.4.4.3 Flash address generation check

For functional safety coverage, Flash provides output sidebands reflecting the re-encoded address used to perform the actual Read operation. In the re-encoding process both word line and column within the selected block are evaluated. For the completion of the re-encoding operation, the addresses related to block selections are also computed. Re-encoding circuitry is able to detect both no-selection as well as multi-selection. Flash provides a self-check of address consistency comparing at each Read the sampling of the current Read address with the re-encoded one, and in case of mismatch an error sideband is set.

37.4.5 Modify mode

All the Modify operations of the Flash Memory module are managed through the Flash User Registers interface. This interface is referred as MAIN.

When a Modify operation is active on some blocks, no Read access is possible on any other block of the same Flash Partition. The Read-While-Modify is supported only among different Flash partitions.

During a Flash Modify operation any attempt to read any Flash location in the same Flash partition outputs invalid data and MCR.RWE is automatically set.

If a reset occurs during a Modify operation, the operation is immediately terminated and the module is reset to Read mode. The data integrity of the Flash section where the Modify operation was terminated is not guaranteed: the interrupted Flash Modify operation must be repeated.

In general each Modify operation is started through a sequence of 3 steps:

1. The first instruction is used to select the desired operation by setting its corresponding selection bit in MCR.PGM or MCR.ERS or UT0.MRE.
2. The second step is the definition of the operands: the address and the data for Programming, or the blocks for Array Integrity Check or Margin Read.
3. The third instruction is used to start the Modify operation by setting MCR.EHV or UT0.AIE.

Once selected, but not yet initiated, one operation can be canceled by resetting the operation selection bit.

A summary of the available Flash Modify operations are shown in *Table 581*.

Table 581. Flash Modify operations

Operation	Select bit	Operands	Start bit
Doubleword Program	MCR.PGM	Address and Data by Interlock Writes	MCR.EHV
Block Erase	MCR.ERS	SEL0, SEL1, SEL2 and SEL3	MCR.EHV
Array Integrity Check	None	SEL0, SEL1, SEL2 and SEL3	UT0.AIE
Margin Read	UT0.MRE	UT0.MRV + SEL0, SEL1, SEL2 and SEL3	UT0.AIE

Once MCR.EHV (or UT0.AIE) is set, all the operands can no longer be modified until MCR.DONE (or UT0.AID) is high.

In general each Modify operation is completed through a sequence of 4 steps:

1. Wait for operation completion: wait for MCR.DONE (or UT0.AID) to go high.
2. Check operation result: check bit MCR.PEG (or compare UM0–UM9 with expected value).
3. Switch-Off FPEC by resetting MCR.EHV (or UT0.AIE).
4. Deselect current operation by clearing MCR.PGM/ERS (or UT0.MRE).

Note: Do not initiate a Modify operation on a Flash Memory module while another Modify operation is in progress on a separate module.

In the following sections, all the possible Modify operations are described with accompanying examples of the sequences needed to activate them.

37.4.5.1 Program

A Flash Program sequence operates on any doubleword within the Flash core. Up to two words within the doubleword may be altered in a single doubleword program operation. A single program operation can also alter one page (4 doublewords, 32 bytes), or 4 pages max (128 bytes in total). It is possible to program only certain doublewords in the page or the quad-page, leaving the others with their original content. Whenever the array is programmed, the ECC bits are also programmed.

ECC is handled on a 64-bit boundary in Data Flash and 128b boundary in Code Flash. Whenever you program, ECC bits are also programmed. Thus, if only one word in any given ECC segment is programmed, the adjoining word/s (in that segment) should not be programmed later on, since ECC calculation has already completed for that segment. Attempts to program the adjoining word will probably result in an operation failure. All programming operations should be from 64 bits to 1024 bits, should be 64-bit aligned for Data Flash and 128-bit aligned for Code Flash. The programming operation should completely fill selected ECC segments.

Due to the fact that Flash functions in a System with an end2end ECC scheme, ECC parity bit are also input during an Interlock Write. After setting MCR.EHV, the FPEC checks the integrity of the data received in terms of ECC. If any doubleword in the doubleword program, the page program or the quad-page program contains an inconsistency between data and parity, the whole operation is rejected and a MCR.PEG=0 is set. The doublewords without data and with parity inconsistency are thus not programmed.

Programming changes the value stored in an array bit from logic ‘1’ to logic ‘0’ only. Programming cannot change a stored logic ‘0’ to a logic ‘1’. If a logic ‘0’ is attempted to be over programmed by a logic ‘1’, the resulting operation fails (MCR.PEG=0), and the 0s that are interlocked are merged (ORed) with 0s that are already present in the 64-bit ECC segment. Extreme care has to be taken when doing such an operation due to the high risk of producing ECC errors.

Addresses in locked blocks cannot be programmed. Register locked blocks cannot be programmed. Hardware locking is also available which only affects the program operation. Hardware locking is enabled by sideband signals, and if enabled, programs are prevented from accessing hardware locked blocks in the same way as program software locking.

The user may program the values in any or all of eight the words in a page with a single program sequence. Page-bound words have addresses which differ only in address bits [4:2]. Up to four pages can be programmed at once on a quad-page boundary, which differ only in address bits [6:5]. Only certain doublewords may be programmed within the page or the quad-page, without altering the content of the ones unselected via interlock.

The Program operation consists of the following sequence of events:

1. Change the value of MCR.PGM from ‘0’ to ‘1’.
2. Ensure the block that contains the address to be programmed is unlocked. Write the first address to be programmed with the program data. The Flash Memory module latches address bits (23:7) at this time, as well as written data. This Write is referred to as a Program Data Interlock Write. An Interlock Write may be as large as 64 bits, and as small as 32 bits (depending on the CPU bus).
3. If more than 1 word is to be programmed, write the additional address in the doubleword with data to be programmed. This is referred to as a Program Data Write.

The Flash Memory modules ignore address bits (23:7) for program data writes. The eventual unwritten data word defaults to 0xFFFFFFFF.

4. Write a logic ‘1’ to the MCR.EHV to start the internal program sequence or terminate via step 9.
5. Wait until the MCR.DONE goes high.
6. Confirm MCR.PEG=1.
7. Write a logic ‘0’ to the MCR.EHV.
8. If more addresses are to be programmed, return to step 2.
9. Write a logic ‘0’ to the MCR.PGM to terminate the program operation.

Programs may be initiated with the 0–1 transition of MCR.PGM or by clearing MCR.EHV at the end of a previous program. The first Write after a program is initiated determines the page address to be programmed. This first Write is referred to as an Interlock Write. The Interlock Write determines if the UTEST or normal array space will be programmed via MCR.PEAS.

In the case of an erase-suspended program, the values in MCR.PEAS may be modified via the Program Interlock Write, enabling erase-suspended programs to UTEST Nonvolatile Memory space and revert back to their original state (related to erase-suspended operation) once the erase-suspended program is completed.

An Interlock Write must be performed before setting MCR.EHV. An Interlock Write performed after setting MCR.EHV is forbidden and data and address passed through interlock is lost.The user may terminate a program sequence by clearing MCR.PGM prior to setting MCR.EHV.

After the Interlock Write, additional writes affect the data to be programmed at the word location determined by address bits [4:2], as well as the page location within a 1024 bit segment (determined by address bits [6:5]). Unwritten locations default to a data value of 0xFFFF_FFFF. If multiple writes are performed on the same location, the data for the last Write is used in programming.

While MCR.DONE is low and MCR.EHV is high, the user may clear EHV, resulting in a Program Abort.

A Program Abort forces the module to step 8 of the program sequence.

An aborted program results in MCR.PEG being set low, indicating a failed operation. MCR.DONE must be checked to know when the aborting command has completed.

The data space being operated on before the Abort contains indeterminate data. This may be recovered by repeating the same program instruction or executing an Erase of the affected blocks.

The user may not abort a program sequence while in Program Suspend.

Figure 453. Doubleword program of data 0x55AA_55AA at address 0x00_AAA8 and data 0xAA55_AA55 at address 0x00_AAAC

```
MCR = 0x0000_0010;           /* Set PGM in MCR: Select Operation */
(0x00_AAA8) = 0x55AA_55AA;   /* Latch Address and 32 LSB data */
(0x00_AAAC) = 0xAA55_AA55;   /* Latch 32 MSB data */
MCR = 0x0000_0011;           /* Set EHV in MCR: Operation Start */
do                           /* Loop to wait for DONE=1 */
{ tmp = MCR;                 /* Read MCR */
} while ( !(tmp & 0x0000_0400) );
status = MCR & 0x0000_0200;  /* Check PEG flag */
MCR = 0x0000_0010;           /* Reset EHV in MCR: Operation End */
MCR = 0x0000_0000;           /* Reset PGM: Deselect Operation */
```

37.4.5.1.1 Program Suspend/Resume

The program sequence may be suspended to allow Read access to the Flash array. It is not possible to program or to erase during a Program Suspend. Read-While-Write operations may also be used to read the array during a program sequence providing the Read is to a different partition.

During Program Suspend, all reads to the doubleword targeted for program return indeterminate data.

A Program Suspend can be initiated by changing the value of MCR.PSUS from ‘0’ to ‘1’. MCR.PSUS can be set to ‘1’ at any time when MCR.PGM and MCR.EHV are high and MCR.ERS is low. 0–1 transition of MCR.PSUS causes the module to start the sequence which places it in Program Suspend.

It is possible to suspend a program performed during an Erase Suspend. The user must wait until MCR.DONE=1 before the module is suspended and further actions are attempted. MCR.DONE takes no more time than t_{PSUS} to go high after MCR.PSUS is set to ‘1’.

Once suspended, the array may be read. Flash memory reads while MCR.PSUS=1 from the doubleword being programmed return indeterminate data.

Figure 454. Doubleword Program Suspend

```
MCR = 0x0000_0019; /* Set PSUS in MCR: Program Suspend */
do                 /* Loop to wait for DONE=1 */
{ tmp = MCR;       /* Read MCR */
} while ( !(tmp & 0x0000_0400) );
```

Note that there is no need to clear MCR.EHV and MCR.PGM in order to perform reads during program suspend.

The Program sequence is resumed by writing a logic ‘0’ to MCR.PSUS.

MCR.EHV must be set to ‘1’ before MCR.PSUS can be cleared to resume the operation.

The module continues the program sequence from one of a set of predefined points. This may extend the time required for the program operation.

Figure 455. Doubleword Program Resume

```
MCR = 0x0000_0011;/* Reset PSUS in MCR: Program Resume */
```

Repeated suspends at a high frequency may result in the operation timing out, and the embedded Flash memory responds by completing the operation with a fail code (MCR.PEG=0). The minimum time between Program Suspends to ensure this does not occur is t_{PSRT}.

37.4.5.1.2 Over-programming protection (OPP) enable

Over Programming Protection can be enabled on a block-by-block basis. In an OPP enabled block of code Flash, any quad-word which has already been programmed cannot be programmed again. The OPP enable does not affect the Erase operation. It is possible to independently set OPP and (hardware) erase protection in order to avoid any kind of overwrite. Using these protection mechanisms in conjunction with the Test mode disable feature gives even greater overwrite protection. Attempts to over-program result in MCR.PEG being cleared.

OPP can be enabled for 16 KB, 32 KB, 64 KB and 256 KB blocks.

OPP is enabled by sideband signals and, if enabled, the program operation is modified to check for programmed bits prior to undertaking a high voltage program event.

37.4.5.1.3 Successful program address

To enable the System to create logic to authenticate operations with a program (like in a diary operation), the Flash Memory module provides the last successful address programmed as a sideband signal. This value is held until the next successful program. A successful program is defined as a program operation of data that is not all 1s, to an unlocked block and the operation results in MCR.PEG returning a ‘1’. An all 1’s value is the default (and invalid) address after reset.

In case of page programming, the last successful address returned is the beginning of the page itself regardless of whether that specific address has not been passed through interlock. In case of quad-page programming, the last successful address returned is the beginning of the quad-page itself regardless of whether that specific address has not been passed through interlock.

37.4.5.2 Erase

Erase changes the value stored in all bits of the selected block(s) to logic ‘1’. An Erase sequence operates on any combination of blocks in the low, mid, high or 256K address space. The Test block cannot be erased.

The Erase sequence is fully automated within the Flash. The user only needs to select the blocks to be erased and initiate the Erase sequence.

Register locked blocks cannot be erased. Hardware locking, which only affects the Erase operation is also available. Hardware locking is enabled by sideband signals and, if enabled, erases are prohibited on hardware locked blocks in the same way as erase software locking. If multiple blocks are selected for erase during an Erase sequence, no specific operation order can be assumed.

The Erase operation consists of the following sequence of events:

1. Change the value in the MCR.ERS bit from ‘0’ to ‘1’.
2. Select the block(s) to be erased by writing 1’s to the appropriate register(s) in SEL0, SEL1, SEL2 and SEL3 registers. Note that Lock and Select are independent. If a block is selected and locked, no erase occurs.
3. Write to any address in Flash. This is referred to as an Erase Interlock Write.
4. Write a logic ‘1’ to MCR.EHV to start the internal Erase sequence or skip to step 9 to terminate.
5. Wait until MCR.DONE goes high.
6. Confirm MCR.PEG=1.
7. Write a logic ‘0’ to MCR.EHV.
8. If more blocks are to be erased, return to step 2.
9. Write a logic ‘0’ to the MCR.ERS bit to terminate the Erase operation.

After setting MCR.ERS, one Write (referred to as an Interlock Write) must be performed before MCR.EHV can be set to ‘1’.

Data words written during Erase sequence Interlock Writes are ignored. As Flash functions in an System with an end2end ECC scheme, ECC parity bits are also input during an Interlock Write. After setting MCR.EHV, the FPEC checks the integrity of the data received in terms of ECC even if the value of the data is not meaningful for the Erase operation. If an inconsistency between data and parity is found, the whole operation is rejected and MCR.PEG=0 is set.

The user may terminate the Erase sequence by clearing ERS before setting EHV.

An Erase operation may be aborted by clearing MCR.EHV assuming MCR.DONE is low, MCR.EHV is high and MCR.ESUS is low. An Erase Abort forces the module to step 8 of the Erase sequence. An aborted Erase results in MCR.PEG being set low, indicating a failed operation. MCR.DONE must be checked to know when the aborting command has completed.

The block(s) being operated on before the Abort contain indeterminate data. This may be recovered by executing an Erase on the affected blocks. The user may not abort an Erase sequence while in Erase-Suspend.

Figure 456. Erase of blocks B0F1 and B0F2

```
MCR = 0x0000_0004;           /* Set ERS in MCR: Select Operation */
SEL0 = 0x0006_0000;          /* Set LSL2-1 in SEL0: Select Blocks */
(0x00_0000) = 0xFFFF_FFFF;   /* Latch a Flash Address */
MCR = 0x0000_0005;           /* Set EHV in MCR: Operation Start */
do                           /* Loop to wait for DONE=1 */
{ tmp = MCR;                 /* Read MCR */
} while ( !(tmp & 0x0000_0400) );
status = MCR & 0x0000_0200;  /* Check PEG flag */
MCR = 0x0000_0004;           /* Reset EHV in MCR: Operation End */
MCR = 0x0000_0000;           /* Reset ERS: Deselect Operation */
```

37.4.5.2.1 Erase Suspend/Resume

The Erase sequence may be suspended to allow Read access to the Flash array. It is possible to program another block in any Flash Memory module during Erase-Suspend.

Programming a location subject to suspended Erase results in the program not being executed and PEG being set to 0.

It is not possible to erase any block during an Erase-Suspend.

During Erase-Suspend, all reads to blocks targeted for Erase return indeterminate data.

An Erase-Suspend can be initiated by changing the value of MCR.ESUS from ‘0’ to ‘1’. MCR.ESUS can be set to 1 at any time when MCR.ERS and MCR.EHV are high and MCR.PGM is low. A 0–1 transition of MCR.ESUS causes the module to start the sequence which places it in Erase-Suspend. The User must wait until MCR.DONE=1 before the module is suspended and further actions are attempted.

MCR.DONE takes no longer than t_{ESUS} to go high after MCR.ESUS is set to ‘1’.

Once suspended, it is not possible to alter the content of SEL0, SEL1, SEL2 and SEL3 registers. Once suspended, the array may be read. Flash memory reads while MCR.ESUS=1 from the block(s) being erased return indeterminate data.

Figure 457. Block Erase Suspend

```
MCR = 0x0000_0007; /* Set ESUS in MCR: Erase Suspend */
do                 /* Loop to wait for DONE=1 */
{ tmp = MCR;       /* Read MCR */
} while ( !(tmp & 0x0000_0400) );
```

Note that there is no need to clear MCR.EHV and MCR.ERS in order to perform reads during Erase-Suspend.

The Erase sequence is resumed by writing a logic ‘0’ to MCR.ESUS.

MCR.EHV must be set to ‘1’ before MCR.ESUS can be cleared to resume the operation. The module continues the Erase sequence from one of a set of predefined points and this may extend the time required for the Erase operation.

Figure 458. Block Erase Resume

```
MCR= 0x0000_0005;/* Reset ESUS in MCR: Erase Resume */
```

Repeated suspends at a high frequency may result in the operation timing out and the embedded Flash memory responds by completing the operation with a fail code (MCR.PEG=0). The minimum time between Erase suspends to ensure this does not occur is t_{ESRT}.

37.4.5.3 Factory Mode

The Factory Mode feature can be used to improve Program and Erse Times and it is managed through a diary location mechanism. If location UTEST at offset +0x0020 is found virgin at the moment of the reset, MCR.FERS can be set, after rising MCR.ERS and before MCR.EHV. After setting MCR.EHV the FPEC recognizes the value of FERS and allows a fast Program and Erase Operation. Under those conditions a factory Mode can start and no Read-While-Write is allowed (all the flh_busy sideband signals are set high). If the diary location has been programmed by the user (having changed from being virgin at reset), the Factory Mode gets permanently disabled.

37.4.6 Alternate program and erase interface

A second Modify interface is provided on the Flash for program and erase operations. The intent is that a dedicated master can have private access to blocks through this interface.

The alternate interface includes an alternate MCR register and alternate lock registers. Any block in low, mid, or high address space may be accessible through the alternate interface. The UTest NVM Block is also accessible for program through the alternate interface. Blocks in 256K Address Space are not accessible through this interface.

The alternate interface supports programming of data sizes from one doubleword (64 bits) to four doublewords within a single page (256 bits). Within a page, up to eight words may be altered in a single program operation. It is recommended that programming operations through the alternate interface be from 64 bits to 256 bits, and be 64-bit aligned. Programming has the same constraints as for the Main interface concerning ECC boundaries: 64b for Data Flash and 128b for Code Flash.

The alternate interface supports erasing a single block at a time with each erase operation.

As with the main interface, erase suspend, program suspend, and erase-suspended programs are supported.

Program and Erase times for the alternate interface are the same as the main interface when programs and erases are done in isolation. If main interface and alternate interface are being utilized simultaneously, bandwidth sharing is utilized.

Note: It is possible for the UTest NVM block to exist in both the main program interface and alternate program interface. In the event of a conflict the first interface to do the program to an address will get priority, and all other attempts after this will result in an OTP over-program failure (MCR[PEG] or MCRA[PEGa] being set to 0).

37.4.6.1 Alternate program interface

The programming procedure follows exactly the same flow as the main interface, except that only page programming is allowed.

The alternate interface program is done at lower priority compared to the main interface. Based on the size of programming allowed, the main interface has a 4:1 ratio advantage over the alternate interface. Program operations are allowed to finish on the active interface prior to allowing the inactive interface to execute its operation. Suspends done on the active interface enable the inactive interface to begin its operation.

For alternate program while erase, there could be a latency of up to 5 ms in the program operation.

For more information on interaction between the alternate program interface and the main interface, see *Section 37.4.6.3: Alternate interface performance*.

If the alternate interface is the only interface active, then normal program times may be expected.

37.4.6.2 Alternate erase interface

For erases done through the alternate interface, the single block to be erased is selected based on the address that is received during the erase interlock write. Thus the Select registers are not valid for the alternate interface (step 2 of the erase flow in *Section 37.4.5.2: Erase*), and the interlock write address is used (step 3 of the erase flow in *Section 37.4.5.2: Erase*). All other features of the main interface apply to the alternate erase interface. The

erase procedure is exactly the same as for the main interface. See *Section 37.4.5.2: Erase* for more information on erase.

The alternate interface erase is done at lower priority compared to the main interface. The time-slice ratio between the two interfaces is 10:1, with 50 ms of time given to the main erase for every 5 ms of alternate interface. Suspends done on the active interface enable the inactive interface to begin its operation.

For alternate erase while program, there can be a latency of up to 5 ms in the main program time. Once the alternate erase is interrupted, it will remain interrupted for main programs for a fixed time of 50 ms, at which time the alternate erase will restart. It should be noted that multiple interruptions of the alternate erase will significantly increase the total erase time.

For alternate erase while erase, the main erase has priority and receives 50 ms of erase for every 5 ms of alternate erase. This may result in a significant increase in the erase time for the alternate interface, and a minor increase in erase time for the main interface.

For more information on interaction between the alternate erase interface and the main interface, see *Section 37.4.6.3: Alternate interface performance*.

If the alternate interface is the only interface active, then normal erase times may be expected.

37.4.6.3 Alternate interface performance

Table 582 highlights the interactions between the alternate interface and the main interface for program, erase, suspend, and idle conditions.

Table 582. Alternate Program and Erase Characteristics

Alt PGM/ERS interface request	Main PGM/ERS interface request	Alt interface result on memory array	Main interface result on memory array
Idle	Idle	Idle	Idle
Idle/SUSP	PGM/ERS	Idle/SUSP	PGM/ERS
PGM/ERS	Idle/SUSP	PGM/ERS	Idle/SUSP
PGM	PGM	PGM	Priority PGM (4:1 ratio)
PGM	ERS	PGM with 5ms latency	ERS (possible minimal increase to total ERS time)
ERS	PGM	ERS (significant increase in total ERS time expected)	Priority PGM (possible latency of 5 ms) Once PGM is granted, 50 ms time slice available for more programs
ERS	ERS	ERS (significant increase in total ERS time expected)	Priority ERS (10:1 ratio) (possible minimal increase to total ERS time) 50 ms Main ERS for every 5 ms alternate ERS

37.4.7 User Test mode

Customers can put the Flash Memory module in User Test mode to perform specific tests for the integrity of the Flash array.

Three kinds of test can be performed:

- Array Integrity Self Check
- Margin Read
- Vth Distribution

User Test mode is an exclusive operation—it cannot be run in conjunction with any other Flash mode (Read, Write, Low-Power and Power-Down), and Read-While-Write functionality is inapplicable. Read accesses attempted by the user during User Test mode generates a Read-While-Write Error (RWE of MCR set). Vth Distribution mode is conditioned by SoC Life Status (mode allowed only as Failure analysis)

It is not permitted to perform User Test operations on the Test blocks (including the UTEST section).

37.4.7.1 Array Integrity Self Check

Array integrity is checked using a predefined address sequence (proprietary), and this operation is executed on selected (SEL0, SEL1, SEL2 and SEL3) blocks. Blocks marked by TMDBS are automatically excluded from MISR signature computation. Any random or non-random code is valid. Once the operation is completed, the results of the reads can be checked by reading the MISR value (stored in UM0–UM9), to determine if an incorrect Read, or ECC detection was noted. Array integrity requires that the Read Wait States and Address Pipelined control registers in the BIU be set to match the frequency being used.

The internal MISR calculator is a 64 + 8 + 1-bit register.

The 256-bit data, the respective ECC parity data, and the double ECC (ERR) errors of the four doublewords are therefore captured by the MISR through 4 different Read accesses at the same location.

The whole check is done through 4 complete scans of the memory address space:

1. The first pass scans only bits 63-0 + 8 ecc + ERR of each doubleword.
2. The second pass scans only bits 128-64 + 8 ecc + ERR of each doubleword.
3. The third pass scans only bits 191-129 + 8 ecc + ERR of each doubleword.
4. The fourth pass scans only bits 255-192 + 8 ecc + ERR of each doubleword.

The 256-bit data and the respective ECC parity data are sampled after any single-bit ECC correction and the double error flag is set after any ECC detection. Single-bit correction is always active and logging may be enabled (UT0.SBCE), resulting in the MCR.SBC flag and related address to be captured in ADR register, but this does not change the final signature in the UM0–UM9.

Only data from existing locations are captured by the MISR. The MISR can be seeded to any value by writing the UM0–UM9 registers.

The Array Integrity Self Check consists of the following sequence of events:

1. Set UTE in UT0 by writing the respective password in UT0.
2. Select the block(s) to be checked by writing 1’s to the appropriate field(s) in SEL0, SEL1, SEL2 and SEL3 registers. Blocks selected for Array Integrity Check do not need

to be unlocked. It is not possible to perform Array Integrity operations on the UTEST Nonvolatile Memory block.

3. If desired, UT0.AIS can be set for sequential addressing only.
4. Seed the MISR UM0–UM9 with desired values.
5. If breakpoints are desired, set UT0.AIBPE to ‘1’, and ensure that MCR.EER and MCR.SBC are cleared. If breaking on single-bit correction is required, ensure that UT0.SBCE is set.
6. Write a logic ‘1’ to UT0.AIE to start the Array Integrity Check. Array integrity operations may be aborted prior to UT0.AID going high. This may be done by clearing UT0.AIE. It should be noted that in the event of an aborted Array Integrity check, the MISR registers contain a signature for the portion of the operation that was completed prior to the Abort, and is not definitive. Prior to performing further Array Integrity operations, the UM0–UM9 registers may require initialization to the desired seed value by performing register writes. Array Integrity operations may be suspended while UT0.AID is low. UT0.AISUS may be set to request an Array Integrity Suspend. Once UT0.AID has been set high, UT0.AISUS may be cleared to resume the Array Integrity sequence.
7. Wait until UT0.AID goes high.
8. Check that UT0.NAIBP is set to ‘1’ if breakpoints were previously enabled so that the MCR.EER, MCR.SBC and ADR registers may be checked to determine the cause and address of the break. Prior to resuming the operation, it necessary to clear MCR.SBC or MCR.EER in order to preserve coherency of MCR.SBC, MCR.EER and ADR updates with respect to any subsequent breakpoints. Operation may then be resumed by clearing UT0.NAIBP. Continue waiting until the UT0.AID bit goes high. If breakpoints were not enabled, or if UT0.NAIBP is low when UT0.AID goes high, then the operation is complete.
9. Compare UM0–UM9 content with the expected result.
10. Write a logic ‘0’ to the UT0.AIE bit.

Use sequential addressing for normal integrity checks of the Flash memory. If a more detailed check of the Read path is required (in diagnostic mode), leave UT0.AIS at ‘0’ and use the proprietary address sequence that checks the Read path more completely, however this sequence takes more time.

Caution: Do not modify the content of Block Select (SEL0, SEL1, SEL2 and SEL3) and Lock (LOCK0, LOCK1, LOCK2 and LOCK3) registers during execution of the Array Integrity Check operation or the MISR value may vary unpredictably.

User mode array reads requested during the Array Integrity test are ignored to ensure that the Array Integrity operation is not corrupted. The memory array does not respond to array Read requests during this time. User mode array reads may be executed if suspended or at a breakpoint.

While UT0.AID is low and UT0.AIE is high, the user may clear AIE, resulting in an Array Integrity Check Abort. UT0.AID must be checked to know when the aborting command has completed.

Array Integrity Check can be suspended by setting UT0.AISUS while Array Integrity Check is running (UT0.AID=0). Once suspended (UT0.AID=1) Array Integrity Check cannot be aborted and clearing UT0.AISUS starts the Resume.

Figure 459. Array Integrity Check of blocks B0F1 and B0F2

```
UM0/9 = 0x0000_0000; /* Reset UM0/9 content */
UT0 = 0xF9F9_9999;   /* Set UTE in UT0: Enable User Test */
SEL0 = 0x0006_0000;  /* Set LSL2-1 in SEL0: Select Blocks */
SEL2 = 0x0000_0006;  /* Set A256KSL2-1 in SEL2: Select Blocks */
UT0 = 0x8000_0002;   /* Set AIE in UT0: Operation Start */
do                   /* Loop to wait for AID=1 */
{ tmp = UT0;         /* Read UT0 */
} while ( !(tmp & 0x0000_0001) );
data0/9 = UM0/9;     /* Read UM0/9 content*/
UT0 = 0x0000_0000;   /* Reset UTE, AIE: Operation End */
```

37.4.7.2 Margin Read

User Margin Read may be achieved using the Array Integrity interface and has all the associated features of the Array Integrity interface (MISR calculation, suspend capability and Breakpoints).

Margin Read procedure (either Margin 0 or Margin 1) can be run on selected blocks in order to unbalance the Sense Amplifiers with respect to standard Read conditions, so that all the Read accesses reduce the margin vs ‘0’ (UT0.MRV=0) or vs ‘1’ (UT0.MRV=1).

User Margin Read is a self timed event and is independent of system clocks or wait states selected. Margin Read can be run on selected (SEL0, SEL1, SEL2 and SEL3) blocks, generating checksum calculations on MISR and ECC flagging. Blocks marked by TMDBS are automatically excluded from MISR signature computation. Once the operations are completed, the results of the Margin Reads can be checked, comparing checksum values in UM0–UM9 and reading MCR.EER and MCR.SBC to determine if zero, one or two bits are being detected by the Margin Read.

The MISR can be seeded to any value by writing the UM0–UM9 registers.

As with Array Integrity Check, the internal MISR calculator is a 64 + 8 + 1-bit register. The 256-bit data, the corresponding ECC parity data and the double ECC (ERR) errors of the four doublewords are therefore captured by the MISR through 4 different Read accesses at the same location. The whole check is accomplished through 4 complete scans of the memory address space:

1. The first pass scans only bits 63-0 + 8 ecc + ERR of each doubleword.
2. The second pass scans only bits 128-64 + 8 ecc + ERR of each doubleword.
3. The third pass scans only bits 191-129 + 8 ecc + ERR of each doubleword.
4. The fourth pass scans only bits 255-192 + 8 ecc + ERR of each doubleword.

The 256-bit data and the respective ECC parity data are sampled after any single-bit ECC correction, and the double error flags are set after any ECC detection. Single-bit logging is always enabled regardless of the value of UT0.SBCE, resulting in the MCR.SBC flag and related address to be captured in ADR register, but this does not alter the final signature in the UM0–UM9.

In any case, the charge losses detected through the Margin Read cannot be considered device failures and no Failure Analysis is initiated on them.

The Margin Read Setup operation consists of the following sequence of events:

1. Set UTE in UT0 by writing the respective password in UT0.
2. Select the block(s) to be checked by writing 1's to the appropriate register(s) in SEL0, SEL1, SEL2 and SEL3 registers. Blocks selected for Margin Read do not need to be unlocked. It is not possible to do Margin Read operations on the UTEST Nonvolatile Memory block.
3. Seed the MISR UM0–UM9 with desired values.
4. Ensure that MCR.EER and MCR.SBC are cleared.
5. Set UT0.SBCE to '1' to detect single bits during Margin Read.
6. Change the value of UT0.MRE from '0' to '1'.
7. Select the Margin level: UT0.MRV=0 for 0's margin, UT0.MRV=1 for 1's margin.
8. Write a logic '1' to UT0.AIE to start the Margin Read.
9. Wait until UT0.AID goes high.
10. Check that UT0.NAIBP is set to '1' if breakpoints were previously enabled so that MCR.EER, MCR.SBC and ADR.ADDR may be checked to determine the cause and address of the break. Prior to resuming operation, clear MCR.SBC or MCR.EER. Operation may then be resumed by clearing UT0.NAIBP. Continue to wait until UT0.AID goes high. If breakpoints were not enabled, or if UT0.NAIBP is low when UT0.AID goes high, then the operation is complete.
11. Compare UM0–UM9 content with the expected results.
12. Check MCR.EER and MCR.SBC.
13. Write a logic '0' to UT0.AIE, UT0.MRE and UT0.MRV.
14. If more blocks are to be checked, return to step 2.

The linear address sequence is used regardless of the value of UT0.AIS.

Caution: Do not modify the content of Block Select (SEL0, SEL1, SEL2 and SEL3) and Lock (LOCK0, LOCK1, LOCK2 and LOCK3) registers during execution of the Margin Read operation, otherwise the MISR value may vary unpredictably.

User mode array reads requested during the Margin Read are ignored to ensure that the Margin Read operation is not corrupted. The memory array does not respond to array Read requests during this time. User mode array reads may be executed if suspended or at a breakpoint.

While UT0.AID is low and UT0.AIE is high, the user may abort Margin Read by clearing AIE. UT0.AID must be checked to know when the aborting command has completed. Margin Read can be suspended by setting UT0.AISUS while Margin Read is running (UT0.AID=0). Once suspended (UT0.AID=1), Margin Read cannot be aborted and clearing UT0.AISUS starts the Resume.

Figure 460. Margin Read Check versus 1's

```
UM0/9 = 0x0000_0000; /* Reset UM0/9 content */
UT0 = 0xF9F9_9999;   /* Set UTE in UT0: Enable User Test */
SEL0 = 0x0006_0000;  /* Set LSL2-1 in SEL0: Select Blocks */
UT0 = 0x8000_0020;   /* Set MRE in UT0: Select Operation */
UT0 = 0x8000_0030;   /* Set MRV: Select Margin versus 1's */
UT0 = 0x8000_0032;   /* Set AIE in UT0: Operation Start */
do                   /* Loop to wait for AID=1 */
{ tmp = UT0;         /* Read UT0 */
} while ( !(tmp & 0x0000_0001) );
data0/9 = UM0/9;     /* Read UM0/9 content*/
UT0 = 0x8000_0030;   /* Reset AIE in UT0: Operation End */
UT0 = 0x0000_0000;   /* Reset UTE, MRE, MRV, AIS in UT0:
                        Deselect Operation */
```

37.4.7.3 Vth Distribution

Vth Distribution functionality can be used in case of Failure Analysis for quick assessment of cells status in terms of Vth. The functionality has bit register UTO.VTD as initiator. Writing of this bit is enabled by the Life Cycle state of the device. Life Cycles has to be in Failure Analysis (FA) condition to allow Vth Distribution mode.

Vth Distribution functionality works on selected blocks on the base of multi internal runs. Each run corresponds to an internal voltage step, where cells read as '1' (Vth lower than internal voltage level) are counted: the completion of each run provides on UMISR0 the count of these cells. Clearing UTO.NAIBP starts next voltage step run, till completion of the intended voltage range. Number of internal voltage step is fixed. Aim of the test is to sweep the range from the erased cells with the highest Vth to the programmed cells with the lowest Vth, therefore to provide an assessment of the effective distribution of the cells in the selected blocks.
Once all the cell counts, one per run, have been collected, distribution can be obtained as difference between counts collected through subsequent steps.
In every run, all the cell, including the ECC parity ones, are counted for the block selected.
Vth Distribution results are expected to be collected at room temperature (25C).

Vth Distribution Setup operation consists of the following sequence of events:

1. Set UTE in UT0 by writing the respective password in UT0.
2. Select the block(s) to be checked by writing 1's to the appropriate register(s) in SEL0, SEL1, SEL2 and SEL3 registers.
3. Unlock blocks Blocks o be checked by writing 0's to the appropriate register(s) in LOCK0, LOCK1, LOCK2 and LOCK3 registers. Blocks selected for Vth Distribution need to be unlocked. It is not possible to do Vth Distribution operations on the UTEST

Nonvolatile Memory block, neither on blocks protected through TMD SEAL Block Select.

4. Set UT0.VTD to ‘1’ to select Vth Distribution mode.
5. Write a logic ‘1’ to UT0.AIE to start the first internal run.
6. Wait until UT0.AID goes high.
7. Check that UT0.NAIBP is set to ‘1’. If NAIBP is at ‘1’ it means current internal run has been completed and device is ready to increase voltage step. If NAIBP is at ‘0’ the overall sweep has been completed.
8. Collect UM0 content for the current voltage step (run).
9. Clear UT0.NAIBP to increase voltage step and return to 6)

Caution: Voltage levels used during Vth distribution sweep may alter the intended reliability of the Cells as data retention. For this reason Vth distribution can be used no more than 10 times.

37.4.8 Protection strategy

Two kinds of protection are available: Modify Protection to avoid unwanted program/erase in Flash blocks and Test Mode Disable to avoid piracy.

37.4.8.1 Modify Protection

The Flash Modify Protection information is stored in nonvolatile Flash cells located in the UTEST region. This information is read once during the Flash Initialization phase following exit from reset and is stored in volatile registers that act as actuators at SoC level. The information is passed to the Flash Memory module through erase_lock and program_lock sidebands, and allows independent protection of each block of Low, Mid, High and 256K Address Space Block.

There are also Address Space Block Locking registers provided to independently lock/unlock each block of Low, Mid, High and 256K Address Space Block against program and erase. Locking is done through the LOCK0, LOCK1, LOCK2 and LOCK3 registers. These registers do not have a nonvolatile image stored in the UTEST area of TestFlash, so the locking information is kept on reset: volatile content is all 1’s, meaning all blocks are locked.

All the volatile registers can be written to ‘0’ or ‘1’ at any time, therefore the user application can lock and unlock blocks when desired. Note, however, that overall protection is achieved by the combination of sidebands with Low, Mid, High and 256K Address Space Block Locking registers in such a way that a block can be programmed/erased only if both (sidebands and Locking registers) indicate an unlocked state. Sidebands independently control program protection, erase protection and over-programming protection (OPP) on a block-by-block basis.

37.4.8.2 Test Mode Disable

In the TestFlash area a mechanism is also available to disable entry into Test mode. Extreme care must be taken when using this feature, as blocks that are selected for protection in this way cannot be analyzed for possible failures by manufacture failure analysts.

Protection of this sort prevents all high voltage operations to the Flash executed by the internal FPEC as well as reads from FPEC, including Array Integrity Check or Margin Read,

when using Test mode interfaces. Protection through other interfaces is managed through normal User mode protection mechanisms, that are System specific.

To disable manufacturer entry into Test mode, first program the Test Mode Disable Seal location to 0x5A4B_3C2D and once the next reset is asserted, Test mode is disabled.

It is possible to create a password to enable manufacturer entry into Test mode. This can be programmed into the Test Mode Disable Override Passcode. It is safest to do this prior to censoring the part to avoid unintended lockout. The passcode may not be 0x0000_0000, 0xFFFF_FFFF, or 0x5555_5555. These are all invalid passcodes and are not accepted for override. If the customer desires that override never be possible, one of the three invalid passcodes should be put into this location. Passcodes may be entered to authenticate entry (if enabled) by performing a register Write to offset address 0x90.

Only blocks selected in the Test Mode Disable Block Select field are controlled by the Test Mode Disable feature. Thus is it possible for customers to select blocks for this type of protection, and render them ineligible for manufacturer failure analysis. Bits programmed to ‘0’ in the Test Mode Disable Block Select field designate blocks that are controlled by Test Mode Disable feature. The TestFlash block, and thus the UTEST and BAF sections, is always protected once the Test Mode Disable Seal password is written. In order to permit two opportunities to select blocks for Test Mode disable, two regions are available, both in UTEST: one at offset 0x50, and the other at offset 0x60; both are 128-bit sized (only some may be effective, depending on actual sectorization). The bits of these two regions are logically ANDed in order to define which blocks are effectively selected for the Test Mode Disable feature.

The bit Block Select field is organized as follows, and aligned in the way blocks are defined in the LOCKx/SELx registers:

Table 583. Test Mode Disable block select

Block	Block selection bits
Blocks in Mid space	Data[16:31]
Blocks in Low space	Data[2:15]
Blocks in High space	Data[48:63]
Blocks in 256K space	Data[64:127]

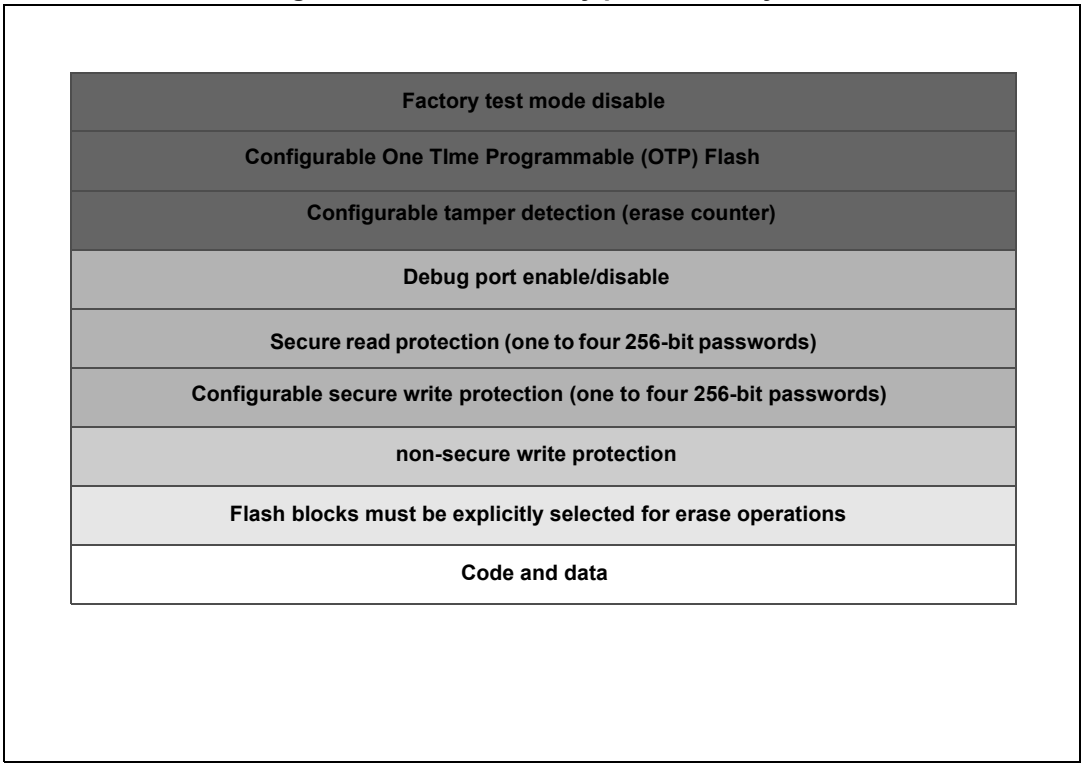
38 Flash Memory Programming and Configuration

This chapter provides basic guidance on activities related to configuring and programming the embedded Flash memory in the SPC58xEx/SPC58xGx microcontroller.

Multiple layers of protection are available to prevent unintended updates to Flash contents and unauthorized reads and writes to Flash contents. The protection mechanisms range from simple write-protect bits for individual Flash blocks to four levels of 256-bit password read/write protection and configuration of Flash memory blocks as One Time Programmable (OTP).

Additionally, the SPC58xEx/SPC58xGx microcontroller includes a tamper detection feature, debug port enable/disable, and the ability to disable the ability for the device to be put into factory test mode.

Figure 461. Flash memory protection layers



At the lowest layer of protection, all Flash erase operations require the explicit selection of one or more blocks of Flash memory before the operation is performed. When executed, the operation is only performed on the selected block(s).

The next layer is the chip's ability to lock individual Flash blocks against unintended programming and erase operations. This level of protection does not offer any password protection and is intended to provide protection against *accidental* changes to code and data.

The first layer of *secure* protection is the microcontroller's ability to support up to four levels of 256-bit password-secured *write* protection for individual Flash blocks.

Similarly, up to four levels of password-secured *read* protection can be implemented. Unlike the password-secured write protection, which is implemented on a per-block basis, password-secured read protection applies to five pre-defined (at the factory) groups of Flash blocks. The read protection applied to a group applies to each block within the group. In contrast, secure write protection is defined at the individual block level.

Secure password protection can also be implemented for the debug port.

The SPC58xEx/SPC58xGx microcontroller also includes a tamper detection feature. In its most basic implementation, tamper detection acts as an erase counter. If tamper detection is implemented, a DCF record in the tamper detection *diary* must be written by software before a protected Flash block can be erased. Multiple tamper detection regions can be defined by the customer, with each region containing one or more blocks. A block can be included in more than one tamper detection region if desired.

Another protection feature related to tamper detection is the ability to configure any Flash block as one-time programmable (OTP).

Finally, the microcontroller includes a feature that allows the customer to disable the ability to place the device in factory test mode.

The remainder of the chapter provides details regarding these protection mechanisms and gives a brief overview of security configuration planning.

Note: Most of the topics in this chapter build on information presented in previous topics. To get a clear overview of how the Flash protection features work and relate to each other, please read the chapter in sequence from beginning to end.

Access protection configuration is applied on a per-block basis (except read protection and tamper detection applied to groups of Flash memory blocks), which means that a custom level of access protection can be provided for each Flash memory block. This is accomplished through the use of registers containing bits that are mapped to individual Flash blocks. Understanding the mapping is key to understanding the implementation and use of the SPC58xEx/SPC58xGx microcontroller's Flash protection features.

Depending on the type of protection to be applied, different sets of registers (or even DCF records) are used, but the mapping scheme is consistent. This mapping scheme is first presented in *Section 38.1: Selection of Flash memory blocks for erase*. It is important to understand the topic before proceeding to other sections.

38.1 Selection of Flash memory blocks for erase

The registers used for selecting Flash blocks for erase operations are the Flash control registers named LOCK0, LOCK1, LOCK2 and LOCK3.

The LOCK0–2 registers are 32-bit registers containing fields associated with specific address spaces: low, mid, high, and 256 KB. Each non-reserved bit in the fields is mapped to a specific Flash block.

For example, *Section 37.3.2.10: Select 0 register (SEL0)* shows the Select 0 (SEL0) register for the Flash module. This register provides a way of selecting blocks in the low and mid address spaces for erase operations.

Table 516: Flash memory partition 0 memory map shows an example of a mapping of SEL0 register's LOWSEL and MIDSEL field bits to individual Flash blocks. In this case, not all bits of the LOWSEL and MIDSEL fields are mapped—only LOWSEL bits 0–11 (register bits 15–4) and MIDSEL bits 0–2 are mapped. Before any mapped block can be erased, it must be

selected by writing a ‘1’ to its corresponding bit in one of the SEL*n* registers. Refer to the device configuration chapter for the complete mapping.

Note: Values written to the SEL registers do not persist through device reset. As indicated by the reset values shown in Figure 419: Select 0 register (SEL0), all bits are reset to ‘0’. As a result, after a device reset all blocks mapped to the register are, by default, not selected for erase.

Example 1 Selecting blocks for erase

Writing the value 0x0005_0000 to the SEL0 register (*Figure 419: Select 0 register (SEL0)*) selects the blocks in *Table 535: SEL0 field descriptions* for erase.

38.2 Non-secure write protection

Non-secure write protection refers to the chip’s ability to lock individual Flash blocks against programming and erase operations. This level of protection is intended to provide protection against *accidental* changes to code and data. Although similar in function, this is *not* associated with the built-in register protection (REG_PROT) feature included with many of the chip’s modules. The non-secure write protection feature detailed here applies only to embedded Flash memory.

Implementing non-secure write protection is almost identical to selecting specific Flash blocks for an erase operation—the only difference is the register used. To provide non-secure write protection for a block, write a ‘1’ to the corresponding mapped bit found in one of the LOCK0–3 Flash control registers.

Note: As indicated by the reset values shown in Section 37.3.2.4: Lock 0 register (LOCK0), after reset all blocks mapped to the register are, by default locked against programming and erase. Values written to the LOCK and SEL registers do not persist through device reset.

The same bit mapping that applies to the SEL0–3 registers also applies to the LOCK0–3 registers.

Example 2 Selecting blocks for write protection

Writing the value 0x0005_0000 (same value used in *Example 1*) to the LOCK0 register (*Section 37.3.2.4: Lock 0 register (LOCK0)*) locks the blocks in *Table 516* (same blocks selected in *Example 1*) so they cannot be programmed or erased, even if selected for erase.

As long as the LOCK0[LOWLOCK] bits are unchanged, the above blocks cannot be modified unless the protection is overridden (discussed in *Section 38.3: Secure write protection*). All other blocks mapped to bits in the LOCK0 *can* be programmed because their corresponding bits have been set to ‘0’.

38.3 Secure write protection

Secure write protection refers to the chip’s ability to implement up to four levels of 256-bit password-secured *write* protection for individual Flash blocks. Use of this feature is optional but the default behavior of the SPC58xEx/SPC58xGx microcontroller is to apply secure write protection. If there is a need to *not* write-protect specific blocks, that needs to be configured by creating the appropriate DCF records. For details, please refer to *Section 92.5.2.1: JTAG Password Challenge* in the PASS chapter.

Note that the *secure* write protection settings override the non-secure write protection detailed in *Section 38.2: Non-secure write protection*.

Implementing a custom secure write protection scheme requires creating DCF records that define the level of write protection to be applied to each Flash block. The DCF records use a bit mapping scheme identical to the mapping in the Flash module's SEL*n* and LOCK*n* registers. The write protection scheme becomes a permanent part of the device but blocks can be temporarily unlocked. Depending on the specific implementation, temporarily unlocking the write protection requires matching up to four 256-bit passwords.

The easiest way to understand the secure write protection functionality is to examine the role played by the PASS_LOCK0_PG*n*–PASS_LOCK3_PG*n* registers.

The PASS_LOCK0_PG*n*–PASS_LOCK3_PG*n* registers are four identical sets of registers contained in the PASS module—they distinguished by the PG*n* suffix. The PG*n* (Password Group *n*) suffix indicates the password required to unlock that set of registers. Password 0 unlocks PASS_LOCK0_PG0, PASS_LOCK1_PG0, PASS_LOCK2_PG0, and PASS_LOCK3_PG0. Similarly, the registers suffixed with PG1 are unlocked by matching password 1, and so on.

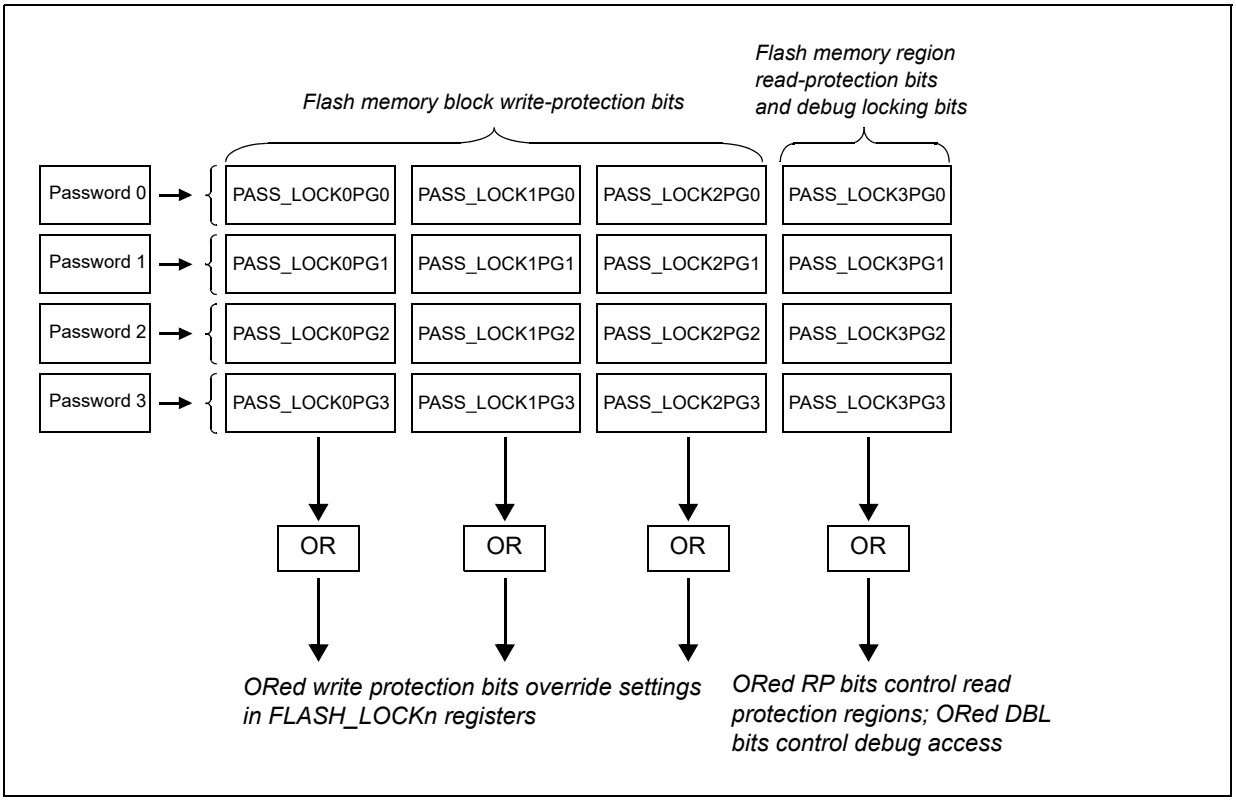
The PASS_LOCK0_PG*n*–PASS_LOCK3_PG*n* register sets have two functions related to secure write protection:

- Reading the registers provides the secure write lock status for each Flash block.
- Changing the value of a mapped bit in a register changes the secure write protect status of the corresponding Flash block.

Each set of registers contains the same mapping of bits to Flash blocks as the Flash module's SEL*n* and LOCK*n* registers. In any of the four PASS_LOCK*x*_PG*n* register sets, if a mapped bit is set to '1', locking a Flash block, that value can only be changed by successfully supplying the password value for that set of registers to gain write access to the register, and setting the bit to '0'. If the same bit is set to '1' in more than one PASS_LOCK*x*_PG*n* register set, each must be unlocked using the corresponding password and the bit must be set to '0' before the Flash block is available for programming or erase.

Figure 462 shows how the registers interact. Note that secure write protection is only affected by the settings in PASS_LOCK0_PG*n*–PASS_LOCK2_PG*n*. The PASS_LOCK3_PG*n* registers have additional security functions, discussed in later topics.

Figure 462. Secure write protection



The next sections discuss implementing secure write protection and overriding secure write protection.

38.3.1 Implementing secure write protection

Implementing secure write protection is a one-time task performed either at the factory or by the customer during chip configuration activities. It normally requires creating a DCF record for each of the PASS_LOCK0_PG*n*–PASS_LOCK2_PG*n* registers[16]. The DCF records have a structure that includes the same mapping of bits to Flash memory blocks as the Flash module's SEL*n* and LOCK*n* registers. The values assigned become the permanent register reset values for the PASS_LOCK0_PG*n*–PASS_LOCK2_PG*n* registers.

Refer to *Chapter 9: Device Configuration Format (DCF) Records* for complete details on DCF records. This section summarizes the contents of a DCF record as required to implement secure write protection.

Even though the mapping of bits to Flash memory blocks is identical to the mapping used for selecting Flash blocks for erase of non-secure write locking, there are two significant differences:

1. Configuring Flash blocks for secure write protection is done only during device configuration to customer specifications, either at the factory or by the customer during

16. The default behavior for secure write protection in the case where passwords are created but no secure write protection DCF records are created, is for all blocks to have four levels of password-secured write protection. To vary the protection among all blocks, a DCF record for each password group must be created.

OEM production activities. The selection becomes a permanent part of the device configuration.

2. Configuration of blocks for secure write protection is done by writing *DCF records*, which is why the selection becomes a permanent part of the device.

Caution: Careful planning is required before writing DCF records. They are created in a write-once area of Flash. Multiple instances of any DCF record can be created, but in some cases only the record created first is used and with others only the most-recently written record is used. Be sure to understand the characteristics of a DCF record before creating it.

The DCF records required for implementing a secure write protection scheme are detailed in the PASS section of DCF client list table (*Table 129: DCF client list* in *Chapter 9: Device Configuration Format (DCF) Records*). Table entries provide the CS and Address value combinations that correspond to PASS_LOCK0_PG*n*–PASS_LOCK3_PG*n* registers.

The format of a secure write protection record is summarized here for convenience.

DCF records appear as contiguous double-word (64-bit) entries programmed in a reserved area of OTP UTEST Flash memory. The structure of a DCF Record is shown in *Figure 463*.

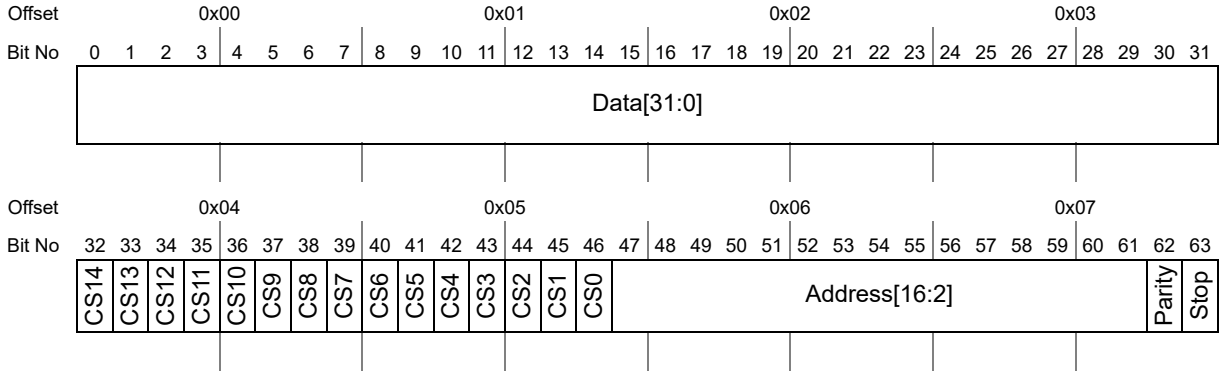


Figure 463. DCF record structure

Note: The values in the PASS_LOCKxPGn registers should mirror the settings in the Flash module's LOCK0–3 registers.

Values written to the PASS_LOCKxPGn registers do not persist through device reset. The reset value is determined by DCF record or, if no relevant DCF record has been created, each mapped bit has a reset value of '1', which causes the associated Flash block to be locked against programming or erase.

The PASS_LOCKxPGn bits take effect when the life cycle is matured to "OEM Production" or older.

Example 3 Implementing secure write protection

Implementation of secure write protection is done during initial Flash setup as follows:

1. Four 256-bit passwords (passwords 0–3) must be stored in a protected OTP area of device Flash called UTEST Flash. Each password secures a set of registers in the PASS module (PASS_LOCK0_PG*n*–PASS_LOCK3_PG*n*) that have the same mapping of register bits to Flash blocks as the Flash module's SEL0–3 registers and LOCK0–3 registers.
2. DCF records must be created to define which Flash memory blocks are to be locked and which blocks are not locked.

The records defined in step 2., determine the reset values of the PASS_LOCK0_PG*n*–PASS_LOCK3_PG*n* registers and therefore the write protection settings that take effect each time the microcontroller is reset.

Assume four passwords have been previously implemented and the following blocks are to be locked in the password 0 group: refer to *Table 516*.

From the PASS section of *Table 129: DCF client list* in *Chapter 9: Device Configuration Format (DCF) Records*:

Table 584. DCF client list of PASS

DCF CS[14:0]	DCF address [16:2] (binary)	DCF client description
000_0000_0000_1000	000_0000_0100_0000	LOCK0_PG0

Referring to *Figure 463*, the values for the DCF record are:

- Data[31:0] is 0x0005_0000 (bits 0–31) (same value used in previous examples)
- CS[14:0] is 0b000_0000_0000_1000 (bits 32–46)
- Address[16:2] is 0b000_0000_0100_0000 (bits 47–61)
- Parity value is irrelevant because it is not used in PASS DCF records (bit 62)
- Preferred Stop value is 0 (refer to footnotes in *Figure 464*).

The following DCF record provides secure write protection for the blocks by setting the reset values PASS_LOCK0_PG0 to the appropriate value to lock the blocks.

Bit No	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bit No	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63
	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	X (1)	X (2)

Figure 464. Sample DCF record for secure write protection

1. Bit 62 indicates parity strategy, which is not used in the PASS DCF records so it is a “don’t care”.
2. Bit 63 indicates whether the current record is the last DCF. Generally, this should be written as ‘0’ so the first unprogrammed (all 1’s) DCF record acts as the stop record.

The remaining blocks mapped to the PASS_LOCK0_PG*n* registers do *not* have secure write protection provided by the password 0 register set because the DATA field of DCF record has only the bits mapped to Flash blocks 4 and 5 set to ‘1’.

Caution: The above example is intended to highlight the need to carefully understand how DCF records work. The protected blocks are secured by password 0 but by default they are also secured by passwords 1, 2, and 3. In contrast the blocks *not* secured by password 0 in this example *are* secured by passwords 1, 2, and 3.

Refer to the PASS section of DCF client list table (*Table 129: DCF client list* in *Chapter 9: Device Configuration Format (DCF) Records*) for details on the DCF records.

38.3.2 Overriding secure write protection

The “PG*n*” (Password Group *n*) suffix on each the register names indicates the password each set of registers is associated with. For example, password 0, is used to unlock

PASS_LOCK0_PG0, PASS_LOCK1_PG0, PASS_LOCK2_PG0, and PASS_LOCK3_PG0. The bits in the first three registers (PASS_LOCK0_PG0, PASS_LOCK1_PG0, and PASS_LOCK2_PG0) are mapped exactly the same as the bits in the Flash module's SEL0–2 registers and LOCK0–2 registers.

Like the SEL0 and LOCK0 registers, the PASS_LOCK0_PG*n* registers all contain bits mapped to the low and mid address spaces, and the mapping scheme is the same. A mapped bit set to '1' in any of the registers causes the corresponding block to be locked against program or erase. A mapped bit set to '0' in any of the registers makes the corresponding block available for programming or erase.

If a bit is set to '1', locking a Flash block, that value can only be changed by successfully supplying the password value for that set of registers to gain write access, and setting the bit to '0'. If the same bit is set to '1' in more than one of the PASS_LOCK*x*PG*n* register sets, each must be unlocked using the corresponding password and the bit must be set to '0' before the Flash block is available for programming or erase.

Figure 465 shows the PASS_LOCK0_PG*n* register. It has the same structure as the Flash module's LOCK0 register. In addition to the main difference of being among a group of four parallel sets of locking registers, the other difference between the PASS_LOCK*x*_PG*n* registers and the Flash LOCK*n* registers is how the reset values are determined.

By default, all mapped bits are set to '1', locking their corresponding Flash blocks from programming and erase. The reset value is user-configurable, though, by writing a DCF record. This is discussed in a later section.

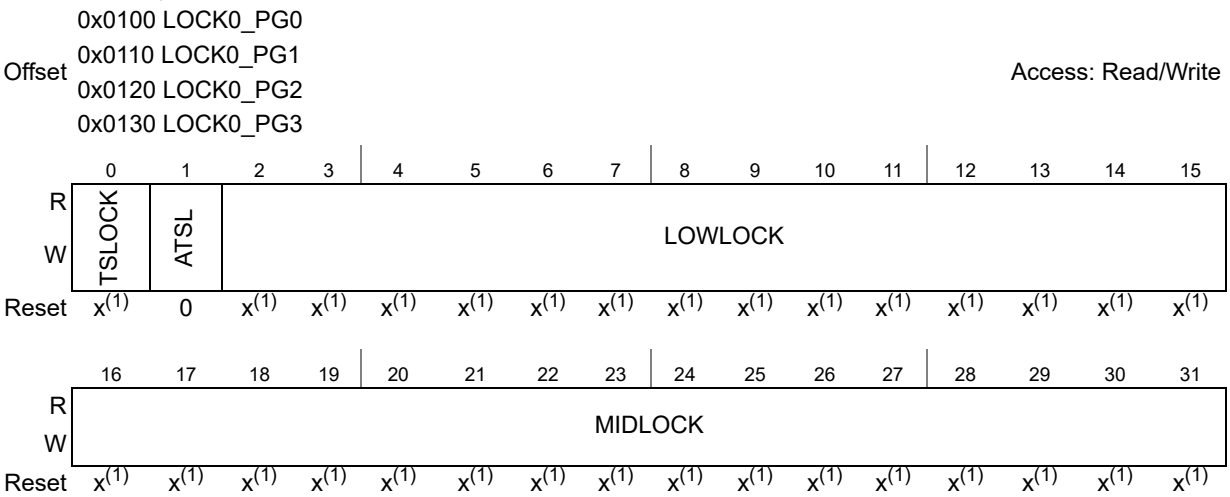


Figure 465. LOCK0_PG*n* register

1. If no DCF records have been written to change the lock state, the bit will reset to '1'

Note: The values in the PASS_LOCKxPGn registers should mirror the settings in the Flash module's LOCK0–3 registers.

Values written to the PASS_LOCKxPGn registers do not persist through device reset. The reset value is determined by DCF record or, if no relevant DCF record has been created, each mapped bit has a reset value of '1', which causes the associated Flash block to be

locked against programming or erase. The purpose of the registers is to indicate secure write protection status and to provide a way to temporarily override secure write protection.

The PASS_LOCKxPGn bits take effect after the life cycle is matured to "OEM Production" or older.

Example 4 Overriding secure write protection

Assume four passwords have been created but no DCF records have been written to set the reset values of the PASS_LOCK*x*_PG*n* registers, so the reset value for each mapped bit defaults to '1', causing all mapped blocks to have four levels of 256-bit password-secured write protection.

Since the settings in the PASS_LOCK*x*_PG*n* registers override the settings in the Flash module's LOCK0–3 registers, the settings in those registers are irrelevant. Each of the four sets of the PASS module's LOCK registers must be unlocked by supplying the correct password for each set and then the appropriate bits must be set to '0' to make the desired Flash blocks available for program or erase.

1. First write 0b00 to the PASS_CHSEL[GRP] register field to indicate registers controlled by password 0 are to be unlocked.
2. Write the 256-bit value for password 0 to the set of eight 32-bit password challenge input registers (PASS_CIN*n*) beginning with PASS_CIN0. The password must be written as a sequence of eight 32-bit writes, with the most significant 32 bits of the password written to PASS CIN0.
3. Write 0xFFCF_FFFF to the PASS_LOCK0_PG0 register and verify a successful write.
4. Repeat the above steps for passwords 1–3.

As in the previous examples, the result is the following blocks are available for program and erase operations but at a much higher level of security.

38.4 Secure read protection

Read Protection is ability of the Flash controller to block read accesses to the Flash memory from all bus masters. This includes the blocking of:

- Instruction fetches and load instructions from the CPU
- Debug reads from an outside tool
- DMA reads
- SIPI reads
- All other types of bus master reads

When secure read protection is active, the affected Flash blocks are readable by any bus master until a debugger is attached to the JTAG port and debug is enabled. If a debugger is attached and debug is enabled while an application is running, the next access to a Flash read protected region terminates with a bus error.

Most instances of Flash memory read protection in the SPC58xEx/SPC58xGx microcontroller are *not* configurable by the customer. An example of this is the 256-bit passwords stored in UTEST Flash memory. Password-based read protection *is* customer-configurable(17) and is similar to the password-based write protection mechanism detailed in *Section 38.3: Secure write protection*.

There are two significant differences between password-based read protection and password-based write protection in the SPC58xEx/SPC58xGx microcontroller.

1. Read protection can only be applied to pre-defined groups of Flash memory blocks. These groups, called “Read Lock groups”, cannot be re-defined, meaning the blocks assigned to those groups are permanently assigned and cannot be assigned to other groups and no new groups can be created.
2. Read protection is only active either when a debugger is attached to the JTAG port and debug is enabled; or the lifecycle has been advanced to “Failure Analysis”.

Another difference is that only the PASS_LOCK3_PG*n* register for each password group is used:

- PASS_LOCK3_PG0
- PASS_LOCK3_PG1
- PASS_LOCK3_PG2
- PASS_LOCK3_PG3

Note: The mappings of PASS_LOCK3_PGn register bits to Flash blocks vary with different microcontrollers. Refer to the Flash memory section in the Device Configuration chapter of this reference manual for the mappings specific to the SPC58xEx/SPC58xGx microcontroller.

The next sections discuss implementing secure read protection and overriding secure write protection.

38.4.1 Implementing secure read protection

Implementation of secure read protection is done during initial Flash setup as follows:

1. Four 256-bit passwords (passwords 0–3) are stored in a protected OTP area of device Flash named UTEST Flash. Each password secures a set of registers in the PASS module (PASS_LOCK0_PG*n*–PASS_LOCK3_PG*n*) that have the same mapping of register bits to Flash blocks as the Flash module’s SEL0–3 registers and LOCK0–3 registers. These are the same passwords used by secure write protection and other secure functions. The passwords are only created once.
2. DCF records must be created to define which Flash memory blocks are to be locked. Those records determine the reset values of the PASS_LOCK3_PG*n* registers.

   Like the other PASS_LOCK*x*_PG*n* registers, the PASS_LOCK3_PG*n* registers have two functions.
   - Reading the registers provides a lock status for each Read Locking group.
   - Changing the value of a mapped bit in a register changes the read protect status of the corresponding Read Locking group.

The layout of the PASS_LOCK3_PG*n* registers is shown in *Figure 466*. The fields relevant to implementing secure read protection are RL4–RL0.

17. Secure read protection is user-configurable in that the default lock/unlock state for each mapped block at reset is controlled by user-created DCF records, giving the user the ability to determine the number of levels of password protection implemented. The read protection is applied to groups of blocks instead of individual blocks and the groups are not user-definable.

Offset
0x010C LOCK3_PG0
0x011C LOCK3_PG1
0x012C LOCK3_PG2
0x013C LOCK3_PG3

Access: Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	PGL	DBL	MO	0	MSTR				0	0	0	RL4	RL3	RL2	RL1	RL0
W																
Reset	x(1)	x(2)	x(2)	0	1	1	1	1	0	0	0	x(2)	x(2)	x(2)	x(2)	x(2)

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	256LCK_U															
W																
Reset	x(2)	x(2)	x(2)	x(2)	x(2)	x(2)	x(2)	x(2)	x(2)	x(2)	x(2)	x(2)	x(2)	x(2)	x(2)	x(2)

Figure 466. PASS_LOCK3_PG*n* Register

1. Reset value depends on life cycle. If life cycle is Customer delivery or earlier, the value will be 0, otherwise 1.
2. If no DCF records have been written to change the lock state, the bit will reset to '1'.

Example 5 Implementing secure read protection

Assume four passwords have been previously implemented.

Following is an example of a mapping of Flash blocks to Read Locking groups.

To provide two levels of password protection for only Read Lock group 3 and leave all other groups unlocked, the RL3 bit can be set in any two of the four PASS_LOCK3_PG*n* registers. For this example, we will choose password groups 0 and 1.

We need to create DCF records that define the reset values for all RL*n* bits as 0, with the exception of PASS_LOCK3_PG0 and PASS_LOCK3_PG1. The information required is from the PASS section of *Table 129: DCF client list* in *Chapter 9: Device Configuration Format (DCF) Records*.

Table 585. DCF client list of PASS

DCF CS[14:0]	DCF address [16:2] (binary)	DCF client description
000_0000_0000_1000	000_0000_0100_0011	LOCK3_PG0
000_0000_0000_1000	000_0000_0100_0111	LOCK3_PG1
000_0000_0000_1000	000_0000_0100_1011	LOCK3_PG2
000_0000_0000_1000	000_0000_0100_1111	LOCK3_PG3

Referring to *Figure 463*, the values for the DCF record are:

- For LOCK3_PG0
  - Data[31:0] is 0x0008_0000 (bits 0–31)
  - CS[14:0] is 0b000_0000_0000_1000 (bits 32–46)
  - Address[16:2] is 0b000_0000_0100_0011 (bits 47–61)
  - Parity value is irrelevant because it is not used in PASS DCF records (bit 62)
  - Preferred Stop value is 0.
- For LOCK3_PG1
  - Data[31:0] is 0x0008_0000 (bits 0–31)
  - CS[14:0] is 0b000_0000_0000_1000 (bits 32–46)
  - Address[16:2] is 0b000_0000_0100_0111 (bits 47–61)
  - Parity value is irrelevant because it is not used in PASS DCF records (bit 62)
  - Preferred Stop value is 0.
- For LOCK3_PG2
  - Data[31:0] is 0x0000_0000 (bits 0–31)
  - CS[14:0] is 0b000_0000_0000_1000 (bits 32–46)
  - Address[16:2] is 0b000_0000_0100_1011 (bits 47–61)
  - Parity value is irrelevant because it is not used in PASS DCF records (bit 62)
  - Preferred Stop value is 0.
- For LOCK3_PG3
  - Data[31:0] is 0x0000_0000 (bits 0–31)
  - CS[14:0] is 0b000_0000_0000_1000 (bits 32–46)
  - Address[16:2] is 0b000_0000_0100_1111 (bits 47–61)
  - Parity value is irrelevant because it is not used in PASS DCF records (bit 62)
  - Preferred Stop value is 0.

38.4.2 Overriding secure read protection

As noted previously, the PASS_LOCK3_PG*n* registers (refer to *Figure 466*) contain five read protection fields, RP0–RP4, that correspond to pre-defined Read Lock groups. To temporarily unlock a Read Lock group, the appropriate PASS_LOCK3_PG*n*[RL*n*] bits must be set to 0.

Like the register bits controlling secure write protection, the reset value of bits controlling secure read protection is determined by DCF record or, in the absence of a relevant DCF record, each bit resets to ‘1’, causing the blocks associated with each read group to be read locked.

Example 6 Overriding secure read protection

Building on *Example 5: Implementing secure read protection*, where Read Lock group 3 was protected by passwords 0 and 1, temporarily overriding secure read protection for that group is accomplished by first unlocking the PASS_LOCK3_PG0 and PASS_LOCK3_PG1 registers and then setting the RL3 bit to ‘0’ in both.

1. First write 0b00 to the PASS_CHSEL[GRP] register field to indicate registers controlled by password 0 are to be unlocked.
2. Write the 256-bit value for password 0 to the set of eight 32-bit password challenge input registers (PASS_CIN*n*) beginning with PASS_CIN0. The password must be written as a sequence of eight 32-bit writes, with the most significant bits of the password written to PASS CIN0.
3. Write a 0 to PASS_LOCK3_PG0[RL3] bit and verify a successful write.
4. Repeat the above steps for password 1.

The result of a successful sequence is that the blocks associated with the unlocked read lock group can be read by a debugger.

38.5 Debug port enable/disable

The mechanism for temporarily enabling the SPC58xEx/SPC58xGx debug port after it has been locked by its life cycle state is similar to the Flash memory read protection mechanism detailed in *Section 38.4: Secure read protection*. The difference is that the PASS_LOCK3_PG*n*[DBL] field is used instead of the RP0–RP4 fields.

38.6 Tamper detection

The Tamper Detection Module (TDM) provides a type of Flash memory write protection mechanism that forces software to write a record associated with one or more blocks in a Tamper Detection Region (TDR) before the block(s) can be erased. An additional feature is provided to enable customers to override the protection mechanism for any TDR so that no record is required to be written before a block within the TDR can be erased.

This section provides details on the mapping between DCF record bits and individual Flash memory blocks to be selected for tamper detection.

Refer to *Chapter 93: Tamper Detection Module (TDM)*, for details on tamper detection, including the DCF clients and two additional memory-mapped registers.

There are two significant differences between selecting blocks for tamper detection and selecting blocks for other activities, such as a Flash erase operation:

1. Selection of blocks for tamper detection protection is done only during device configuration to customer specifications, either at the factory or by the customer during OEM production activities. The selection becomes a permanent part of the device configuration.
2. The selection of blocks for tamper detection is done by writing *DCF records*, which is why the selection becomes a permanent part of the device.

The Tamper Detect section of DCF client list table (*Table 129: DCF client list* in *Chapter 9: Device Configuration Format (DCF) Records*) gives the CS and Address value combinations that correspond to groups of LOCK0–LOCK3 registers, similar to the Flash module's LOCK0–LOCK3 registers.

The mapping of individual Flash blocks to TDM DCF record data field bits is shown included in the detailed Flash block map in the Flash section of the Device Configuration chapter.

38.6.1 Implementing tamper detection

Implementing tamper detection requires:

1. Creating the tamper detect diary
2. Assigning blocks to Tamper Detection Regions (TDRs)

The tasks are detailed in the following sections.

38.6.2 Creating the tamper detect diary

Tamper detection in the SPC58xEx/SPC58xGx microcontroller is user-defined. The basic functionality provided is an erase counter, or diary, for 6 user-defined regions, called “tamper detect regions”, or TDRs, within the Flash memory. Each TDR is defined using a set of DCF records that is similar in function and layout to the Flash module’s SEL*n* and LOCK*n* registers, which contain bits matched to specific Flash blocks.

For a program or erase operation to be performed on a block included in a TDR, a record must be written to a “diary” that tracks Flash modification operations. The format and size of records is customer-defined. The only hardware constraints are that each diary has a maximum size of 4 KB and the minimum size of any programming operation depends on the Flash sector, 8 bytes for Data sector and 16 bytes for the remaining sector.

Note: The limiting factor for tamper detection records is the diary size. The total diary size is given by the number of TDRs (6 for SPC58xEx/SPC58xGx) multiplied by 4 KB, the portion of the diary allotted to records associated with each TDR.

The diary can be placed within any Flash block in the Flash array but to maintain the security of the TDR, there are two constraints:

- The block where the diary is placed must be assigned as OTP within the OTP registers (covered in the next section).
- The diary base address must be on a 4 KB boundary, therefore the least significant 12 bits of the base address must all be ‘0’.

Caution: The tamper detect diary base address DCF record is a “write once” record. All subsequent tamper detect diary base address DCF records created are ignored.

Example 7 Creating the tamper detect diary

If the total space required for a diary is 24 KB (as in the case of 6 TDRs), any Flash block in the SPC58xEx/SPC58xGx microcontroller but B0F0 and B0F1 whose size is not enough to allocate the diary total space can be used. Since the diary base address must be on a 4 KB, boundary, though, if the diary base address is within a 32 KB Flash block, it can be assigned to the base address of the block or the address given by the base address plus 4 KB or the address given by the base address plus 8 KB.

If the base address of the 32 KB block is 0x00FD000, the base address of the diary is given by:

32’b0000_0000_1111_1101_0000_0000_0000_0000 (32 KB base address)

or

32’b0000_0000_1111_1101_0001_0000_0000_0000 (32 KB base address + 4 KB)

or

32’b0000_0000_1111_1101_0010_0000_0000_0000 (32 KB base address + 8 KB)

The base address of each TDR's section of the diary is:

- TDR0 = Diary_Base_Address + 16'b0000_0000_0000_0000
- TDR1 = Diary_Base_Address + 16'b0001_0000_0000_0000
- TDR2 = Diary_Base_Address + 16'b0010_0000_0000_0000
- TDR3 = Diary_Base_Address + 16'b0011_0000_0000_0000
- TDR4 = Diary_Base_Address + 16'b0100_0000_0000_0000
- TDR5 = Diary_Base_Address + 16'b0101_0000_0000_0000
- End of the diary = Diary_Base_Address + 16'b0110_0000_0000_0000

We need to create one DCF record that defines the base address for the diary. The information required is from the Tamper Detect section of *Table 129: DCF client list* in *Chapter 9: Device Configuration Format (DCF) Records*.

Table 586. DCF client list of Tamper Detect

DCF CS[14:0]	DCF address [16:2] (binary)	DCF client description
000_0000_0001_0000	000_0000_0000_0000	Diary Base Address

Referring to *Figure 463*, the values for the DCF record are:

- For LOCK0_PG0
  - Data[31:0] is 0x00FC4000 (the base address).
  - CS[14:0] is 0b000_0000_0001_0000 (bits 32–46).
  - Address[16:2] is 0b000_0000_0000_0000 (bits 47–61).
  - Parity value is irrelevant because it is not used in Tamper Detect DCF records (bit 62).
  - Preferred Stop value is 0.

Refer to the Tamper Detect section of DCF client list table (*Table 129: DCF client list* in *Chapter 9: Device Configuration Format (DCF) Records*) for details.

Note: User software must check whether the diary is full. If the diary area for a TDR is full, user software must program a DCF record to override blocking of erase from TDR, disabling tamper detect, so that erases can be done to that TDR without any diary entry. This is detailed in Section 38.6.4: Overriding tamper detection.

38.6.3 Assigning blocks to Tamper Detection Regions (TDRs)

Tamper detection regions are defined by creating TDR*x*_LOCK*n* DCF records. For example, in a microcontroller with six tamper detection regions, the following records define the TDRs:

- TDR0_LOCK0, TDR0_LOCK1, TDR0_LOCK2, TDR0_LOCK3
- TDR1_LOCK0, TDR1_LOCK1, TDR1_LOCK2, TDR1_LOCK3
- TDR2_LOCK0, TDR2_LOCK1, TDR2_LOCK2, TDR2_LOCK3
- TDR3_LOCK0, TDR3_LOCK1, TDR3_LOCK2, TDR3_LOCK3
- TDR4_LOCK0, TDR4_LOCK1, TDR4_LOCK2, TDR4_LOCK3
- TDR5_LOCK0, TDR5_LOCK1, TDR5_LOCK2, TDR5_LOCK3

Each set of TDR*x*LOCK*n* DCF records mapped in exactly the same way as the Flash module's SEL*n* and LOCK*n* registers (refer to the detailed Flash block map in the Flash section of the Device Configuration chapter for the mapping).

In the TDR*x*LOCK*n* DCF records:

- A bit set to '1' means the corresponding Flash block is assigned to this TDR.
- A bit set to '0' means the corresponding Flash block is NOT assigned to this TDR.

The default state (no TDRs defined) is defined as "Block NOT assigned to the TDR".

38.6.4 Overriding tamper detection

When a TDR's diary section is full, a Tamper Override (TO) record must be created before a block assigned to the TDR can be programmed or erased.

The details of the data section of the DCF record used to override tamper detection is shown in *Figure 467*.

Caution: Bits in this DCF record are writable from 0 to 1 only. Attempted writes of bits from 1 to 0 are ignored.

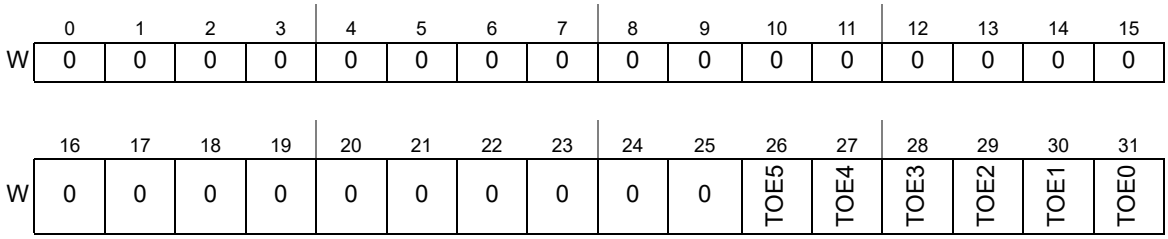


Figure 467. Data portion of Tamper Region Override (TO) DCF record

Example 8 Implementing secure write protection

Assume the diary Tamper Detection Region 3 is full.

We need to create one DCF record that defines that overrides tamper detection for the tamper detection region whose diary is full. The information required is from the Tamper Detect section of *Table 129: DCF client list* in *Chapter 9: Device Configuration Format (DCF) Records*.

Table 587. DCF client list of Tamper Detect

DCF CS[14:0]	DCF address [16:2] (binary)	DCF client description
000_0000_0001_0000	000_0000_0000_0001	Tamper Region Override

Referring to *Figure 463*, the values for the DCF record are:

- Data[31:0] is 0x0000_0008 (TOE3).
- CS[14:0] is 0b000_0000_0001_0000 (bits 32–46).
- Address[16:2] is 0b000_0000_0000_0001 (bits 47–61).
- Parity value is irrelevant because it is not used in Tamper Detect DCF records (bit 62)
- Preferred Stop value is 0.

Refer to the Tamper Detect section of DCF client list table (*Table 129: DCF client list* in *Chapter 9: Device Configuration Format (DCF) Records*) for details.

38.7 Implementing OTP

Any Flash block within the Flash array can be assigned, at any time, to be OTP. OTP means that Flash erase of the entire block is disabled and only 128-bit quad-words that are already erased (Flash content is 0xFFFF_FFFF_FFFF_FFFF) can be programmed. Over-programming is not possible.

Flash blocks are assigned as OTP by writing a DCF record. The block becomes OTP after the next reset.

Caution: After a Flash block is configured as OTP, it cannot be changed back.

This capability resides within the Tamper Detection Module (TDM). The records are detailed in *Table 129: DCF client list* in *Chapter 9: Device Configuration Format (DCF) Records*. Refer to the entries for the OTP_EN0, OTP_EN1, OTP_EN2, and OTP_EN3 DCF records in the Tamper Detect section of the table.

38.8 Implementing test mode disable

The SPC58xEx/SPC58xGx Flash memory module includes a mechanism to disable manufacturer entry into test mode. Extreme care must be taken when using this feature, as blocks that are selected to be protected in this method are able to have possible failures analyzed by manufacturer's failure analysts.

Test mode disable prevents all high voltage operations to the Flash executed by the internal state-machine, as well as reads through the state machine, and reads through the Array Integrity state machine when using test mode interfaces.

Optionally, the customer can create a password to enable manufacturer entry into test mode.

38.8.1 Unconditional test mode disable seal

Warning: This process is not reversible. After completing these steps the SPC58xEx/SPC58xGx microcontroller is permanently prevented from entering test mode for factory analysis.

The steps for unconditionally disabling test mode are the same steps required for implementing passcode-protected entry into test mode with one exception—the passcode specified must be one of the following three *invalid* passcodes:

- 0x0000_0000
- 0xFFFF_FFFF
- 0x5555_5555

To unconditionally disable test mode entry:

1. Select individual Flash blocks to be protected by programming the appropriate values into the Test Mode Disable Block Select Group A and Test Mode Disable Block Select Group B areas shown in the UTEST Flash memory map.
2. Program one of the three *invalid* passcodes shown above into the Test Mode Override Passcode area shown in the UTEST Flash memory map.
3. Program the value 0x5A4B_3C2D into the Test Mode Disable Seal location specified in the UTEST Flash memory map.

After the next reset is asserted, Test Mode is permanently disabled.

Note: Blocks selected for test mode disable can be selected in either or both Test Mode Disable Block Select Groups (A and B) because those fields are logically ORed to determine the blocks to be protected.

Note: The lifecycle state must be programmed to 'Production at OEM' or later for the disable seal to be in effect.

38.8.2 Passcode-protected test mode disable seal

The steps for implementing passcode-protected entry into test mode are the same required for unconditionally disabling test mode with the exception that —the passcode specified must NOT be one of the following three *invalid* passcodes:

- 0x0000_0000
- 0xFFFF_FFFF
- 0x5555_5555

To implement passcode-protected test mode entry:

1. Select individual Flash blocks to be protected by programming the appropriate values into the Test Mode Disable Block Select Group A and Test Mode Disable Block Select Group B areas shown in the UTEST Flash memory map.
2. Program a valid passcode into the Test Mode Override Passcode area shown in the UTEST Flash memory map.
3. Program the value 0x5A4B_3C2D into the Test Mode Disable Seal location specified in the UTEST Flash memory map.

After the next reset is asserted, Test Mode is permanently passcode protected.

Note: Blocks selected for test mode disable can be selected in either or both Test Mode Disable Block Select Groups (A and B) because those fields are logically ORed to determine the blocks to be protected.

Note: The lifecycle state must be programmed to 'Production at OEM' or later for the disable seal to be in effect.

38.8.3 Selecting Flash memory blocks for test mode disable seal

Selecting Flash memory blocks for test mode disable seal protection is similar to selecting blocks for tamper detection in that it involves writing to non-register Flash locations. In the case of the test mode disable seal function, there are two areas of UTEST Flash memory that have identical mapping to Flash memory blocks. When the test mode disable seal takes effect, the mapped bits are logically ORed to determine the blocks to be protected.

The mapping of bits to blocks for this function is identical to the mapping used for other functions. Refer to *Chapter 37: Embedded Flash Memory* for the map.

Note: Selecting a block for test mode disable requires a '0' in the mapped bit—not a '1' as with other functions.

38.9 Security configuration planning

At minimum, designing the Flash security framework consists of selecting the security features to be implemented and determining the scope of implementation for each selected feature.

In devices containing a Hardware Security Module (HSM), significant planning must go into the implementation of HSM functions. The HSM is an independent software platform within the microcontroller. It is intended to be used to implement advanced security functions that are defined by customer software.

Note: Details of advanced security features, including the advanced security architecture is and the Hardware Security Module (HSM) are contained in the SPC58xEx/SPC58xGx Microcontroller Security Reference Manual. Distribution of the document is restricted to qualified customers.

The basic security features available in the SPC58xEx/SPC58xGx microcontroller include:

- Secure write protection
- Secure read protection
- Debug port enable/disable
- One-time-programmable (OTP) Flash memory block assignment
- Factory test mode disable

38.9.1 Hardware Security Module (HSM)

Details of the HSM are included in the *SPC58xEx/SPC58xGx Microcontroller Security Reference Manual*. It is mentioned here because the Flash memory module includes blocks that can be configured for exclusive access by the HSM. If the HSM is to be used in an application it is advisable to protect HSM code by configuring it for exclusive access by the HSM. The blocks configurable for exclusive access by the HSM appear in *Figure 7* in *Chapter 3: Embedded memories*. They are labeled as secure code Flash or secure EEPROM emulation blocks. They are not configured for HSM-exclusive access by default, though—they are available to act as secure Flash areas but must be explicitly configured for that purpose. Refer to *SPC58xEx/SPC58xGx Microcontroller Security Reference Manual* for details.

38.9.2 Creating password groups

Most basic security features in the SPC58xEx/SPC58xGx microcontroller are password-based. Therefore, one of the early tasks that must be performed is the creation of four 256-bit passwords. They are programmed by the customer into the locations identified in the UTEST memory map (refer to the Memory Map chapter).

The UTEST memory map fields to be written are:

- PASS Password Group 0
- PASS Password Group 1
- PASS Password Group 2
- PASS Password Group 3

Passwords must not be one of the following, which are considered insecure and therefore invalid:

- 0x0000_0000
- 0xFFFF_FFFF
- 0x5555_5555

Caution: The password fields are in OTP UTEST Flash, so they can only be written once.

38.9.3 Planning secure write protection

Up to four levels of password-secured write protection can be assigned to any block. Note that by default, all blocks of user Flash are write protected, so this could be more accurately viewed as planning which blocks of user Flash will *not* have four levels of password-secured write protection. The write protection desired will depend on the usage of the Flash blocks. Before creating DCF records to implement secure write protection, please thoroughly review the implementation steps detailed in *Section 38.3.1: Implementing secure write protection* and *Table 129: DCF client list* in *Chapter 9: Device Configuration Format (DCF) Records*.

38.9.4 Planning secure read protection

Up to four levels of password-secured write protection can be assigned to Flash blocks in pre-defined Read Locking groups. Note that by default, all blocks of user Flash are read protected, so this could be more accurately viewed as planning which blocks of user Flash will *not* have four levels of password-secured read protection. The read protection desired will depend on the usage of the Flash blocks. Before creating DCF records to implement secure read protection, please thoroughly review the implementation steps detailed in *Section 38.4.1: Implementing secure read protection* and *Table 129: DCF client list* in *Chapter 9: Device Configuration Format (DCF) Records*.

38.9.5 Planning debug port enable/disable

Refer to *Section 38.5: Debug port enable/disable* and *Table 129: DCF client list* in *Chapter 9: Device Configuration Format (DCF) Records*.

38.9.6 Planning OTP Flash memory block assignment

Any block of user Flash can be configured as OTP. The need to configure blocks as OTP will depend on the data or code to be stored in the block. It is important to note that configuring a Flash block as OTP is not reversible.

Before creating DCF records to implemented OTP Flash blocks, please thoroughly review the implementation steps detailed in *Section 38.7: Implementing OTP* and *Table 129: DCF client list* in *Chapter 9: Device Configuration Format (DCF) Records*.

38.9.7 Planning factory test mode disable

Refer to *Section 38.8: Implementing test mode disable* and *Table 129: DCF client list* in *Chapter 9: Device Configuration Format (DCF) Records*.

39 Decorated Storage Memory Controller (DSMC)

39.1 Introduction

This section details hardware support for atomic read-modify-write memory operations. In the Power Architecture, these capabilities are called “*decorated storage*”. It is supported by capabilities in the processor cores plus instantiations of a Decorated Storage Memory Controller (DSMC).

The Decorated Storage APU defines instructions for providing load and store operations to memory addresses that require *additional semantics* beyond just the reading and writing of data values to the addressed memory locations. Decorated storage operations are intended to be used for specific devices or memory targets that require these additional semantics. A “decoration” is the additional semantic information to be applied to the decorated storage operation by the DSMC.

Consider the basic mnemonics, syntax and formats for the decorated load instructions.

For loads, the syntax is `l[b,h,w]d{cb}x rT,rA,rB` where the data size specifier is defined as 8-bit (`b` = byte), 16-bit (`h` = halfword) or 32-bit (`w` = word), and the three register specifiers include `rT` as the destination target register, `rB` as the effective address and `rA` as the decoration. The optional `{cb}` specifier defines a version of the instruction which is treated as a cache bypass operation.

The syntax for store instruction is `st[b,h,w]d{cb}x rS,rA,rB` where the same data size specifiers are used, and the three register specifiers are `rS` as the source data register, `rB` as the effective address and `rA` as the decoration. The cache bypass attribute is again specified with the use of the optional `{cb}` in the instruction mnemonic.

For all decorated loads and stores, the memory effective address is simply defined by the rB register (hence the `“x”` mnemonic suffix, signaling an “indexed” addressing mode) and the decoration is specified by the rA register. The core transmits the contents of the rA register as the decoration value along with the access address (register rB) to the DSMC.

The decorated load and store instruction formats are shown in *Figure 468*.

Figure 468. Power Architecture Decorated Load and Store Instruction Formats

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
lbdx	0	1	1	1	1	1	**r**T						**r**A				**r**B				1	0	0	0	0	0	0	0	1	1	/
lhdx	0	1	1	1	1	1	**r**T						**r**A				**r**B				1	0	0	0	1	0	0	0	1	1	/
lwdx	0	1	1	1	1	1	**r**T						**r**A				**r**B				1	0	0	1	0	0	0	0	1	1	/
lbdcbx	0	1	1	1	1	1	**r**T						**r**A				**r**B				1	0	0	0	0	0	0	0	1	0	/
lhdcbx	0	1	1	1	1	1	**r**T						**r**A				**r**B				1	0	0	0	1	0	0	0	1	0	/
lwdcbx	0	1	1	1	1	1	**r**T						**r**A				**r**B				1	0	0	1	0	0	0	0	1	0	/
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
stbdx	0	1	1	1	1	1	**r**S						**r**A				**r**B				1	0	1	0	0	0	0	0	1	1	/
sthdx	0	1	1	1	1	1	**r**S						**r**A				**r**B				1	0	1	0	1	0	0	0	1	1	/
stwdx	0	1	1	1	1	1	**r**S						**r**A				**r**B				1	0	1	1	0	0	0	0	1	1	/
stbdcbx	0	1	1	1	1	1	**r**S						**r**A				**r**B				1	0	1	0	0	0	0	0	1	0	/
sthdcbx	0	1	1	1	1	1	**r**S						**r**A				**r**B				1	0	1	0	1	0	0	0	1	0	/
stwdcbx	0	1	1	1	1	1	**r**S						**r**A				**r**B				1	0	1	1	0	0	0	0	1	0	/

The basic decorated load or store memory reference supplements the access address and attribute information with a 32-bit “decoration”. The DSMC decoration defines the special memory operation to be performed and optionally includes bit specifiers and operand data for the command or both.

The default format for the 32-bit decoration field defines a 4-bit command in the most significant bits (decoration[0:3]).

The cache bypass decoration load (`l[b,h,w]dcbx`) and store (`st[b,h,w]dcbx`) instructions do not distinguish between cacheable and cache-inhibited storage attributes, and instead have semantics to effectively emulate a cache-inhibited operation to storage regardless of the actual storage attributes. This is desired in order to allow a small portion of an otherwise cacheable storage area to be treated as non-cacheable using instruction supplied semantics, regardless of storage attributes provided by a memory management, protection or other memory control scheme. This allows decoration or other accesses to be applied to a portion of memory that is normally treated as cacheable according to its storage attributes, as well as to ensure that any stale copies of storage locations present in the cache are not used.

The combination of the decorated load and store instruction support in the e200zX cores plus the decorated storage memory controller in the core platform adds a robust atomic read-modify-write capability to the Power Architecture. The architecture capability defined by these core and platform functions is targeted at manipulation of *n*-bit fields in peripheral registers (and is consistent with I/O hardware addressing in the Embedded C standard) as well as software synchronization data structures needed in multi-core systems (mutexes, semaphores, test-and-set and compare-and-swap structures).

For additional information on the decorated load and store instructions from the core's perspective, refer to the appropriate e200zX core reference manual.

39.2 Decorated Stores: st[b,h,w]d{cb}x rS,rB,rA

The next sections present descriptions of the specific operations, based on the 4-bit command field defined in decoration[0:3]. These descriptions include the bit pattern definitions for the 32-bit decoration value and include pseudo-code detailing the sequence of operations. The decoration formats are defined using a <command>.<size> syntax where the operand size specifier is b (byte, 8-bit), h (halfword, 16-bit) or w (word, 32-bit). The operand size is specified directly in the decorated instruction executed in the core, and this attribute is driven to the system bus as part of the decorated data transfer. Additionally, the write data is taken from the right-justified 8, 16 or 32 bits of the rS register, that is:

```
 8-bit wdata = rS[24:31]
16-bit wdata = rS[16:31]
32-bit wdata = rS[ 0:31]
```

Likewise, read data is loaded into the right-justified 8, 16 or 32 bits of the rT register:

```
 8-bit rdata = rT[24:31]
16-bit rdata = rT[16:31]
32-bit rdata = rT[ 0:31]
```

For the byte and halfword decorated load instructions, the upper bits of the rT register are zero filled.

The starting bit (SRTBIT) position follows the Power Architecture convention where the MSB is bit 0 and the LSB is bit 31. The bit field width (BFW) value defines the width and BFW = 0 specifies the maximum width, that is, the container size.

The basic decorated store includes three data transfer fields sourced from the core: the access address (rB), the decoration (rA) and the write data (wdata) operand (rS). There are five operations defined and most of these transactions convert a single core AHB write bus cycle into an atomic read-modify-write, that is, an indivisible read followed by write sequence. Support for three basic boolean logic functions (AND, OR, XOR) is provided along with a compare-and-store operation plus a bit field insert operation. These operations do not support any type of bit field wrapping.

In the next sections detailing the decorated store operations, the following associations between the pseudocode variables and the core registers apply.

- decoration = rA
- accessAddress = rB
- wdata = rS

39.2.1 Bit Field Insert (BFINS) into an 8, 16 or 32-bit Memory Container

Figure 469. Decoration Format: BFINS

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
bfins.b	0	0	0	0	0	0	0	SRTBIT			0	0	0	BFW			-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
bfins.h	0	0	0	0	0	0	SRTBIT				0	0	BFW				-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
bfins.w	0	0	0	0	0	SRTBIT					0	BFW					-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

This command inserts a bit field defined by SRTBIT and BFW into the memory “container” defined by the access size associated with the decorated store instruction using an atomic read-modify-write sequence.

```
tmp = mem[accessAddress, size] // memory read

if bfw == 0
   then bfw = container

if ((srtbit + bfw) <= container)// generate bit mask
    mask = ((1 << bfw) - 1) << (container - (srtbit + bfw))
else
    mask = ((1 << bfw) - 1) >> ((srtbit + bfw) - container)

tmp = tmp   & ~mask// modify
    | wdata &  mask

mem[accessAddress, size] = tmp// memory write
```

The write data operand (wdata) associated with the decorated store instruction contains the bit field to be inserted. *It must be properly aligned within a right-justified container in the source register (rS)*, that is, within the lower 8 bits for a byte operation, the lower 16 bits for a halfword or the entire 32 bits for a word operation.

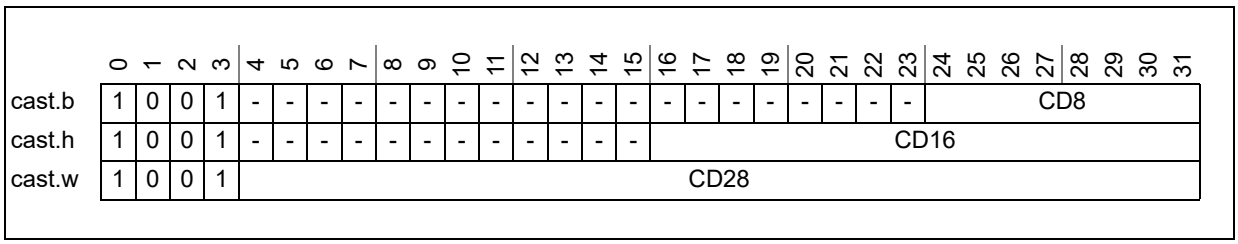
To illustrate, consider the following example of the insertion of the 3-bit field “xyz” into an 8-bit memory container, initially set to “abcd_efgh”. For all cases, the bit field width (BFW) is 3.

```
if SRTBIT = 0 and the decorated store rS register[24:31] = xyz-_----,
   then destination is "xyzd_efgh"
if SRTBIT = 1 and the decorated store rS register[24:31] = -xyz_----,
   then destination is "axyz_efgh"
if SRTBIT = 2 and the decorated store rS register[24:31] = --xy_z---,
   then destination is "abxy_zfgh"
if SRTBIT = 3 and the decorated store rS register[24:31] = ---x_yz--,
   then destination is "abcx_yzgh"
if SRTBIT = 4 and the decorated store rS register[24:31] = ---_xyz-,
   then destination is "abcd_xyzh"
if SRTBIT = 5 and the decorated store rS register[24:31] = ---_-xyz,
   then destination is "abcd_exyz"
if SRTBIT = 6 and the decorated store rS register[24:31] = ---_--xy,
   then destination is "abcd_efxy"
if SRTBIT = 7 and the decorated store rS register[24:31] = ---_---x,
   then destination is "abcd_efgx"
```

If the bit field insert operation specifies SRTBIT as 0 and BFW as 0 (indicating the width matches the container width), then the operation is logically equivalent to a simple register store operation.

39.2.2 Compare-and-Store (CAST)

Figure 470. Decoration Format: CAST



This command begins by performing a read of the referenced memory address and comparing that data with the operand included in the decoration word. If the read data is equal to the compare operand, the write data associated with the decorated store instruction is placed into the memory location, else the original read data is rewritten into the memory location. The write operand is selected from the appropriate data byte lanes in the same manner as any memory store operation; this implies the write data operand is right justified in the rS source register.

```
tmp = mem[accessAddress, size] // memory read

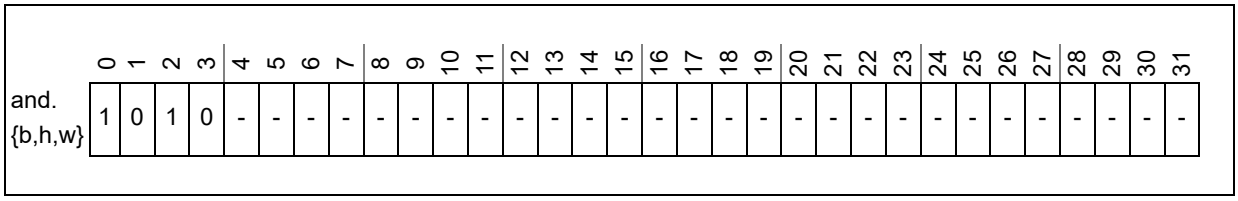
if (size == 8)// define compare_operand
    compare_operand = CD8
else if (size == 16)
          compare_operand = CD16
     else compare_operand = {0x0, CD28}// zero-filled data

if (tmp == compare_operand)// compare - "modify"
     mem[accessAddress, size] = wdata// memory write
else mem[accessAddress, size] = tmp
```

Note for the word size operation (cast.w), the data value specified in the low-order 28 bits of the decoration is zero filled in the most significant bits to create the required 32 bit compare operand.

39.2.3 Logical AND (AND)

Figure 471. Decoration Format: AND

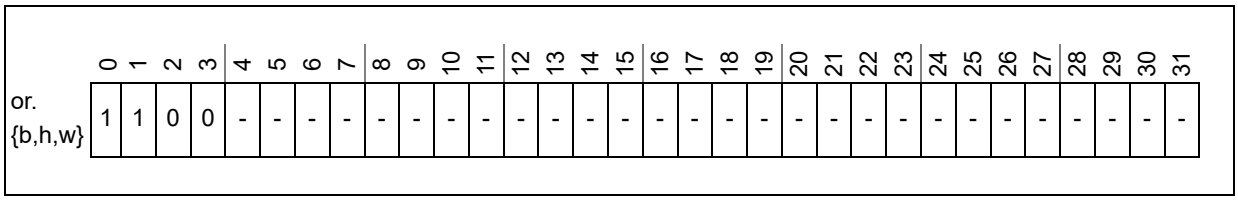


This command performs an atomic read-modify-write of the referenced memory location. First, the location is read; it is then modified by performing a logical AND operation using the write operand defined by the rS register; finally, the result of the AND operation is written back into the referenced memory location.

```
tmp = mem[accessAddress, size] // memory read
tmp = tmp & wdata// modify
mem[accessAddress, size] = tmp// memory write
```

39.2.4 Logical OR (OR)

Figure 472. Decoration Format: OR

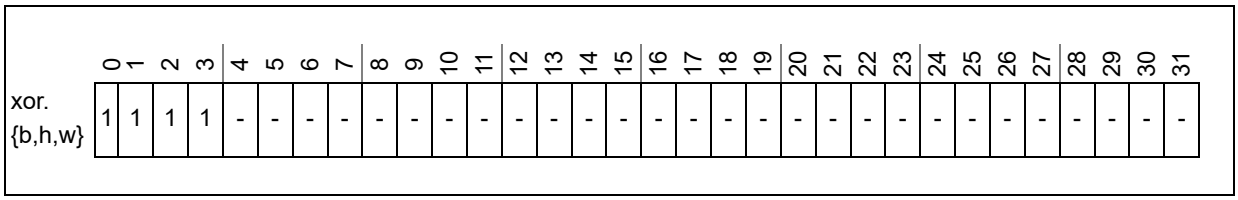


This command performs an atomic read-modify-write of the referenced memory location. First, the location is read; it is then modified by performing a logical OR operation using the write operand defined by the rS register; finally, the result of the OR operation is written back into the referenced memory location.

```
tmp = mem[accessAddress, size] // memory read
tmp = tmp | wdata// modify
mem[accessAddress, size] = tmp// memory write
```

39.2.5 Logical Exclusive-OR (XOR)

Figure 473. Decoration Format: XOR



This command performs an atomic read-modify-write of the referenced memory location. First, the location is read; it is then modified by performing a logical XOR operation using the write operand defined by the rS register; finally, the result of the XOR operation is written back into the referenced memory location.

```
tmp = mem[accessAddress, size] // memory read
tmp = tmp ^ wdata// modify
mem[accessAddress, size] = tmp// memory write
```

39.3 Decorated Loads: `l[b,h,w]d{cb}x rT,rB,rA`

The basic decorated load includes two data transfer fields sourced from the core: the access address (rB), the decoration (rA) plus the read data (rdata) operand returned from DSMC to the core and loaded into the destination register (rT). There are 3 operations defined and two of these transactions convert a single core AHB read bus cycle into an atomic read-modify-write. Support for a swap function, a load-and-set-1-bit functions are provided along with a simple load (aka an “extract”) operation.

In the next sections detailing the decorated load operations, the following associations between the pseudocode variables and the core registers apply.

- decoration = rA
- accessAddress = rB
- rdata = rT

39.3.1 Simple Load (SLD)

Figure 474. Decoration Format: SLD

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
sld. {b,h,w}	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

This command performs a simple memory load of the reference size from the access address. For this operation, the 12-bit field defined in decoration[4:15] must be zeroes else the transfer is not performed and error terminated.

```
rdata = mem[accessAddress, size] // memory read
```

Support for generic bit field extract operations is best handled using existing core and compiler capabilities involving standard load instructions followed by left and right shift operations to emulate these functions.

39.3.2 Registers-and-Memory Exchange (SWAP)

Figure 475. Decoration Format: SWAP

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
cast.b	0	1	0	1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	WD8							
cast.h	0	1	0	1	-	-	-	-	-	-	-	-	-	-	-	-	WD16															
cast.w	0	1	0	1	WD28																											

This command loads the contents of the referenced memory location into the core's rT register and stores the source operand from the core's rA register (the decoration) into the memory location.

```
tmp   = mem[accessAddress, size] // memory read
rdata = tmp// rT = rdata = tmp

if (size == 8)// store_operand - "modify"
    store_operand = WD8
else if (size == 16)
          store_operand = WD16
     else store_operand = {0x0, WD28}// zero-filled write data

mem[accessAddress, size] = store_operand// memory write
```

Note for the word size operation (swap.w), the data value specified in the low-order 28 bits of the decoration is zero filled in the most significant bits to create the required 32-bit store operand.

39.3.3 Load-and-Set-1(Bit) (LAS1)

Figure 476. Decoration Format: LAS1

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
las1.b	0	1	1	0	0	0	0	BIT			-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
las1.h	0	1	1	0	0	0	BIT				-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
las1.w	0	1	1	0	0	BIT					-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

This command first reads the referenced memory location; it then modifies the read operand by setting the single bit position defined in the decoration (BIT); the modified data is then written back to the referenced memory location and the original read data returned to the initiating processor core.

```
tmp   = mem[accessAddress, size] // memory read
rdata = tmp// rT = rdata = tmp
mask  = 1 << (container - bit - 1)// generate bit mask
tmp   = tmp | mask// modify
mem[accessAddress, size] = tmp// memory write
```

39.4 DSMC Instantiations

The decorated storage memory controller which performs these operations is instantiated multiple times within the core platform and physically resides between the data port of the cores and the master ports of the crossbar.

The module includes error checking logic that validates the decoration field is properly defined (all illegal commands are rejected and the transfer error terminated). Additionally, the decorated memory controller only operates on aligned, single data transfers (misalignment and bursts are not supported and error terminated if attempted).

As the DSMC generates the atomic read-modify-write bus transactions to the targeted slave memory controller, the two transactions (read, write) are fully pipelined with no idle cycles introduced. During these transactions, the AHB hlock control signal is asserted to the slave during the entire read-modify-write.

40 Analog-to-Digital Converters (ADC) Configuration

This chapter is organized as follows:

- *Section 40.1: ADC overview*
  - *Section 40.1.1: ADC subsystem block diagram*
  - *Section 40.1.2: Analog input pin multiplexing*
- *Section 40.2: Configuration of ADC modules*
  - *Section 40.2.1: Sigma-Delta Analog-to-Digital Converter (SDADC)*
  - *Section 40.2.2: Successive Approximation Register Analog-to-Digital Converter (SAR ADC)*

40.1 ADC overview

SPC58xEx/SPC58xGx contains the following analog-to-digital converters (ADCs):

- five independent 12-bit and three independent 10-bit Successive Approximation Register Analog-to-Digital Converters (SARADCs)
- six independent 16-bit Sigma-Delta Analog-to-Digital Converters (Sigma-Delta, or SDADCs)

Table 588. ADC implementation

Converter Type	12-bit SAR ADC					10-bit SAR ADC			16-bit SD ADC					
Special Function	Supervisor SAR ADC	Fast SAR ADC				Fast Comparator		STDBY & Fast Comparator	Sigma-Delta					
Name	SAR_ADC_12bit_SUPERVISOR (SAR_ADC_12bit_B0)	SAR_ADC_12bit_0	SAR_ADC_12bit_1	SAR_ADC_12bit_2	SAR_ADC_12bit_3	SAR_ADC_10bit_0	SAR_ADC_10bit_1	SAR_ADC_10bit_STDBY	SD_ADC_0	SD_ADC_1	SD_ADC_2	SD_ADC_3	SD_ADC_4	SD_ADC_5
Number of internal channels(1)	82	8	8	15	15	4	4	8	2	2	4	8	2	2

1. This is the maximum number of available channels per dedicated silicon but the available number of channels per package might be different (see the IO_Definition document)

Note: SDADC and SARADC instances naming in some figures of this chapter may differ from the naming provided in Table 588 in order to simplify these figures.

The internal channels of each ADC are connected to analog input pads. The analog pads include programmable pull-up / pull-down resistors.

These ADCs interface to external analog input pins. The number of external analog input pins for each package is available in the IO_Definition.xls file attached to the IO_Definition document.

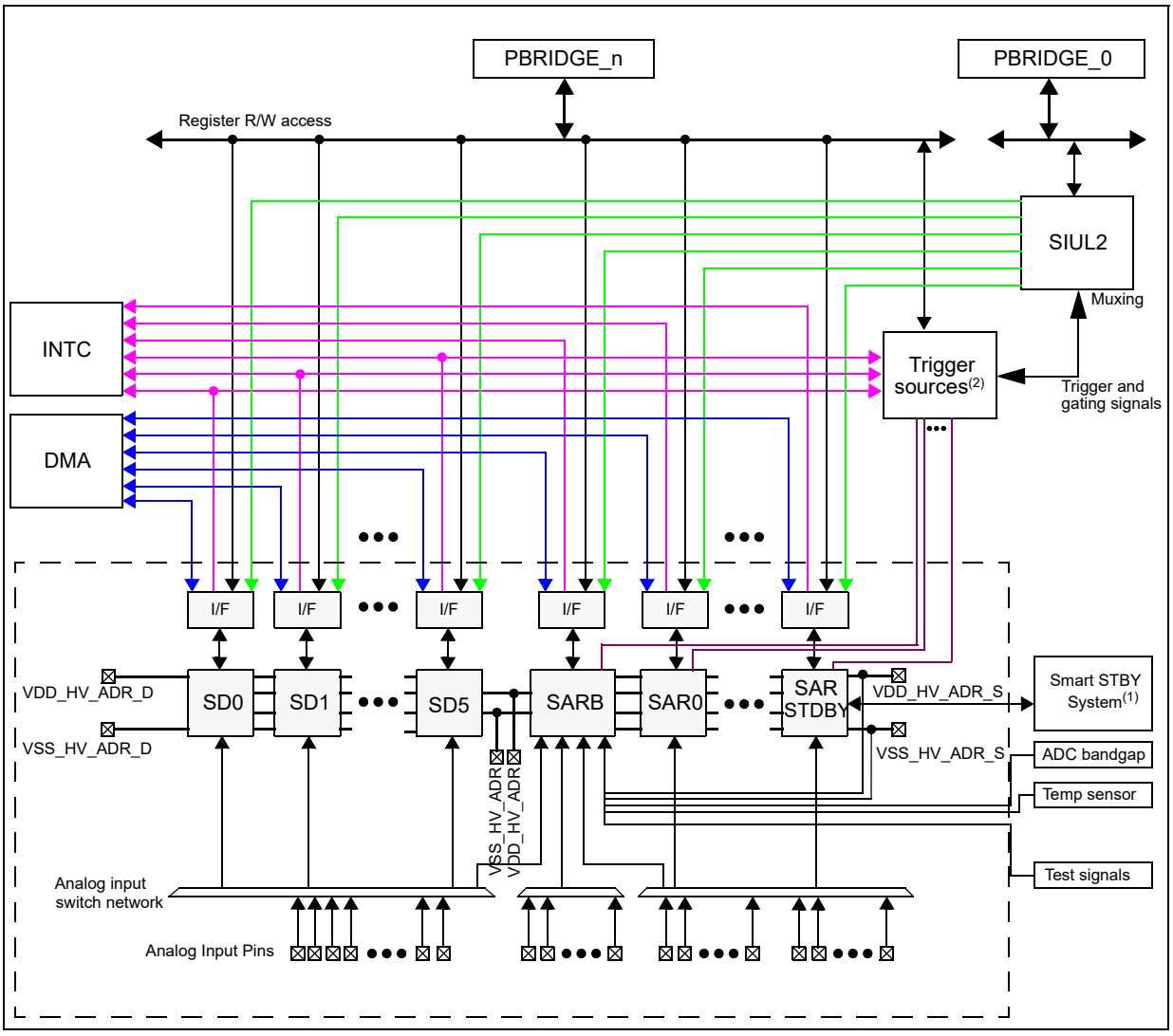
Each ADC instance, whether SAR or Sigma-Delta, has independent analog input , register interface, trigger and clock inputs, independent interrupt, and DMA capability. Each SAR and Sigma-Delta ADC has a dedicated peripheral bridge (PBRIDGE) interface for CPU access to the registers.

All SAR and Sigma-Delta ADCs share the same supply pins (VDD_HV_ADV/VSS_HV_ADV) and ground, but have separate reference pins. This is true for all packages.

40.1.1 ADC subsystem block diagram

Figure 477 gives the block diagram for the high-level integration of the ADC sub-system for SPC58xEx/SPC58xGx. The primary intent of the diagram is to show that each ADC, whether SAR or Sigma-Delta, has independent register interfaces, interrupt/DMA requests, and trigger source selection.

Figure 477. SPC58xEx/SPC58xGx ADC subsystem



1. SAR_ADC_10bit_STDBY interactions with the Smart STBY System are described in the *Chapter 11: Smart Stand-by Wake-up Unit (SSWU)*
2. The different trigger sources are detailed in the *Section 40.2.2.5: SAR ADC triggering*.

40.1.2 Analog input pin multiplexing

For analog input pin multiplexing please refer to the IO_Definition.xls file attached to the IO_Definition document.

40.2 Configuration of ADC modules

The following sections describe device-specific details for the following modules:

- *Section 40.2.1: Sigma-Delta Analog-to-Digital Converter (SDADC)*
- *Section 40.2.2: Successive Approximation Register Analog-to-Digital Converter (SAR ADC)*

40.2.1 Sigma-Delta Analog-to-Digital Converter (SDADC)

There are six independent Sigma-Delta ADCs on SPC58xEx/SPC58xGx. Refer to *Figure 477* for the ADC subsystem block diagram.

The following sections describe configuration details:

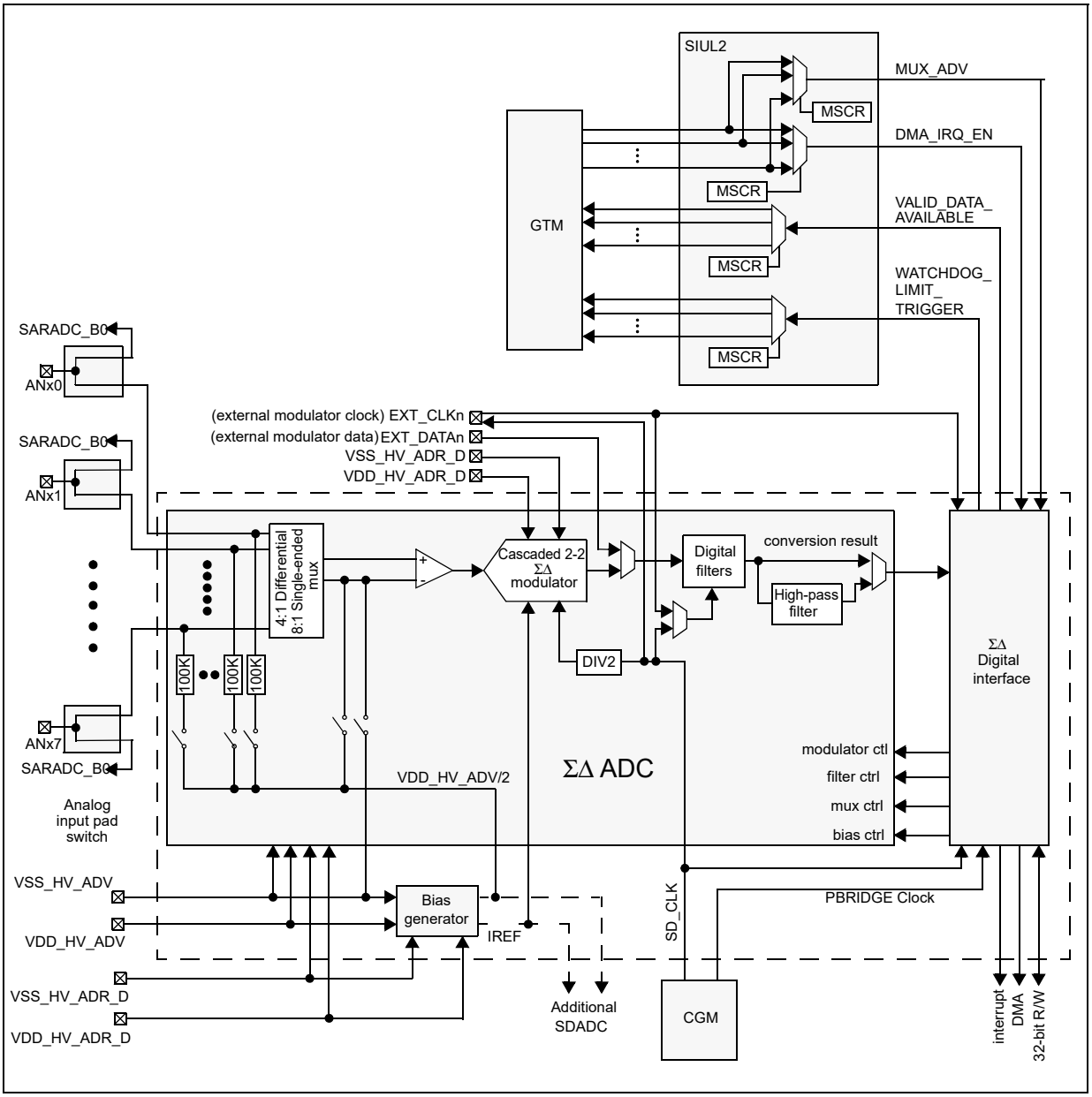
- *Section 40.2.1.1: Sigma-Delta ADC integration diagram*
- *Section 40.2.1.2: Sigma-Delta ADC analog input multiplexing*
- *Section 40.2.1.3: Sigma-Delta ADC clock sources*
- *Section 40.2.1.4: Sigma-Delta ADC starting/triggering*
- *Section 40.2.1.5: Sigma-Delta ADC external modulator*
- *Section 40.2.1.6: Conversion result interrupt/DMA*
- *Section 40.2.1.9: Sigma-Delta ADC analog input channel selection*

For detailed module information, refer to the Sigma-Delta Analog-to-Digital Converter (SDADC) Digital Interface chapter.

40.2.1.1 Sigma-Delta ADC integration diagram

Figure 478 shows the block diagram of the Sigma-Delta ADC integration on SPC58xEx/SPC58xGx. Note that in the diagram any unused Sigma-Delta ADC input can be read by SAR_ADC_12bit_B0. This is true for the unused input in single-ended operation.

Figure 478. Sigma-Delta ADC integration diagram



40.2.1.2 Sigma-Delta ADC analog input multiplexing

The analog input multiplexing switches are contained within the Sigma-Delta ADCs and within the SAR_ADC_B. The connection to the Sigma-Delta ADC inputs is made to the pad pin through the analog pad cell, shown in *Figure 478*, and is controlled as follows:

- SAR_ADC_12bit_B0 analog inputs have no SIUL2_MSCRn[APC] bit-control, but are directly connected to the pad.
- SDADC analog inputs have no SIUL2_MSCRn[APC] bit control, but are directly connected to the pad.

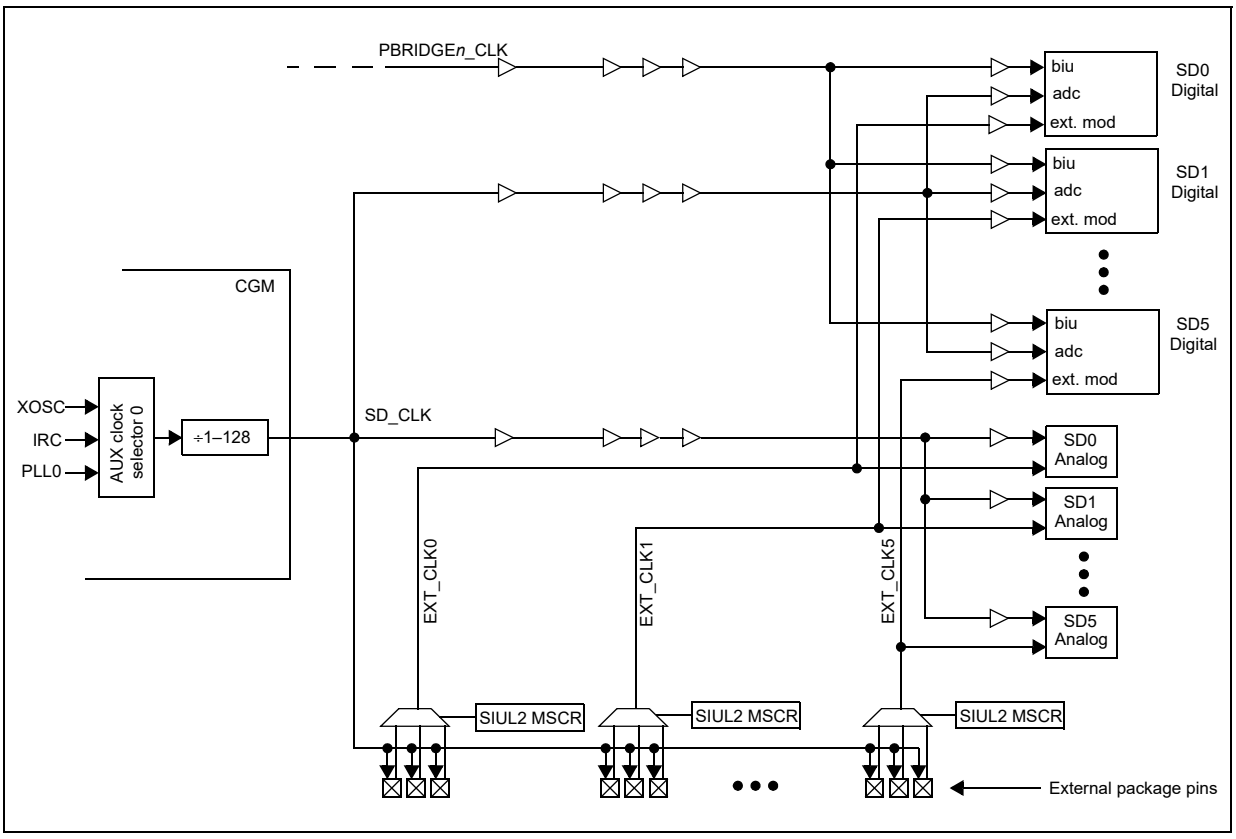
The mapping of analog input pins to the Sigma-Delta converters is given in *Section 40.2.1.9: Sigma-Delta ADC analog input channel selection*.

40.2.1.3 Sigma-Delta ADC clock sources

Each Sigma-Delta ADC digital interface on the SPC58xEx/SPC58xGx has a register clock input, which is separate from the module clock input. The register interface is clocked by the PBRIDGE*n*_CLK. The module clock is sourced by the modulator clock (SD_CLK). The input clock for the Sigma-Delta modulator is the SD_CLK. *Figure 479* gives a block diagram of the required Sigma-Delta ADC clock architecture. See the *Section 27.6.7: Sigma-Delta ADC clocking* in the *Chapter 27: Clocking*, for more detail on the clock sources.

Both the analog and digital blocks of the ADC contain an internal clock mux for external modulator mode. The external modulator clock (EXT_CLK) can be input to the device, or generated from SD_CLK from the CGM. The pad input/output settings in the corresponding SIUL2 MSCR pad control registers determine the external modulator clock source to the ADC blocks. The EXT_CLK pins are multiplexed on multiple external pins on the device. Selection of the EXT_CLK pin to use for the external modulator clock for each ADC is done in the SIUL2 MSCR input multiplexing registers. Each Sigma-Delta ADC has three external pins to select from for an external modulator clock.

Figure 479. Sigma-Delta ADC clock diagram



40.2.1.3.1 Clock Skew

To avoid crosstalk issues, all Sigma-Delta ADCs run at the same clock frequency. Independent programmable oversampling rates are provided to support the varying bandwidths of interest on each ADC in the application. Therefore, all Sigma-Delta analog blocks are clocked from the output of a single divider of the peripheral clock at the chip level.

40.2.1.4 Sigma-Delta ADC starting/triggering

Each Sigma-Delta ADC on SPC58xEx/SPC58xGx supports starting the converter with a software write to an enable bit in a register in the digital interface.

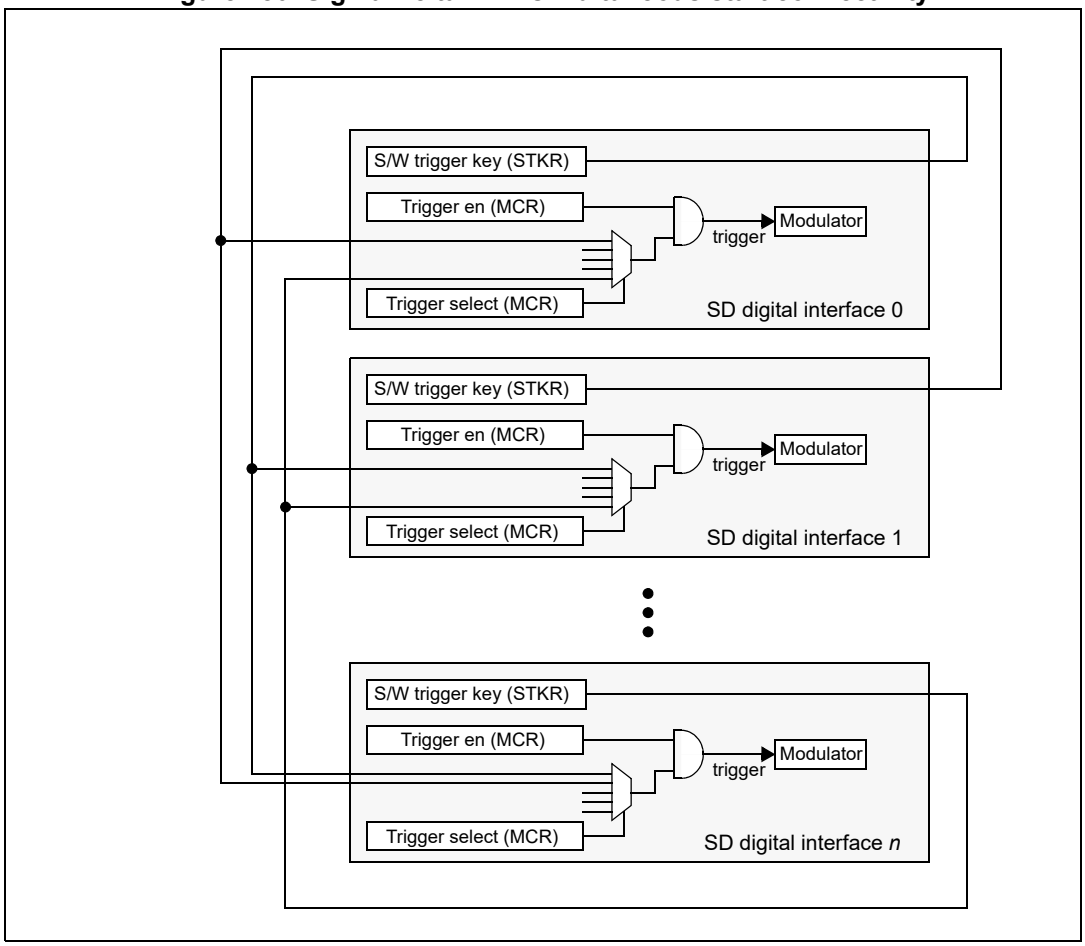
40.2.1.4.1 Simultaneous start of multiple Sigma-Delta converters

A method to start multiple Sigma-Delta converters simultaneously by a single software write to a control register is available on the device. Simultaneously in this case means that all selected converters start sampling their inputs on the same SD_CLK clock edge.

The digital interface of the Sigma-Delta ADC contains the control register for simultaneous start of the ADCs. When the register is written to start multiple converters, an output from the digital interface is asserted. The hardware trigger input of each Sigma-Delta ADC is connected to this simultaneous start output. This way the user controls which ADCs are selected by enabling the hardware trigger of desired ADCs before writing to the starting simultaneous converters.

The connections for the simultaneous start of the converters is given in *Figure 480*.

Figure 480. Sigma-Delta ADC simultaneous start connectivity



40.2.1.4.2 Sigma-Delta mux input advance and wraparound

The mux wraparound feature is enabled and controlled in the digital interface of the Sigma-Delta ADC. The advancing of the mux can be controlled in the interface by software or by external GTM channels. The mapping of GTM output channels to the mux advance input for each Sigma-Delta ADC for each device is given in the SIUL2 MSCR registers.

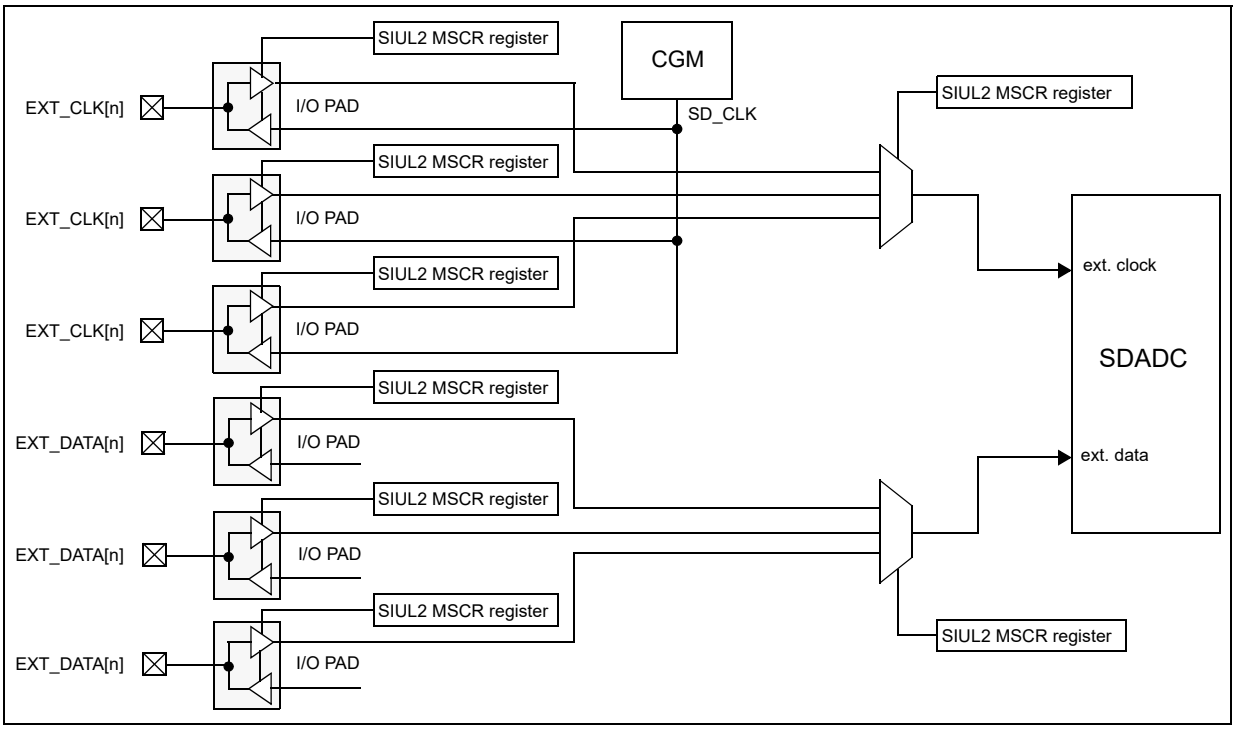
40.2.1.5 Sigma-Delta ADC external modulator

40.2.1.5.1 External modulators

The Sigma-Delta ADC supports an interface to external Sigma-Delta modulators. In this mode, the Sigma-Delta ADC internal modulator is disabled, and a serial bit stream from an external modulator is clocked into the back-end digital filters of the on-chip Sigma-Delta ADC. Currently, the AMC1203 external modulator is supported by the Sigma-Delta ADC. It is possible that other external 2nd order modulators with clock rates less than 10 MHz and a compatible electrical interface (see SPC58xEx/SPC58xGx Data Sheet) could be supported.

The external modulator interface for each Sigma-Delta ADC on SPC58xEx/SPC58xGx is shown in *Figure 481*.

Figure 481. Sigma-Delta ADC external modulator interface



40.2.1.5.2 External modulator clock/data inputs

Two external pins are required for each Sigma-Delta ADC to support an external modulator, an input data pin EXT_DATA (bitstream data), and an input/output clock pin EXT_CLK (bitstream clock). When the external modulator is used, the clock and data inputs to the Sigma-Delta ADC digital filters come from the EXT_CLK and EXT_DATA pins, instead of the internal modulator as shown in *Figure 481*. External modulator operation is enabled in the ADC digital interface registers. The internal modulator is disabled. The EXT_CLK and

EXT_DATA inputs are multiplexed on port input pins. For each Sigma-Delta ADC, there are three sets of EXT_CLK/EXT_DATA pins, and selection of those pins for each ADC is done in the SIUL2 MSCR registers.

40.2.1.5.3 External modulator clock output

SPC58xEx/SPC58xGx provides an option for an output clock from the device to the external modulator. When providing an output clock, the SD_CLK from the CGM is connected to the EXT_CLK output pin. Both the input and output buffer of the EXT_CLK pin must be enabled in the corresponding SIUL2 MSCR register, in order for both the external modulator and internal filters to be clocked from the SD_CLK. This is shown in *Figure 481*.

40.2.1.6 Conversion result interrupt/DMA

A DMA or interrupt can be generated for each conversion for each ADC. The Sigma-Delta ADC has an 8-deep result FIFO.

The Sigma-Delta ADC digital interface has an input signal that is used to disable interrupt and DMA requests for conversion results. The input signal is connected to outputs from the GTM timer block. The digital interface has a register bit for enabling and disabling the interrupt/DMA gating feature.

When the signal from the GTM is asserted, no interrupt or DMA request is generated for any conversion results. The polarity of the gating signal is such that when the signal is high, interrupt and DMA requests are passed. When the gating signal is low, interrupt and DMA requests are blocked.

The gating signal for each ADC is driven by several GTM channels. See *Chapter 17: System Integration Unit Lite2 (SIUL2)* for the specific mapping of GTM channels to Sigma-Delta interrupt/DMA gating signals.

40.2.1.7 Conversion Limit Watchdog Output to GTM

Each Sigma-Delta ADC provides an optional conversion limit check for upper and lower bounds. The enable bit and the two 16-bit limit registers for the feature are included in the SDADC wrapper. If a conversion result exceeds one of the limits, a flag bit is asserted in a status register in the SDADC wrapper. The flag bit is output to the GTM for each SDADC. The mapping of conversion limit watchdog outputs to GTM channels is defined in the SIUL2 MSCR registers.

40.2.1.8 SDADC_MCR[TRIGSEL] definitions

For this chip, the following input triggers apply.

Table 589. Input triggers

TRIGSEL	Trigger input selection(1)
0000	SD_ADC_0
0001	SD_ADC_1
0010	SD_ADC_2
0011	SD_ADC_3
0100	SD_ADC_4
0101	SD_ADC_5
0110	MUX_ADV
0110–1111	Reserved

1. This field selects which input will be used for hardware-triggered conversions.

Note: Selecting the input trigger source to be from the output trigger of the same SDADC is not prohibited by hardware, but is not supported, and will result in the SDADC never responding to the input trigger.

40.2.1.9 Sigma-Delta ADC analog input channel selection

For the Sigma-Delta mux inputs AN[x], the positive and negative input terminals are selected in the Sigma-Delta ADC digitial interface Channel Selection Register (CSR).

This section shows analog channel selections for each combination of Module Configuration Register (MCR) field values MCR_MODE and MCR_VCOMSEL. The following table provides a quick reference key to the MCR fields used in the SD ADC analog input selection tables that follow it.

Table 590. Register fields for SD ADC analog input channel selection

Field	Description
MCR_VCOMSEL	Common Voltage Bias Selection This bit selects the common voltage bias for the negative input terminal of SDADC during single-ended mode (MODE=1). 0 Negative input terminal is biased with VREFN. 1 Negative input terminal is biased with VREFP/2 (half scale bias).
MCR_MODE	Mode selection 0 Differential input mode is selected. 1 Single-ended input mode is selected.
CSR_ANCHSEL	Analog Channel Selection Based on single-ended or differential mode of operation (MODE bit of MCR), common mode voltage selection (VCOMSEL bit of MCR), this bit-field defines the connectivity of analog inputs to either positive or negative polarity terminals of Sigma-Delta ADC, as shown in the following tables: – *Table 591* for SD_ADC_0 – *Table 592* for SD_ADC_1 – *Table 593* for SD_ADC_2 – *Table 594* for SD_ADC_3 – *Table 595* for SD_ADC_4 – *Table 596* for SD_ADC_5

Table 591. SDADC_0 analog input AN[x] selection

MCR_MODE	MCR_VCOMSEL	CSR_ANCHSEL	INP (positive terminal)	INM (Negative terminal)
1	0	000	AN[0]	VREFN
		001	AN[1]	
		010	Reserved	
		011	Reserved	
		100	Reserved	
		101	Reserved	
		110	Reserved	
		111	Reserved	
	1	000	AN[0]	VREFP/2
		001	AN[1]	
		010	Reserved	
		011	Reserved	
		100	Reserved	
		101	Reserved	
		110	Reserved	
		111	Reserved	
0	0/1	000	AN[0]	AN[1]
		001	Reserved	
		010	Reserved	
		011	Reserved	
		100	VREFN	VREFN
		101	VREFP/2	VREFP/2
		110	VREFP	VREFN
		111	VREFN	VREFP

Table 592. SDADC_1 analog input AN[x] selection

MCR_MODE	MCR_VCOMSEL	CSR_ANCHSEL	INP (positive terminal)	INM (Negative terminal)
1	0	000	AN[0]	VREFN
		001	AN[1]	
		010	Reserved	
		011	Reserved	
		100	Reserved	
		101	Reserved	
		110	Reserved	
		111	Reserved	
	1	000	AN[0]	VREFP/2
		001	AN[1]	
		010	Reserved	
		011	Reserved	
		100	Reserved	
		101	Reserved	
		110	Reserved	
		111	Reserved	
0	0/1	000	AN[0]	AN[1]
		001	Reserved	
		010	Reserved	
		011	Reserved	
		100	VREFN	VREFN
		101	VREFP/2	VREFP/2
		110	VREFP	VREFN
		111	VREFN	VREFP

Table 593. SDADC_2 analog input AN[x] selection

MCR_MODE	MCR_VCOMSEL	CSR_ANCHSEL	INP (positive terminal)	INM (Negative terminal)
1	0	000	AN[0]	VREFN
		001	AN[1]	
		010	AN[2]	
		011	AN[3]	
		100	Reserved	
		101	Reserved	
		110	Reserved	
		111	Reserved	
	1	000	AN[0]	VREFP/2
		001	AN[1]	
		010	AN[2]	
		011	AN[3]	
		100	Reserved	
		101	Reserved	
		110	Reserved	
		111	Reserved	
0	0/1	000	AN[0]	AN[1]
		001	AN[2]	AN[3]
		010	Reserved	
		011	Reserved	
		100	VREFN	VREFN
		101	VREFP/2	VREFP/2
		110	VREFP	VREFN
		111	VREFN	VREFP

Table 594. SDADC_3 analog input AN[x] selection

MCR_MODE	MCR_VCOMSEL	CSR_ANCHSEL	INP (positive terminal)	INM (Negative terminal)
1	0	000	AN[0]	VREFN
		001	AN[1]	
		010	AN[2]	
		011	AN[3]	
		100	AN[4]	
		101	AN[5]	
		110	AN[6]	
		111	AN[7]	
	1	000	AN[0]	VREFP/2
		001	AN[1]	
		010	AN[2]	
		011	AN[3]	
		100	AN[4]	
		101	AN[5]	
		110	AN[6]	
		111	AN[7]	
0	0/1	000	AN[0]	AN[1]
		001	AN[2]	AN[3]
		010	AN[4]	AN[5]
		011	AN[6]	AN[7]
		100	VREFN	VREFN
		101	VREFP/2	VREFP/2
		110	VREFP	VREFN
		111	VREFN	VREFP

Table 595. SDADC_4 analog input AN[x] selection

MCR_MODE	MCR_VCOMSEL	CSR_ANCHSEL	INP (positive terminal)	INM (Negative terminal)
1	0	000	AN[0]	VREFN
		001	AN[1]	
		010	Reserved	
		011	Reserved	
		100	Reserved	
		101	Reserved	
		110	Reserved	
		111	Reserved	
	1	000	AN[0]	VREFP/2
		001	AN[1]	
		010	Reserved	
		011	Reserved	
		100	Reserved	
		101	Reserved	
		110	Reserved	
		111	Reserved	
0	0/1	000	AN[0]	AN[1]
		001	Reserved	
		010	Reserved	
		011	Reserved	
		100	VREFN	VREFN
		101	VREFP/2	VREFP/2
		110	VREFP	VREFN
		111	VREFN	VREFP

Table 596. SDADC_5 analog input AN[x] selection

MCR_MODE	MCR_VCOMSEL	CSR_ANCHSEL	INP (positive terminal)	INM (Negative terminal)
1	0	000	AN[0]	VREFN
		001	AN[1]	
		010	Reserved	
		011	Reserved	
		100	Reserved	
		101	Reserved	
		110	Reserved	
		111	Reserved	
	1	000	AN[0]	VREFP/2
		001	AN[1]	
		010	Reserved	
		011	Reserved	
		100	Reserved	
		101	Reserved	
		110	Reserved	
		111	Reserved	
0	0/1	000	AN[0]	AN[1]
		001	Reserved	
		010	Reserved	
		011	Reserved	
		100	VREFN	VREFN
		101	VREFP/2	VREFP/2
		110	VREFP	VREFN
		111	VREFN	VREFP

40.2.2 Successive Approximation Register Analog-to-Digital Converter (SAR ADC)

There are five independent 12-bit and three independent 10-bit SAR ADCs on SPC58xEx/SPC58xGx. Refer to *Figure 477* for the ADC sub-system block diagram.

The following sections describe configuration details:

- *Section 40.2.2.1: SAR ADC configuration overview*
- *Section 40.2.2.2: SAR ADC integration diagram*
- *Section 40.2.2.3: SAR ADC analog input*
- *Section 40.2.2.4: SAR ADC clock sources*
- *Section 40.2.2.5: SAR ADC triggering*
- *Section 40.2.2.6: SAR ADC fast comparator*
- *Section 40.2.2.7: SAR ADC Standby*
- *Section 40.2.2.8: SAR absolute voltage reference*
- *Section 40.2.2.9: SAR ADC diagnostic*
- *Section 40.2.2.10: SAR ADC channel assignment*

For detailed module information, refer to the Successive Approximation Register Analog-to-Digital Converter (SAR ADC) Digital Interface chapter.

40.2.2.1 SAR ADC configuration overview

40.2.2.1.1 12-bit Fast SAR ADC

Four 12-bit Fast SAR ADC instances are implemented with:

- 12-bit and 10-bit conversion mode selectable by software on a per conversion basis
  - 8 analog input channels for each SAR_ADC_12bit_0 and SAR_ADC_12bit_1
  - 15 analog input channels for each SAR_ADC_12bit_2 and SAR_ADC_12bit_3
- 4 internal test channels (called "test channels") to be used for self-diagnosis of the converter and diagnosis of the analog signal path up to the external circuitry. This includes:
  - four internal voltage references with 20 kOhm source impedance
- 4 watchdogs

40.2.2.1.2 12-bit SAR ADC B0

One 12-bit SARADC B0 (SAR_ADC_12BIT_B0) instance with diagnosis is implemented with:

- 12-bit and 10-bit conversion mode selectable by software on a per conversion basis
- 82 analog input channels (called internal channels)
- 64 external multiplexed channels (called external channels)
- All analog internal channels from other ADCs are also connected to SAR_ADC_12bit_B0
- 9 internal test channels (called "test channels") to be used for self-diagnosis of the converter and diagnosis of the analog signal path up to the external circuitry. This includes:
  - an internal bandgap reference from temperature sensor
  - an internal analog ground signal for software offset cancellation
  - four internal voltage references with 20 kOhm source impedance
- 8 watchdogs
- Body Cross Triggering Unit (BCTU) support to trigger ADC conversion

40.2.2.1.3 10-bit Fast SAR ADC

Two fast 10-bit SAR ADC instances with four internal channels and fast comparator capability are implemented with:

- 10-bit and 8-bit conversion mode selectable by software on a per conversion basis
- Very fast detection of analog threshold violations
- 4 internal test channels (called "test channels") to be used for self-diagnosis of the converter and diagnosis of the analog signal path up to the external circuitry. This includes:
  - four internal voltage references with 20 kOhm source impedance
- 4 watchdogs
- Comparator mode:
  - the detection is done against two programmable 10-bit thresholds
  - two violation events (against upper and lower thresholds) are available. These events are combined in one signal connected to the GTM.

40.2.2.1.4 10-bit Fast SAR ADC STDBY

One fast 10-bit SAR ADC STDBY instance with 8 internal channels with fast comparator capability and supporting STAND-BY feature is implemented with:

- 10-bit and 8-bit conversion mode selectable by software on a per conversion basis (all standard SAR ADC functionalities are present)
- 8 external channels needed for data register and watchdog functionality
- 4 analog watchdogs
- Cross Triggering Unit (STBY eCTU) support. In Standby mode, ADC operation is controlled by STBY eCTU. Refer to the *Chapter 11: Smart Stand-by Wake-up Unit (SSWU)* for details on STAND-BY feature
- In RUNTIME, 10-bit Fast ADC STDBY can be used as the other 10-bit Fast SAR ADC instances

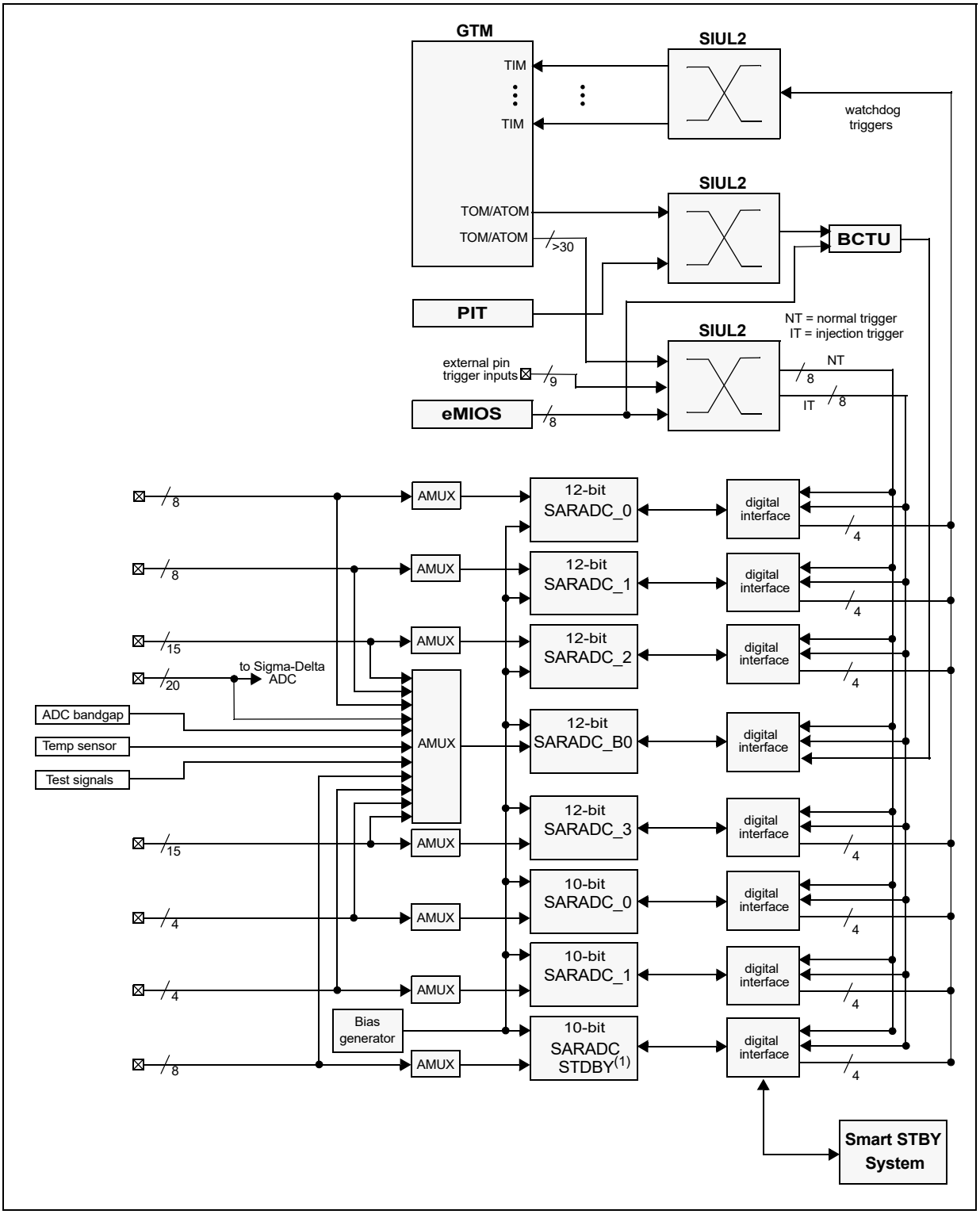
Note: Refer to SPC58xEx/SPC58xGx datasheet for SAR ADC analog performances.

40.2.2.2 SAR ADC integration diagram

Figure 482 shows the block diagram of the integration of the five 12-bit and three 10-bit SAR ADCs on SPC58xEx/SPC58xGx.

The bias generator provides 4 reference points (VDD_HV_ADR_S, 1/3 VDD_HV_ADR_S, 2/3 VDD_HV_ADR_S, and VSS_HV_ADR_S) which are provided to SAR_ADC_12bit_B0 via an internal 20 kΩ impedance. See the SAR ADC block diagram in the *Chapter 42: Successive Approximation Register Analog-to-Digital Converter (SARADC)*.

Figure 482. SAR ADC integration diagram(1)



1. SAR_ADC_10bit_STDBY instance is implemented on a special low-power domain. Further information is provided in the *Chapter 11: Smart Stand-by Wake-up Unit (SSWU)*

40.2.2.3 SAR ADC analog input

All analog input pins are routed to SAR_ADC_12bit_B0 and another fast SAR ADC. Analog input (AMUX) is implemented in the SAR ADC subsystem.

Note: Simultaneous sampling of two SAR ADCs on the same analog channel is possible under certain conditions:

- both ADCs need synchronized start of sampling (both ADCs have to be triggered by the same trigger signal)

- both ADCs need synchronized end of sampling (both ADCs have to be configured with the same sampling time)

- external circuitry on the analog input has to be chosen accordingly (effective internal capacity on analog signal path during sampling time is doubled versus single conversion as given in the device data sheet)

The mapping of analog input pins to the SAR converters is given in *Section 40.1.2: Analog input pin multiplexing*.

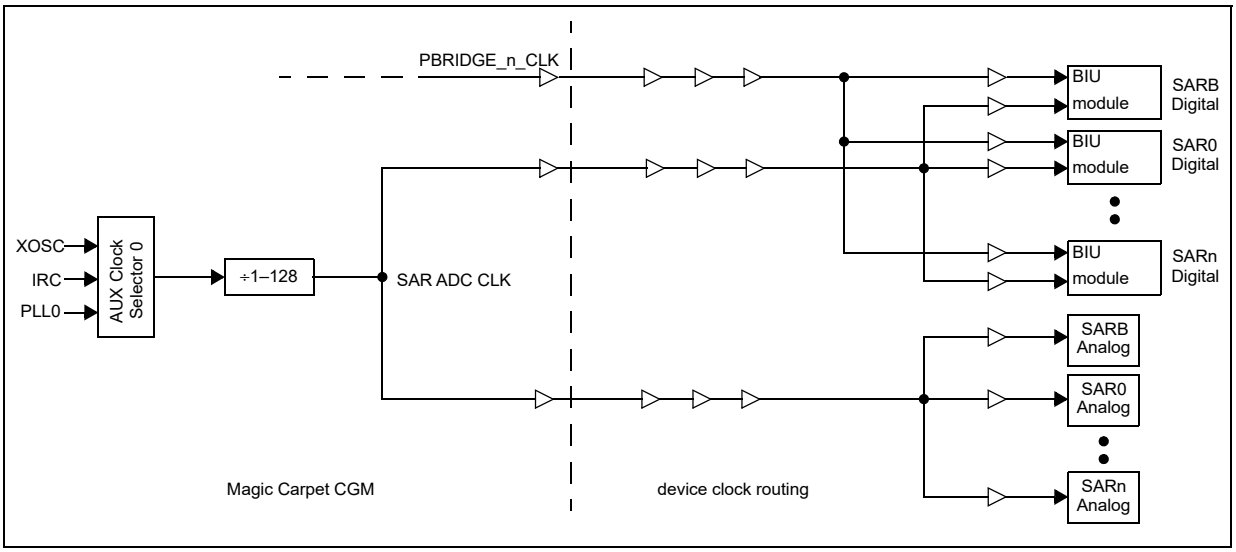
40.2.2.4 SAR ADC clock sources

Each SAR ADC digital interface on SPC58xEx/SPC58xGx has a register clock input, which is separate from the module clock input. The register interface is clocked by the PBRIDGE*n*_CLK. The module clock is sourced by the ADC clock (SAR ADC CLK). See *Chapter 27: Clocking* for more detail on the clock sources.

The input clock for the SAR ADC analog blocks is clocked by the SAR ADC CLK. *Figure 483* shows a block diagram of the SAR ADC clock architecture.

To avoid crosstalk issues, all SAR ADCs run at the same clock frequency. The SAR digital interface is clocked from the PBRIDGE*n*_CLK clock. All SAR analog blocks are clocked from the output of a single divider of the SAR ADC CLK at the device level.

Figure 483. SAR ADC clock diagram



40.2.2.5 SAR ADC triggering

Each SAR ADC has 2 independent trigger inputs for starting normal conversion or injected conversion on an analog input. One trigger source can be selected among a list of available signals and connected to the trigger inputs independently for each SAR ADC.

The trigger sources for each SAR ADC on the device are either external input pins, eMIOS or GTM channels. Each external input can selectively trigger any of the SAR ADCs on the device. The synchronization to the PBRIDGE clock and digital filtering of the external input pin triggers is done in the system integration logic.

Each trigger input supports software selectable options for rising edge, falling edge, rising and falling edge, or level-based trigger assertion. The trigger selection for each SAR ADC and edge/level selection is located in the SIUL2.

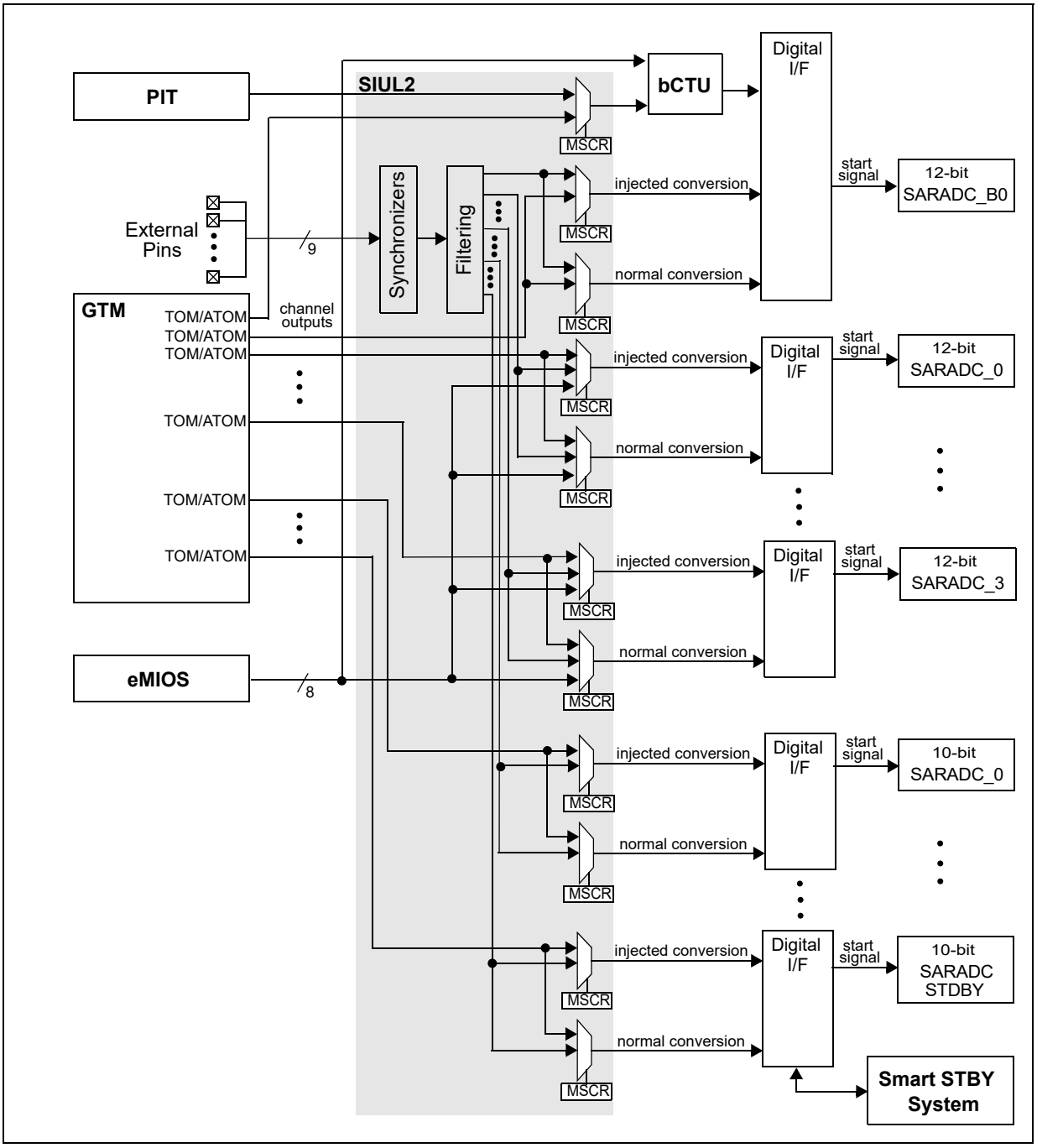
The BCTU module is connected to SAR_ADC_12bit_B0 (SAR_ADC parameter CTSEN = 0). This module receives trigger requests from the timer module (PIT_RTI), from the generic timer module (GTM-TOM) or from EMIOS. It translates those trigger requests into ADC measurement commands.

The STBY eCTU module is connected to SAR_ADC_10bit_STDBY (SAR_ADC parameter CTSEN = 2). It receives trigger requests from the real time clock module (RTC) and translates them into ADC commands. The STBY eCTU is used for the Smart Standby feature. Please refer to the *Chapter 11: Smart Stand-by Wake-up Unit (SSWU)* for more data about this feature.

The SPC58xEx/SPC58xGx SoC diagram for the SAR ADC trigger inputs is given in *Figure 484*.

Note: Please refer to the MS Excel file attached to SPC58xEx/SPC58xGx IO_Definition document (I/O signal table) to get the exhaustive list of signals source able to trigger ADC conversion.

Figure 484. SAR ADC input trigger diagram



Trigger source selection for each SAR ADC is given in the *Chapter 17: System Integration Unit Lite2 (SIUL2)*.

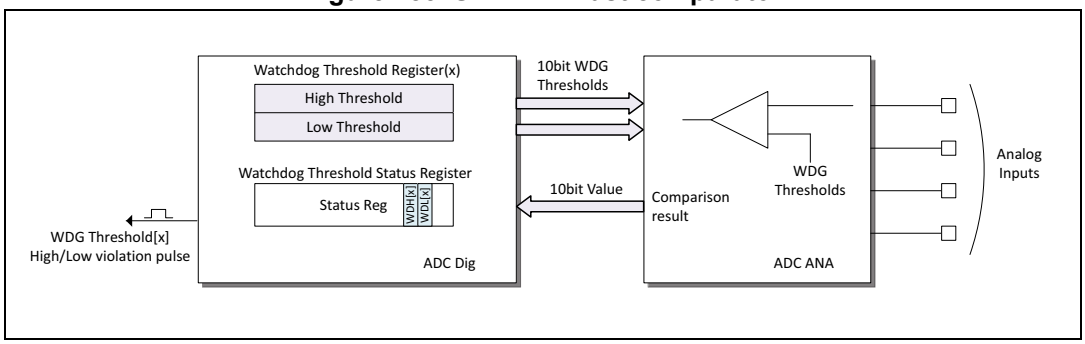
Each SAR ADC digital interface, except SAR_ADC_12bit_B0, provides 4 watchdog Threshold Crossover event trigger signals. These signals can be routed to the GTM module (TIM interface) via SIUL2. Selection of watchdog trigger event for each GTM TIM interface is provided in the *Chapter 17: System Integration Unit Lite2 (SIUL2)*.

40.2.2.6 SAR ADC fast comparator

The fast comparator mode feature is only available with the 10-bit SAR ADCs.

The fast comparator mode provides fast signal evaluation with respect to two thresholds configurable by the user. The result of the comparison is reflected by two status bits in the Watchdog Threshold Status Register and a watchdog pulse that is generated out of the ADC and forwarded to the GTM. SAR ADC fast comparator scheme is shown below.

Figure 485. SAR ADC fast comparator



40.2.2.7 SAR ADC Standby

SAR_ADC_10bit_STDBY interactions with the Smart STBY System are described in the *Chapter 11: Smart Stand-by Wake-up Unit (SSWU)*

40.2.2.8 SAR absolute voltage reference

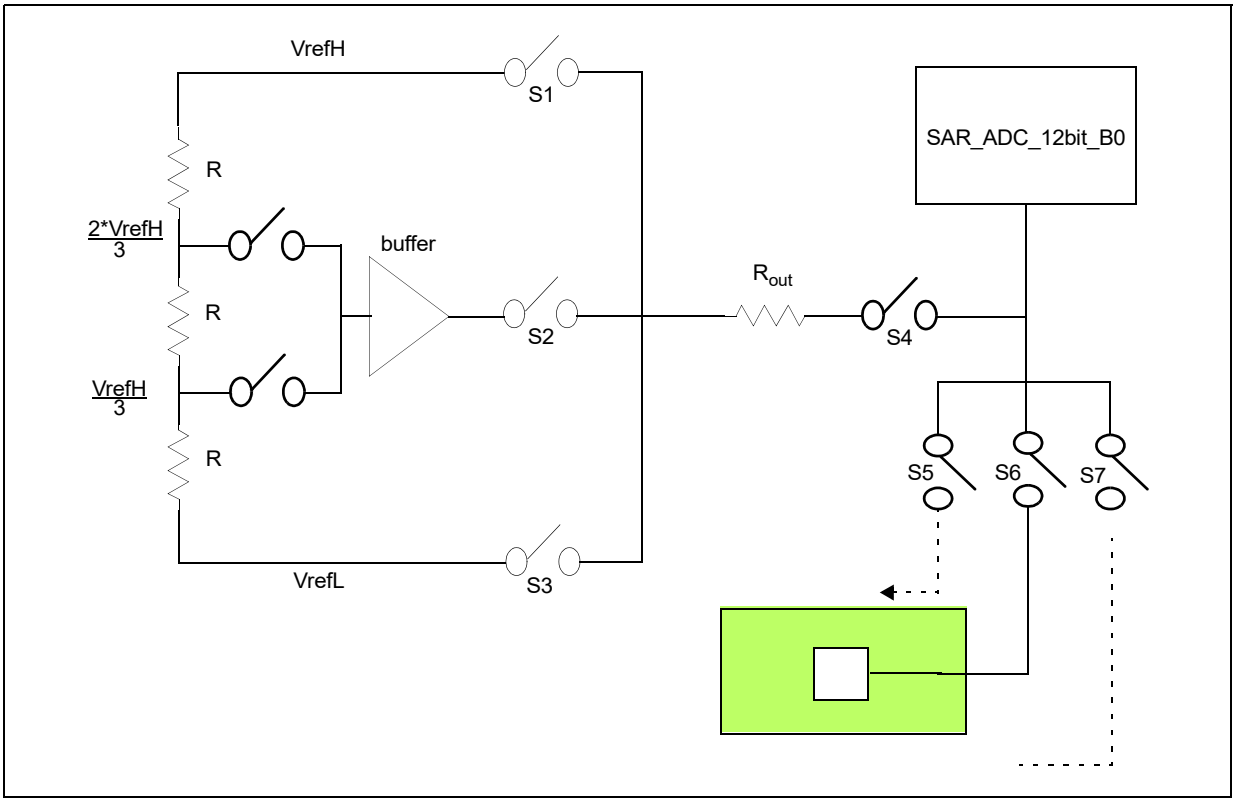
SPC58xEx/SPC58xGx provides an internal, absolute voltage reference that can be calculated from two digital temperature sensor signal values. The temperature sensor signal values can be read by SAR_ADC_12bit_B0. For further information refer to *Chapter 43: Temperature Sensor*. The ADC conversion channel numbers for the temperature sensor signals are given in *Table 598*.

40.2.2.9 SAR ADC diagnostic

40.2.2.9.1 Self Test features

- The ADCBIAS provides four different voltage levels defined as GND, $V_{ref}/3$, $2*V_{ref}/3$ and V_{ref}. The self test internal reference voltage precision is provided in the SAR ADC electrical specification table of the device datasheet.
- For the impedance requirements please refer to SPC58xEx/SPC58xGx datasheet.
- The impedance of the switches that are connecting the different references to the ADC input has not been considered in the impedance calculation.
- The impedance of the switches from the reference generation to the ADC input has been matched with the one of the switches that are connecting the device pin to the ADC input.
- The sampling time of SAR_ADC_12bit_B0 needs to be extended due to long settling time induced by the high impedance of the source. For sampling time adjustment, please refer to SPC58xEx/SPC58xGx datasheet.
- The SAR ADC Digital Interface provides the necessary control to select the required switches S1–S7 (shown in *Figure 486*) to perform the Self Test.

Figure 486. Self Test implementation



40.2.2.9.2 Internal reference

SAR_ADC_12bit_B0 provides the ability to sample and convert four internal voltages through a 20 kΩ source impedance. These voltages are as follows:

- $V_{DD_HV_ADR_S}$
- 1/3 $V_{DD_HV_ADR_S}$
- 2/3 $V_{DD_HV_ADR_S}$
- $V_{SS_HV_ADR_S}$

Conversion of an internal reference voltage works the same as that for a normal conversion, with the exception of the conversion time. Due to the 20 kΩ source impedance, the sampling time is much higher than the normal time.

Enabling the sampling and selection of an internal reference is independent of input channel selection for SAR_ADC_12bit_B0.

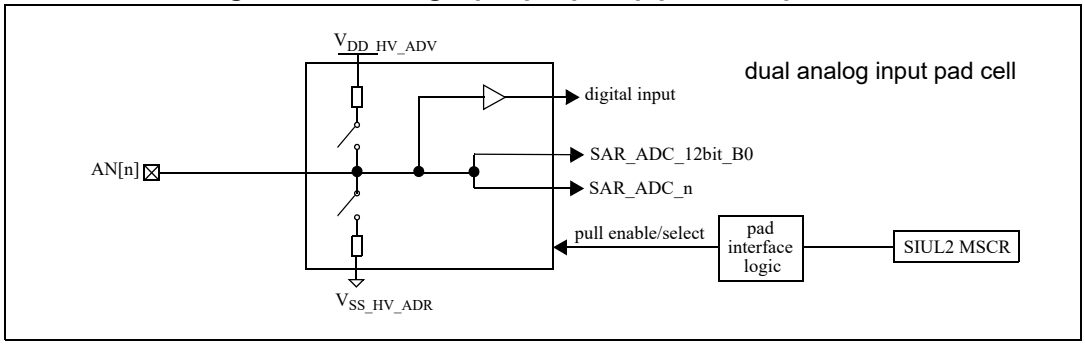
40.2.2.9.3 Analog input pin programmable pullup/pulldown

An open or short circuit condition is detectable within the device for analog input pins. There is a programmable pullup/pulldown pad cell at the input pad. This allows for application testing of the mux input logic. The analog input pin diagram with the pullup/pulldown is shown in *Figure 487*. The pullup/pulldown is included inside the analog input pad cell.

Control of the pullup/pulldown is independently done in the SIUL2 MSCR registers for each analog input pin.

Only the analog input pins specified in *Section 40.1.2: Analog input pin multiplexing* have the pullup/pulldown feature.

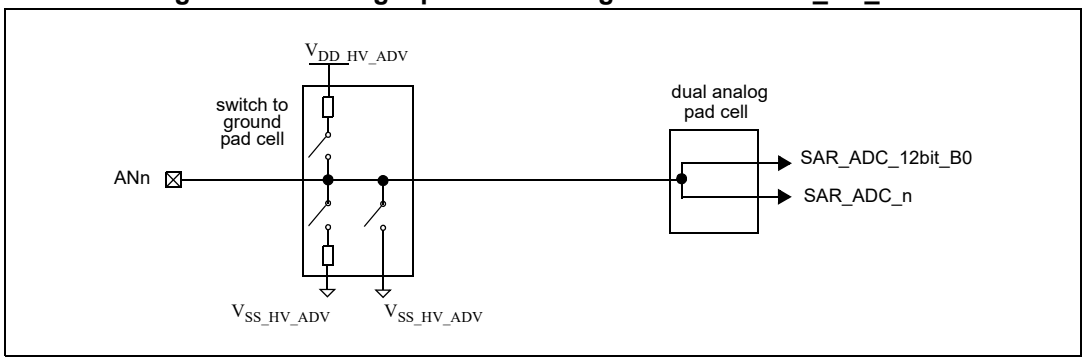
Figure 487. Analog input pin pullup/pulldown pad cell



40.2.2.9.4 Analog input pin switch to $V_{SS_HV_ADV}$

All analog input-only pins on SPC58xEx/SPC58xGx have in addition to the weak pullup/pulldown a programmable switch to $V_{SS_HV_ADV}$ to discharge the input. The analog input diagram is given in *Figure 488*. The default state of the switch for both inputs is OFF.

Figure 488. Analog input — discharge switch to VSS_HV_ADV



The ground switches are controlled independently from the input switch to the ADC. The ground switch is controlled in input multiplexing MSCR registers (MSCR[WPDE], MSCR[WPUE]) in the SIUL2. The MSCR register provides selection between open or closed switch, with open being the default state.

The ground switches can be used to periodically discharge the associated channels.

40.2.2.10 SAR ADC channel assignment

40.2.2.10.1 Internal channel assignment

The input multiplexer channel assignments for the SAR ADCs is provided in *Table 597*.

Table 597. SAR analog input channel assignment

Analog input pin	SAR_ADC_12bit_B0 input channel	Fast SAR	Fast SAR channel	Description
AN[3:0]	3–0	—	—	Analog input pin
AN[7:4]	7–4	SAR_ADC_12bit_3	7–4	Analog input pin

Table 597. SAR analog input channel assignment (continued)

Analog input pin	SAR_ADC_12bit_B0 input channel	Fast SAR	Fast SAR channel	Description
AN[15:8]	15–8	SAR_ADC_12bit_0	15–8	Analog input pin
AN[19:16]	19–16	SAR_ADC_10bit_0	19–16	Analog input pin
AN[21:20]	21–20	—	—	Analog input pin
AN[23:22]	23–22	SAR_ADC_12bit_1	23–22	Analog input pin
AN[25:24]	25–24	—	—	Analog input pin
AN[31:26]	31–26	SAR_ADC_12bit_1	31–26	Analog input pin
AN[35:32]	35–32	SAR_ADC_10bit_1	35–32	Analog input pin
AN[39:36]	39–36	—	—	Analog input pin
AN[43:40]	43–40	SAR_ADC_12bit_2	43–40	Analog input pin
AN[47:44]	47–44	—	—	Analog input pin
AN[51:48]	51–48	SAR_ADC_12bit_2	51–48	Analog input pin
AN[53:52]	53–52	—	—	Analog input pin
AN[59:54]	59–54	SAR_ADC_12bit_2	59–54	Analog input pin
AN[61:60]	61–60	—	—	Analog input pin
—	62	—	—	Reserved
AN[63]	63	SAR_ADC_12bit_2	63	Analog input pin
—	64	—	—	Reserved
AN[65]	65	SAR_ADC_12bit_3	65	Analog input pin
—	67–66	—	—	Reserved
AN[73:68]	73–68	SAR_ADC_12bit_3	73–68	Analog input pin
—	75–74	—	—	Reserved
AN[79:76]	79–76	SAR_ADC_12bit_3	79–76	Analog input pin
—	87–80	—	—	Reserved
AN[95:88]	95–88	SAR_ADC_10bit_STDBY	95–88	Analog input pin

40.2.2.10.2 Test channel assignments

Each test channel has a unique channel assignment, and is treated by software like any other A/D conversion. Access to the test channels is required for SAR_ADC_12bit_B0 only. The test channel assignments for SAR_ADC_12bit_B0 are given in *Table 598*.

Table 598. SAR_ADC analog test channel assignment

SAR_ADC_12bit_B0 input channel	Description
96	Reserved
97	Reserved

Table 598. SAR_ADC analog test channel assignment (continued)

SAR_ADC_12bit_B0 input channel	Description
98	Reserved
99	Reserved
100	Reserved
101	Reserved
102	Reserved
103	Reserved
104	VDD_LV
105	Reserved
106	Reserved
107	Reserved
108	Reserved
109	VSS_HV_ADV (Ground Supply for ADC)
110	Reserved
111	Reserved
112	Reserved
113	Reserved
114	Reserved
115	Reserved
116	Reserved
117	Reserved
118	Reserved
119	Reserved
120	PTAT (Proportional To Absolute Temperature)
121	Temperature sensor CTAT (Complementary To Absolute Temperature)
122	VSS_LV
123	Reserved
124	SAR BIAS 0 — VSS_HV_ADR_S through 20 KΩ source impedance
125	SAR BIAS 1 — 1/3 (VDD_HV_ADR_S - VSS_HV_ADR_S) through 20 KΩ source impedance
126	SAR BIAS 2 — 2/3 (VDD_HV_ADR_S - VSS_HV_ADR_S) through 20 KΩ source impedance
127	SAR BIAS 3 — (VDD_HV_ADR_S - VSS_HV_ADR_S) through 20 KΩ source impedance

41 Sigma-Delta Analog-to-Digital Converter (SDADC) Digital Interface

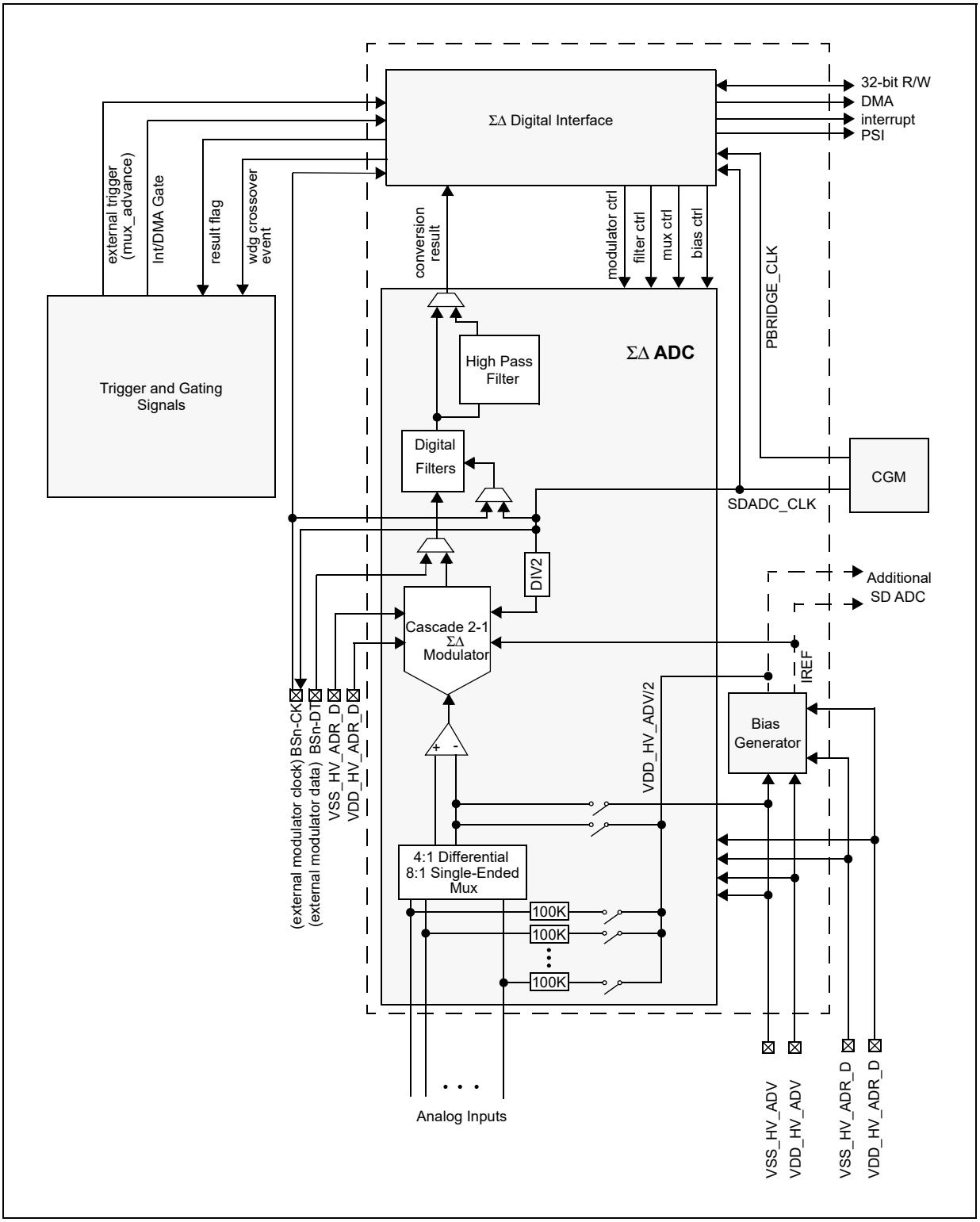
41.1 Introduction

The Sigma-Delta Analog-to-Digital Converter (SDADC) digital interface block controls the on-chip SDADC and holds control and status registers accessible for application. It provides accurate conversion data for a wide range of applications.

41.2 Overview

Figure 489 depicts the block diagram of an SDADC and its digital control block. The SDADC digital interface block provides all control information for operating mode selection (single-ended or differential), analog input gain, decimation rate, high-pass filter enable, and so on. It also generates the bias controls for common mode voltage selection, to bias the positive and negative terminals to different voltage levels (for example, half scale bias, reference ground, reference supply) for diagnostics. It generates the analog multiplexer control signals for the selected channels. It also provides external modulator support and keeps the internal modulator in low consumption mode. To assist synchronous operation across multiple ADC blocks, hardware trigger support is implemented to start data conversion.

Figure 489. SDADC block diagram



41.3 Features

- 16-bit data resolution output
- Single-ended input or differential input mode of operation
- Programmable wraparound mechanism for both modes of operation
  - Configurable trigger sources: hardware or software
  - Configurable initial entry and wraparound values for the loop
- Configurable biasing for negative input terminal in single-ended mode
- Gain and offset calibration support using fixed bias for input terminals
- Programmable decimation rate
- Cascaded COMB + FIR decimation filter with an optional FIR filter for the SD ADC data path
- Programmable gain for analog inputs
- Optional external modulator support
- Optional high-pass filter for pure AC application
- Hardware trigger support for synchronous operation of multiple SDADC blocks
- Trigger event output generation by software
- Programmable FIFO structure for storing 16-bit converted data
- Left/right data alignment
- Signed/unsigned format of converted data output
- Interrupt/DMA request generation based on various conditions:
  - Data buffer at or above threshold
  - Data buffer overrun
  - WDG crossover event
- Programmable threshold range WDG support
- Low consumption mode
- Optional conversion process halt mechanism on SoC debug request

41.4 Modes of operation

This section describes the different operation modes of the SDADC. When exiting reset, the mode of operation is determined by the Module Configuration Register (MCR).

41.4.1 Differential input mode

This mode is entered by negating the MCR[MODE] bit. In this mode, a pair of analog inputs are connected to positive and negative terminals of ADC modulator. In order to support gain and offset calibration for diagnostics, it is possible in this mode to select fixed bias voltages to both input terminals.

41.4.2 Single-ended input mode

This mode is entered by asserting the MCR[MODE] bit. In this mode, the negative input terminal is biased with a fixed voltage based on selection information in MCR, and the other analog input is connected to the selected external analog channel.

41.4.3 External modulator mode

This mode is entered by asserting the MCR[EMSEL] bit. In this mode, the data stream and clock outputs from the external modulator available on BSEXT/CLKEXT input pins are routed directly to the SDADC block. The internal modulator is bypassed in this mode. Only the digital filters of the SDADC will be used for result calculation. The source for the external modulator clock can be the internal on-chip clock source or the direct clock output coming from the external modulator. The external pin function control for these pins must be configured in the integration logic on a device, and is not included in the SDADC digital interface.

41.5 External signal description

Table 599 shows the list of signals driven by external pins. At the chip integration level, some of the digital and analog signals might share pins or may not be available external to the chip.

Table 599. Signal properties

Name[1]	Function	I/O	Reset	Pull-up
AIN[0:7]	Single-ended analog inputs or Differential pairs of analog inputs	I	—	—
BSEXT	External modulator data input	I	0	—
CLKEXT	External modulator clock input	I	0	—
VREFP = VDD_HV_ADR_D	Voltage reference for SDADC	I	—	—
VREFN = VSS_HV_ADR_D	Ground reference for SDADC	I	—	—
VDD_HV_ADV	Analog voltage supply for SDADC	I	—	—
VSS_HV_ADV	Analog ground supply for SDADC	I	—	—
VDD_LV	Digital voltage supply for SDADC	I	—	—
VSS_LV	Digital ground supply for SDADC	I	—	—

1. See *Section 40.2.1.1: Sigma-Delta ADC integration diagram*.

41.5.1 Detailed signal descriptions

Table 600 provides a detailed description of the external signals.

Table 600. Detailed signal descriptions

Signal	I/O	Description	
AIN[0:7]	I	Each analog input can be used as a single-ended analog input or one of a differential pair of analog inputs which is connected to either positive or negative terminals of the SDADC, depending on the mode of operation and polarity selection.	
		State meaning	Analog input
		Timing	Can be sampled at any time by the ADC.
BSEXT	I	Data stream input provided by external modulator.	
		State meaning	Asserted—Data input is 1 Negated—Data input is 0
		Timing	Assertion—May occur at any time, but data stream is valid only when external modulator is selected by MCR[MODSEL] = 1. Negation—Ignored when external modulator is not selected.
CLKEXT	I	Clock input provided by external modulator.	
		State meaning	Asserted—clock high level Negated—clock low level
		Timing	Assertion—May occur at any time, but clock is valid only when external modulator is selected by MCR[MODSEL] = 1. Negation—Ignored when external modulator is not selected.

41.6 Memory map and register descriptions

41.6.1 Memory map

This section provides the memory map for the SDADC block.

Table 601. SDADC memory map

Address Offset	Register name	Width (in bits)	Access	Reset value	Section
0x0000	Module Configuration Register (SDADC_MCR)	32	R/W	0x0000_0000	*Section 41.6.2*
0x0004	Channel Selection Register (SDADC_CSR)	32	R/W	0x0000_0000	*Section 41.6.3*
0x0008	Reset Key Register (SDADC_RKR)	32	R/W	0x0000_A50F	*Section 41.6.4*
0x000C	Status Flag Register (SDADC_SFR)	32	R/W	0x0000_0100	*Section 41.6.5*
0x0010	Request Select and Enable Register (SDADC_RSER)	32	R/W	0x0000_0000	*Section 41.6.6*
0x0014	Output Settling Delay Register (SDADC_OSDR)	32	R/W	0x0000_0000	*Section 41.6.7*
0x0018	FIFO Control Register (SDADC_FCR)	32	R/W	0x0000_0006	*Section 41.6.8*
0x001C	Software Trigger Key Register (SDADC_STKR)	32	R/W	0x0000_0000	*Section 41.6.9*
0x0020	Converted Data Register (SDADC_CDR)	32	R	0x0000_0000	*Section 41.6.10*
0x0024	WDG Threshold Register (SDADC_WTHHLR)	32	R/W	0x0000_0000	*Section 41.6.11*

41.6.2 Module Configuration Register (SDADC_MCR)

The Module Configuration register (MCR) consists of different control bits to select the operating modes and various configuration settings which define the behavior of the SDADC block.

Offset 0x0000 Access: User Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	Reserved	Reserved	0	PDR					0	PGAN			ODF	ODA	EMSEL	HPFEN
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	WDGEN	TRIGEDSEL		TRIGEN	TRIGSEL				FRZ	Reserved	Reserved	VCOMSEL	WRMODE	GECEN	MODE	EN
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 490. Module Configuration Register (SDADC_MCR)

Table 602. SDADC_MCR field descriptions

Field	Description
3:7 PDR	Programmable Decimation Rate This field selects the over-sampling ratio to be applied to support different pass-bands with a fixed input sampling clock. The output data rate f_d is equal to f_s / (2×OSR), where f_s is the input sampling clock frequency. When external modulator is selected, the output data rate is independent of this field and is fixed to f_s / 256. 00000 OSR = 24 00001 OSR = 28 00010 OSR = 32 00011 OSR = 36 00100 OSR = 40 00101 OSR = 44 00110 OSR = 48 00111 OSR = 56 01000 OSR = 64 01001 OSR = 72 01010 OSR = 75 01011 OSR = 80 01100 OSR = 88 01101 OSR = 96 01110 OSR = 112 01111 OSR = 128 10000 OSR = 144 10001 OSR = 160 10010 OSR = 176 10011 OSR = 192 10100 OSR = 224 10101 OSR = 256 10110 OSR = 512 10111 OSR = 1024 11000-11111 Reserved
9:11 PGAN	Programmable Gain This field selects the gain to be applied to the analog input stage of the SDADC. The effective analog input becomes the input voltage level multiplied by the gain factor. **Note:** When SDADC_MCR[PGAN]=111b (programmable gain is 16), the least significant bit of the conversion data in SDADC_CDR[CDATA] is always 0. 000 Gain = 1 001 Gain = 2 010 Gain = 4 011 Gain = 8 100 Reserved 101 Reserved 110 Reserved 111 Gain = 16
12 ODF	Output Data Format ODF selects the representation for digital part. 0 Output data is unsigned. 1 Output data is signed and sign extended to 16 bits.

Table 602. SDADC_MCR field descriptions (continued)

Field	Description
13 ODA	Output Data Alignment 0 Output data is right-aligned. 1 Output data is left-aligned.
14 EMSEL	External Modulator Selection 0 External modulator data and clock inputs are ignored. 1 External modulator data stream and clock inputs are provided to SDADC.
15 HPFEN	High Pass Filter Enable 0 High-pass (DC removal) filter is disabled. 1 High-pass (DC removal) filter is enabled.
16 WDGEN	Watchdog Enable This bit enables the WDG monitor. 0 WDG disabled 1 WDG is enabled
17:18 TRIGEDSEL	Trigger Edge Selection 00 Falling edge of trigger input is selected. 01 Rising edge of trigger input is selected. 10 Both edges of trigger input are selected. 11 Both edges of trigger input are selected.
19 TRIGEN	Trigger Enable This field enables the hardware trigger input to initiate a fresh conversion. Upon receiving a trigger event, SDADC reset input is asserted and deasserted synchronously with respect to the peripheral clock. The reset pulse width is fixed to four peripheral clock cycles. The trigger event is ignored if the SDADC internal modulator is not enabled (MCR[EN] = 0) or the external modulator is selected (MCR[EMSEL] = 1). 0 Trigger input is disabled 1 Trigger input is enabled

Table 602. SDADC_MCR field descriptions (continued)

Field	Description
20:23 TRIGSEL	Trigger Input Selection This field selects which input will be used for hardware-triggered conversions. 0000 Input trigger 0 is selected. See chip-specific information for trigger source. 0001 Input trigger 1 is selected. See chip-specific information for trigger source. 0010 Input trigger 2 is selected. See chip-specific information for trigger source. 0011 Input trigger 3 is selected. See chip-specific information for trigger source. 0100 Input trigger 4 is selected. See chip-specific information for trigger source. 0101 Input trigger 5 is selected. See chip-specific information for trigger source. 0110 MUX_ADV 0111 Reserved 1000 Reserved 1001 Reserved 1010 Reserved 1011 Reserved 1100 Reserved 1101 Reserved 1110 Reserved 1111 Reserved **Note:** MUX_ADV is GTM trigger selectable vie SIUL MCR. Refer to the SPC58xEx/SPC58xGx pinout Microsoft Excel file attached to the IO_Definition document, for more details.
24 FRZ	Freeze This field enables stopping the SDADC conversions at the end of the current channel conversion when the SoC enters debug mode. When the SoC enters debug mode, further conversions by the SDADC analog block will be stopped, and converted data output from the SDADC analog block are ignored. The ongoing channel conversion will be completed and the converted data is stored. When the SoC exits debug mode, SDADC resumes the job that was left before halting the conversions. 0 Conversions are not stopped 1 Conversions are stopped
27 VCOMSEL	Common Voltage Bias Selection VCOMSEL selects the reference for analog part. This field selects the common voltage bias for the negative input terminal of the SDADC during single-ended mode (MODE = 1). 0 Negative input terminal is biased with VREFN. 1 Negative input terminal is biased with VREFP/2 (half-scale bias).
28 WRMODE	Wrap-Around Mode This bit selects the wraparound mechanism for conversion of programmed sequence of channels. 0 Wraparound mechanism disabled (in this case the default software control mechanism will be enabled) 1 Wraparound mechanism enabled

Table 602. SDADC_MCR field descriptions (continued)

Field	Description
29 GECEN	Accurate Gain Error Mode Enable This bit selects the accurate gain error mode, and must be set during gain error calibration as well as during conversions requiring accurate gain error mode. It can be kept '0' if gain error calibration mode is not needed. 0 Gain error calibration mode disabled 1 Gain error calibration mode enabled
30 MODE	Mode selection 0 Differential input mode selected 1 Single-ended input mode selected
31 EN	Enable for SDADC block Refer to *Section 41.8: Initialization information* for details on configuration sequence. 0 SDADC internal modulator placed in low consumption mode 1 SDADC internal modulator enabled

41.6.3 Channel Selection Register (SDADC_CSR)

The Channel Selection Register (CSR) selects analog inputs AIN[0:7] to be connected to positive and negative terminals of the SDADC and controls the biasing of analog inputs to half-scale bias through 100 Kohm resistance.

Offset 0x0004 Access: User Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	BIASEN							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	ANCHSEL_WRAP			0	0	0	0	0	ANCHSEL		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 491. Channel Selection Register (SDADC_CSR)

Table 603. SDADC_CSR field descriptions

Field	Description
8:15 BIASEN	Bias enable If BIASEN[i] = 1, analog input AIN[i] (could be single-ended input or differential analog input) is connected to half-scale bias through a 100 KΩ resistor.

Table 603. SDADC_CSR field descriptions (continued)

Field	Description
21:23 ANCHSEL_WRAP	Analog Channel Selection Wraparound Value (ANCHSEL_WRAP) When in wraparound mode, this field indicates the maximum value of the wraparound counter, after which it will wrap around back to initial value 000. This value should be programmed before enabling wraparound mechanism (MCR[WRMODE] = 1).
29:31 ANCHSEL	Analog Channel Selection (ANCHSEL) Based on single-ended or differential mode of operation (MCR[MODE]), common mode voltage selection (MCR[VCOMSEL]), this field defines the connectivity of analog inputs to either positive or negative polarity terminals of the SDADC according to *Table 604*. Wraparound Mode: – This field also indicates the initial entry value for the first loop of the wraparound sequence from where the wraparound counter will start incrementing when wraparound mode is entered (when wraparound mechanism enabled (MCR[WRMODE] = 1)). – If this field is written any time during wraparound mode (when MCR[WRMODE] = 1), the present value of the wraparound counter will be updated with the value of ANCHSEL — the next trigger will act on the updated channel, and the next increment will be on the updated channel. Here, the value of ANCHSEL chosen should be < ANCHSEL_WRAP; otherwise it will be ignored by hardware. – Wraparound will always be from initial counter value 000.

Table 604. Analog input AIN[0:7] selection

MODE	VCOMSEL	ANCHSEL	INP (positive terminal)	INM (negative terminal)
1	0	000	AIN[0]	VREFN
		001	AIN[1]	
		010	AIN[2]	
		011	AIN[3]	
		100	AIN[4]	
		101	AIN[5]	
		110	AIN[6]	
		111	AIN[7]	
	1	000	AIN[0]	VREFP/2
		001	AIN[1]	
		010	AIN[2]	
		011	AIN[3]	
		100	AIN[4]	
		101	AIN[5]	
		110	AIN[6]	
		111	AIN[7]	
0	0/1	000	AIN[0]	AIN[1]
		001	AIN[2]	AIN[3]
		010	AIN[4]	AIN[5]
		011	AIN[6]	AIN[7]
		100	VREFN	VREFN
		101	VREFP/2	VREFP/2
		110	VREFP	VREFN
		111	VREFN	VREFP

41.6.4 Reset Key Register (SDADC_RKR)

The Reset Key Register (RKR) is the target for a predefined key write operation used to reset the SDADC to start a fresh conversion.

Offset 0x0008 Access: User Read/Write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	RESET_KEY															
W																
Reset	1	0	1	0	0	1	0	1	0	0	0	0	1	1	1	1

Figure 492. Reset Key Register (SDADC_RKR)

Table 605. SDADC_RKR field descriptions

Field	Description
16:31 RESET_KEY	Reset Key This field, when written with 0x5AF0, is used to generate the reset for the SDADC block and to reload the internal counter with the start value programmed in OSDR. Any write access to this register which is different from the predefined key is ignored. Read access will always return the inverted key.

41.6.5 Status Flag Register (SDADC_SFR)

The Status Flag Register (SFR) contains status and flag bits. These bits are set by hardware and reflect the status of the SDADC and indicate the occurrence of events that may generate an interrupt or a DMA request. Software can clear some flags by writing ‘1’ to those bits. Writing a ‘0’ to a flag has no effect.

Offset 0x000C Access: User Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	ANCHSEL_CNT		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	DFEF	0	0	0	WTHH	WTHL	CDVF	DFORF	DFFF
W																
Reset	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0

Figure 493. Status Flag Register (SDADC_SFR)

Table 606. SDADC_SFR field descriptions

Field	Description
13:15 ANCHSEL_CNT	Analog Channel Selection Counter (ANCHSEL_CNT) This field reflects the current value of the internal Analog Channel Selection Counter. It indicates the present number of the channel selected through the analog multiplexer.
23 DFEF	Data FIFO Empty Flag This field is asserted when there is no data word in the FIFO. 0 Data FIFO is not empty. 1 Data FIFO is empty.
27 WTHH	Watchdog Upper Threshold Cross Over Event This field is set when Converted Data crosses Upper Threshold of WDG. This bit automatically gets cleared on arrival of ipd_done when current ongoing DMA transfer completes, It can also be cleared by software writing ‘1’ to clear. 0 Watchdog Upper Threshold Cross Over Event did not occur. 1 Watchdog Upper Threshold Cross Over Event occurred.
28 WTHL	Watchdog Lower Threshold Cross Over Event This field is set when Converted Data crosses Lower Threshold of WDG. This bit automatically gets cleared on arrival of ipd_done when current ongoing DMA transfer completes. It can be cleared by software writing ‘1’ to clear. 0 Watchdog Lower Threshold Cross Over Event did not occur. 1 Watchdog Lower Threshold Cross Over Event occurred.
29 CDVF	Converted Data Valid Flag This field gives the actual status of data coming from the SDADC block. The converted data is not valid or becomes corrupted whenever there is a change in configuration settings determined by MCR, channel selection programmed in CSR. Any write access to the registers MCR and CSR will reset this flag. This flag is set automatically after generating SDADC reset and internal timer counts OSD cycles of output clock f_d. 0 Data output from SDADC is not valid. 1 Data output from SDADC is valid.

Table 606. SDADC_SFR field descriptions (continued)

Field	Description
30 DFORF	Data FIFO Overrun Flag This field indicates that software or DMA failed to prevent FIFO from overflowing with converted data words. Software can clear this bit by writing ‘1’ to it after reading the FIFO contents through CDR. This flag remains set if software or DMA failed to clear the condition that caused this flag setting. If DFORF is set, further datawords will not be received into FIFO even if sufficient room exists. 0 No overrun has occurred since the last time the flag was cleared. 1 Overrun has occurred or the DFORF has not been cleared since the last overrun occurred.
31 DFFF	Data FIFO Full Flag This bit is set when the number of converted data words in the FIFO is equal to o greater than the number of data words indicated by FCR[FTHLD] value. Software can clear this bit by writing ‘1’ to it. This bit is also cleared by an acknowledgment from the DMA controller when DMA request generation is selected. This bit remains set if software or DMA failed to clear the condition that caused this flag setting. Even if DFFF is set, data words will continue to be received until an overrun condition occurs. Data words will not be received if the converted data is not valid (CDVF = 0). 0 The number of data words in FIFO is less than or equal to the number indicated by FCR[FTHLD]. 1 The number of data words in FIFO is greater than the number indicated by FCR[FTHLD] at some point in time since this flag was last cleared. **Note:** Whenever MCR[EN] is cleared in order to stop the SDADC or change the channel configuration, it is necessary to clear the RSER[DFFDIRE] bit as well. Similarly if DMA or software is too slow in not reading the FIFO data words resulting in DFFF condition it is advisable to stop the SDADC by clearing the MCR[EN] followed by clearing of RSER[DFFDIRE] bit. Both these action will ensure safe operation.

41.6.6 Request Select and Enable Register (SDADC_RSER)

The Request Select and Enable Register (RSER) serves two purposes:

- Enables the flag bits in SFR to generate DMA requests or interrupt requests.
- Selects the type of request to generate.

Refer to the field descriptions for the type of requests that are supported.

Offset 0x0010 Access: User Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	WTHDIRS	DFFDIRS
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	GDIGE	0	0	0	0	0	0	0	0	0	0	0	WTHDIRE	CDVEE	DFORIE	DFFDIRE
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 494. Request Select and Enable Register (SDADC_RSER)

Table 607. SDADC_RSER field descriptions

Field	Description
14 WTHDIRS	WDG Threshold Cross Over Event DMA/Interrupt Request Select When SFR[WTHH/WTHL] and RSER[WTHDIRE] fields are set, this field selects between generating a DMA request or an interrupt request. 0 Interrupt request is selected. 1 DMA request is selected.
15 DFFDIRS	Data FIFO Full DMA/Interrupt Request Select When the SFR[DFFF] and RSER[DFFDIRE] fields are set, this field selects between generating a DMA request or an interrupt request. 0 Interrupt request is selected. 1 DMA request is selected.
16 GDIGE	Global DMA/Interrupt Gating Enable This field determines whether all module interrupts and DMA requests need to be gated by an external gating signal or not. 0 No impact of external gating signal on module DMA/interrupt requests. 1 All module DMA/interrupt requests are qualified only when external gating signal is asserted.
28 WTHDIRE	WDG Threshold Cross Over Event DMA/Interrupt Request Enable This field enables the SFR[WTHH/WTHL] field to generate a request. The WTHDIRS field determines which is selected, a DMA request or an interrupt request. The final DMA/interrupt request also depends on the gating signal if enabled by the GGE and DIRFWGS fields. 0 Interrupt/DMA request is disabled on WDG Threshold Cross Over Event. 1 Interrupt/DMA request is enabled on WDG Threshold Cross Over Event.
29 CDVEE	Conversion Data Valid Event Enable The status of SFR[CDVF] gated with RSER[CDVEE] is passed on as an event on “conversion_flag" output port, which can be used at SoC level for indication of valid data conversion window. Once enabled, the event output will remain asserted as long as valid data conversions are ongoing. 0 Event output disabled 1 Event output assertion/deassertion based on the SFR[CDVF] field

Table 607. SDADC_RSER field descriptions (continued)

Field	Description
30 DFORIE	Data FIFO Overrun Interrupt Enable This bit enables the SFR[DFORF] field to generate an interrupt request. The final interrupt request also depends on the gating signal if enabled by the GGE and DIRFWGS fields. 0 Interrupt request is disabled when data FIFO overrun condition occurs. 1 Interrupt request is enabled when data FIFO overrun condition occurs.
31 DFFDIRE	Data FIFO Full DMA/Interrupt Request Enable This field enables the SFR[DFFF] field to generate a request. The DFFDIRS field determines which is selected, a DMA request or an interrupt request. The final DMA/interrupt request also depends on the gating signal if enabled by the GGE and DIRFWGS fields. To have safe operation this bit should not be set when MCR[EN]=0. Whenever MCR[EN] is cleared this bit should also be cleared if already set and FCR[FRST] can be set to clear the previous FIFO content. 0 Interrupt/DMA request is disabled when data FIFO full condition occurs. 1 Interrupt/DMA request is enabled when data FIFO full condition occurs.

41.6.7 Output Settling Delay Register (SDADC_OSDR)

The Output Settling Delay Register (OSDR) provides a delay value to qualify the converted output data coming from the SDADC.

Offset 0x0014 Access: User Read/Write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	OSD							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 495. Output Settling Delay Register (SDADC_OSDR)

Table 608. SDADC_OSDR field descriptions

Field	Description
24:31 OSD	Output Settling Delay This field defines the delay to qualify the conversion data stored in CDR. Whenever the SDADC block is reset in order to start the conversion from a fresh state, an internal timer is loaded with this start value. The counter counts down with output clock f_d until it reaches zero, and then it generates a flag which qualifies the converted data. The output clock f_d is equal to f_s / (2 x OSR), where f_s is the input sampling clock frequency. When external modulator is selected, the output clock is fixed to f_s / 256. See the electrical specifications for the settling delays.

41.6.8 FIFO Control Register (SDADC_FCR)

The FIFO Control Register (FCR) provides the ability to enable or disable FIFO functionality, and to control the amount of available FIFO depth to be used.

Note: It is important to note that the FIFO depth is set at the factory and cannot be changed but the amount of available FIFO space actually used can be set by specifying the value in the FTHLD field. The FSIZE field indicates the amount of FIFO space available. By setting the value of FTHLD to be less than the FIFO depth indicated by FSIZE, the FIFO is considered to be full when the number of data words in the FIFO is equal or greater than FTHLD.

This register may be read at any time, but should only be written when SDADC is disabled (MCR[EN] = 0).

Offset 0x0018 Access: User Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																FRST
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	FTHLD				0	0	0	0	FOWEN	FSIZE		FE
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0

Figure 496. FIFO Control Register (SDADC_FCR)

Table 609. SDADC_FCR field descriptions

Field	Description
15 FRST	FIFO Flush Reset This is functional soft reset to flush the current FIFO content and initialize the internal FIFO pointers. This is W1S (write one shot) bit which can be written as ‘1’ but always read as ‘0’. Use this option only after disabling the ADC that is MCR[EN]=’0’. 0 No effect 1 Generate a single cycle reset event to flush the FIFO
20:23 FTHLD	FIFO Threshold It sets the threshold to generate DMA/Interrupt event. For example: FTHLD = "0000" indicates 1 data word, FTHLD = "0011" indicates 4 data words and so on. When the number of data words in the data FIFO is greater than or equal to the number of data words indicated by the FTHLD field, the FIFO full event is flagged. An interrupt or a DMA request will be generated as determined by DFFDIRE and DFFDIRS fields of RSER. For proper operation, number of data words indicated by the FTHLD field must be set less than or equal to the FIFO size as indicated by FSIZE.

Table 609. SDADC_FCR field descriptions (continued)

Field	Description
28 FOWEN	FIFO Over Write Enable When set the built-in FIFO structure allows the overwriting of new converted data over older converted data after the FIFO has number of entries equal to the FSIZE. In this case overrun will not be seen. 0 Data FIFO OW option disabled. 1 Data FIFO OW option enabled.
29:30 FSIZE	FIFO Size The maximum number of converted datawords that can be stored in the data FIFO before an overrun occurs. This field is read-only. The reset value of this field depends on the implementation parameter to define the FIFO size. 00 FIFO depth = 1 data word 01 FIFO depth = 4 data words 10 FIFO depth = 8 data words 11 FIFO depth = 16 data words
31 FE	FIFO Enable When this field is set the built-in FIFO structure is enabled. The size of the FIFO structure is indicated by the FSIZE field. If this field is not set then the FIFO depth is set to one data word regardless of the value in the FSIZE field. 0 Data FIFO is not enabled for multiple data words; FIFO depth is one data word. 1 Data FIFO is enabled; FIFO depth is indicated by FSIZE.

41.6.9 Software Trigger Key Register (SDADC_STKR)

The Software Trigger Key Register (STKR) allows generating the trigger event output by a software write operation.

Offset 0x001C — Access: User Read/Write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	ST_KEY															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 497. Software Trigger Key Register (SDADC_STKR)

Table 610. SDADC_STKR field descriptions

Field	Description
16:31 ST_KEY	Software Trigger Key This bit field, when written with 0xFFFF, is used to generate a trigger event output which can be used to trigger conversions of multiple SDADC blocks synchronously depending on the SoC implementation. Any write access to this register which is different from the predefined key is ignored. Read access will always return 0x0000.

41.6.10 Converted Data Register (SDADC_CDR)

The Converted Data Register (CDR) stores the converted data in the data FIFO.

Note: When SDADC_MCR[PGAN]=111b (programmable gain is 16), the least significant bit of the conversion data in SDADC_CDR[CDATA] is always 0.

Offset 0x0020 Access: User Read

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	CDATA															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 498. Converted Data Register (SDADC_CDR)

If the output data format is right-aligned (ODA is reset), then the register bit map will be as follows.

Table 611. SDADC_CDR field descriptions

Field	Description
16:31 CDATA	Converted Data Register The converted data words can be read from FIFO by reading this register. The data width is 16 bits which has actual 16-bit data (bits 16 up to 31) coming from ADC. If the data width is less than 16 (for example: 14) and if ODF is set, then: – If bit 18 is ‘1’, then the result is negative and bits 16 and 17 will also return ‘1’. – If bit 18 is ‘0’, then the result is positive and bits 16 and 17 will return ‘0’. If ODF is reset, then bits 16 and 17 will always return ‘0’.

Note: If the output data format is left-aligned (ODA is set), then the register bit map will be as follows.

Table 612. SDADC_CDR field descriptions

Field	Description
16:31 CDATA	Converted Data Register The converted data words can be read from FIFO by reading this register. If the actual data width is less than 16 bits (for example: 14), then bits 16 up to 29 will store the data coming from SDADC and bits 30 and 31 will always return ‘0’.

41.6.11 WDG Threshold Register (SDADC_WTHHLR)

This is the Converted Data Range WDG Threshold register.

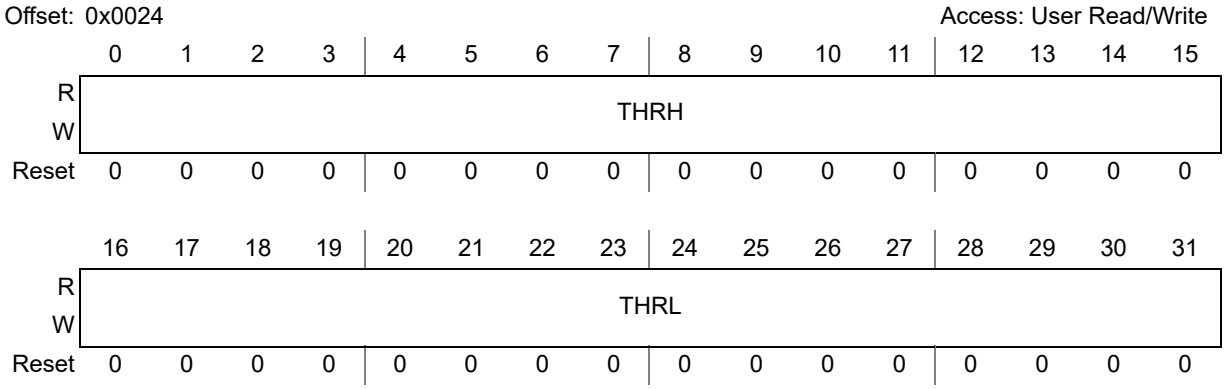


Figure 499. WDG Threshold Register (SDADC_WTHHLR)

Note: A single WDG monitor will check for lower and upper threshold crossover events.

Table 613. SDADC_WTHHLR field descriptions

Field	Description
0:15 THRH	WDG Upper Threshold Value This indicates the upper threshold for the converted data on any channel, above which threshold a crossover event wold be flagged through SFR[WTHH].
16:31 THRL	WDG Lower Threshold Value This indicates the lower threshold for the converted data on any channel, below which threshold cross over event wold be flagged through SFR[WTHL].

41.7 Functional description

The SDADC has four distinct available modes determined by fields in MCR.

- Differential input mode
- Single-ended input mode
- External modulator mode
- Low consumption mode

41.7.1 Differential input mode

After system reset exit, this is the default mode of operation if SDADC is enabled by asserting MCR[EN]. This mode is entered by negating MCR[MODE]. In differential input mode, a pair of analog inputs among AIN[0:7] is connected to the input terminals of the SDADC modulator block. The module supports four differential pairs: AIN[0,1], AIN[2,3], AIN[4,5], and AIN[6,7].

The modulator block has two input terminals, positive (INP) and negative (INM). The selection of input pair depends on the ANCHSEL field of CSR. The connectivity of each analog input in a given differential pair, whether it is connected to positive or negative terminal, is determined according to *Table 604*.

The differential input mode can also be used to perform gain and offset calibration, where the input terminals are biased with fixed voltages coming from the bias block.

41.7.2 Single-ended input mode

Single-ended input mode is entered by asserting MCR[MODE]. In this mode, one of the analog inputs among AIN[0:7] is selected by the ANCHSEL field of CSR, and is connected to the positive terminal. The negative input terminal is biased with a fixed internal voltage. There are two internal bias voltages available: reference ground (VREFN) and half-scale bias (VREFP/2). The selection of this common mode voltage is controlled by MCR[VCOMSEL].

41.7.3 External modulator mode

This mode is selected by asserting the MCR[EMSEL] field. When external modulator is selected, the internal modulator can be put in low consumption mode by deasserting the MCR[EN] field. In this mode, the data stream and clock outputs from the external modulator available on BSEXT/CLKEXT input pins are multiplexed with internal modulator data output and input sampling clock (f_s). External modulator data must be synchronized with respect to the falling edge of CLKEXT. In this mode, hardware trigger input is ignored even if it is enabled by asserting the MCR[TRIGEN] field. All other configuration settings (such as gain, decimation rate, filter) programmed in MCR have no effect on the conversion.

41.7.4 Low consumption mode

This is the default mode after exiting from reset. This mode is exited by asserting the MCR[EN] field to enable the SDADC block. When data conversion is not required, the SDADC internal modulator can be put in low consumption mode to reduce power consumption.

41.7.5 Analog input multiplexing and bias support

The complete selection and connectivity of analog inputs AIN[0:7] is based on the ANCHSEL field of CSR and the MODE and VCOMSEL bits of MCR, according to *Table 604*. Each analog input can be biased to half scale (VDD_HV_ADV/2) via on-chip 100 kΩ resistance. This allows simple AC coupling to the external analog inputs through a capacitor. This biasing is applicable irrespective of whether analog channel is selected through multiplexer or not.

41.7.6 Programmable gain and decimation rate

All analog inputs can be configured to have a selectable input gain as defined in the PGAN field description in MCR. This means the input signal is sampled and the result is amplified by the factor determined by PGAN before providing the same to modulator.

The SDADC module is provided with a fixed input sampling clock f_s. The input signal is grossly oversampled by the modulator in order to reduce quantization noise. To support different pass-bands with a fixed sampling clock, a programmable decimation rate is implemented. The process of decimation, to eliminate redundant data at the output while retaining the necessary information, is controlled by the field PDR in MCR.

41.7.7 High-pass filter support

For pure AC applications, it is useful to remove any DC component in the input signal. An optional high-pass filter is implemented which can be enabled by asserting the MCR[HPFEN] field. The –3 dB frequency of the filter is fixed (not programmable) and is available in the device data sheet. This high-pass filter is implemented in the decimation filter logic of the SDADC and is applicable only to internal modulator mode.

41.7.8 Data conversion

Software Control Mechanism: The SDADC block will be in continuous conversion mode as soon as it is enabled. The conversion data is not accurate enough for some time, depending on the latency and output settling time specifications. This is true even when there is any change in the configuration settings provided by MCR, change in the analog channel selection due to mux switching time. To account for these delays, an internal timer is implemented which counts down from a start value determined by the OSD field of OSDR. This timer is reloaded whenever the RESET_KEY of RKR is written with 0x5AF0, which is required to start SDADC conversion from a fresh state with a changed configuration. The timer is stalled on reaching count ‘0’ and asserts the SFR[CDVF] field. Once this flag is asserted, converted data is loaded into data FIFO on every rising edge of the output clock (f_d).

Any write access to the MCR and CSR registers will deassert the CDVF flag bit to indicate that data coming from SDADC is not valid or corrupted. If the CDVF flag is reset, the flags DFFF and DFORF are blocked from becoming asserted, that is transitioning from inactive state to active state. If DFFF bit is already set, this means the data FIFO is already full before CVDF deassertion.

The DFF flag bit (which not conditioned by the interrupt or DMA selection) is provided outside which can be routed to timer module at SoC level.

Wraparound Control Mechanism: Here a wrap-around logic shall include all analog channels of the input multiplexer from Ain(“000“) to Ain(CSR[ANCHSEL_WRAP]) in both Single ended and Differential ended mode in wraparound sequence triggered by Hardware or software source:

- MCR[WRMODE] bit would enable or disable the wraparound mechanism. Clearing the WRMODE bit automatically switches FSM back to Software control mode.
- Every valid trigger advances the wraparound counter to next channel to be converted.
- Wraparound Counter: An internal wraparound counter which points to analog channel to be converted takes the initial entry value for the first loop of execution from CSR[ANCHSEL] while entering wraparound mode. From next loop onwards the counter wraparounds to default initial value “000“. Maximum value of the counter

(wraparound value) should be programmed in CSR[ANCHSEL_WRAP] before entering wraparound mode, definition of ANCHSEL and ANCHSEL_WRAP is provided in *Table 604*.

- Upon entering or during wraparound mode user should take care to program ANCHSEL value < ANCHSEL_WRAP value, otherwise hardware would reject the new ANCHSEL value.
- Possible trigger sources:
  – SOURCE1 (Software Bit): writing to STKR.ST_KEY at any time during the wraparound mode (irrespective of MCR[TRIGEN or TRIGSEL] setting) generates a trigger pulse, each such trigger initiates a fresh conversion and advances the wraparound counter by one.

Note: Care should be taken while programming the TRIGEN/TRIGSEL of each SDADC instance that no unintended simultaneous SW triggering of multiple instances happen.

  – SOURCE2 (Hardware Trigger): Here external trigger output (for example mux_advance of GTM) enabled by MCR[TRIGEN or TRIGEDSEL or TRIGSEL] before entering wraparound mode for selective or all SDADC instances provides trigger, each such valid trigger initiates a fresh conversion and advances the wraparound counter by one. One of the hw_trig_in ports can be used to connect the external trigger output.
- Next channel number to be converted can be updated by software even in the middle of wraparound sequence execution, this can be done by rewriting CSR[ANCHSEL]. After CSR[ANCHSEL] is rewritten, the next trigger takes updated value (wraparound counter would increment from the updated value).
- Current wraparound counter value can be read from SFR[ANCHSEL_CNT] bit field.

41.7.9 Hardware triggering

Software Control Mechanism: SDADC conversion can be triggered by various hardware events coming from either external pins or internal timers. The field TRIGSEL of MCR selects which hardware event is to be used for triggering conversions. This trigger event, which is synchronous to the peripheral clock, is enabled by asserting MCR[TRIGEN]. As soon as a trigger event is detected, reset input of SDADC block is asserted synchronously for four cycles. This will ensure that SDADC starts from a fresh state when reacting to the input trigger event. The SDADC internal modulator must be enabled before enabling the trigger event — otherwise the trigger event will be ignored. A hardware trigger event is also ignored when external modulator is selected (MCR[EMSEL] = 1).

If the same trigger event is provided to multiple SDADC digital interface modules and the respective enable controls (TRIGEN bit MCR) are asserted, synchronous operation of multiple SDADC blocks can be achieved with output results being updated at the same time. Once the reset input is deasserted, the first valid conversion output is stored in data FIFO after a configurable delay determined by OSDR[OSD].

It is also possible to generate a software trigger event output by a write access of STKR with a predefined key. This event output can be used as a trigger to achieve synchronous software start of multiple SDADC blocks.

In case of a hardware trigger, the skew/jitter will be less than 100 ns. It will be more than one ADC clock cycle due to synchronization. In case of a software trigger, all SDADCs start synchronous sampling.

Wraparound Control Mechanism: In wraparound control mode, hardware and software triggers are equally valid. The hardware trigger source should be selected before enabling the wraparound mode.

41.7.10 Interrupt/DMA request support

The SDADC has one condition that generates interrupts only and another that generates interrupts or DMA request.

The DFFF field is asserted when the data FIFO full condition is detected, indicating that the number of converted data words stored in data FIFO is equal to or greater than the number of data words indicated by FCR[FTHLD] value. The interrupt or DMA request is generated only when RSER[DFFDIRE] is asserted. RSER[DFFDIRS] selects whether a DMA request or an interrupt request is generated.

The data FIFO overrun flag (DFORF) indicates the FIFO overflow condition. Further converted data words cannot be received and data is lost when FIFO overwrite functionality is disabled. The interrupt request is generated only when RSER[DFORIE] is asserted.

Apart from above, a WDG threshold crossover event can also generate an interrupt or DMA request.

If RSER[GDIGE] is set, all module interrupt/DMA requests are qualified only when an external global gating signal is asserted.

41.7.11 Gain calibration support

To perform gain calibration, the following sequence needs to be applied.

1. Select differential mode of operation by writing MCR[MODE] to ‘0’.
2. Enable gain error calibration mode by writing MCR[GECEN] to ‘1’.
3. Configure the mux selection ANCHSEL field of CSR to ‘110’. This configuration will apply VREFP on positive terminal and VREFN on negative terminal.
4. Disable the bias on all input analog channels by writing ENBIAS field of CSR to 0x00.
5. Disable the high-pass filter by deasserting MCR[HPFEN].
6. Generate a reset event by writing 0x5AF0 to RESET_KEY of RKR.
7. Read the digital output (CDR[CDATA]) stored in FIFO after the output settling time. The measurement should be repeated to reduce contribution of noise during calibration process. D_p is the average value of attenuated positive full scale given by D_p = AVERAGE(CDR[CDATA]).
8. Change the mux selection ANCHSEL field of CSR to ‘111’. This configuration will apply VREFN on positive terminal and VREFP on negative terminal.
9. Generate a reset event by writing 0x5AF0 to RESET_KEY of RKR.
10. Read the digital output (CDR[CDATA]) stored in FIFO after the output settling time. The measurement should be repeated to reduce contribution of noise during calibration process. Dn is the average value of attenuated negative full scale given by Dn = AVERAGE(CDR[CDATA]).
11. The SDADC calibrated gain can be calculated as:
    $$\text{Gain} = (D_p - D_n) / 2^{16}$$
12. The measured gain value can be used to nullify the gain errors in the digital output. For calibrated conversion, the data CDR[CDATA] provided by the SDADC should be

normalized using the calculated gain: $CDATA_{norm}$ is given by $CDATA_{norm}$ = CDR[CDATA]/Gain.

Note: The gain error calibration should only be done with gain equal to '1' (in other words, PGAN = 000). Nevertheless, the calculated gain can be applied to normalized data also for conversion with GAIN > 1.

MCR[GECEN] = 1 ensures the accurate gain error mode is enabled. This bit should be set both during calibration process and after calibration, for application data conversion. Application data conversion must then be normalized using the measured gain.

The higher the number of full-scale conversion (D_p, D_n) done, the higher the rejection of noise during the calibration. The number of conversion is dependent on application noise. It is recommended to run at least 16 conversions when calculating an average value.

Gain calibration should be done using the same OSR configuration as the target application, since full-scale values may vary slightly with these settings .

Refer to Digital output codes in full scale table in the SPC58xEx/SPC58xGx microcontroller Datasheet for Full-Scale values (with MCR[GECEN] = 1) with different OSR settings.

41.7.12 Offset calibration support

To perform offset calibration, the following sequence must be applied.

1. Select differential mode of operation by writing MCR[MODE] to '0'.
2. Configure the mux selection ANCHSEL field of CSR to '100' or '101' as required.
   a) CSR = '100' in case of data conversion after calibration in "single ended mode with negative input = $V_{SS_HV_ADR_D}$"
   b) CSR = '101' in case of data conversion after calibration in "differential mode" and "single ended mode with negative input = ($V_{DD_HV_ADR_D}$–$V_{SS_HV_ADR_D}$)/2"
3. Disable the bias on all input analog channels by writing ENBIAS field of CSR to 0x00.
4. Disable the high pass filter by deasserting MCR[HPFEN].
5. Generate a reset event by writing 0x5AF0 to RESET_KEY of RKR.
6. Read the digital output stored in FIFO after the output settling time.
7. The measured offset can be used to nullify the offset error in the digital output. Expected output is 0b00_0000_0000_0000. The SDADC offset can be calculated as:

   Offset = Expected Output – Actual Output

Note: The offset must be calculated for each PGAN field setting since it is expected to vary with gain configuration of SDADC.

41.8 Initialization information

To initialize the SDADC registers for data conversion, the following sequence is required.

1. Configure MCR to select the required mode, polarity, common mode voltage, input gain, and decimation rate.
2. Enable high-pass filter if required.
3. Select the required analog channel for data conversion. It is possible to select the bias for each channel for AC coupling applications.
4. Configure OSD delay according to SDADC startup time or latency from reset exit.
5. Enable the SDADC by asserting MCR[EN].
6. Generate a reset event by writing 0x5AF0 to RESET_KEY of RKR. Otherwise, if data conversion need to be triggered by a hardware event, assert MCR[TRIGEN].
7. DFFF interrupt or DMA request will be generated after the data FIFO has reached threshold to indicate that data can be transferred.

The following sequence needs to be followed when external modulator mode of SDADC is selected for data conversions.

1. Disable the SDADC internal modulator by deasserting MCR[EN].
2. Configure MCR to select the external modulator mode.
3. Select the required external pins BSEXT/CLKEXT in the system integration logic (SIUL) of the device.
4. Configure OSD delay according to external modulator startup time or latency from reset exit.
5. Generate a reset event by writing 0x5AF0 to RESET_KEY of RKR.
6. DFFF interrupt or DMA request will be generated after the data FIFO has reached threshold to indicate that data can be transferred.

41.8.1 Data conversion step

To acquire a data from SDADC, the following sequence is required:

1. Enable the SDADC by asserting MCR[EN].
2. Configure MCR to select the required mode, polarity, common mode voltage, input gain, and decimation rate.
3. Enable high-pass filter is required.
4. Select the required analog channel for data conversion. It is possible to select the bias for each channel for AC coupling applications.
5. Configure OSD delay according to SDADC required startup time or latency from reset exit.
6. Generate a reset event writing 0x5AF0 in RESET_KEY field of RKR register.
7. Wait till FIFO empty flag DFEF of SFR register is clear.
8. Read data by CDATA of CDR register.

Repeat steps from 6 to 8 for new acquisitions.

Note: The time elapsed between reset event and Read data by CDATA field of CDR register are:

- *Reset event <- -> Data Valid flag (CDVS) = Output settling Time delay (OSDR has to be set at least to 16)*
- *Data Valid flag <- -> Data FIFO is not empty = 0.5*CLK_Out + 3*CLK_In*
- *Data FIFO is not empty <- -> read Data (safety point) = 2*CLK_In*

*where: CLK_Out = CLK_In/(2*OSR)*

42 Successive Approximation Register Analog-to-Digital Converter (SARADC)

42.1 Introduction

The Successive Approximation Register Analog-to-Digital Converter (SARADC) digital interface block controls the on-chip SARADC analog block and holds control and status registers accessible for the application. It provides accurate and fast conversion data for a wide range of applications. Each SARADC analog block has its corresponding digital interface implemented at the chip level.

42.2 Overview

The SARADC digital interface contains advanced features for normal or injected conversion modes of operation. The conversion can be triggered by software or hardware. For advanced cross triggering functions, it provides interface with an on-chip Crosstriggering Unit (CTU) which can support two different CTU Modes: CTU “trigger mode” and CTU “control mode” (refer to details in *Section 42.4.5: Cross Triggering Unit (CTU) interface*).

The digital interface can be configured to control up to 256 multiplexed analog input channels. There are three types of input channels:

- Internal channels, mapped to index ranging from 0 to 95.
- Test channels, mapped to index ranging from 96 to 127.
- External channels, mapped to index ranging from 128 to 255.

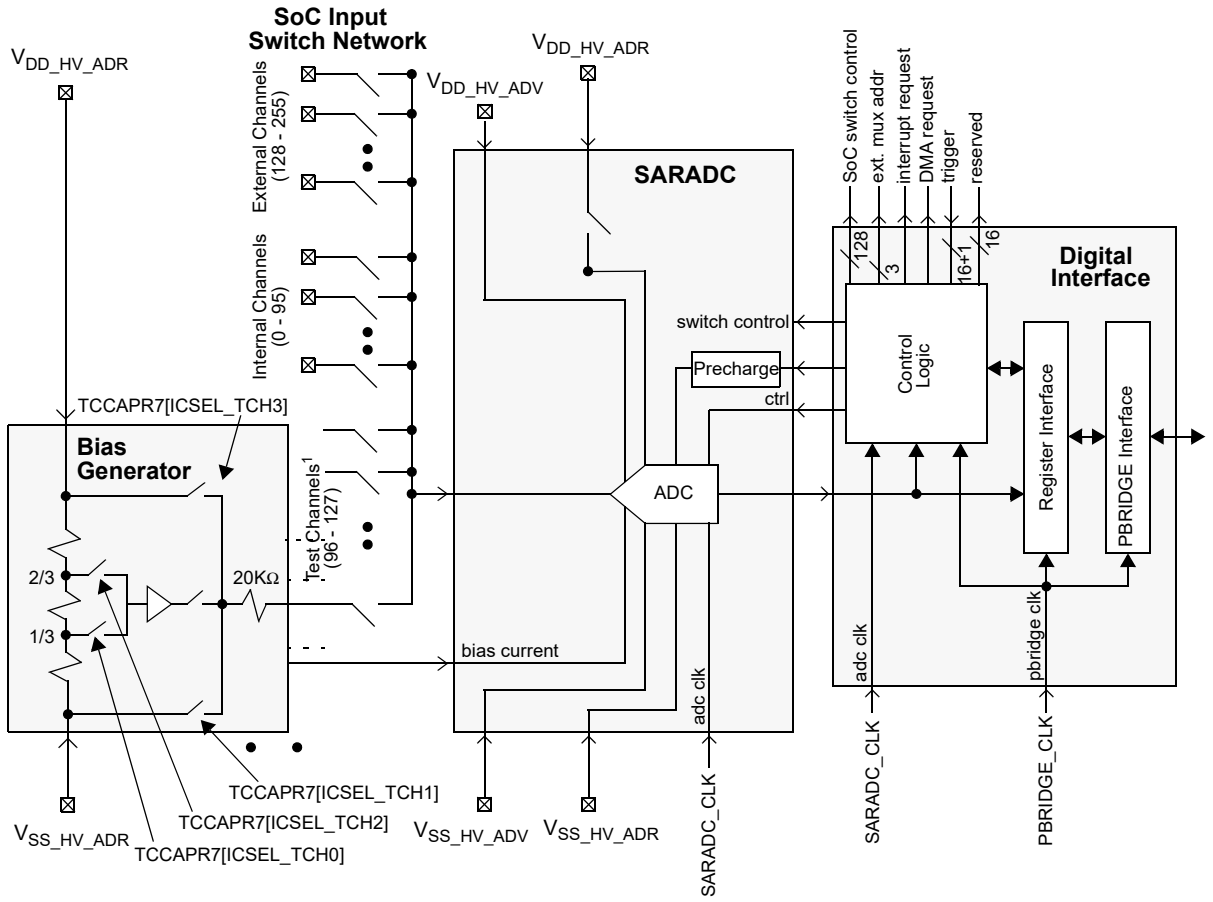
Note: The number of input channels and their mapping depend on the device configuration. Refer to the device-specific Analog-to-Digital Converters (ADC) Configuration section (analog input pin multiplexing and SAR ADC channel assignment) for a channel mapping table.

The mask registers present within the digital interface can be properly programed to configure which channel has to be converted. External channel selection is provided through external decode signals and are available as alternate functions on GPIO. Each set of eight external channels can be mapped to any internal analog channel via static software programming in specific registers.

Four conversion timing registers allow configuration of different precharging and sampling durations, one of which can then be assigned to each channel. Analog Watchdogs allow continuous hardware monitoring of analog input channels. The digital interface also provides for interrupt/DMA support for various conditions related to end of channel conversions.

Figure 500 shows the SARADC block diagram.

Figure 500. SARADC block diagram



Note: Test channels are mapped to internal analog channels via the TCCAPRn registers of SAR ADC supervisor.

42.3 Feature description

Note: Not all features are implemented on all SARADC instances: refer to the Chapter 40: Analog-to-Digital Converters (ADC) Configuration for further information.

42.3.1 Main features

- Selectable 10-bit or 12-bit data resolution output for 12-bit SARADC
- Selectable 8-bit or 10-bit data resolution output for 10-bit SARADC
- Up to 96 internal channels, 32 test channels, 128 external channels supported; variable number of analog channels of each type controlled by parameters
- Four different conversion timing registers selectable for any channel
- Mapping of external channel to any internal channel through static programming by software
- Shorting of test channel with internal channel through static programming by software
- External decode signals (3 signals) for selection of external analog mux inputs
- Normal conversion with One Shot/Scan modes
- Injected conversion with One Shot mode
- Normal conversion with dedicated trigger input
- Injected conversion with dedicated trigger input
- Two different abort features that allow to abort either a single channel conversion or chain conversion
- Power Down Mode
- Dedicated data register for each channel, containing the following information:
  - Conversion result in one half-word:
    10-bit or 12-bit for 12-bit SARADC.
    8-bit or 10-bit for 10-bit SARADC.
  - Status byte which provides some conversion information such as mode of operation (Normal, Injected, or CTU), data valid, data overwritten status
  - Control byte for conversion timing parameter selection
- Configurable number of analog watchdogs.
- Two different CTU modes (CTU Control mode and CTU “trigger mode”) available when CTU feature is present in SARADC instance.
- Fast Comparator Mode Conversion available when Fast Comparator feature is present in SARADC instance
- Smart Standby Wakeup Support available when Smart Standby Wakeup feature is present on SARADC instance
- Interrupt/DMA support for the following conditions
  - End of conversion of single channel for both normal, injected conversions
  - End of conversion chain for both normal, injected conversions
  - End of CTU conversion
  - Watchdog thresholds crossover

42.4 Functional description

Two main conversion types are available within the SARADC digital interface:

- Normal conversion
- Injected conversion

Note: "precharge" option on this device means "de-charge" to reference ground VSS_HV_ADR_S.

42.4.1 Normal channel conversion

This is the normal conversion that the user programs by configuring the normal conversion mask registers (ICNCMR0–2, TCNCMR, ECNCMR0–3). Each channel can be individually enabled by setting '1' in the corresponding field of these registers. Mask registers must be programed before starting the conversion and cannot be changed until the conversion of all the selected channels ends (MSR[NSTART] bit is reset).

42.4.1.1 Start of normal conversion

The normal conversion can be started as follows:

- By software
  - The normal conversion chain starts when the MCR[NSTART] bit is set. It is recommended that the normal trigger enable MCR[NTRGEN] bit is reset during conversions started by software. Once normal conversion has started, any further write operation of MCR[NSTART] bit has no effect during ongoing conversion, but the conversion is enqueued after completion of current conversion in oneshot mode.

- By normal trigger
  - A normal trigger valid edge or level is detected to start the conversion.The normal trigger is enabled by setting the NTRGEN bit in MCR and two options are available. A programed event (rising, falling or both edges depending on NEDGESEL bit field of MCR) on the normal trigger input starts the normal conversion. Once normal conversion has started, any further valid trigger edge will be ignored during ongoing conversion. If the trigger mode is enabled (MCR[NTRGEN] bit is set) and a conversion is in progress, no conversion request should be given by programming MCR[NSTART] bit.
  - An appropriate gap should be maintained between any two consecutive triggers such that next trigger is received only after the completion of the current conversion. This gap value can be calculated as a sum of total conversion time as programed using CTR0-3 registers for each channel in the chain.

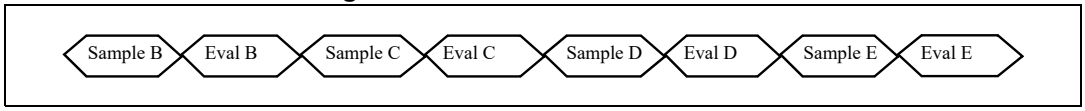
42.4.1.2 Normal conversion operating modes

Two operating modes are available for the normal conversion:

- One Shot mode
- Scan mode

To enter one of these modes, it is necessary to program the MCR[MODE] bit. The first phase of the conversion process involves sampling the analog channel and the next phase involves the conversion phase when the sampled analog value is converted to digital as shown in *Figure 501*.

Figure 501. Normal conversion flow



In **One Shot mode** (MODE = 0), a sequential conversion specified in the ICNCMR0–2, TCNCMR, ECNCMR0–3 mask registers is performed only once. At the end of each conversion, the digital result of the conversion is stored into the corresponding data register.

For Example, channels A-B-C-D-E-F-G-H are present in the device where channels B-D-E are to be converted in the One Shot Mode. MODE = 0 is set for One Shot mode. Conversion starts from the channel B followed by conversion of channels D-E. At the end of conversion of channel E, the scanning of channels stops.

The MSR[NSTART] status bit is automatically set when the normal conversion starts. At the same time MCR[NSTART] bit is reset if the conversion is started by software.

Once the normal conversion operation starts in oneshot mode and another software conversion request is made writing the MCR[NSTART] bit and before MSR[NSTART] is cleared, one additional conversion is initiated with the same channel mask register settings after the current conversion is finished. This is called **enqueuing**. Enqueuing can help in removing the delay between two consecutive one shot chains and uncertainty caused by metastability between MCR[NSTART] bit written at system clock and generation of start pulse on ADC clock before every chain execution.

In **Scan mode** (MODE = 1), a sequential conversion of N channels specified in the ICNCMR0–2, TCNCMR, ECNCMR0–3 registers is continuously performed. As in the previous case, at the end of each conversion the digital result of the conversion is stored into the corresponding data register.

In this mode, the software can start conversion by writing ‘1’ to MCR[NSTART] bit or a valid normal trigger edge is detected which also sets MCR[NSTART] bit. The MSR[NSTART] status bit is automatically set when the normal conversion starts. Unlike One Shot Mode, the MCR[NSTART] bit is not reset. It can be reset by software when the user needs to stop scan mode. In this case, the current scan conversion is completed and MSR[NSTART] bit is also reset after the last conversion of the chain.

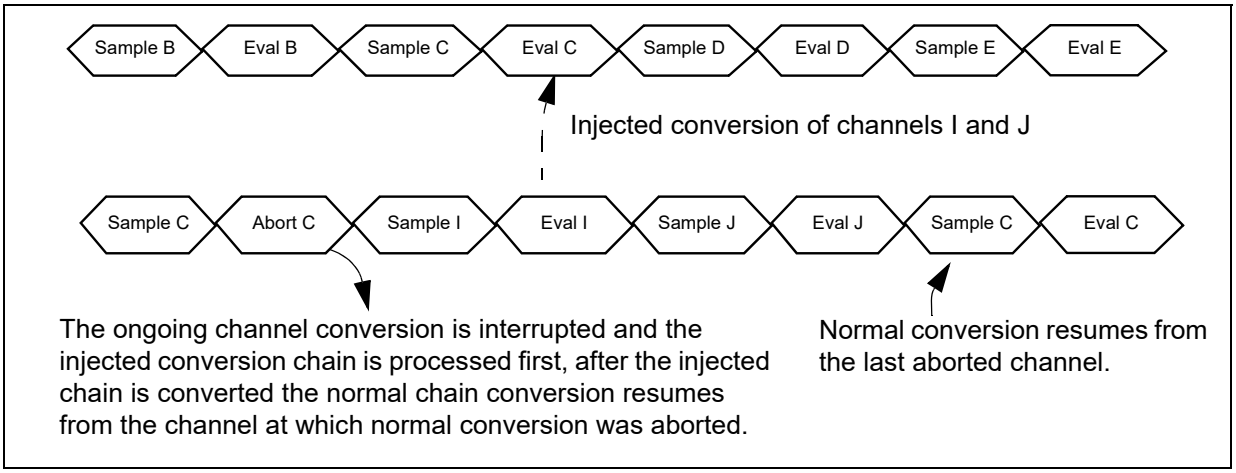
For Example, channels A-B-C-D-E-F-G-H are present in the device where channels B-D-E are to be converted in the *Scan Mode*. MODE = 1 is set for Scan Mode. Conversion starts from the channel B followed by conversion of the channels D-E. At the end of conversion of channel E the scanning of channel B starts followed by conversion of the channels D-E. This sequence repeats itself until the MCR[NSTART] bit is reset by software.

In both modes, at the end of each channel conversion an End Of Conversion interrupt is issued (if enabled by the corresponding mask bit), and at the end of the conversion sequence an End Of Chain interrupt is issued (if enabled by the corresponding mask bit). The End Of Chain interrupt is issued for each pass of scan mode.

42.4.2 Injected channel conversion

A conversion chain can be injected into the current normal conversion by configuring the injected mask registers (ICJCMR0–2, TCJCMR, ECJCMR0–3). As in normal conversion, each internal, test or external channel can be individually selected. This injected conversion can only be in one shot mode and interrupts the normal conversion. When an injected conversion is inserted, ongoing channel conversion is aborted and the injected channel request is processed. After the last channel in the injected chain is converted, normal conversion resumes from the channel at which the normal conversion was stopped, as shown in *Figure 502*.

Figure 502. Injected sample/conversion sequence



The injected conversion can be started as follows:

- By software
  - The injected conversion chain starts when the MCR[JSTART] bit is set. The current conversion is suspended and the injected chain is converted. At the end of the chain, the MSR[JSTART] bit is reset and the normal chain conversion is resumed. It is recommended that the injection trigger enable MCR[JTRGEN] bit is reset during conversions started by software. Once injected conversion has started, any further write operation of MCR[JSTART] bit will not have any effect during ongoing conversion and the new request will get enqueued and executed after current injected chain finishes.

- By injection trigger
  - The injection trigger is enabled by setting the JTRGEN bit in MCR. A programed event (rising, falling or both edges depending on JEDGESEL bit field of MCR) on the injection external trigger starts the injected conversion and also sets the MSR[JSTART] bit. At the end of the chain, the MSR[JSTART] bit is reset and the normal chain conversion is resumed.
  - When JTRGSEQ bit is set, the injection trigger sequence feature is enabled. In this mode, each valid edge of injection trigger input will convert one analog channel. First valid edge will convert the first analog channel enabled through ICJCMR0–2, TCJCMR, ECJCMR0–3 registers. The next valid edge converts the next enabled analog input and so on. Once all the enabled channels are converted, the channel selection loops back to the first enabled channel for next trigger edge. If the JTRGSEQ is reset, the channel selection is reset again to first channel enabled through ICJCMR0–2, TCJCMR, ECJCMR0–3 registers. If JTRGSEQ is reset before completing the injected chain, the remaining channels are completed in one-shot after receiving next valid trigger edge.
  - An appropriate gap should be maintained between any two consecutive injected triggers such that next injected trigger is received only after the completion of ongoing injected conversion. This gap value can calculated as a sum of total conversion time as programed using CTR0-3 registers for each channel in the injected chain.

The MSR[JSTART] status bit is automatically set when the injected conversion starts. At the same time MSR[JSTART] is reset if the conversion is started by software.

At the end of each injected conversion, an End Of Injected Conversion (JEOC) interrupt is issued (if enabled by the corresponding mask bit) and at the end of the sequence an End Of Injected Chain (JECH) interrupt is issued (if enabled by the corresponding mask bit).

42.4.3 Abort conversion

Two different abort functions are provided:

- The user can abort the ongoing conversion by setting the ABORT bit in the MCR. The current conversion is aborted and the conversion of the next channel of the chain is immediately started (generating a new start pulse to the SARADC). In the case of an abort operation, the NSTART/JSTART bit remains set and the ABORT bit is reset after the conversion of the next channel starts. The EOC corresponding to the aborted channel is not generated. This behavior is true for normal or triggered/Injected conversion modes. If the last channel of a chain is aborted, the end of chain is reported generating an ECH interrupt. ABORT function should not be used when only single channel is selected in a conversion chain, instead ABORTCHAIN can be used.
- It is also possible to abort the current chain conversion by setting the ABORTCHAIN bit in the MCR. In that case the behavior of the ADC depends on the MODE bit. In fact, if scan mode is disabled, the NSTART bit is automatically reset together with the ABORTCHAIN bit. Otherwise, if the MODE bit is set to ‘1’, a new chain conversion is started. The EOC of the current aborted conversion is not generated but an ECH interrupt is generated to signal the end of the chain.

  When a chain conversion abort is requested (ABORTCHAIN bit is set) while an injected conversion is running over a suspended Normal conversion, there are 2 cases. If One Shot Mode is enabled (MODE=0 in MCR register) then both injected chain and Normal conversion chain are aborted (both the NSTART and JSTART bits are also reset). If the Scan Mode is enabled (MODE=1 in MCR register) then injected conversion is stopped and the suspended Normal conversion is restarting. (only JSTART bit is reset).

Note: ABORT/ABORTCHAIN functions are not supported during CTU initiated conversions.

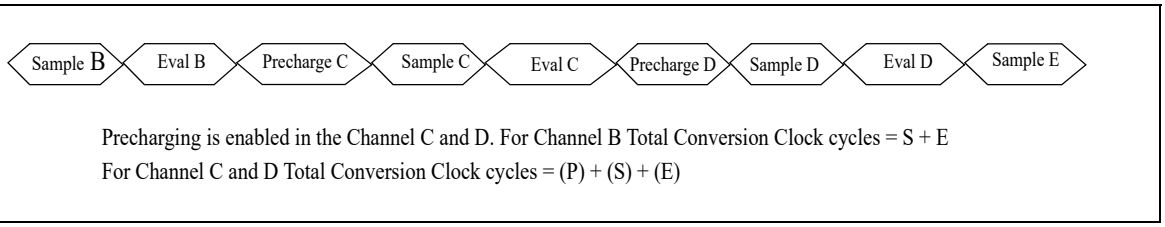
42.4.4 Analog conversion timings

42.4.4.1 Conversion timings

In order to support different loadings and switching times (in particular for the external channel types) four different Conversion Timing registers are present (CTR0–3). Each conversion timing register contains PRECHG and INPSAMP bit fields to program the required duration for precharging and sampling phases. The selection of these registers for each channel is done by the CTSEL bit field of corresponding channel data register.

Precharging phase allows to precharge or discharge the internal capacitor to a fixed internal voltage for duration determined by PRECHG bit field prior to actual analog channel sampling start. This is useful for resetting information regarding the last converted data. Precharging feature can be enabled or disabled for each channel by programming the PCE bit of corresponding channel data register. If precharging is enabled for a given channel, the normal sequence of operation is precharging, then sampling, then evaluation, as shown in *Figure 503*. Refer to device datasheet for the minimum precharge and sample durations required.

Figure 503. Analog conversion sequence



Bit fields PRECHG and INPSAMP are used to define the total conversion duration (t_{conv}) and in particular the partition among precharge duration (t_{prechg}), sampling phase duration (t_{sample}) and evaluation phase duration (t_{eval}).

The precharging phase duration is given by

$$t_{prechg} = PRECHG * t_{ck}$$

where PRECHG is the configured precharging phase duration and PCE bit of channel data register is ‘1’. If the PCE bit is ‘0’, the precharging phase is skipped and conversion starts with sampling phase directly.

The sampling phase duration is given by the following equation:

$$t_{sample} = INPSAMP * t_{ck}$$

where INPSAMP must be greater than or equal to 5 (hardware requirement). In case the value of INPSAMP is found to be less than 5, it is automatically set to 5 inside SARADC. The total evaluation phase duration is given by the following equation:

$$t_{eval} = EVAL * t_{ck}$$

In this phase, the internal sampling capacitor is disconnected from the analog channel and ADC estimates the digitized value of the sampled input using successive approximation algorithm. In the evaluation phase, all the bits are estimated sequentially to provide the conversion result.

The total conversion duration is (not including external multiplexing) is given by the following:

$$t_{conv} = t_{prechg} + t_{sample} + t_{eval}$$

The timings refer to the unit t_{ck} which is the inverse of f_{ck}, where f_{ck} = SARADC peripheral clock. Refer to the device data sheet for the total conversion time.

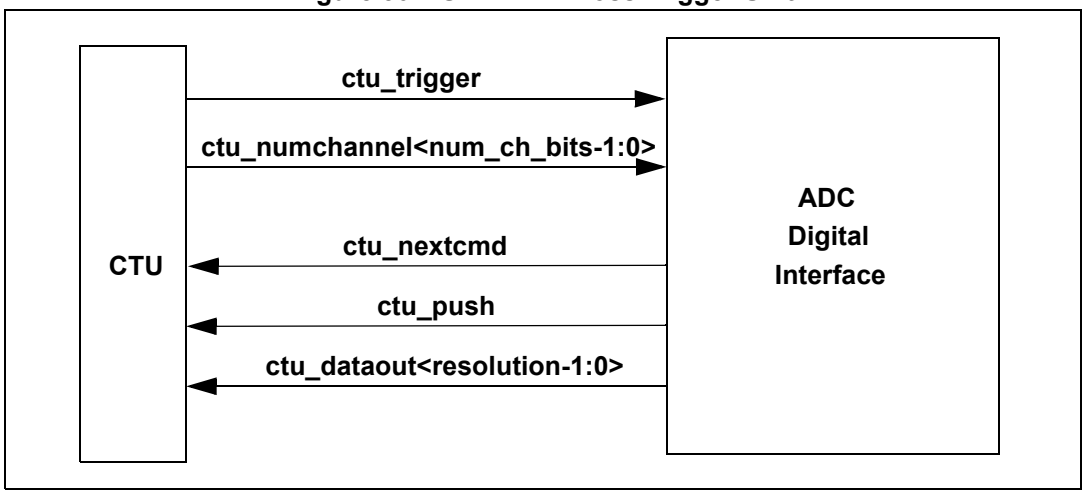
Note: The current implementation of ADCDIG does not support hardware restriction over values of PRECHG and INPSAMP fields. As minimum values for these fields are product specific and need to be managed through software.

42.4.5 Cross Triggering Unit (CTU) interface

The Cross Triggering Unit (CTU) is added to enhance the injected conversion capability of the SARADC. The CTU contains multiple event inputs that can be used to select the channels to be converted from the appropriate event configuration register. In *Figure 504* the CTU/ADC interface is shown.The CTU generates a trigger output pulse of one clock cycle and outputs onto a data bus which channel has to be converted. A single channel is converted for each request. After performing the conversion, the digital interface returns the conversion result to CTU. The conversion result is also saved in the corresponding channel data register and it is compared with watchdog thresholds if requested. Together with the converted data, it also sends a request for next command and a control which can be used to push the conversion result into CTU data storage registers/FIFO.

These two pulses named *ctu_nextcmd* and *ctu_push* spaced of one or two clock cycle (of analog ADC clock). The ctu_nextcmd is set to '1' during the last clock cycle or last but one of ADC evaluation phase (depending on ADC type used). The ctu_push is asserted when the conversion is finished meaning that the ctu_dataout is valid.

Figure 504. SARADC Cross Trigger Unit



Different CTU modes can be selected programming the CTUEN and CTU_MODE fields of the MCR with respect of CTSEN SARADC parameter as described in the following table.

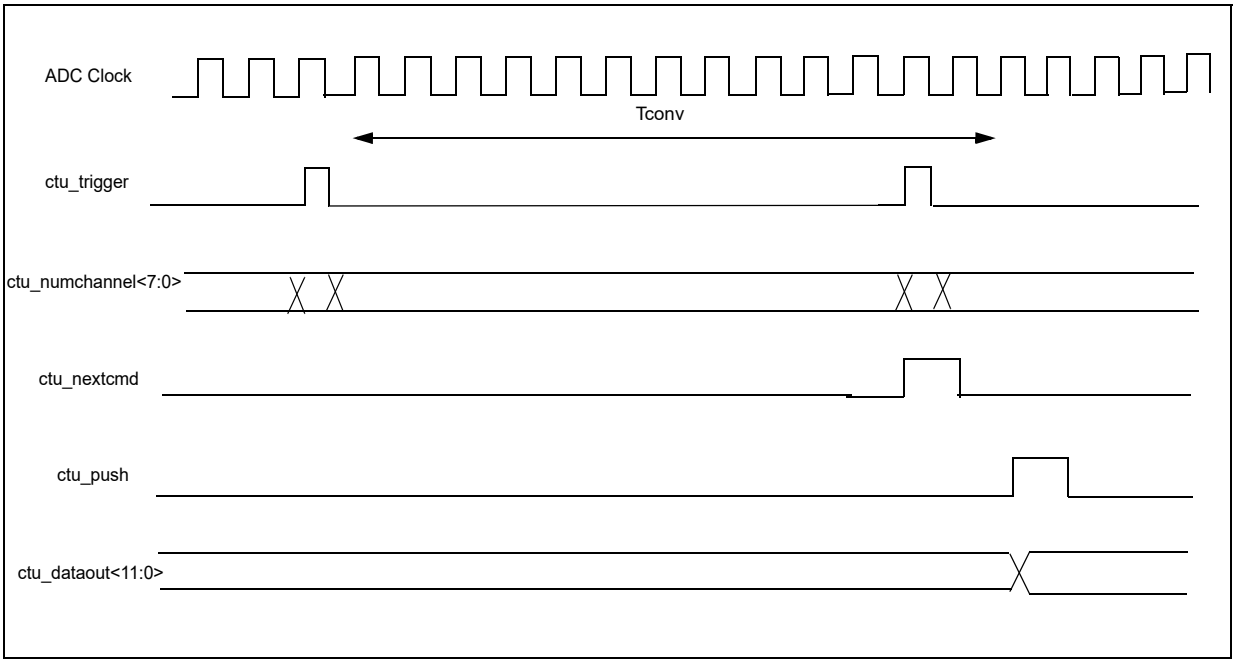
Table 614. CTU modes

CTSEN(1)	CTUEN	CTU_MODE	CTU mode
—	0	—	CTU is disabled
0	1	—	CTU trigger mode is enabled
1	1	—	CTU control mode is enabled
2	1	1	CTU trigger mode is selected
2	1	0	CTU control mode is selected

1. Refer to *Section 40.2.2.5: SAR ADC triggering* of *Chapter 40: Analog-to-Digital Converters (ADC) Configuration* for CTSEN value applicable on each SAR_ADC instance. CTSEN is hardware implemented, no bit register available.

The CTU, SARADC analog block and digital Interface are synchronous to each other, the CTU clock and ADC clock post division should be synchronous and phase aligned to ensure jitter free start pulses to ADC analog. Timings of CTU Interface signals are described in *Figure 505*.

Figure 505. CTU interface timings



42.4.5.1 CTU trigger mode

In CTU trigger mode, normal and injected conversions triggered by the CPU are still enabled. When a trigger pulse and the injected channel are received internally by the ADCDig, the conversion starts. The CTUSTART bit in the MSR is set automatically at this point and it is also automatically reset when the triggered injected conversion is completed.

In CTU trigger mode if an injected conversion (programed by the user by setting the JSTART bit) is ongoing and a pulse is received on the CTU trigger input line, then the injected channel conversion chain is aborted and only the triggered CTU injected conversion proceeds. Aborting the injected conversion, the JSTART bit in the MSR is reset to ‘0’. The JECH and JEOC are also reset to '0'. That abort is signaled through the status bit JABORTCHAIN in the MSR.

IN CTU trigger mode if normal conversion is ongoing and a pulse is received on the CTU trigger input line, then ongoing channel conversion is aborted and the triggered injected conversion is processed. When it is finished, normal conversion resumes from the channel at which the normal conversion was aborted.

If another CTU trigger pulse is received before the end of conversion, that request is discarded and this error condition is reported by CTUTRGERR bit in ISR.

When a normal conversion is requested during CTU conversion (CTUSTART bit = ‘1’), the normal conversion starts when CTU conversion is completed (CTUSTART reset). Otherwise, when an Injected conversion is requested during CTU conversion, that conversion is discarded and the JSTART bit in the MCR is immediately reset.

CTU trigger mode can be entered by setting CTUMODE=’1’ during power-down and CTUEN=’1’ any time after the power-down is exited. CTU trigger mode can be exited by clearing the CTUEN bit, But care must be taken not to exit the CTU mode when some CTU initiated conversion is ongoing. Also power down request (by setting MCR[PWDN] bit) should not be made while CTU transactions are ongoing.

42.4.5.2 CTU control mode

When CTU control mode is enabled, the CPU is able to write in the ADC registers but it cannot start any conversion. Conversion requests can be generated only by the CTU trigger pulse. If a normal or injected conversion is requested, it is automatically discarded.

When a CTU sends trigger pulse and channel number to be converted to SARADC digital interface, triggered injected conversion starts. The MSR[CTUSTART] bit is set automatically on the first received trigger pulse and it can be reset only when CTU is disabled (CTUEN bit to ‘0’).

CTU control mode can be entered by setting CTUMODE=’0’ during power-down and CTUEN=’1’ any time after the power-down is exited. CTU control mode can be exited by clearing the CTUEN bit, But care should be taken not to enter or exit the CTU mode when some conversion is ongoing. Also power down request (by setting MCR[PWDN] bit) should not be made while CTU transactions are ongoing.

42.4.6 Test channel connection with internal analog channel

The test channels are meant for monitoring various on-chip analog signals coming from temperature sensor, bandgap, voltage regulator and different voltage levels available on the chip. These analog signals can be interfaced to SARADC SUPERVISOR through test channels for testing purpose. Test channels are mapped from 96 to 127. Each test channel is controlled in the same way as normal internal channel.

FAST SAR ADC self-testing method: the test channel[127:124] is also mapped to various V_ref voltages (4 voltages) selection switches in the ADCBIAS block. The decoding of these channels to switch on and select a particular V_ref voltage is done in SoC level muxing logic. Refer to ADC configuration for detailed mapping/control description of Self Test feature.(SAR_ADC_10bit_STDBY has no test channels)

It is also possible to perform test channel conversions with and without shorting the normal analog channel pin with the internal test voltage.

For analog test channel assignment, refer to the device configuration or ADC configuration section.

After the conversion is performed on a test channel the converted data is stored at two CDR locations at the end of conversion, first location is the CDR belonging to the test channel on which conversion is performed and second location is the CDR belonging to internal channel on which test channel is mapped.

42.4.7 External channel mapping to internal analog channel

External channels are supposed to be used in conjunction with a given internal analog channel along with an 8:1 external mux outside the device. Each input of the external mux can be mapped to a unique channel number, thus allowing it to perform conversions like any other channel of the ADC. External channels are mapped from 128 to 255.

External channels are implemented in the multiples of 8. Each slice of eight channels can be associated to any physical internal analog channel mapped from 0 to 95. This external-to-internal channel mapping is determined by the ICSEL_ECx_x bit field of ECMICR0–3 registers. For example, when external multiplexer is planned to be used on internal channel 5, and the external inputs need to be mapped to channels 128 to 135, then this channel number 5 must be written to the ICSEL_ECH128_135 bit field of the ECMICR0 register. After reset, all external channels are mapped to internal analog channel 0. If a conversion

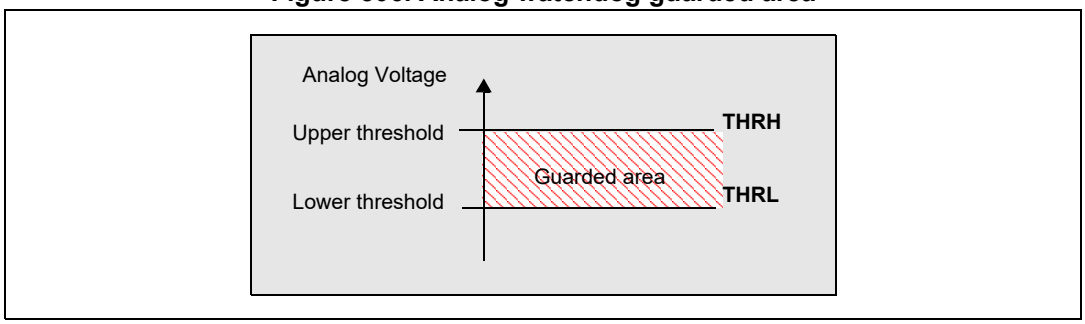
on a single external channel (out of group of 8) mapped to a particular internal channel has to be performed than only that particular channel should be enabled in the Conversion Mask Register (ECNCMR or ECJCMR).

After the conversion is performed on a external channel, the converted data is stored at two CDR locations at the end of conversion. The first location is the CDR belonging to the external channel on which conversion is performed and the second location is the CDR belonging to the internal channel on which the external channel is mapped.

42.4.8 Programmable analog watchdog

Analog watchdogs are used for determining whether the result of the conversion of a channel lies within a given guard area (as shown in the *Figure 506*) bounded by upper and a lower threshold values named THRH and THRL respectively.

Figure 506. Analog watchdog guarded area



Following conversion of the selected channel, a comparison is performed between the converted value and the threshold values. If the converted value lies outside that guard area, then corresponding threshold violation interrupts are generated. The comparison result is stored as WDGxH and WDGxL bits in the WTISR as explained in *Table 615*. Depending on the MSKWDGxL and MSKWDGxH mask bits in the WTIMR, an interrupt is generated on threshold violation.

Table 615. Values of WDGxH and WDGxL fields

WDGxH	WDGxL	Converted data
1	0	converted data > THRH
0	1	converted data < THRL
0	0	THRH ≤ converted data ≤ THRL

The lower and higher threshold values for the analog watchdog are programed using THRHLR0–15 registers. Analog watchdog functionality for each channel can be enabled independently from ICWENR0–2, TCWENR, ECWEN0–3 registers and can select the watchdog threshold registers (THRHLR0–15) to be used by programming ICWSELR0–11, TCWSELR0–3, ECWSELR0–15 registers. The threshold registers selected by WSEL_CHx field of watchdog threshold selection registers will provide the threshold values.

For example, if channel 30 is to be monitored with the threshold values in register THRHLR12, then WSEL_CH30 is programed to select THRHLR12 register to provide threshold values. Enabling is done by setting the bit corresponding to channel 30 in ICWENR0 register.

If the converted value for a particular channel lies outside the range specified by threshold values, then the corresponding bit is set in ICAWORR0–2, TCAWORR, ECAWORR0–3 registers. In this case, a set of threshold registers (THRHLR0–15) can be linked to several analog channels. The threshold values to be selected for a channel needs be programed only once in ICWSELR, TCWSELR, ECWSELR registers.

Note: If the higher threshold for the analog watchdog is programed lower than the lower threshold and the converted value is less than the lower threshold, then the WDGxL interrupt for the low threshold violation is set, else if the converted value is greater than the lower threshold (consequently also greater than the higher threshold) then the interrupt WDGxH for high threshold violation is set. The user should therefore take care to avoid this situation as it could lead to misinterpretation of the watchdog interrupts.

42.4.9 DMA functionality

A Direct Memory Access (DMA) request can be programed after the conversion of every channel by setting the respective masking bit in ICDSR0–2, TCDSR and ECDSR0–3 registers. The DMA masking registers must be programed before starting any conversion. The DMA transfers can be enabled using the DMAE[DMAEN] bit. When the DMAE[DCLR] bit is set, the DMA request is cleared on the reading of the register for which DMA transfer has been enabled.

42.4.10 Interrupts

The SARADC digital interface generates the following maskable interrupt signals:

- End Of Conversion interrupts
  - NEOC (end of normal conversion of each channel)
  - NECH (end of normal chain)
  - JEOC (end of injected conversion of each channel)
  - JECH (end of injected chain)
  - EOCTU (End of CTU Conversion)

The EOC_CH[x] bit of interrupt pending registers (ICIPR0–2, TCIPR, ECIPR0–3) is set when channel x completes the conversion (normal or injection). This pending status is qualified when the corresponding mask bit IM_CH[x] is also set in the interrupt mask registers (ICIMR0–2, TCIMR, ECIMR0–3).

The NEOC interrupt is generated only if the following conditions are met

- ISR[NEOC] bit is set after the normal conversion completion for any channel x.
- IMR[MSKNEOC] bit is set
- EOC_CH[x], IM_CH[x] bits are set at least for one channel x

The JEOC interrupt is generated only if the following conditions are met

- ISR[JEOC] bit is set after the injected conversion completion for any channel x.
- IMR[MSKJEOC] bit is set
- EOC_CH[x], IM_CH[x] bits are set at least for one channel x

The EOCTU interrupt is generated only if the following conditions are met

- ISR[EOCTU] bit is set after the CTU triggered conversion completion for any channel x.
- IMR[MSKEOCTU] bit is set
- EOC_CH[x], IM_CH[x] bits are set at least for one channel x

It is recommended to clear the NEOC/JEOC/EOCTU bits of ISR and all the individual channel pending bits which are not masked while servicing the interrupt.

The ISR[NECH] bit is set when a normal channel conversion chain is completed. This pending status is qualified when the corresponding mask bit IMR[MSKNECH] is also set.

The ISR[JECH] bit is set when an injected channel conversion chain is completed. This pending status is qualified when the corresponding mask bit IMR[MSKNECH] is also set.

- WDGxL and WDGxH (Watchdog threshold) interrupt requests

The interrupts generated due to the analog watchdog are handled by registers WTISR and WTIMR in order to check and enable the interrupt requests to external interrupt controller module. The watchdog interrupt source sets two pending bits WDGxH and WDGxL for each channel being monitored in the WTISR. The interrupt request is generated if at least one of the following conditions are met:

- WTISR[WDGxL] bit is set and the corresponding mask bit MSKWDGxL is also set in WTIMR for at least one channel.
- WTISR[WDGxH] bit is set and the corresponding mask bit MSKWDGxH is also set in WTIMR for at least one channel.

The ISR and WTISR contain the interrupt pending request status flags. In case the user wants to clear a particular interrupt event status, writing a ‘1’ to the corresponding status bit clears the pending interrupt flag. (At this write operation, all other bits of the ISR must be maintained at ‘0’.)

The ADC module provides dedicated interrupt outputs for all interrupt sources. Also, all interrupt lines are OR-ed together to provide a single output to the external interrupt controller if a combined interrupt is required.

Conversion should not be initiated unless the required conversion mask bits (ECNCMR or ECJCMR) are set.

42.4.11 External decode signals selection and delay

The ADC provides three external decode signals used as select lines for 8:1 external mux. For each set of 8 external channels (for example [CH128:CH135], [CH136:CH143], and so on), the same 3-bit decode signals output are used to select the inputs of external mux. These external mux select lines change in grey code sequence for each increment of channel selection. This ensures that only one line changes and avoids any current spikes or leakage if more than one line changes simultaneously.

In order to take into account the control switching time of the external analog mux, a Decode Signals Delay register is provided. Writing that register allows programming of the delay between the decoding signal selection and the actual start of conversion.

42.4.12 Power down mode

The SARADC analog block can be put in low current consumption mode by setting the PWDN bit in MCR. This is the default mode after exiting from reset. This state must be

exited before starting any operation by resetting the PWDN bit in the MCR. When in power-down mode, no conversion should be started. Bit ADCSTATUS[0] in the MSR is set only when ADC enters power-down mode.

The power-down mode can be requested by setting the PWDN bit in the MCR. If a one-shot conversion is ongoing, the SARADC cannot immediately enter the power-down mode. In fact, the SARADC enters power-down mode only after completing the ongoing conversion. If a scan conversion is ongoing, the ongoing operation shall be first aborted manually by resetting the NSTART bit and using the ABORTCHAIN bit. A power Down request while scan mode conversion is ongoing is ignored.

After the power down phase is completed, the process ongoing before the power down phase must be restarted manually (by setting the appropriate START bit). Resetting PWDN bit and setting NSTART or JSTART bit during the same cycle is forbidden.

CTU triggered conversion should not be initiated during power down mode. In both CTU modes, to request power down mode the application has to wait for the end of the current CTU conversion (CTUSTART bit automatically reset when conversion finishes). Additionally, when SARADC is in CTU control mode the application needs to exit the CTU control mode (by resetting the CTUEN bit) before requesting the power down mode.

42.4.13 Stop mode

The SARADC can be requested to enter this low power mode anytime during the operation. If stop mode is requested while some Normal/Injected mode chained conversions are ongoing, the current channel conversion will be completed and upon stop mode exit the conversion will be resumed from next channel. However, stop mode should not be requested while CTU triggered conversions are ongoing.

42.4.14 Fast comparator mode conversion

In this mode the SARADC analog block/sequencer can be requested to work as a fast comparator instead of a regular data converter for the initiated conversion (single/chain). Here, the sample phase remains unchanged, but the evaluation phase is reduced to just two cycles of fast comparison of the sampled channel value with regards to the corresponding threshold pair values (THRL,THRH) provided at the start of each conversion.

42.4.14.1 Execution order

The programming sequence and execution order when SARADC works in Fast Comparator mode is as follows:

1. The user must configure and program the required SARADC configuration and control registers like MCR, Channel Conversion Mask (CMRs), CTRs in the same way and order as if normal or injected conversion were initiated. CDR/CTR programming constraint follows:

   CTR[INPSAMP] >= 2. Value of 2 for the minimum sampling duration.
2. Enable the Fast Comparator feature by programming the Fast Comparator Enable (MCR[FCE]) bit.
3. Associate and select (program WSELR) 1 out of maximum 16 watchdog threshold pairs (program WTHRHLR[THRL,THRH]) for each of the channels selected for fast

comparison and enable (WENR). WTHTHLR[THRL,THRH] programming constraints are as follows:

a) THRH > THRL

b) THRH < Maximum value of converted analog signal, that is $2^{resolution}$-1

c) THRL > Minimum value of converted analog signal, that is 0

4. Initiate the conversion by setting MCR[NSTART/JSTART].
5. The result of fast comparison from ADC analog block {THRH+1, THRL-1 or (THRH+THRL)/2} and the comparator error status are stored as normal conversion results in the CDR registers.
6. The conversion results are again compared by the watchdog monitor inside the ADC digital interface and resulting violation is reported back to the system by WTISR[WDG0-15L/H] status bits, corresponding interrupts and two external signals (upper threshold violation, lower threshold violation).
7. For additional requirements about smart standby wake up using fast comparator mode, refer to *Section 42.4.15: Smart standby wake up support* for more details.

42.4.15 Smart standby wake up support

Here, the SARADC in conjunction with the CTU and the smart standby wake up unit (SSWU) can be used to wake up the system by regularly checking the sampled values. This is done with respect to the programed thresholds on certain selected channels.

42.4.15.1 Execution order

The programming sequence and execution order when SARADC works in conjunction with the SSWU is as follows:

1. User must configure and program the required SARADC configuration and control registers like MCR, Channel Conversion Mask (CMRs), CTRs.
2. Associate and select (program WSELR) 1 out of maximum 16 watchdog threshold pairs (program WTHRHLR[THRL,THRH]) for each of the channels selected through CMRs and enable (WENR).
3. Set MCR[FCE]; if fast comparator mode conversion is required (optional), the steps 1, 2 and 3 should be performed before the system enters in standby mode.
4. The CTU and SSWU exit the power down mode of SARADC by de-asserting the power down request signal.
5. CTU Initiates conversion providing CTU triggers. The minimum gap between the power down request signal pulse and the first CTU trigger pulse should be greater than 2 to 3 ADC clock cycles.
6. The result of conversion is available in CDR after the end of conversion pulse arrives from SARADC analog block; If MCR[FCE] is set, comparison results are stored as either THRH+1, THRL-1 or (THRH+THRL)/2 in the CDR registers.
7. The conversion results are again compared by the watchdog monitor inside the ADC digital interface and the resulting violation is reported back to the SSWU by two external signals (upper threshold violation, lower threshold violation).
8. SSWU can request for SARADC power-down mode entry by asserting the power down request signal (rising edge) port.

42.5 Memory map and registers

This section provides memory map for the SARADC digital interface.

Note: The actual availability and number of registers and their configuration are chip-specific. For this information, refer to the Chapter 40: Analog-to-Digital Converters (ADC) Configuration.

Table 616. SARADC digital interface register map

Offset	Register	Location
0x000	Main Configuration Register (MCR)	*Section 42.5.1.1*
0x004	Main Status Register (MSR)	*Section 42.5.1.2*
0x008–0x00C	Reserved	
0x010	Interrupt Status Register (ISR)	*Section 42.5.1.3*
0x014–0x01C	Internal channel Interrupt Pending Registers 0–2 (ICIPR0–ICIPR2)	*Section 42.5.1.4*
0x020	Interrupt Mask Register (IMR)	*Section 42.5.1.5*
0x024–0x02C	Internal Channel Interrupt Mask Registers 0–2 (ICIMR0–ICIMR2)	*Section 42.5.1.6*
0x030	Watchdog Threshold Interrupt Status Register (WTISR)	*Section 42.5.1.7*
0x034	Watchdog Threshold Interrupt Mask Register (WTIMR)	*Section 42.5.1.8*
0x038–0x03C	Reserved	
0x040	DMA Enable Register (DMAE)	*Section 42.5.1.9*
0x044–0x04C	Internal Channel DMA Select Registers 0–2 (ICDSR0–ICDSR2)	*Section 42.5.1.10*
0x050–0x05C	Reserved	
0x060–0x06C	Watchdog Threshold Registers 0–3 (WTHRHLR0–WTHRHLR3)	*Section 42.5.1.11*
0x070–0x090	Reserved	
0x094–0x0A0	Conversion Timing Registers 0–3 (CTR0–3)	*Section 42.5.1.12*
0x0A4–0x0AC	Internal Channel Normal Conversion Mask Registers 0–2 (ICNCMR0–ICNCMR2)	*Section 42.5.1.13*
0x0B0	Reserved	
0x0B4–0x0BC	Internal Channel Injected Conversion Mask Registers 0–2 (ICJCMR0–ICJCMR2)	*Section 42.5.1.14*
0x0C0–0x0C4	Reserved	
0x0C8	Power Down Exit Delay Register (PDEDR)	*Section 42.5.1.15*
0x0CC–0x0FC	Reserved	
0x100–0x27C	Internal Channel Data Registers 0–95 (ICDR0–ICDR95)	*Section 42.5.1.16*
0x280–0x2AC	Watchdog Threshold Registers 4-15 (WTHRHLR4-WTHRHLR15)	*Section 42.5.1.17*
0x2B0–0x2DC	Internal Channel Watchdog Selection Registers 0–11 (ICWSELR0–ICWSELR11)	*Section 42.5.1.18*
0x2E0–0x2E8	Internal Channel Watchdog Enable Registers 0–2 (ICWENR0–ICWENR2)	*Section 42.5.1.19*
0x2EC–0x2EF	Reserved	

Table 616. SARADC digital interface register map (continued)

Offset	Register	Location
0x2F0–0x2F8	Internal Channel Analog Watchdog Out of Range Registers 0–2 (ICAWORR0–ICAWORR2)	*Section 42.5.1.20*
0x2FC–0x3FC	Reserved	
0x400	Test Channel Interrupt Pending Register (TCIPR)	*Section 42.5.1.21*
0x404	Test Channel Interrupt Mask Register (TCIMR)	*Section 42.5.1.22*
0x408	Test Channel DMA Select Register (TCDSR)	*Section 42.5.1.23*
0x40C	Test Channel Normal Conversion Mask Register (TCNCMR)	*Section 42.5.1.24*
0x410	Test Channel Injected Conversion Mask Register (TCJCMR)	*Section 42.5.1.25*
0x414–0x420	Test Channel Watchdog Selection Registers 0–3 (TCWSELR0–TCWSELR3)	*Section 42.5.1.26*
0x424	Test Channel Watchdog Enable Register (TCWENR)	*Section 42.5.1.27*
0x428	Test Channel Analog Watchdog Out of Range Register (TCAWORR)	*Section 42.5.1.28*
0x42C	Reserved	
0x430–0x44C	Test Channel Connection with Analog Pin Registers 0–7 (TCCAPR0–TCCAPR7)	*Section 42.5.1.29*
0x450–0x4CC	Test Channel Data Registers 96-127 (TCDR96-TCDR127)	*Section 42.5.1.30*
0x4D0–0x4FC	Reserved	
0x500	External Channel Decode Signals Delay Register (ECDSDR)	*Section 42.5.1.31*
0x504–0x50C	Reserved	
0x510–0x51C	External Channel Interrupt Pending Registers 0–3 (ECIPR0–ECIPR3)	*Section 42.5.1.32*
0x520–0x52C	External Channel Interrupt Mask Registers 0–3 (ECIMR0–ECIMR3)	*Section 42.5.1.33*
0x530–0x53C	External Channel DMA Select Registers 0–3 (ECDSR0–ECDSR3)	*Section 42.5.1.34*
0x540–0x54C	External Channel Normal Conversion Mask Registers 0–3 (ECNCMR0–ECNCMR3)	*Section 42.5.1.35*
0x550–0x55C	External Channel Injected Conversion Mask Registers 0–3 (ECJCMR0–ECJCMR3)	*Section 42.5.1.36*
0x560–0x59C	External Channel Watchdog Selection Registers 0–15 (ECWSELR0–ECWSELR15)	*Section 42.5.1.37*
0x5A0–0x5AC	External Channel Watchdog Enable Registers 0–3 (ECWENR0–ECWENR3)	*Section 42.5.1.38*
0x5B0–0x5BC	External Channel Analog Watchdog Out of Range Registers 0–3 (ECAWORR0–ECAWORR3)	*Section 42.5.1.39*
0x5C0–0x5CC	External Channel Mapping to Internal Channel Registers 0–3 (ECMICR0–ECMICR3)	*Section 42.5.1.40*
0x5D0–0x7CC	External Channel Data Registers 128-255 (ECDR128-ECDR255)	*Section 42.5.1.41*

42.5.1 Register descriptions

42.5.1.1 Main Configuration Register (MCR)

Offset: 0x000 Access: User Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	OWREN	WLSIDE	MODE	Reserved	NSTART	NTRGEN(3)	NEDGESEL(3)		JSTART	JTRGEN(2)	JEDGESEL(2)		JTRGSEQ(2)	0	CTUEN(1)	CTU_MODE(1)
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	Reserved	0	0	0	JTRGSEL(2)				ABORTCHAIN	ABORT	0	FRZ	0	FCE(4)	EDCSELF(5)	PWDN
W																
Reset	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

1. Only available with cross-triggering unit (CTU) feature.
2. Only available with injection trigger feature.
3. Only available with normal trigger feature.
4. Only available with Fast Comparator feature.
5. Only available with External decode format selectable feature.

Figure 507. Main Configuration Register (MCR)

Table 617. MCR field descriptions

Field	Description
0 OWREN	Overwrite enable This bit enables or disables the functionality to overwrite unread converted data in ICDR, TCDR, ECDR registers. 0 Prevents overwrite of unread converted data, new result is discarded. 1 Enables converted data to be overwritten by a new conversion.
1 WLSIDE	Write left/right-aligned 0 The conversion data in ICDR, TCDR, ECDR registers is written right-aligned. 1 Conversion data is left-aligned (from 15 to (15 – resolution + 1))
2 MODE	One Shot/Scan 0 One Shot Mode: Configures the normal conversion of one chain. 1 Scan Mode: Configures continuous chain conversion mode; when the programed chain conversion is finished it restarts immediately

Table 617. MCR field descriptions (continued)

Field	Description

4
NSTART
	Normal Start conversion
	Setting this bit starts the chain or scan conversion. Resetting this bit during scan mode causes the current chain conversion to finish, then stops the operation.
	0 Causes the current chain conversion to finish and stops the operation.
	1 Starts the chain or scan conversion
	Note: The application relies on the corresponding bit in MSR (MSR[NSTART]) to confirm that any changes to this bit has taken effect.
	Example with NSTART bit:
	MCRR[NSTART] = 1; // Register to start normal conversion
	while(MSR[NSTART]==0); // Ensure normal conversion is completed before raising new request

5 NTRGEN	Normal trigger enable 0 Normal trigger disabled to start a normal conversion. 1 Normal trigger enabled to start a normal conversion.
6:7 NEDGESEL	Normal trigger edge selection 00 Falling edge of normal trigger is selected. 01 Rising edge of normal trigger is selected. 10 Both rising and falling edges of normal trigger are selected. (*x* = 0,1). 11 Both rising and falling edges of normal trigger are selected. (*x* = 0,1)
8 JSTART	Injected Start conversion Setting this bit will start the configured injected analog channels to be converted by software. Resetting this bit has no effect, as the injected chain conversion cannot be interrupted. 0 Writing ‘0’ has no effect. 1 Starts the injected chain conversion. **Note:** The application should rely on the corresponding bit in MSR (MSR[JSTART]) to confirm that any changes to this bit have taken effect.
9 JTRGEN	Injection trigger enable 0 Injection trigger disabled for channel injection (injected conversion cannot be started using an injection trigger) 1 Injection trigger enabled for channel injection
10:11 JEDGESEL	Injection trigger edge selection 00 Falling edge of injection trigger is selected 01 Rising edge of injection trigger is selected 10 Both rising and falling edges of injection trigger are selected. (*x* = 0,1) 11 Both rising and falling edges of injection trigger are selected. (*x* = 0,1)
12 JTRGSEQ	Injection trigger sequence enable 0 Injection trigger sequence mode is disabled 1 Injection trigger sequence mode is enabled
14 CTUEN	Cross Trigger Unit Enable 0 The cross triggering unit is disabled and the triggered injected conversion cannot take place 1 The cross triggering unit is enabled and the triggered injected conversion can take place **Note:** Can be set or cleared anytime with some restriction mentioned in CTU trigger/control mode section.

Table 617. MCR field descriptions (continued)

Field	Description
15 CTU_MODE	Cross Trigger Unit Mode This bit is present only if generic parameter CTSEN = 2, otherwise this bit is a reserved bit and always read as 0. This should be set only when ADC Digital is in power-down mode. 0 CTU control mode is selected 1 CTU trigger mode is selected
20:23 JTRGSEL	Injection Trigger Input Selection This bit field selects which trigger input to be selected to start injected conversion. 0000 Trigger input 0 is selected. 0001 Trigger input 1 is selected ... 1111 Trigger input 15 is selected **Note:** JTRGSEL[3:0] must be 0000 when only one trigger input (input 0) is supported
24 ABORTCHAIN	Abort Chain When this bit is set, the ongoing Chain Conversion is aborted. This bit is reset by hardware as soon as a new conversion is requested. 0 Conversion is not affected 1 Aborts the ongoing chain conversion **Note:** Behavior of this request during CTU mode is mentioned in CTU trigger/control mode section.
25 ABORT	Abort Conversion When this bit is set, the ongoing conversion is aborted and a new conversion is invoked. This bit is reset by hardware as soon as a new conversion is invoked. 0 Conversion is not affected 1 Aborts the ongoing conversion **Note:** Behavior of this request during CTU mode is mentioned in CTU trigger/control mode section.
27 FRZ	Freeze This bit enables to stop the SARADC conversions at the end of current channel conversion when SoC enters debug mode. 0 Conversions are not stopped 1 When SoC enters debug mode, further conversions by SARADC analog block will be stopped. The ongoing channel conversion will be completed and the converted data is stored. When the SoC exits debug mode, SARADC resumes the job that was left before halting the conversions. Changing configuration registers during halted condition may cause a pending operation to fail when normal operation is resumed
29 FCE	Fast Comparator Enable This bit enables the fast comparison mode for upcoming channel conversions. 0 Fast Comparison mode disabled 1 Fast Comparison mode enabled

Table 617. MCR field descriptions (continued)

Field	Description
30 EDCSELF	External Decode Channel Format Select (EDCSELF) This bit selects the format for 3-bit output port ipp_decode_extch used as select line for external decode channel 8:1 muxes. The last three bits of external channel number are passed as is (if value 0) or converted to gray code (if value 1) and passed to this port. 0 Binary format 1 Gray Code format
31 PWDN	Power-down enable When this bit is set, the analog module is requested to enter Power Down mode. When SARADC status is PWDN, resetting this bit starts SARADC transition to IDLE mode. 0 SARADC is in normal mode. 1 SARADC has been requested to power down. **Note:** When requesting a power mode transition, the application should rely on the MSR[ADCSTATUS] bit field in order to confirm that the transition has taken effect.

42.5.1.2 Main Status Register (MSR)

Offset: 0x004 Access: User Read Only

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	NSTART	0	0	0	JSTART	0	0	0	0	JABORTCHAIN	0	CTUSTART(1)
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	CHADDR								0	0	0	0	0	ADCSTATUS		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

1. Only available with cross-triggering unit (CTU) feature.

Figure 508. Main Status Register (MSR)

Table 618. MSR field descriptions

Field	Description
4 NSTART	This status bit is used to signal that a normal conversion is ongoing. 0 Normal conversion is not taking place. 1 Normal conversion is ongoing or the normal conversion is pending due to injected conversion or CTU triggered injected conversion.
8 JSTART	This status bit is used to signal that an injected conversion is ongoing. 0 Injected conversion is not taking place. 1 Injected conversion is ongoing.
13 JABORTCHAIN	This status bit is used to signal that an Injected conversion chain has been aborted. This bit is reset when a new conversion starts. 0 Last injected conversion chain has not been aborted. 1 Last injected conversion chain has been aborted.
15 CTUSTART	This status bit is used to signal that a CTU conversion is ongoing. This bit is set when a CTU trigger pulse is received and the CTU conversion starts. When CTU trigger mode is enabled this bit is automatically reset when the conversion is completed. Otherwise, if Control Mode is enabled this bit can be reset only when the CTU is disabled (CTUEN set to ‘0’). 0 CTU triggered injected conversion is not taking place. 1 CTU triggered injected conversion is ongoing.
16:23 CHADDR	Channel under measure address This status bit is used to signal which channel is under measure.
29:31 ADCSTATUS	The value of this parameter depends on ADC status. While requesting an external channel conversion, SARADC digital interface needs to wait for a fixed time determined by DSD bit field of DSDR before proceeding to actual sampling phase. This is known as wait state. 000 IDLE 001 Power-down 010 Wait state 011 — 100 Sample 101 — 110 Conversion 111 —

42.5.1.3 Interrupt Status Register (ISR)

Offset: 0x010 Access: User Read/Write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	CTUTRGERR(1)	EOCTU(1)	JEOC	JECH	NEOC	NECH
W											w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. Only available with cross-triggering unit (CTU) feature.

Figure 509. Interrupt Status Register (ISR)

Table 619. ISR field descriptions

Field	Description
26 CTUTRGERR	CTU Trigger Error Flag This Error Flag indicates that the next CTU trigger has been discarded as it overlapped during execution of current CTU conversion. This indicated that CTU sampling duration is less than programed conversion time of ADC. 0 No CTU trigger overlap error condition since last CTU conversion. 1 CTU trigger overlap error has occurred.
27 EOCTU	End of CTU channel conversion interrupt flag This bit indicates the end of conversion for a CTU injected channel. 0 End of conversion of CTU triggered channel has not occurred since the last time the flag is cleared. 1 End of conversion of CTU triggered channel has occurred or the flag has not been cleared since the last end of conversion occurrence.
28 JEOC	End of injected channel conversion interrupt flag This bit indicates the end of conversion for an injected channel. 0 End of conversion of injected channel has not occurred since the last time the flag is cleared. 1 End of conversion of injected channel has occurred or the flag has not been cleared since the last end of conversion occurrence.
29 JECH	End of injected chain conversion interrupt flag This bit indicates the end of conversion for an injected chain. 0 End of conversion of injected chain has not occurred since the last time the flag is cleared. 1 End of conversion of injected chain has occurred or the flag has not been cleared since the last end of conversion occurrence.

Table 619. ISR field descriptions (continued)

Field	Description
30 NEOC	End of normal channel conversion interrupt flag This bit indicates the end of conversion for a normal channel. 0 End of conversion of normal channel has not occurred since the last time the flag is cleared. 1 End of conversion of normal channel has occurred or the flag has not been cleared since the last end of conversion occurrence.
31 NECH	End of normal chain conversion interrupt flag This bit indicates the end of conversion for an normal chain. 0 End of conversion of normal chain has not occurred since the last time the flag is cleared. 1 End of conversion of normal chain has occurred or the flag has not been cleared since the last end of conversion occurrence.

42.5.1.4 Internal Channel Interrupt Pending Register *n* (ICIPR*n*)

The interrupt channel register to channel association is described in *Table 621*.

Offset: 0x014 + *n**0x4 (*n* = 0 to 2) Access: User Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	EOC_CH[x]															
W	w1c															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	EOC_CH[x]															
W	w1c															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 510. Internal Channel Interrupt Pending Register *n* (ICIPR*n*)

Table 620. ICIPR*n* field descriptions

Field	Description
0:31 EOC_CH[x]	End of conversion interrupt pending bit for channel x 0 End of conversion for CH[x] has not occurred. 1 End of conversion for CH[x] has occurred.

Table 621. ICIPR0–2 Registers to channel association

Register	Register bits 0:31
ICIPR0	EOC_CH[31:0]
ICIPR1	EOC_CH[63:32]
ICIPR2	EOC_CH[95:64]

Note: If channel CH[x] is not existing then the corresponding bit field EOC_CH[x] is reserved and returns 0.

If all channels CH[x] are not existing, the full register is not existing and any attempt to access this space generates a transfer error.

42.5.1.5 Interrupt Mask Register (IMR)

The description of the Interrupt Mask Register is given below.

Offset: 0x020 Access: User Read/Write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	MSKEOCTU(1)	MSKJEOC	MSKJECH	MSKNEOC	MSKNECH
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. Only available with cross-triggering unit (CTU) feature.

Figure 511. Interrupt Mask Register (IMR)

Table 622. IMR field descriptions

Field	Description
27 MSKEOCTU	Mask bit for EOCTU 0 EOCTU interrupt is disabled. 1 EOCTU interrupt is enabled.
28 MSKJEOC	Mask bit for JEOC 0 JEOC interrupt is disabled. 1 JEOC interrupt is enabled.
29 MSKJECH	Mask bit for JECH 0 JECH interrupt is disabled. 1 JECH interrupt is enabled.
30 MSKNEOC	Mask bit for NEOC 0 NEOC interrupt is disabled. 1 NEOC interrupt is enabled.
31 MSKNECH	Mask bit for NECH 0 NECH interrupt is disabled. 1 NECH interrupt is enabled.

42.5.1.6 Internal Channel Interrupt Mask Register *n* (ICIMR*n*)

The interrupt mask register to channel association is described in *Table 624*.

Offset: 0x024 + *n**0x4 (*n* = 0 to 2) Access: User Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	IM_CH[x]															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	IM_CH[x]															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 512. Internal Channel Interrupt Mask Register *n* (ICIMR*n*)

Table 623. ICIMR*n* field descriptions

Field	Description
0:31 IM_CH[x]	IM_CH[x]: Interrupt mask bit for channel x 0 Interrupt for CH[x] is disabled. 1 Interrupt for CH[x] is enabled.

Table 624. Interrupt Mask Registers to Channel Association

Register	Register bits 0:31
ICIMR0	IM_CH[31:0]
ICIMR1	IM_CH[63:32]
ICIMR2	IM_CH[95:64]

Note: If channel CH[x] is not existing then the corresponding bit field IM_CH[x] is reserved and returns 0.

If all channels CH[x] are not existing, the full register is not existing and any attempt to access this space generates a transfer error.

42.5.1.7 Watchdog Threshold Interrupt Status Register (WTISR)

This register gives the interrupt status information for the 16 possible upper and 16 possible lower threshold limits monitored by up to 16 watchdog monitors which can be selected for each channel.

Offset: 0x030 Access: User Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	WDG15H	WDG15L	WDG14H	WDG14L	WDG13H	WDG13L	WDG12H	WDG12L	WDG11H	WDG11L	WDG10H	WDG10L	WDG9H	WDG9L	WDG8H	WDG8L
W	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	WDG7H	WDG7L	WDG6H	WDG6L	WDG5H	WDG5L	WDG4H	WDG4L	WDG3H	WDG3L	WDG2H	WDG2L	WDG1H	WDG1L	WDG0H	WDG0L
W	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 513. Watchdog Threshold Interrupt Status Register (WTISR)

Table 625. WTISR field descriptions

Field	Description
30-2x WDGxH [x = 0–15]	This corresponds to the interrupt generated on the converted value being higher than the programed higher threshold as reported by WDG monitor ‘x’. 0 Converted data is not higher than the programed higher threshold. 1 Converted data is higher than the programed higher threshold.
31-2x WDGxL [x = 0–15]	This corresponds to the interrupt generated on the converted value being lower than the programed lower threshold as reported by WDG monitor ‘x’. 0 Converted data is not lower than the programed lower threshold. 1 Converted data is lower than the programed lower threshold.

42.5.1.8 Watchdog Threshold Interrupt Mask Register (WTIMR)

This register gives the interrupt mask information for the 16 possible upper and 16 possible lower threshold limits which can be selected for each channel.

Offset: 0x034 Access: User Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	MSKWDG15H	MSKWDG15L	MSKWDG14H	MSKWDG14L	MSKWDG13H	MSKWDG13L	MSKWDG12H	MSKWDG12L	MSKWDG11H	MSKWDG11L	MSKWDG10H	MSKWDG10L	MSKWDG9H	MSKWDG9L	MSKWDG8H	MSKWDG8L
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	MSKWDG7H	MSKWDG7L	MSKWDG6H	MSKWDG6L	MSKWDG5H	MSKWDG5L	MSKWDG4H	MSKWDG4L	MSKWDG3H	MSKWDG3L	MSKWDG2H	MSKWDG2L	MSKWDG1H	MSKWDG1L	MSKWDG0H	MSKWDG0L
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 514. Watchdog Threshold Interrupt Mask Register (WTIMR)

Table 626. WTIMR field descriptions

Field	Description
30-2x MSKWDGxH [x = 0–15]	This corresponds to the mask bit for the interrupt generated on the converted value being higher than the programed higher threshold. 0 Interrupt for WDGxH is disabled. 1 Interrupt for WDGxH is enabled.
31-2x MSKWDGxL [x = 0–15]	This corresponds to the mask bit for the interrupt generated on the converted value being lower than the programed lower threshold. 0 Interrupt for WDGxL is disabled. 1 Interrupt for WDGxL is enabled.

42.5.1.9 DMA Enable Register (DMAE)

The DMA Enable (DMAE) register sets up the DMA for use with the SARADC.

Offset: 0x040 Access: User Read/Write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	DCLR	DMAEN
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 515. DMA Enable Register (DMAE)

Table 627. DMAE register field descriptions

Field	Description
30 DCLR	DMA clear sequence enable 0 DMA request cleared by Acknowledge from DMA controller. 1 DMA request cleared on read of data registers.
31 DMAEN	DMA global enable 0 DMA feature disabled. 1 DMA feature enabled.

42.5.1.10 Internal Channel DMA Select Register *n* (ICDSR*n*)

The DMA select registers to channel association is described in *Table 629*.

Offset: 0x044 + *n**0x4 (*n* = 0 to 2) Access: User Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	DS_CH[x]															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	DS_CH[x]															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 516. Internal Channel DMA Select Register *n* (ICDSR*n*)

Table 628. ICDSR*n* field descriptions

Field	Description
0:31 DS_CH[x]	DMA select for channel x 0 CH[x] is disabled to transfer data in DMA mode. 1 CH[x] is enabled to transfer data in DMA mode.

Table 629. Internal Channel DMA Select Registers to channel association

Register	Register bits 0:31
ICDSR0	DS_CH[31:0]
ICDSR1	DS_CH[63:32]
ICDSR2	DS_CH[95:64]

Note: If channel CH[x] is not existing then the corresponding bit field DS_CH[x] is reserved and returns 0.

If all channels CH[x] are not existing, the full register is not existing and any attempt to access this space generates a transfer error.

42.5.1.11 Watchdog Threshold Register *n* (WTHRHLR*n*)

Note: The number of WTHRHLR registers corresponds to the number of analogs watchdogs implemented in the ADC module.

The length of the threshold field depends on the SARADC resolution. For 10-bit SARADC, the bits 4, 5, 20 and 21 are reserved.

Note: The reset value of the bits 4 and 5 is 0 instead of 1 in case SARADC resolution is 10-bit.

Offset: 0x060 + *n**0x4 (*n* = 0 to 3) Access: User Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	THRH											
W																
Reset	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	THRL											
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 517. Watchdog Threshold Register *n* (WTHRHLR*n*)

Table 630. WTHRHLR*n* field descriptions

Field	Description
4:15 THRH	High threshold value for channel x
20:31 THRL	Low threshold value for channel x

42.5.1.12 Conversion Timing Register *n* (CTR*n*)

Offset: 0x094 + *n**0x4 (*n* = 0 to 3) Access: User Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	CRES	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	PRECHG				INPSAMP							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 518. Conversion Timing Register *n* (CTR*n*)

Table 631. CTR*n* field descriptions

Field	Description
0 CRES	Conversion Resolution select This bit selects the conversion resolution for the conversion. 0 12-bit resolution in case of 12-bit SARADC. 10-bit resolution in case of 10-bit SARADC 1 10-bit resolution in case of 12-bit SARADC. 8-bit resolution in case of 10-bit SARADC
20:23 PRECHG	Precharging phase duration This bit field defines the precharging phase duration to be applied when this phase is enabled by PCE bit of channel data register. This duration is calculated as (PRECHG X 1/Frequency of SARADC clock). Refer to device datasheet for the minimum duration required for precharging phase.
24:31 INPSAMP	Sampling phase duration This bit field defines the sampling phase duration to be applied. This duration is calculated as (INPSAMP X 1/Frequency of SARADC clock). Refer to device datasheet for the minimum duration required for sampling phase.

42.5.1.13 Internal Channel Normal Conversion Mask Register *n* (ICNCMR*n*)

The normal conversion mask registers to channel association is described in *Table 633*.

Offset: 0x0A4 + *n**0x4 (*n* = 0 to 2) Access: User Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	NCE_CH[x]															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	NCE_CH[x]															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 519. Internal Channel Normal Conversion Mask Register *n* (ICNCMR*n*)

Table 632. ICNCMR*n* field descriptions

Field	Description
0:31 NCE_CH[x]	Normal conversion enable for channel x 0 Normal conversion is disabled for CH[x]. 1 Normal conversion is enabled for CH[x].

Table 633. Internal Channel Normal Conversion Mask Registers to channel association

Register	Register bits 0:31
ICNCMR0	NCE_CH[31:0]

Table 633. Internal Channel Normal Conversion Mask Registers to channel association (continued)

Register	Register bits 0:31
ICNCMR1	NCE_CH[63:32]
ICNCMR2	NCE_CH[95:64]

Note: If channel CH[x] is not existing then the corresponding bit field NCE_CH[x] is reserved and returns 0.

If all channels CH[x] are not existing, the full register is not existing and any attempt to access this space generates a transfer error.

42.5.1.14 Internal Channel Injected Conversion Mask Register *n* (ICJCMR*n*)

The injected conversion mask registers to channel association is described in *Table 635*.

Offset: 0x0B4 + n*0x4 (n = 0 to 2) Access: User Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	JCE_CH[x]															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	JCE_CH[x]															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 520. Internal Channel Injected Conversion Mask Register *n* (ICJCMR*n*)

Table 634. ICJCMR*n* field descriptions

Field	Description
0:31 JCE_CH[x]	Injected conversion enable for channel x 0 Injected conversion is disabled for CH[x]. 1 Injected conversion is enabled for CH[x].

Table 635. Internal Channel Normal Conversion Mask Registers to channel association

Register	Register bits 0:31
ICJCMR0	JCE_CH[31:0]
ICJCMR1	JCE_CH[63:32]
ICJCMR2	JCE_CH[95:64]

Note: If channel CH[x] is not existing then the corresponding bit field JCE_CH[x] is reserved and returns 0.

If all channels CH[x] are not existing, the full register is not existing and any attempt to access this space generates a transfer error.

42.5.1.15 Power Down Exit Delay Register (PDEDR)

Offset: 0x0C8 Access: User Read/Write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	PDED							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 521. Power Down Exit Delay Register (PDEDR)

Table 636. PDEDR field descriptions

Field	Description
24:31 PDED	Power down exit delay duration Defines the delay between the power-down bit reset and the starting of conversion. The power-down delay is calculated as [PDED × 1/(SARADC clock frequency)]. The minimum required power down delay is the sum of $t_{ADCBIASINIT}$ (time required by ADCBIAS to generate correct value of bias current for all ADCs) and $t_{ADCINIT}$ (initialization time for ADC after the generation of bias current). Both values are provided in the device datasheet.

42.5.1.16 Internal Channel Data Register n (ICDR*n*)

The conversion results for each channel are loaded into data registers. Each data register will contain the conversion result, status information related to ADC mode, data valid, timing parameter selection for each channel.

This register should be accessed via 32-bit R/W.

Offset: 0x100 + *n**0x4 (*n* = 0 to 95) Access: User Read/Write

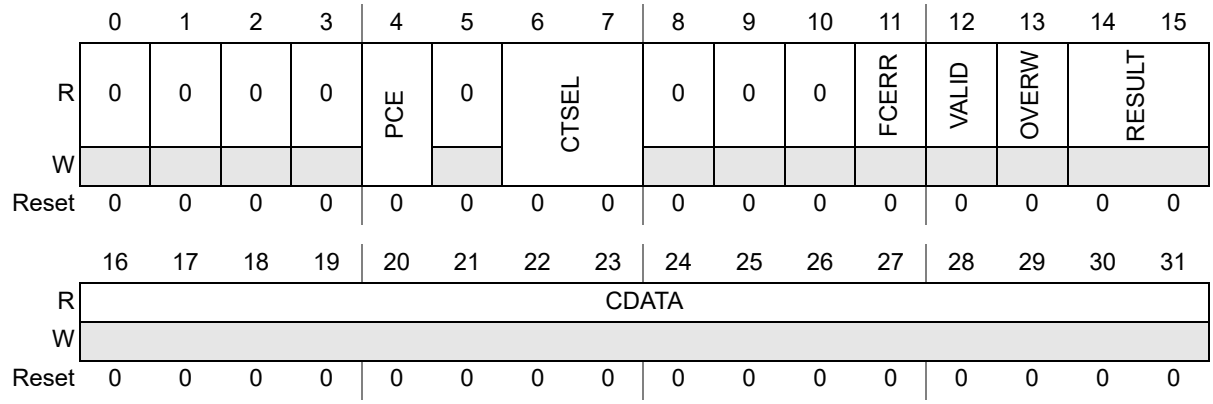


Figure 522. Internal Channel Data Register *n* (ICDR*n*)

Table 637. ICDR*n* field descriptions

Field	Description
4 PCE	Precharge Enable This bit enables the precharging phase during channel conversion. 0 Precharge phase is disabled. 1 Precharge phase is enabled.
6:7 CTSEL	This bit field selects the conversion timing register for each channel to select different precharge and sampling phase durations. 00 CTR0 is selected. 01 CTR1 is selected. 10 CTR2 is selected. 11 CTR3 is selected.
11 FCERR	Fast Comparator Error Flag This bit is used to notify error condition on current fast comparator operation inside ADC analog block. It is automatically cleared when data is read. 0 Cleared by reading the Converted by software. 1 Fast Comparator Error Condition has occurred in current conversion.
12 VALID	Data valid flag This bit is used to notify when the data is valid (a new value has been written). It is automatically cleared when data is read. 0 Converted data has been read by software. 1 Converted data is valid and has not been read yet.
13 OVERW	Data overwritten flag This bit signals that the previous converted data has been overwritten by a new conversion. This functionality depends on the value of MCR[OWREN]: – When OWREN = 0, OVERW is frozen to 0 and CDATA field is protected against being overwritten until being read. – When OWREN = 1, OVERW flags the CDATA field overwrite status. 0 Converted data has not been overwritten. 1 Previous converted data has been overwritten before having been read.

Table 637. ICDR*n* field descriptions (continued)

Field	Description
14:15 RESULT	Conversion result mode status This bit reflects the mode of conversion for the corresponding channel. 00 Data is a result of Normal conversion mode. 01 Data is a result of Injected conversion mode. 10 Data is a result of CTU conversion mode. 11 Reserved
16:31 CDATA	Channel converted data **Note:** The data organization inside the CDATA field is dependent from: – The SARADC type which can either be a 12-bit type, a 10-bit type or a 9-bit type. – The setting of CTRx.CRES which selects if conversion is done with full resolution or with reduced resolution. – The setting of MCR.WLSIDE which selects the read data alignment. The valid data bits position is summarized in *Table 638*.

Table 638. Valid data bit organization

SARADC type	CTRx.CRES setting	MCR.WLSIDE setting	Effective Resolution	Data Alignment	Data Bit valid	Data Bit read as zero	Data Bit to be ignored
12-bit	0	0	12-bit	Right	CDATA[20:31]	CDATA[16:19]	—
		1		Left	CDATA[16:27]	CDATA[28:31]	—
	1	0	10-bit	Right	CDATA[22:31]	CDATA[16:21]	—
		1		Left	CDATA[16:25]	CDATA[26:31]	—
10-bit	0	0	10-bit	Right	CDATA[22:31]	CDATA[16:21]	—
		1		Left	CDATA[16:25]	CDATA[26:31]	—
	1	0	8-bit	Right	CDATA[24:31]	CDATA[16:23]	—
		1		Left	CDATA[16:23]	CDATA[24:31]	—
9-bit	0	0	9-bit	Right	CDATA[22:30]	CDATA[16:21]	CDATA[31]
		1		Left	CDATA[16:24]	CDATA[26:31]	CDATA[25]
	1	0	8-bit	Right	CDATA[24:31]	CDATA[16:23]	—
		1		Left	CDATA[16:23]	CDATA[24:31]	—

42.5.1.17 Watchdog Threshold Register *n* (WTHRHLR*n*)

Note: The number of WTHRHLR registers corresponds to the number of analogs watchdogs implemented in the ADC module.

The length of the threshold field depends on the SARADC resolution. For 10-bit SARADC, the bits 4, 5, 20 and 21 are reserved.

Offset: 0x280 + n*0x4 (n = 4 to 15) Access: User Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	THRH											
W																
Reset	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	THRL											
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 523. Watchdog Threshold Register *n* (WTHRHLR*n*)

Table 639. WTHRHLR*n* field descriptions

Field	Description
4:15 THRH	High threshold value for channel x
20:31 THRL	Low threshold value for channel x

42.5.1.18 Internal Channel Watchdog Select Register *n* (ICWSELR*n*)

These registers contain WSEL_CHx[0:3] bit fields to select the threshold register which provides the values to be used for upper and lower bounds for channel x.

The Watchdog select registers to channel association is described in *Table 641*.

Offset: 0x2B0 + n*0x4 (*n* = 0 to 11) Access: User Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	WSEL_CH[(8*n)+7]				WSEL_CH[(8*n)+6]				WSEL_CH[(8*n)+5]				WSEL_CH[(8*n)+4]			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	WSEL_CH[(8*n)+3]				WSEL_CH[(8*n)+2]				WSEL_CH[(8*n)+1]				WSEL_CH[8*n]			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 524. Internal Channel Watchdog Select Register *n* (ICWSELR*n*)

Table 640. ICWSELR*n* field descriptions

Field	Description
0:3 WSEL_CH[(8*n)+7]	Watchdog select for channel x 0000 THRHLR0 register is selected. 0001 THRHLR1 register is selected. 0010 THRHLR2 register is selected. … 1110 THRHLR14 register is selected. 1111 THRHLR15 register is selected.
4:7 WSEL_CH[(8*n)+6]	Watchdog select for channel x 0000 THRHLR0 register is selected. 0001 THRHLR1 register is selected. 0010 THRHLR2 register is selected. … 1110 THRHLR14 register is selected. 1111 THRHLR15 register is selected.
8:11 WSEL_CH[(8*n)+5]	Watchdog select for channel x 0000 THRHLR0 register is selected. 0001 THRHLR1 register is selected. 0010 THRHLR2 register is selected. … 1110 THRHLR14 register is selected. 1111 THRHLR15 register is selected.
12:15 WSEL_CH[(8*n)+4]	Watchdog select for channel x 0000 THRHLR0 register is selected. 0001 THRHLR1 register is selected. 0010 THRHLR2 register is selected. … 1110 THRHLR14 register is selected. 1111 THRHLR15 register is selected.
16:19 WSEL_CH[(8*n)+3]	Watchdog select for channel x 0000 THRHLR0 register is selected. 0001 THRHLR1 register is selected. 0010 THRHLR2 register is selected. … 1110 THRHLR14 register is selected. 1111 THRHLR15 register is selected.
20:23 WSEL_CH[(8*n)+2]	Watchdog select for channel x 0000 THRHLR0 register is selected. 0001 THRHLR1 register is selected. 0010 THRHLR2 register is selected. … 1110 THRHLR14 register is selected. 1111 THRHLR15 register is selected.

Table 640. ICWSELR*n* field descriptions (continued)

Field	Description
24:27 WSEL_CH[(8*n)+1]	Watchdog select for channel x 0000 THRHLR0 register is selected. 0001 THRHLR1 register is selected. 0010 THRHLR2 register is selected. … 1110 THRHLR14 register is selected. 1111 THRHLR15 register is selected.
28:31 WSEL_CH[8*n]	Watchdog select for channel x 0000 THRHLR0 register is selected. 0001 THRHLR1 register is selected. 0010 THRHLR2 register is selected. … 1110 THRHLR14 register is selected. 1111 THRHLR15 register is selected.

Table 641. Internal Channel Watchdog Enable Registers to channel association

Register	Register bits 0:31
ICWSELRn (n = 0 to 11)	WSEL_CH[(8*n)+7] … WSEL_CH[8*n]

The number of bits implemented for each WSEL_CHx depends on Watchdog number:

- If Watchdog number > 8, the number of bits implemented is 4 (WSEL_CHx[3:0])
- If 4 < Watchdog number <= 8, the number of bits implemented is 3 (WSEL_CHx[3] is reserved).
- If 1 < Watchdog number <= 4, the number of bits implemented is 2 (WSEL_CHx[3:2] is reserved).
- If Watchdog number = 1, only 1 bit is present.

Each such nibble (WSEL_CHx) has positional correspondence to Internal Channel number (each channel has a 1:1 nibble correspondence), so each word ICWSELRx corresponds to eight internal Channels for a total of twelve 32-bit registers to covering all 96 test channels.

If an Internal channel is not defined, the corresponding WSEL_CHx nibble is not implemented. For the implemented nibbles, width of WSEL_CHx field within the nibble is dependent on the Watchdog number. For example, if Watchdog number is 4 than only two LSBs of each WSEL_CHx are implemented and remaining two will remain unimplemented. If both nibbles in a byte-access, all four nibbles in a halfword-access, and all eight nibbles in a word-access are unimplemented, a transfer error is generated.

42.5.1.19 Internal Channel Watchdog Enable Register *n* (ICWENR*n*)

The Watchdog enable registers to channel association is described in *Table 643*.

Offset: 0x2E0 + *n**0x4 (*n* = 0 to 2) Access: User Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	WEN_CH[x]															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	WEN_CH[x]															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 525. Internal Channel Watchdog Enable Register *n* (ICWENR*n*)

Table 642. ICWENRn field descriptions

Field	Description
0:31 WEN_CH[x]	Watchdog enable for channel x 0 Watchdog feature is disabled for CH[x]. 1 Watchdog feature is enabled for CH[x].

Table 643. Internal Channel Watchdog Enable Registers to channel association

Register	Register bits 0:31
ICWER0	WEN_CH[31:0]
ICWER1	WEN_CH[63:32]
ICWER2	WEN_CH[95:64]

Note: If channel CH[x] is not existing then the corresponding bit field WEN_CH[x] is reserved and returns 0.

If all channels CH[x] are not existing, the full register is not existing and any attempt to access this space generates a transfer error.

42.5.1.20 Internal Channel Analog Watchdog Out of Range Register *n* (ICAWORR*n*)

The analog watchdog registers to channel association is described in *Table 645*.

Offset: 0x2F0 + n*0x4 (*n* = 0 to 2) Access: User Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	AWOR_CH[x]															
W	w1c															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	AWOR_CH[x]															
W	w1c															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 526. Internal Channel Analog Watchdog Out of Range Register *n* (ICAWORR*n*)

Table 644. ICAWORR*n* field descriptions

Field	Description
0:31 AWOR_CH[x]	Analog watchdog out of range status for channel x, provided corresponding WEN_CHx bit is set. 0 CH[x] converted data is not out of range determined by its thresholds. 1 CH[x] converted data is out of range determined by its thresholds.

Table 645. Internal Channel Analog Watchdog Registers to channel association

Register	Register bits 0:31
ICAWORR0	AWOR_CH[31:0]
ICAWORR1	AWOR_CH[63:32]
ICAWORR2	AWOR_CH[95:64]

Note: If channel CH[x] is not existing then the corresponding bit field AWOR_CH[x] is reserved and returns 0.

If all channels CH[x] are not existing, the full register is not existing and any attempt to access this space generates a transfer error.

42.5.1.21 Test Channel Interrupt Pending Register (TCIPR)

The interrupt channel register to channel association is described in *Table 647*.

Offset: 0x400 Access: User Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	EOC_CH[x]															
W	w1c															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	EOC_CH[x]															
W	w1c															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 527. Test Channel Interrupt Pending Register (TCIPR)

Table 646. TCIPR field descriptions

Field	Description
0:31 EOC_CH[x]	End of conversion interrupt pending bit for channel x 0 End of conversion for CH[x] has not occurred. 1 End of conversion for CH[x] has occurred.

Table 647. TCIPR Register to channel association

Register	Register bits 0:31
TCIPR	EOC_CH[127:96]

Note: If channel CH[x] is not existing then the corresponding bit field EOC_CH[x] is reserved and returns 0.

If all channels CH[x] are not existing, the full register is not existing and any attempt to access this space generates a transfer error.

42.5.1.22 Test Channel Interrupt Mask Register (TCIMR)

The interrupt mask register to channel association is described in *Table 649*.

Offset: 0x404 Access: User Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	IM_CH[x]															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	IM_CH[x]															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 528. Test Channel Interrupt Mask Register (TCIMR)

Table 648. TCIMR field descriptions

Field	Description
0:31 IM_CH[x]	Interrupt mask bit for channel x 0 Interrupt for CH[x] is disabled. 1 Interrupt for CH[x] is enabled.

Table 649. Test Channel Interrupt Mask Register to channel association

Register	Register bits 0:31
TCIMR	IM_CH[127:96]

Note: If channel CH[x] is not existing then the corresponding bit field IM_CH[x] is reserved and returns 0.

If all channels CH[x] are not existing, the full register is not existing and any attempt to access this space generates a transfer error.

42.5.1.23 Test Channel DMA Select Register (TCDSR)

The DMA select registers to channel association is described in *Table 651*.

Offset: 0x408 Access: User Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	DS_CH[x]															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	DS_CH[x]															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 529. Test Channel DMA Select Register (TCDSR)

Table 650. TCDSR field descriptions

Field	Description
0:31 DS_CH[x]	DMA select for channel x 0 CH[x] is disabled to transfer data in DMA mode. 1 CH[x] is enabled to transfer data in DMA mode.

Table 651. Test Channel DMA Select Registers to channel association

Register	Register bits 0:31
TCDSR	DS_CH[127:96]

Note: If channel CH[x] is not existing then the corresponding bit field DS_CH[x] is reserved and returns 0.

If all channels CH[x] are not existing, the full register is not existing and any attempt to access this space generates a transfer error.

42.5.1.24 Test Channel Normal Conversion Mask Register (TCNCMR)

The normal conversion mask register to channel association is described in *Table 653*.

Offset: 0x40C Access: User Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	NCE_CH[x]															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	NCE_CH[x]															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 530. Test Channel Normal Conversion Mask Register (TCNCMR)

Table 652. TCNCMR field descriptions

Field	Description
0:31 NCE_CH[x]	Normal conversion enable for channel x 0 Normal conversion is disabled for CH[x]. 1 Normal conversion is enabled for CH[x].

Table 653. Test Channel Normal Conversion Mask Register to channel association

Register	Register bits 0:31
TCNCMR	NCE_CH[127:96]

Note: If channel CH[x] is not existing then the corresponding bit field NCE_CH[x] is reserved and returns 0.

If all channels CH[x] are not existing, the full register is not existing and any attempt to access this space generates a transfer error.

42.5.1.25 Test Channel Injected Conversion Mask Register (TCJCMR)

The injected conversion mask register to channel association is described in *Table 655*.

Offset: 0x410 Access: User Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	JCE_CH[x]															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	JCE_CH[x]															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 531. Test Channel Injected Conversion Mask Register (TCJCMR)

Table 654. TCJCMR field descriptions

Field	Description
0:31 JCE_CH[x]	JCE_CH[x]: Injected conversion enable for channel x 0 Injected conversion is disabled for CH[x]. 1 Injected conversion is enabled for CH[x].

Table 655. Test Channel Normal Conversion Mask Register to channel association

Register	Register bits 0:31
TCJCMR	JCE_CH[127:96]

Note: If channel CH[x] is not existing then the corresponding bit field JCE_CH[x] is reserved and returns 0.

If all channels CH[x] are not existing, the full register is not existing and any attempt to access this space generates a transfer error.

42.5.1.26 Test Channel Watchdog Select Register *n* (TCWSELR*n*)

This register contains WSEL_CHx[0:3] bit fields to select the threshold register which provides the values to be used for upper and lower bounds for channel x.

The Watchdog select register to channel association is described in *Table 657*.

Offset: 0x414 + *n**0x4 (*n* = 0 to 3) Access: User Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	WSEL_CH[(8*n)+103]				WSEL_CH[(8*n)+102]				WSEL_CH[(8*n)+101]				WSEL_CH[(8*n)+100]			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	WSEL_CH[(8*n)+99]				WSEL_CH[(8*n)+98]				WSEL_CH[(8*n)+97]				WSEL_CH[(8*n)+96]			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 532. Test Channels Watchdog Select Register *n* (TCWSELR*n*)

Table 656. TCWSELR*n* field descriptions

Field	Description
4:7 WSEL_CH[(8*n)+103]	Watchdog select for channel x 0000 THRHLR0 register is selected. 0001 THRHLR1 register is selected. 0010 THRHLR2 register is selected. … 1110 THRHLR14 register is selected. 1111 THRHLR15 register is selected.
0:3 WSEL_CH[(8*n)+102]	Watchdog select for channel x 0000 THRHLR0 register is selected. 0001 THRHLR1 register is selected. 0010 THRHLR2 register is selected. … 1110 THRHLR14 register is selected. 1111 THRHLR15 register is selected.
8:11 WSEL_CH[(8*n)+101]	Watchdog select for channel x 0000 THRHLR0 register is selected. 0001 THRHLR1 register is selected. 0010 THRHLR2 register is selected. … 1110 THRHLR14 register is selected. 1111 THRHLR15 register is selected.

Table 656. TCWSELR*n* field descriptions (continued)

Field	Description
12:15 WSEL_CH[(8*n)+100]	Watchdog select for channel x 0000 THRHLR0 register is selected. 0001 THRHLR1 register is selected. 0010 THRHLR2 register is selected. … 1110 THRHLR14 register is selected. 1111 THRHLR15 register is selected.
16:19 WSEL_CH[(8*n)+99]	Watchdog select for channel x 0000 THRHLR0 register is selected. 0001 THRHLR1 register is selected. 0010 THRHLR2 register is selected. … 1110 THRHLR14 register is selected. 1111 THRHLR15 register is selected.
20:23 WSEL_CH[(8*n)+98]	Watchdog select for channel x 0000 THRHLR0 register is selected. 0001 THRHLR1 register is selected. 0010 THRHLR2 register is selected. … 1110 THRHLR14 register is selected. 1111 THRHLR15 register is selected.
24:27 WSEL_CH[(8*n)+97]	Watchdog select for channel x 0000 THRHLR0 register is selected. 0001 THRHLR1 register is selected. 0010 THRHLR2 register is selected. … 1110 THRHLR14 register is selected. 1111 THRHLR15 register is selected.
28:31 WSEL_CH[(8*n)+96]	Watchdog select for channel x 0000 THRHLR0 register is selected. 0001 THRHLR1 register is selected. 0010 THRHLR2 register is selected. … 1110 THRHLR14 register is selected. 1111 THRHLR15 register is selected.

Table 657. Test Channel Watchdog Select Registers to channel association

Register	Register bits 0:31
TCWSELR0	WSEL_CH[103] … WSEL_CH[96]
TCWSELR1	WSEL_CH[111] … WSEL_CH[104]
TCWSELR2	WSEL_CH[119] … WSEL_CH[112]
TCWSELR3	WSEL_CH[127] … WSEL_CH[120]

The number of bits implemented for each WSEL_CHx depends on Watchdog number:

- If Watchdog number > 8, the number of bits implemented is 4 (WSEL_CHx[3:0])
- If 4 < Watchdog number <= 8, the number of bits implemented is 3 (WSEL_CHx[3] is reserved).
- If 1 < Watchdog number <= 4, the number of bits implemented is 2 (WSEL_CHx[3:2] is reserved).
- If Watchdog number = 1, only 1 bit is present.

Each such nibble (WSEL_CHx) has positional correspondence to Test Channel number (each channel has a 1:1 nibble correspondence), so each word TCWSELRx corresponds to eight test Channels, thus requiring total four 32-bit registers to cover all 32 test channels.

If a Test channel is not defined, the corresponding WSEL_CHx nibble is not implemented. For the implemented nibbles, width of WSEL_CHx field within the nibble is dependent on the Watchdog number. For example, if Watchdog number is 4 than only two LSBs of each WSEL_CHx are implemented and remaining two will remain unimplemented. If both nibbles in a byte-access, all four nibbles in a halfword-access, and all eight nibbles in a word-access are unimplemented, a transfer error is generated.

42.5.1.27 Test Channel Watchdog Enable Register (TCWENR)

The Watchdog enable registers to channel association is described in *Table 659*.

Offset: 0x424 Access: User Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	WEN_CH[x]															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	WEN_CH[x]															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 533. Test Channel Watchdog Enable Register (TCWENR)

Table 658. TCWENR field descriptions

Field	Description
0:31 WEN_CH[x]	Watchdog enable for channel x 0 Watchdog feature is disabled for CH[x]. 1 Watchdog feature is enabled for CH[x].

Table 659. Test Channel Watchdog Enable Registers to channel association

Register	Register bits 0:31
TCWENR	WEN_CH[127:96]

Note: If channel CH[x] is not existing then the corresponding bit field WEN_CH[x] is reserved and returns 0.

If all channels CH[x] are not existing, the full register is not existing and any attempt to access this space generates a transfer error.

42.5.1.28 Test Channel Analog Watchdog Out of Range Register (TCAWORR)

The analog watchdog registers to channel association is described in *Table 661*.

Offset: 0x0428 Access: User Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	AWOR_CH[x]															
W	w1c															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	AWOR_CH[x]															
W	w1c															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 534. Test Channel Analog Watchdog Out of Range Register (TCAWORR)

Table 660. TCAWORR field descriptions

Field	Description
0:31 AWOR_CH[x]	Analog watchdog out of range status for channel x, provided corresponding WEN_CHx bit is set. 0 CH[x] converted data is not out of range determined by its thresholds. 1 CH[x] converted data is out of range determined by its thresholds.

Table 661. Analog Watchdog Register to channel association

Register	Register bits 0:31
TCAWORR	AWOR_CH[127:96]

Note: If channel CH[x] is not existing then the corresponding bit field AWOR_CH[x] is reserved and returns 0.

If all channels CH[x] are not existing, the full register is not existing and any attempt to access this space generates a transfer error.

42.5.1.29 Test Channel Connection with Analog Pin Register *n* (TCCAPR*n*)

Test channels connected through a 20 kΩ source impedance can be shorted with any internal analog channel using the configuration programed in these registers.

Test channel short registers to channel association is described in *Table 663*.

For analog test channel assignment, refer to the table SAR_ADC analog test channel assignment of *Chapter 40: Analog-to-Digital Converters (ADC) Configuration*.

Offset: 0x430 + *n**0x4 (*n* = 0 to 7) Access: User Read/Write

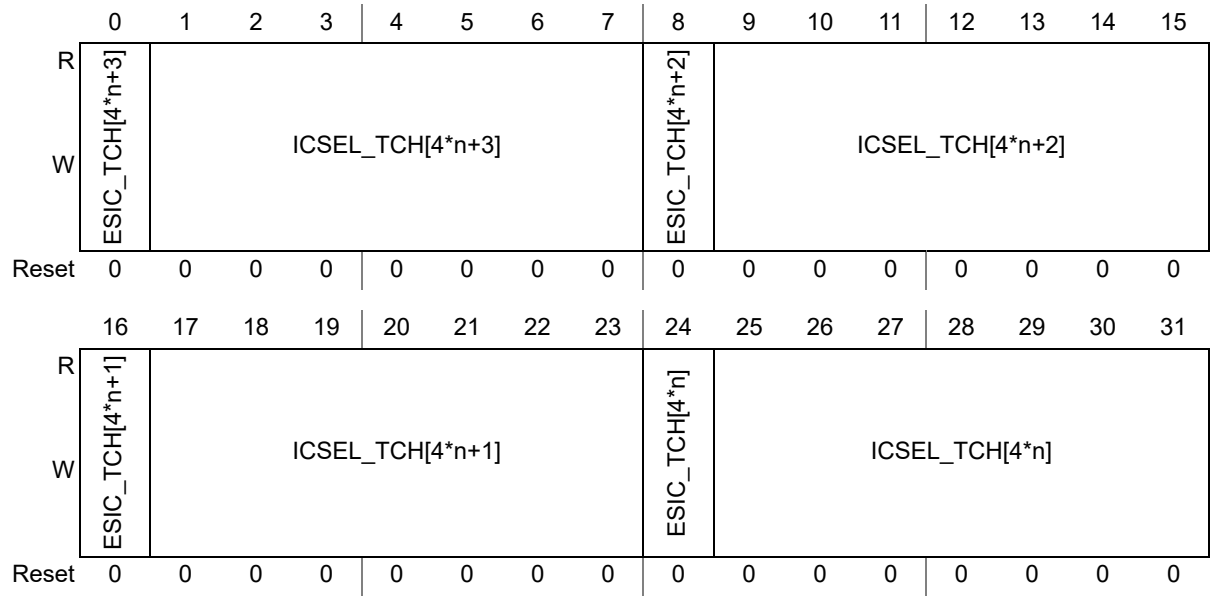


Figure 535. Test Channel Connection with Analog Pin Register *n* (TCCAPR*n*)

Table 662. TCCAPR*n* field descriptions

Field	Description
0 ESIC_TCH[4*n+3]	Enable short with internal channel for test channel x=4*n+3 0 Test channel x cannot be shorted with internal channel. 1 Test channel x can be shorted with internal channel.
1:7 ICSEL_TCH[4*n+3]	Internal channel selection for short with test channel x=4*n+3
8 ESIC_TCH[4*n+2]	Enable short with internal channel for test channel x=4*n+2 0 Test channel x cannot be shorted with internal channel. 1 Test channel x can be shorted with internal channel.
9:15 ICSEL_TCH[4*n+2]	Internal channel selection for short with test channel x=4*n+2
16 ESIC_TCH[4*n+1]	Enable short with internal channel for test channel x=4*n+1 0 Test channel x cannot be shorted with internal channel. 1 Test channel x can be shorted with internal channel.
17:23 ICSEL_TCH[4*n+1]	Internal channel selection for short with test channel x=4*n+1
24 ESIC_TCH[4*n]	Enable short with internal channel for test channel x=4*n 0 Test channel x cannot be shorted with internal channel. 1 Test channel x can be shorted with internal channel.
25:31 ICSEL_TCH[4*n]	Internal channel selection for short with test channel x=4*n

Table 663. Test Channel Short Registers to channel association

Register	Register bits 0:31 (arrangement of bit fields left to right)
TCCAPR0	ESIC_TCH3, ICSEL_TCH3, ESIC_TCH2, ICSEL_TCH2, ESIC_TCH1, ICSEL_TCH1, ESIC_TCH0, ICSEL_TCH0
TCCAPR1	ESIC_TCH7, ICSEL_TCH7, ESIC_TCH6, ICSEL_TCH6, ESIC_TCH5, ICSEL_TCH5, ESIC_TCH4, ICSEL_TCH4
TCCAPR2	ESIC_TCH11, ICSEL_TCH11, ESIC_TCH10, ICSEL_TCH10, ESIC_TCH9, ICSEL_TCH9, ESIC_TCH8, ICSEL_TCH8
TCCAPR3	ESIC_TCH15, ICSEL_TCH15, ESIC_TCH14, ICSEL_TCH14, ESIC_TCH13, ICSEL_TCH13, ESIC_TCH12, ICSEL_TCH12
TCCAPR4	ESIC_TCH19, ICSEL_TCH19, ESIC_TCH18, ICSEL_TCH18, ESIC_TCH17, ICSEL_TCH17, ESIC_TCH16, ICSEL_TCH16
TCCAPR5	ESIC_TCH23, ICSEL_TCH23, ESIC_TCH22, ICSEL_TCH22, ESIC_TCH21, ICSEL_TCH21, ESIC_TCH20, ICSEL_TCH20
TCCAPR6	ESIC_TCH27, ICSEL_TCH27, ESIC_TCH26, ICSEL_TCH26, ESIC_TCH25, ICSEL_TCH25, ESIC_TCH24, ICSEL_TCH24
TCCAPR7	ESIC_TCH31, ICSEL_TCH31, ESIC_TCH30, ICSEL_TCH30, ESIC_TCH29, ICSEL_TCH29, ESIC_TCH28, ICSEL_TCH28

42.5.1.30 Test Channel Data Register *n* (TCDR*n*)

The conversion results for each channel are loaded into data registers. Each data register contains:

- The conversion result
- Status information related to ADC mode,
- Data valid
- Timing parameter selection for each channel.

These registers are accessed 32-bit R/W.

Offset: 0x2D0 + *n**0x4 (*n* = 96 to 127) Access: User Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	PCE	0	CTSEL		0	0	0	FCERR	VALID	OVERW	RESULT	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	CDATA															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 536. Test Channel Data Register *n* (TCDR*n*)

Table 664. TCDR*n* field descriptions

Field	Description
4 PCE	Precharge Enable This bit enables the precharging phase during channel conversion. 0 Precharge phase is disabled 1 Precharge phase is enabled
6:7 CTSEL	This bit field selects the conversion timing register for each channel to select different precharge and sampling phase durations. 00 CTR0 is selected 01 CTR1 is selected 10 CTR2 is selected 11 CTR3 is selected
11 FCERR	Fast Comparator Error Flag This bit is used to notify error condition on current fast comparator operation inside ADC analog block. It is automatically cleared when data is read. 0 Cleared by reading the Converted by software. 1 Fast Comparator Error Condition has occurred in current conversion.
12 VALID	Data valid flag This bit is used to notify when the data is valid (a new value has been written). It is automatically cleared when data is read. 0 Converted data has been read by software 1 Converted data is valid and has not been read yet
13 OVERW	Data overwritten flag This bit signals that the previous converted data has been overwritten by a new conversion. This functionality depends on the value of MCR[OWREN]: – When OWREN = 0, OVERW is frozen to 0 and CDATA field is protected against being overwritten until being read. – When OWREN = 1, OVERW flags the CDATA field overwrite status. 0 Converted data has not been overwritten 1 Previous converted data has been overwritten before having been read
14:15 RESULT	Conversion result mode status This bit reflects the mode of conversion for the corresponding channel. 00 Data is a result of Normal conversion mode 01 Data is a result of Injected conversion mode 10 Data is a result of CTU conversion mode 11 Reserved
16:31 CDATA	Channel converted data **Note:** If MCR[WLSIDE] is set to ‘0’ than the converted data is read right aligned with lower 12/10 bits representing the actual converted data (width of actual converted data depends on selected resolution of current conversion, selectable by CTRx[CRES] bit) and remaining upper 4/6 bits are read as zeros. Whereas if the MCR[WLSIDE] is set as ‘1’ than the converted data is read left aligned with upper 12/10 bits representing the actual converted data (width of actual converted data depends on selected resolution of current conversion, selectable by CTRx[CRES] bit) and remaining lower 4/6 bits are read as zeros.

42.5.1.31 External Channel Decode Signals Delay Register (ECDSDR)

Offset: 0x500 Access: User Read/Write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	DSD											
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 537. External Channel Decode Signals Delay Register (ECDSDR)

Table 665. ECDSDR field descriptions

Field	Description
20:31 DSD	Decode signals delay This bit field defines the delay between the external decode signals and the start of the sampling phase. It is used to take into account of the settling time required for the external mux. The decode signal delay is calculated as (DSD X 1/Frequency of SARADC clock).

42.5.1.32 External Channel Interrupt Pending Register *n* (ECIPR*n*)

The interrupt channel register to channel association is described in *Table 667*.

Offset: 0x510 + n*0x4 (n = 0 to 3) Access: User Read Only

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	EOC_CH[x]															
W	w1c															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	EOC_CH[x]															
W	w1c															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 538. External Channel Interrupt Pending Register *n* (ECIPR*n*)

Table 666. ECIPR*n* register field descriptions

Field	Description
0:31 EOC_CH[x]	End of conversion interrupt pending bit for channel x 0 End of conversion for CH[x] has not occurred. 1 End of conversion for CH[x] has occurred.

Table 667. External Channel Interrupt Pending Registers to channel association

Register	Register bits 0:31
ECIPR0	EOC_CH[159:128]
ECIPR1	EOC_CH[191:160]
ECIPR2	EOC_CH[223:192]
ECIPR3	EOC_CH[255:224]

Note: If channel CH[x] is not existing then the corresponding bit field EOC_CH[x] is reserved and returns 0.

If all channels CH[x] are not existing, the full register is not existing and any attempt to access this space generates a transfer error.

42.5.1.33 External Channel Interrupt Mask Register *n* (ECIMR*n*)

The interrupt mask register to channel association is described in *Table 669*.

Offset: 0x520 + n*0x4 (n = 0 to 3) Access: User Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	IM_CH[x]															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	IM_CH[x]															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 539. External Channel Interrupt Mask Register *n* (ECIMR*n*)

Table 668. ECIMR*n* field descriptions

Field	Description
0:31 IM_CH[x]	Interrupt mask bit for channel x 0 Interrupt for CH[x] is disabled. 1 Interrupt for CH[x] is enabled.

Table 669. External Channel Interrupt Mask Registers to channel association

Register	Register bits 0:31
ECIMR0	IM_CH[159:128]
ECIMR1	IM_CH[191:160]
ECIMR2	IM_CH[223:192]
ECIMR3	IM_CH[255:224]

Note: If channel CH[x] is not existing then the corresponding bit field IM_CH[x] is reserved and returns 0.

If all channels CH[x] are not existing, the full register is not existing and any attempt to access this space generates a transfer error.

42.5.1.34 External Channel DMA Select Register *n* (ECDSR*n*)

The DMA select registers to channel association is described in *Table 671*.

Offset: 0x530 + n*0x4 (n = 0 to 3) Access: User Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	DS_CH[x]															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	DS_CH[x]															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 540. External Channel DMA Select Register *n* (ECDSR*n*)

Table 670. ECDSR*n* field descriptions

Field	Description
0:31 DS_CH[x]	DMA select for channel x 0 CH[x] is disabled to transfer data in DMA mode. 1 CH[x] is enabled to transfer data in DMA mode.

Table 671. DMA Select Registers to channel association

Register	Register bits 0:31
ECDSR0	DS_CH[159:128]
ECDSR1	DS_CH[191:160]
ECDSR2	DS_CH[223:192]
ECDSR3	DS_CH[255:224]

Note: If channel CH[x] is not existing then the corresponding bit field DS_CH[x] is reserved and returns 0.

If all channels CH[x] are not existing, the full register is not existing and any attempt to access this space generates a transfer error.

42.5.1.35 External Channel Normal Conversion Mask Register *n* (ECNCMR*n*)

The normal conversion mask registers to channel association is described in *Table 673*.

Offset: 0x540 + n*0x4 (n = 0 to 3) Access: User Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	NCE_CH[x]															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	NCE_CH[x]															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 541. External Channel Normal Conversion Mask Register *n* (ECNCMR*n*)

Table 672. ECNCMR*n* field descriptions

Field	Description
0:31 NCE_CH[x]	Normal conversion enable for channel x 0 Normal conversion is disabled for CH[x]. 1 Normal conversion is enabled for CH[x].

Table 673. External Channel Normal Conversion Mask Registers to channel association

Register	Register bits 0:31
ECNCMR0	NCE_CH[159:128]
ECNCMR1	NCE_CH[191:160]
ECNCMR2	NCE_CH[223:192]
ECNCMR3	NCE_CH[255:224]

Note: If channel CH[x] is not existing then the corresponding bit field NCE_CH[x] is reserved and returns 0.

If all channels CH[x] are not existing, the full register is not existing and any attempt to access this space generates a transfer error.

42.5.1.36 External Channel Injected Conversion Mask Register *n* (ECJCMR*n*)

The injected conversion mask registers to channel association are described in *Table 675*.

Offset: 0x550 + n*0x4 (n = 0 to 3) Access: User Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	JCE_CH[x]															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	JCE_CH[x]															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 542. External Channel Injected Conversion Mask Register *n* (ECJCMR*n*)

Table 674. ECJCMR*n* field descriptions

Field	Description
0:31 JCE_CH[x]	Injected conversion enable for channel x 0 Injected conversion is disabled for CH[x]. 1 Injected conversion is enabled for CH[x].

Table 675. External Channel Injected Conversion Mask Registers to channel association

Register	Register bits 0:31
ECJCMR0	JCE_CH[159:128]
ECJCMR1	JCE_CH[191:160]
ECJCMR2	JCE_CH[223:192]
ECJCMR3	JCE_CH[255:224]

Note: If channel CH[x] is not existing then the corresponding bit field JCE_CH[x] is reserved and returns 0.

If all channels CH[x] are not existing, the full register is not existing and any attempt to access this space generates a transfer error.

42.5.1.37 External Channel Watchdog Select Register *n* (ECWSELR*n*)

These registers contain WSEL_CHx[0:3] bit fields to select the threshold register which provides the values to be used for upper and lower bounds for channel x.

The Watchdog select registers to channel association is described in *Table 677*.

Offset: 0x0560 + n*0x4 (n = 0 to 15) Access: User Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R/W	WSEL_CH[(8*n)+135]				WSEL_CH[(8*n)+134]				WSEL_CH[(8*n)+133]				WSEL_CH[(8*n)+132]			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R/W	WSEL_CH[(8*n)+131]				WSEL_CH[(8*n)+130]				WSEL_CH[(8*n)+129]				WSEL_CH[(8*n)+128]			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 543. External Channel Watchdog Select Register *n* (ECWSELR*n*)

Table 676. ECWSELR*n* field descriptions

Field	Description
0:3 WSEL_CH[(8*n)+135]	Watchdog select for channel x 0000 THRHLR0 register is selected. 0001 THRHLR1 register is selected. 0010 THRHLR2 register is selected. ... 1110 THRHLR14 register is selected. 1111 THRHLR15 register is selected.
4:7 WSEL_CH[(8*n)+134]	Watchdog select for channel x 0000 THRHLR0 register is selected. 0001 THRHLR1 register is selected. 0010 THRHLR2 register is selected. ... 1110 THRHLR14 register is selected. 1111 THRHLR15 register is selected.
8:11 WSEL_CH[(8*n)+133]	Watchdog select for channel x 0000 THRHLR0 register is selected. 0001 THRHLR1 register is selected. 0010 THRHLR2 register is selected. ... 1110 THRHLR14 register is selected. 1111 THRHLR15 register is selected.
12:15 WSEL_CH[(8*n)+132]	Watchdog select for channel x 0000 THRHLR0 register is selected. 0001 THRHLR1 register is selected. 0010 THRHLR2 register is selected. ... 1110 THRHLR14 register is selected. 1111 THRHLR15 register is selected.
16:19 WSEL_CH[(8*n)+131]	Watchdog select for channel x 0000 THRHLR0 register is selected. 0001 THRHLR1 register is selected. 0010 THRHLR2 register is selected. ... 1110 THRHLR14 register is selected. 1111 THRHLR15 register is selected.

Table 676. ECWSELR*n* field descriptions (continued)

Field	Description
20:23 WSEL_CH[(8*n)+130]	Watchdog select for channel x 0000 THRHLR0 register is selected. 0001 THRHLR1 register is selected. 0010 THRHLR2 register is selected. ... 1110 THRHLR14 register is selected. 1111 THRHLR15 register is selected.
24:27 WSEL_CH[(8*n)+129]	Watchdog select for channel x 0000 THRHLR0 register is selected. 0001 THRHLR1 register is selected. 0010 THRHLR2 register is selected. ... 1110 THRHLR14 register is selected. 1111 THRHLR15 register is selected.
28:31 WSEL_CH[(8*n)+128]	Watchdog select for channel x 0000 THRHLR0 register is selected. 0001 THRHLR1 register is selected. 0010 THRHLR2 register is selected. ... 1110 THRHLR14 register is selected. 1111 THRHLR15 register is selected.

Table 677. External Channel Watchdog Enable Registers to channel association

Register	Register bits 0:31
ECWSELRn (n = 0 to 15)	WSEL_CH[(8*n)+135] ... WSEL_CH[(8*n)+128]

The number of bits implemented for each WSEL_CHx depends on Watchdog number:

- If Watchdog number > 8, the number of bits implemented is 4 (WSEL_CHx[3:0])
- If 4 < Watchdog number <= 8, the number of bits implemented is 3 (WSEL_CHx[3] is reserved).
- If 1 < Watchdog number <= 4, the number of bits implemented is 2 (WSEL_CHx[3:2] is reserved).
- If Watchdog number = 1, only 1 bit is present.

Each such nibble (WSEL_CHx) has positional correspondence to the External Channel number (each channel has a 1:1 nibble correspondence), so each word ECWSELRx corresponds to eight external channels, thus requiring a total of sixteen 32-bit registers to cover all 128 external channels.

If an External channel is not defined, the corresponding WSEL_CHx nibble is not implemented. For the implemented nibbles, width of WSEL_CHx field within the nibble is dependent on the Watchdog number. For example, if Watchdog number is 4 than only two LSBs of each WSEL_CHx are implemented and remaining two will remain unimplemented. If both nibbles in a byte-access, all four nibbles in a halfword-access, and all eight nibbles in a word-access are unimplemented, a transfer error is generated.

42.5.1.38 External Channel Watchdog Enable Register *n* (ECWENR*n*)

The Watchdog enable registers to channel association is described in *Table 679*.

Offset: 0x5A0 + n*0x4 (n = 0 to 3) Access: User Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	WEN_CH[x]															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	WEN_CH[x]															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 544. External Channel Watchdog Enable Register *n* (ECWENR*n*)

Table 678. ECWENR*n* field descriptions

Field	Description
0:31 WEN_CH[x]	Watchdog enable for channel x 0 Watchdog feature is disabled for CH[x]. 1 Watchdog feature is enabled for CH[x].

Table 679. External Channel Watchdog Enable Registers to channel association

Register	Register bits 0:31
ECWENR0	WEN_CH[159:128]
ECWENR1	WEN_CH[191:160]
ECWENR2	WEN_CH[223:192]
ECWENR3	WEN_CH[255:224]

Note: If channel CH[x] is not existing then the corresponding bit field WEN_CH[x] is reserved and returns 0.

If all channels CH[x] are not existing, the full register is not existing and any attempt to access this space generates a transfer error.

42.5.1.39 External Channel Analog Watchdog Out of Range Register *n* (ECAWORR*n*)

The analog watchdog registers to channel association is described in *Table 681*.

Offset: 0x05B0 + n*0x4 (n = 0 to 3) Access: User Read Only

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	AWOR_CH[x]															
W	w1c															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	AWOR_CH[x]															
W	w1c															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 545. External Channel Analog Watchdog Out of Range Register *n* (ECAWORR*n*)

Table 680. ECAWORR*n* field descriptions

Field	Description
0:31 AWOR_CH[x]	Analog watchdog out of range status for channel x, provided corresponding WEN_CHx bit is set. 0 CH[x] converted data is not out of range determined by its thresholds. 1 CH[x] converted data is out of range determined by its thresholds.

Table 681. External Channel Analog Watchdog Registers to channel association

Register	Register bits 0:31
ECAWORR0	AWOR_CH[159:128]
ECAWORR1	AWOR_CH[191:160]
ECAWORR2	AWOR_CH[223:192]
ECAWORR3	AWOR_CH[255:224]

Note: If channel CH[x] is not existing then the corresponding bit field AWOR_CH[x] is reserved and returns 0.

If all channels CH[x] are not existing, the full register is not existing and any attempt to access this space generates a transfer error.

42.5.1.40 External Channel Mapping to Internal Channel Registers *n* (ECMICR*n*)

Each set of eight external channels can be mapped with any internal analog channel using the configuration programed in these registers.

Offset: 0x5C0 + n*0x4 (n = 0 to 3) Access: User Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	ICSEL_ECH[159+32*n]_[152+32*n]							0	ICSEL_ECH[151+32*n]_[144+32*n]						
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	ICSEL_ECH[143+32*n]_[136+32*n]							0	ICSEL_ECH[135+32*n]_[128+32*n]						
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 546. External Channel Mapping to Internal Channel Registers *n* (ECMICR*n*)

Table 682. ECMICR*n* field descriptions

Field	Description
1:7 ICSEL_ECH[159+32*n]_[152+32*n]	Internal channel selection for external channels (159+32*n) to (152+32*n)
9:15 ICSEL_ECH[151+32*n]_[144+32*n]	Internal channel selection for external channels (151+32*n) to (144+32*n)
17:23 ICSEL_ECH[143+32*n]_[136+32*n]	Internal channel selection for external channels (143+32*n) to (136+32*n)
25:31 ICSEL_ECH[135+32*n]_[128+32*n]	Internal channel selection for external channels (135+32*n) to (128+32*n)

Table 683. External Channel Mapping to Internal Channel Registers to channel association

Register	Register bits 0:31 (arrangement of bit fields left to right)
ECMICR0	ICSEL_ECH159_152, ICSEL_ECH151_144, ICSEL_ECH143_136, ICSEL_ECH135_128
ECMICR1	ICSEL_ECH191_184, ICSEL_ECH183_176, ICSEL_ECH175_168, ICSEL_ECH167_160
ECMICR2	ICSEL_ECH223_216, ICSEL_ECH215_208, ICSEL_ECH207_200, ICSEL_ECH199_192
ECMICR3	ICSEL_ECH255_248, ICSEL_ECH247_240, ICSEL_ECH239_232, ICSEL_ECH231_224

42.5.1.41 External Channel Data Register *n* (ECDR*n*)

The conversion results for each channel are loaded into data registers. Each data register will contain the conversion result, status information related to ADC mode, data valid and some control information to select the required reference voltage, timing parameter selection for each channel.

This register should be accessed 32-bit R/W.

Offset: 0x3D0 + n*0x4 (n = 128 to 255) Access: User Read/Write

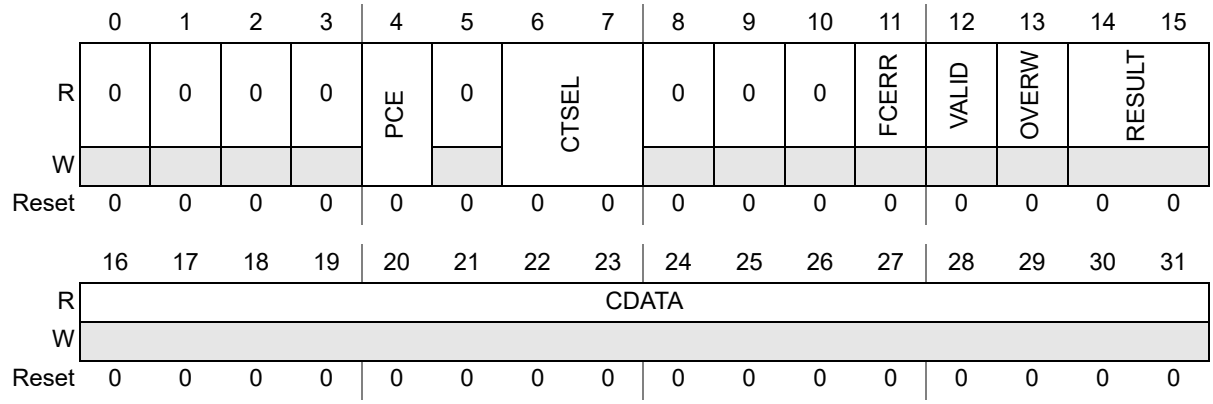


Figure 547. External Channel Data Register *n* (ECDR*n*)

Table 684. ECDR*n* field descriptions

Field	Description
4 PCE	Precharge Enable This bit enables the precharging phase during channel conversion. 0 Precharge phase is disabled. 1 Precharge phase is enabled.
6:7 CTSEL	This bit field selects the conversion timing register for each channel to select different precharge and sampling phase durations. 00 CTR0 is selected. 01 CTR1 is selected. 10 CTR2 is selected. 11 CTR3 is selected.
11 FCERR	Fast Comparator Error Flag This bit is used to notify error condition on current fast comparator operation inside ADC analog block. It is automatically cleared when data is read. 0 Cleared by reading the Converted by software. 1 Fast Comparator Error Condition has occurred in current conversion.
12 VALID	Data valid flag This bit is used to notify when the data is valid (a new value has been written). It is automatically cleared when data is read. 0 Converted data has been read by software. 1 Converted data is valid and has not been read yet.
13 OVERW	Data overwritten flag This bit signals that the previous converted data has been overwritten by a new conversion. This functionality depends on the value of MCR[OWREN]: – When OWREN = 0, OVERW is frozen to 0 and CDATA field is protected against being overwritten until being read. – When OWREN = 1, OVERW flags the CDATA field overwrite status. 0 Converted data has not been overwritten. 1 Previous converted data has been overwritten before having been read.

Table 684. ECDR*n* field descriptions (continued)

Field	Description
14:15 RESULT	Conversion result mode status This bit reflects the mode of conversion for the corresponding channel. 00 Data is a result of Normal conversion mode. 01 Data is a result of Injected conversion mode. 10 Data is a result of CTU conversion mode. 11 Reserved
16:31 CDATA	Channel converted data **Note:** If MCR[WLSIDE] is set to '0' than the converted data is read right aligned with lower 12/10 bits representing the actual converted data (width of actual converted data depends on selected resolution of current conversion, selectable by CTRx[CRES] bit) and remaining upper 4/6 bits are read as zeros. Whereas if the MCR[WLSIDE] is set as '1' than the converted data is read left aligned with upper 12/10 bits representing the actual converted data (width of actual converted data depends on selected resolution of current conversion, selectable by CTRx[CRES] bit) and remaining lower 4/6 bits are read as zeros.

42.6 Start of conversion pulse delay

The following table lists the delays between various trigger points (initiation request for conversion as registered at system clock) and the actual issue of start pulse (adc_start[18]) to the ADC hard-macrocell. Here, each possible trigger source is described with the number of clock cycles required for different operations until the trigger of adc_start. This table does not take into account any software delay involved in programming or polling of any bit in order to measure conversion timing.

Table 685. Start of conversion pulse delay table

Trigger source	Register control (PBRIDGE_CLK)	Inter domain synchronization (PBRIDGE_CLK)	ADC state machine (SARADC_CLK)	Inter domain synchronization (SARADC_CLK)	Comments
Normal conversion triggered by software (setting of NSTART of MCR)	1	—	2	1-2[3]	Cumulative delay from wen (byte_en) assertion to actual issue of adc_start pulse = 3-4 SARADC_CLK cycles + 1 PBRIDGE_CLK cycle.
Normal conversion triggered by external trigger (hardware trigger through)	—	—	2	1–2[3]	Cumulative delay from the triggering edge (rising or falling) to the actual issue of adc_start pulse = 3–4 SARADC_CLK cycles.

18. adc_start is the start of conversion indication to ADC hard-macrocell to start the conversion.

Table 685. Start of conversion pulse delay table (continued)

Trigger source	Register control (PBRIDGE_CLK)	Inter domain synchronization (PBRIDGE_CLK)	ADC state machine (SARADC_CLK)	Inter domain synchronization (SARADC_CLK)	Comments
Normal conversion within a chain	—	—	0 (from EOC[1] of previous conversion)	—	Delay from the EOC[1] of last conversion, The start of next conversion is issued exactly on the same cycle when EOC[1] is expected for previous conversion.
Injected conversion triggered by software (setting of JSTART bit of MCR)	1	—	3	1–2[3]	Cumulative delay from wen (byte_en) assertion. Here delay for first start pulse of the injected chain execution when FSM was originally in IDLE state is considered = 4–5 SARADC_CLK cycles + 1 PBRIDGE_CLK cycle.
Injected conversion triggered by external Trigger (hardware trigger)	—	—	3	1–2[3]	Cumulative delay from triggering edge (rising or falling) to actual issue of adc_start = 4–5 SARADC_CLK cycles.
Injected conversion within a chain (delay between consecutive conversion of injected chain)	—	—	0 (from EOC[1] of previous conversion)	—	Delay from EOC[1] of last injected channel and adc_start pulse of next injected channel. The start of next conversion is issued exactly on the same cycle when EOC[1] is expected for previous conversion
Injected conversion over normal conversion (software injected conversion chain)	1	1–2[3]	3–5[2]	1–2[3]	Cumulative delay from wen (byte_en) assertion to actual trigger of first injected channel, when the FSM was in conversion cycle of previous normal conversion = 4–7 SARADC_CLK cycles + 2-3 PBRIDGE_CLK cycles.

Table 685. Start of conversion pulse delay table (continued)

Trigger source	Register control (PBRIDGE_CLK)	Inter domain synchronization (PBRIDGE_CLK)	ADC state machine (SARADC_CLK)	Inter domain synchronization (SARADC_CLK)	Comments
Injection of conversion over normal conversion in JTRGSEQ mode	—	—	3	1–2(3)	Cumulative delay from triggering edge (rising or falling) to actual issue of adc_start = 4–5 SARADC_CLK cycles.
Abort of a Normal/Injected Channel conversion during Normal Chain/Injected Chain execution (setting of MCR[ABORT] bit)	1	1–2(3)	3–5(2)	1–2(3)	Delay from writing of ABORT bit (wen/byte_en assertion) to actual issue of adc_start pulse for the next channel conversion = 4–7 SARADC_CLK cycles + 2–3 PBRIDGE_CLK cycles.

1. EOC is the end of conversion from ADC hard macrocell
2. The above delay values are calculated over a range of system and ADC clock frequencies. With PBRIDGE_CLK(max) = 50 MHz to PBRIDGE_CLK(min) = 40 MHz and SARADC_CLK(max) = 16 MHz to SARADC_CLK(min) = 14 MHz. L.H.S value corresponds to max frequencies and RHS corresponds to min frequencies.
3. The above variation is owing to inter domain synchronization.

42.7 Initialization information

To initialize the SARADC registers for data conversion, the following programming sequence is required:

1. Program the PDEDR register with appropriate power down exit delay value and required CTU mode (MCR[CTU_MODE]) in case CTU mode has to be entered later on by setting CTUEN bit.
2. Exit the power down mode by clearing the MCR[PWDN] bit. After the power down mode is exited the SARADC enters the IDLE state, this can be checked by polling for MSR[ADCSTATUS] bits. Note that steps 3-7 can also be executed before step 2.
3. Program the required conversion mode and control options for example one-shot mode or scan mode, hardware trigger enable and selection for normal or injected conversions, hardware trigger edge select, overwrite enable, and so on, in MCR register.
4. Program the conversion mask registers ({I,T,E}C{N,J}CMR) for channels to be converted. If required, enable and select watchdog monitors ({I,T,E}CWENR, {I,T,E}CWSELR) as applicable for each channel, program the watchdog threshold values (WTHRHLR) for each watchdog monitor.
5. If required, Enable and select the External to Internal channel mapping by programming the ECMICR registers, Test to internal channel mapping by programming the TCCAPR registers.
6. Program the required Interrupt/DMA mask registers (DMAE, {I,T,E}CDSR, IMR, {I,T,E}C{N,J}MR, WTIMR) for each of the selected channels.

7. Program the Conversion timing registers(CTR0-3) for the required conversion durations. Enable the precharge phase, CTR selection per channel basis by programming the {I,T,E}CDR registers.
8. To initiate CTU mode conversion enter the selected CTU mode by setting the MCR[CTUEN] bit and program the CTU module accordingly to start the conversions. To initiate Normal/Injected mode conversions by software set the MCR[NSTART/JSTART] bit. To initiate hardware triggered Normal/Injected conversions program the external trigger module to provide the required triggers.
9. Read the conversion data and status in {I,T,E}CDR registers.

43 Temperature Sensor

43.1 Introduction

The device includes an onboard temperature sensor that monitors device temperature and delivers two analog outputs signals and three digital output signals.

The analog outputs consist of two voltage signals which vary linearly with the internal junction temperature: a voltage signal that is linearly increasing (PTAT: proportional to absolute temperature) and a voltage signal that is linearly decreasing (CTAT: complementary proportional to absolute temperature). The analog outputs are connected to an input channel of an ADC on the device. The internal junction temperature must be calculated by software based on the converted temperature values.

The three digital outputs, connected to the PMC module, are used to signal under- and over-temperature operating conditions. These signals notify the device to take action to appropriately adjust the device temperature in response to an out of specification low or high temperature operating condition. Calibration parameter values associated with the temperature threshold detection feature are determined and stored in internal flash memory during production testing at the factory.

43.1.1 Signal Descriptions

This module has no external signals.

43.2 Functional description

The temperature sensor generates two analog outputs voltage, PTAT and CTAT which are proportional to the absolute current junction temperature of the device.

An on-chip ADC module is used to convert the analog outputs voltage, PTAT and CTAT, into a digital representation. These values, along with parameter values stored in onboard flash memory, are used by software to calculate the device junction temperature.

43.2.1 Linear temperature sensor (analog output generation)

The temperature sensor outputs two voltage proportional and complementary proportional to the internal junction temperature of the chip. These analog voltage signals are converted into digital values by an on-chip ADC. The temperature value is obtained from a linear voltage-temperature relation with coefficients adjusted by calibration parameters extracted during factory test and programed into flash memory.

43.3 Temperature formula

43.3.1 Equations for converting TSENS voltages to junction temperature

Table 686. Calibration constants (1) (2)

Constant	Description
P1	Code from the ADC converting PTAT output voltage at 150^0C
P2	Code from the ADC converting PTAT output voltage at -40^0C
C1	Code from the ADC converting CTAT output voltage at 150^0C
C2	Code from the ADC converting CTAT output voltage at -40^0C

1. It is mandatory to keep the ADC reference constant while measuring P1, P2, C1, and C2
2. Flash locations for P2, C2, P1 and C1 are respectively 0x400000, 0x400002, 0x400004 and 0x400006.

In the equations below:

- Pn and Cn are the calibration constants described in *Table 686*
- T is the unknown device temperature in ^{0}C
- Px is the code from the ADC converting PTAT output voltage at the temperature T with any ADC reference voltage Vref
- Cx is the code from the ADC converting CTAT output voltage at the temperature T with the same ADC reference voltage Vref.
- T2 = -40^0C
- T1 = 150^0C

Equation 24 $$A = Px \cdot C2 - (Cx \cdot P2)$$

Equation 25 $$B = Cx \cdot P1 - (Px \cdot C1)$$

The temperature T is calculated as

Equation 26 $$T = T2 + \frac{A \cdot (T1 - T2)}{A + B}$$

Caution: It is mandatory to have the same value as the ADC reference, while measuring Px and Cx.

43.3.2 Equations for converting TSENS voltages into constant reference (Digital Bandgap Voltage)

By converting the two outputs of the Temperature Sensor through ADC, one can find out a constant reference code (REFCODE) which does not change with temperature.

- Pn and Cn are the calibration constants described in *Table 686*.
- Px is the code from the ADC converting PTAT output voltage at any temperature T with any ADC reference voltage Vref.
- Cx is the code from the ADC converting CTAT output voltage at the same temperature T with the same ADC reference voltage Vref.

Equation 27

$$REFCODE = Px + \left(\frac{(P2 - P1)}{(C1 - C2)}\right) \times Cx$$

Caution: It is mandatory to measure Px and Cx at the same ADC reference voltage and at the same temperature.

Note: Software calculation must be made with negligible precision loss associated with limited bit representation of intermediate calculated results.

44 Body Cross Triggering Unit (BCTU)

44.1 Introduction

The Body Cross Triggering Unit (BCTU) allows to synchronize an ADC conversion with a timer event from eMIOS (every mode which can generate a DMA request can trigger BCTU), PIT_RTI or GTM. To select which ADC channel must be converted on a particular timer event, the BCTU provides the ADC with a 7-bit channel number. This channel number can be configured for each timer channel event by the application.

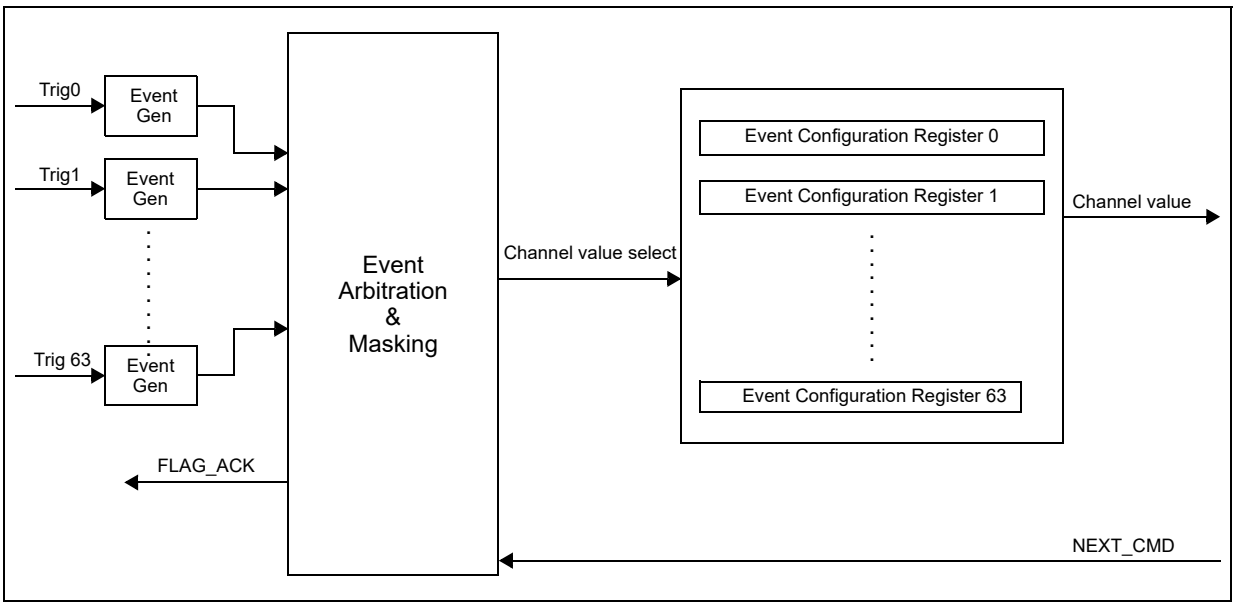
44.2 Main features

- Single cycle delayed trigger output. The trigger output is a combination of 64 (generic value) input flags/events connected to different timers in the system.
- One event configuration register dedicated to each timer event allows to define the corresponding ADC channel.
- Acknowledgment signal to eMIOS/PIT_RTI/GTM for clearing the flag
- Synchronization with ADC to avoid collision

44.3 Block diagram

The BCTU block diagram is shown in *Figure 548*.

Figure 548. Body Cross Triggering Unit block diagram



44.4 Memory map and register descriptions

The BCTU registers are listed in *Table 687*.

Table 687. BCTU memory map

Offset	Register	Location
0x000	Control Status Register (BCTU_CSR)	*Section 44.4.1*
0x004–0x02C	Reserved	
n = 0 to 63		
0x030 + n*0x4	Event Configuration Register *n* (BCTU_EVTCFGR*n*)	*Section 44.4.2*

44.4.1 Control Status Register (BCTU_CSR)

Offset: 0x000 Access: Read/write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	TRGIEN(1)	TRGI	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. BCTU is not able to trigger interrupt to CPU because interrupt signal is not routed to the interrupt controleur (INTC).

Figure 549. Control Status Register (BCTU_CSR)

Table 688. BCTU_CSR field descriptions

Field	Description
24 TRGIEN	Trigger Interrupt Request Enable 0 Trigger interrupt request disabled 1 Trigger interrupt request enabled. A request is generated if the TRGI flag is set.
25 TRGI	Trigger Interrupt Flag This flag is set by hardware when the trigger output request is generated after a valid input event is detected. It is cleared by software. 0 No trigger output request 1 Trigger output request is generated

44.4.2 Event Configuration Register *n* (BCTU_EVTCFGR*n*)

Offset: 0x030 + n*0x4 (n=0 to 63) Access: Read/write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	TM	CLR_FLAG(1)	0	0	0	0	0	0	0	CHANNEL_VALUE						
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. This bit implementation is generic based and implemented only for inputs mapped to PIT_RTI event flags.

Figure 550. Event Configuration Register *n* (BCTU_EVTCFGR*n*)

Table 689. BCTU_EVTCFGR*n* field descriptions

Field	Description
16 TM	Trigger Mask 0 Trigger masked 1 Trigger enabled
17 CLR_FLAG	To provide flag_ack through software 1 Flag_ack is forced to ‘1’ for the particular event 0 Flag_ack is dependent on flag servicing
25:31 CHANNEL_VALUE	0xxxxxx ADC internal channel number from 0 to 63 1xxxxxx ADC external channel number from 128 to 191 **Note:** When bit 25 = 1 an offset of 128 is added to the value defined by bits 26:31 : that allows to address external channels inside ADC module.

These registers contain the ADC channel number to be converted when the timer event occurs.

When CLR_FLAG is set, BCTU ignores any event from PIT_RTI if some triggered conversion is going on in ADC (conversion from BCTU to ADC). But if BCTU is not busy in sending any trigger to ADC, then it will send the trigger to ADC for conversion of configured channel in BCTU_EVTCFGR register. In that case, setting CLR_FLAG bit would not provide flag_ack through software.

The CLR_FLAG bit has to be used cautiously as setting this bit may result in a loss of events.

The event input can be masked by writing ‘0’ to bit TM of the BCTU_EVTCFGR register. Writing ‘1’ to bit TM enables the BCTU triggering and automatically disables the DMA connection for the corresponding eMIOS channel.

Note: The BCTU tracks issued conversion requests to the ADC. When the ADC is being triggered by the BCTU and there is a need to shut down the ADC, the ADC must be allowed to complete conversions before being shut down. This ensures that the BCTU is notified of completion; if the ADC is shut down while performing a BCTU-triggered conversion, the BCTU is not notified and will not be able to trigger further conversions until the device is reset.

44.5 Functional description

This peripheral is used to synchronize ADC conversions with timer events (from eMIOS or PIT_RTI). When a timer event occurs, the BCTU triggers an ADC conversion providing the ADC channel number to be converted. In case concurrent events occur the priority is managed according to the index of the timer event. The trigger output is a single cycle pulse used to trigger ADC conversion of the channel number provided by the BCTU.

Each trigger input from the BCTU is connected to the Event Trigger signal of an eMIOS channel. The assignment between eMIOS outputs and BCTU trigger inputs of the device is defined in the *Chapter 7: Device configuration*.

Each event has a dedicated configuration register (BCTU_EVTCFGR). These registers store a channel number which is used to communicate which channel needs to be converted.

In case several events are pending for ADC request, the priority is managed according to the timer event index. The lowest index has the highest priority. Once an event has been serviced (conversion requested to ADC) the eMIOS flag is cleared by the BCTU and next prior event is handled.

The acknowledgment signal can be forced to ‘1’ by setting the CLR_FLAG bit of the BCTU_EVTCFGR register. These bits are implemented for only those input flags to which PIT_RTI flags are connected. Providing these bits offers the option of clearing PIT_RTI or GTM flags by software.

45 System Timer Module (STM)

45.1 Introduction

This section provides an overview, list of features, and modes of operation for the STM.

Note: For chip-specific implementation details of this module's instances, see the chip configuration information.

45.1.1 Overview

The System Timer Module (STM) is a 32-bit timer designed to support commonly required system and application software timing functions. The STM includes a 32-bit up counter and four 32-bit compare channels with a separate interrupt source for each channel. The counter is driven by the FBRIDGE clock divided by an 8-bit prescale value (1 to 256).

45.1.2 Features

The STM has the following features:

- One 32-bit up counter with 8-bit prescaler
- Four 32-bit compare channels
- Independent interrupt source for each channel
- Counter can be stopped in debug mode

45.1.3 Modes of operation

The STM supports two device modes of operation: Normal and Debug. When the STM is enabled in Normal mode, its counter runs continuously. In Debug mode, operation of the counter is controlled by the FRZ bit in the STM Control Register (STM_CR). If STM_CR[FRZ] is set, the counter is stopped in Debug mode, otherwise it continues to run.

45.2 External signal description

The STM does not have any external interface signals.

45.3 Memory map and register definition

The STM programming model includes a group of 32-bit registers: a module control register, a counter value register, and three registers for each channel.

The STM registers can only be accessed using 32-bit (word) accesses. Attempted references using a different size or to a reserved address generates a bus error termination.

45.3.1 Memory map

The STM memory map is shown in *Table 690*.

Table 690. STM memory map

Address offset	Register	Location
0x0000	STM Control Register (STM_CR)	*Section 45.3.2.1*
0x0004	STM Counter Value (STM_CNT)	*Section 45.3.2.2*
0x0008–0x000B	Reserved	
0x000C–0x000F	Reserved	
STM channel registers per channel (n=0 to 3)		
0x0010 + n*0x10	STM Channel n Control Register (STM_CCRn)	*Section 45.3.2.3*
0x0014 + n*0x10	STM Channel n Interrupt Register (STM_CIRn)	*Section 45.3.2.4*
0x0018 + n*0x10	STM Channel n Compare Register (STM_CMPn)	*Section 45.3.2.5*
0x001C + n*0x10	Reserved	
0x004C–0x3FFF	Reserved	

45.3.2 Register descriptions

45.3.2.1 STM Control Register (STM_CR)

The STM Control Register (STM_CR) includes the prescale value, freeze control and timer enable bits.

Offset: 0x000 Access: Read/Write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	CPS								0	0	0	0	0	0	FRZ	TEN
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 551. STM Control Register (STM_CR)

Table 691. STM_CR field descriptions

Field	Description
16:23 CPS	Counter Prescaler. Selects the clock divide value for the prescaler (1 - 256). 0x00 Divide FBRIDGE clock by 1 0x01 Divide FBRIDGE clock by 2 ... 0xFF Divide FBRIDGE clock by 256
30 FRZ	Freeze. Allows the timer counter to be stopped when the device enters debug mode. **Note:** When the microcomputer enters debug mode, the STM is notified and uses the FRZ bit to determine counter mode. 0 STM counter continues to run in debug mode 1 STM counter is stopped in debug mode
31 TEN	Timer Counter Enabled. 0 Counter is disabled 1 Counter is enabled

45.3.2.2 STM Count Register (STM_CNT)

The STM Count Register (STM_CNT) holds the timer count value.

Offset: 0x004 Access: Read/Write

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

Figure 552. STM Count Register (STM_CNT)

Table 692. STM_CNT field descriptions

Field	Description
0:31 CNT	Timer count value used as the time base for all channels. When enabled, the counter increments at the rate of the FBRIDGE clock divided by the prescale value.

45.3.2.3 STM Channel Control Register (STM_CCR*n*)

The STM Channel Control Register (STM_CCR*n*) has the enable bit for channel n of the timer.

Offset: 0x10 + *n**0x10 (*n* = 0 to 3) Access: Read/Write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	CEN
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 553. STM Channel Control Register (STM_CCR*n*)

Table 693. STM_CCR*n* field descriptions

Field	Description
31 CEN	Channel Enable. 0 The channel is disabled 1 The channel is enabled

45.3.2.4 STM Channel Interrupt Register (STM_CIR*n*)

The STM Channel Interrupt Register (STM_CIR*n*) has the interrupt flag for channel n of the timer.

Offset: 0x14 + *n**0x10 (*n* = 0 to 3) Access: Read/Write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	CIF
W																w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 554. STM Channel Interrupt Register (STM_CIR*n*)

Table 694. STM_CIR*n* field descriptions

Field	Description
31 CIF	Channel Interrupt Flag 0 No interrupt request 1 Interrupt request due to a match on the channel

45.3.2.5 STM Channel Compare Register (STM_CMP*n*)

The STM channel compare register (STM_CMPn) holds the compare value for channel n.

Offset: 0x18 + *n**0x10 (*n* = 0 to 3) Access: Read/Write

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

Figure 555. STM Channel Compare Register (STM_CMP*n*)

Table 695. STM_CMP*n* field descriptions

Field	Description
0:31 CMP	Compare value for channel n. If the STM_CCR*n*[CEN] bit is set and the STM_CMP*n* register matches the STM_CNT register, a channel interrupt request is generated and the STM_CIR*n*[CIF] bit is set.

45.4 Functional description

The System Timer Module (STM) is a 32-bit timer designed to support commonly required system and application software timing functions. The STM includes a 32-bit up counter and four 32-bit compare channels with a separate interrupt source for each channel.

The STM has one 32-bit up counter (STM_CNT) that is used as the time base for all channels. When enabled, the counter increments at the FBRIDGE clock frequency divided by a prescale value. The STM_CR[CPS] field sets the divider to any value in the range from 1 to 256. The counter is enabled with the STM_CR[TEN] bit. When enabled in normal mode the counter continuously increments. When enabled in debug mode the counter operation is controlled by the STM_CR[FRZ] bit. When enabled in normal mode the counter continuously increments. When enabled in debug mode, the counter operation is controlled by the STM_CR[FRZ] bit. When the STM_CR[FRZ] bit is set, the counter is stopped in debug mode, otherwise it continues to run in debug mode. The counter rolls over at 0xFFFF_FFFF to 0x0000_0000 with no restrictions at this boundary.

The STM has four identical compare channels. Each channel includes a channel control register (STM_CCR*n*), a channel interrupt register (STM_CIR*n*) and a channel compare register (STM_CMP*n*). The channel is enabled by setting the STM_CCR*n*[CEN] bit. When enabled, the channel will set the STM_CIR*n*[CIF] bit and generate an interrupt request when the channel compare register matches the timer counter. The interrupt request is cleared by writing a 1 to the STM_CIR*n*[CIF] bit. A write of 0 to the STM_CIR*n*[CIF] bit has no effect.

Note: The STM counter does not advance when the FBRIDGE clock is stopped.

46 Software Watchdog Timer (SWT)

46.1 Introduction

This section provides an overview, list of features, and modes of operation for the SWT.

Note: For chip-specific implementation details of this module's instances, refer to the chip configuration information.

46.1.1 Overview

The Software Watchdog Timer (SWT) is a peripheral module that can prevent system lockup in situations such as software getting trapped in a loop or if a bus transaction fails to terminate. When enabled, the SWT requires periodic execution of a watchdog servicing operation. The servicing operation resets the timer to a specified time-out period. If this servicing action does not occur before the timer expires the SWT generates an interrupt or hardware reset. The SWT can be configured to generate a reset or interrupt on an initial time-out. A reset is always generated on a second consecutive time-out.

Note: The SWT runs over all device mode except the STANDBY mode where it is not powered.

46.1.2 Features

The SWT has the following features:

- 32-bit time-out register to set the time-out period
- Programmable selection of window mode or regular servicing
- Programmable selection of reset or interrupt on an initial time-out
- Programmable selection of the servicing mode
- Master access protection
- Hard and soft configuration lock bits
- Reset configuration inputs allow timer to be enabled out of reset

46.1.3 Modes of operation

The SWT supports three device modes of operation: normal, debug and stop. When the SWT is enabled in normal mode, its counter runs continuously. In debug mode, operation of the counter is controlled by the FRZ bit in the SWT_CR *Section 46.3.1.1: SWT Control Register (SWT_CR)*. If the FRZ bit is set, the counter is stopped in debug mode, otherwise it continues to run. In stop mode, operation of the counter is controlled by the STP bit of the SWT_CR register. Refer to *Section 46.3.1.1: SWT Control Register (SWT_CR)*. If the STP bit is set, the counter is stopped in stop mode, otherwise it continues to run.

46.2 External signal description

The SWT module does not have any external interface signals.

46.3 Memory map and register definition

The SWT programming model has seven 32-bit registers. The programming model can only be accessed using 32-bit (word) accesses. References using a different size are invalid. Other types of invalid accesses include: writes to read-only registers, incorrect values written to the service register when enabled, accesses to reserved addresses and accesses by masters without permission. If the RIA bit in the SWT_CR is set then the SWT generates a system reset on an invalid access, otherwise a bus error is generated. If either the HLK or SLK bits in the SWT_CR are set, then the SWT_CR, SWT_TO, SWT_WN, and SWT_SK registers are read-only.

The SWT memory map is shown in *Table 696*.

Table 696. SWT memory map

Offset	Register	Location
0x0000	SWT Control Register (SWT_CR)	*Section 46.3.1.1*
0x0004	SWT Interrupt Register (SWT_IR)	*Section 46.3.1.2*
0x0008	SWT Time-Out Register (SWT_TO)	*Section 46.3.1.3*
0x000C	SWT Window Register (SWT_WN)	*Section 46.3.1.4*
0x0010	SWT Service Register (SWT_SR)	*Section 46.3.1.5*
0x0014	SWT Counter Output Register (SWT_CO)	*Section 46.3.1.6*
0x0018	SWT Service Key Register (SWT_SK)	*Section 46.3.1.7*
0x001C–0x3FFF	Reserved	

46.3.1 Register descriptions

The following sections detail the individual registers within the SWT programming model.

Note: Refer to the memory map for the base address of SWT registers.

46.3.1.1 SWT Control Register (SWT_CR)

The SWT_CR contains fields for configuring and controlling the SWT. This register is read-only if either the SWT_CR[HLK] or SWT_CR[SLK] bits are set.

Offset: 0x0000 Access: Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	MAP0	MAP1	MAP2	MAP3	MAP4	MAP5	MAP6	MAP7	0	0	0	0	0	0	0	0
W																
Reset(1)	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	SMD		RIA	WND	ITR	HLK	SLK	1	STP	FRZ	WEN
W																
Reset(1)	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*

1. The reset value for the SWT_CR is device specific. Refer to the device configuration chapter.

Figure 556. SWT Control Register (SWT_CR)

Table 697. SWT_CR field descriptions

Field	Description
0:7 MAP[0:7]	Master Access Protection for Master n The platform bus master assignments are device specific. Not all MAPn fields are implemented. Refer to the device configuration chapter for master ports that are implemented on the SMPU. 0 Access for the master is not enabled 1 Access for the master is enabled
21:22 SMD	Service Mode 00 Fixed Service Sequence, the watchdog is serviced by writing the fixed sequence 0xA602, 0xB480 to the SWT_SR. 01 Keyed Service Sequence, the watchdog is serviced by writing two pseudo-random key values to the SWT_SR. 10 Fixed Address Execution, the watchdog is serviced by executing code at the address loaded into the designated IAC register which can not be updated while the watchdog is enabled. Support for this mode is device and instance specific. 11 Incremental Address Execution, the watchdog is serviced by executing code at the address loaded into the designated IAC register which can be updated. Support for this mode is device and instance specific.
23 RIA	Reset on Invalid Access 0 Invalid access to the SWT generates a bus error 1 Invalid access to the SWT causes a system reset if WEN=1
24 WND	Window Mode 0 Regular mode, service sequence can be done at any time 1 Windowed mode, the service sequence is only valid when the down counter is less than the value in the SWT_WN register.
25 ITR	Interrupt Then Reset 0 Generate a reset on a time-out 1 Generate an interrupt on an initial time-out, reset on a second consecutive time-out
26 HLK	Hard Lock This bit is only cleared at reset. 0 SWT_CR, SWT_TO, SWT_WN and SWT_SK are read/write registers if SLK=0 1 SWT_CR, SWT_TO, SWT_WN and SWT_SK are read-only registers

Table 697. SWT_CR field descriptions (continued)

Field	Description
27 SLK	Soft Lock This bit is cleared by writing the unlock sequence to the service register. 0 SWT_CR, SWT_TO SWT_WN and SWT_SK are read/write registers if HLK=0 1 SWT_CR, SWT_TO, SWT_WN and SWT_SK are read-only registers
29 STP	Stop Mode Control Allows the watchdog timer to be stopped when the device enters stop mode. 0 SWT counter continues to run in stop mode 1 SWT counter is stopped in stop mode
30 FRZ	Debug Mode Control Allows the watchdog timer to be stopped when the device enters debug mode. 0 SWT counter continues to run in debug mode 1 SWT counter is stopped in debug mode
31 WEN	Watchdog Enabled 0 SWT is disabled 1 SWT is enabled

46.3.1.2 SWT Interrupt Register (SWT_IR)

The SWT_IR contains the time-out interrupt flag.

Offset: 0x0004 Access: Read/Write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	TIF
W																w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 557. SWT Interrupt Register (SWT_IR)

Table 698. SWT_IR field descriptions

Field	Description
31 TIF	Time-out Interrupt Flag The flag and interrupt are cleared by writing a 1 to this bit. Writing a 0 has no effect. 0 No interrupt request. 1 Interrupt request due to an initial time-out.

46.3.1.3 SWT Time-Out Register (SWT_TO)

The SWT Time-Out (SWT_TO) register contains the 32-bit time-out period. The reset value for this register is device specific. This register is read-only if either the SWT_CR[HLK] or SWT_CR[SLK] bits are set.

Offset: 0x0008 Access: Read/Write

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

1. The reset value of the SWT_TO register is device specific. Refer to the device configuration chapter.

Figure 558. SWT Time-Out Register (SWT_TO)

Table 699. SWT_TO field descriptions

Field	Description
0:31 WTO	Watchdog time-out period in clock cycles An internal 32-bit down counter is loaded with this value when the service sequence is written or when the SWT is enabled. The initial value is 0x0003_FDE0. The minimum value is 0x0000_0100.

46.3.1.4 SWT Window Register (SWT_WN)

The SWT Window (SWT_WN) register contains the 32-bit window start value. This register is cleared on reset. This register is read-only if either the SWT_CR[HLK] or SWT_CR[SLK] bits are set.

Offset: 0x000C Access: Read/Write

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

Figure 559. SWT Window Register (SWT_WN)

Table 700. SWT_WN field descriptions

Field	Description
0:31 WST	Window start value When window mode is enabled, the service sequence can only be written when the internal down counter is less than this value.

46.3.1.5 SWT Service Register (SWT_SR)

The SWT Service (SWT_SR) register is the target for service operation writes used to reset the watchdog timer.

Offset: 0x0010 Access: Write

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

Figure 560. SWT Service Register (SWT_SR)

Table 701. SWT_SR field descriptions

Field	Description
16:31 WSC	Watchdog Service Code This field is used to service the watchdog and to clear the soft lock bit (SWT_CR[SLK]). If the SWT_CR[SMD] = 01, two pseudo-random key values are written to service the watchdog, refer to *Section 46.3.1.1: SWT Control Register (SWT_CR)* for details. Otherwise, the sequence 0xA602 followed by 0xB480 is written to the WSC field. To clear the soft lock bit (SWT_CR[SLK]), the value 0xC520 followed by 0xD928 is written to the WSC field. When read, the WSC field always returns zero.

46.3.1.6 SWT Counter Output Register (SWT_CO)

The SWT Counter Output (SWT_CO) register is a read-only register that shows the value of the internal down counter when the SWT is disabled.

Offset: 0x0014 Access: Read-Only

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

1. SWT_CO will reflect the internal value of the counter when WEN=0, and will be read as 0x0000_0000 when WEN=1

Figure 561. SWT Counter Output Register (SWT_CO)

Table 702. SWT_CO field descriptions

Field	Description
0:31 CNT	Watchdog Count When the watchdog is disabled (SWT_CR[WEN]=0), this field shows the value of the internal down counter. When the watchdog is enabled (SWT_CR[WEN]=1), this field is cleared (the value is 0x0000_0000). Values in this field can lag behind the internal counter value for up to six system plus eight counter clock cycles. Therefore, the value read from this field immediately after disabling the watchdog may be higher than the actual value of the internal counter.

46.3.1.7 SWT Service Key Register (SWT_SK)

The SWT Service Key (SWT_SK) register holds the previous (or initial) service key value. This register is read-only if either the SWT_CR[HLK] or SWT_CR[SLK] bits are set.

Offset: 0x0018 Access: Read/Write

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

Figure 562. SWT Service Key Register (SWT_SK)

Table 703. SWT_SK field descriptions

Field	Description
16:31 SK	Service Key This field is the previous (or initial) service key value used in keyed service mode. If SWT_CR[SMD] = 01, the next key value to be written to the SWT_SR is (17*SK+3) mod 2^16.

46.4 Functional description

46.4.1 Introduction

The SWT is a 32-bit timer designed to enable the system to recover in situations such as software getting trapped in a loop or if a bus transaction fails to terminate. It includes:

- A control register (SWT_CR)
- An interrupt register (SWT_IR)
- A time-out register (SWT_TO)
- A window register (SWT_WN)
- A service register (SWT_SR)
- A counter output register (SWT_CO)
- A service key register (SWT_SK)

Accesses to SWT registers occur with no peripheral bus wait states. (The peripheral bus bridge may add one or more system wait states.) However, due to synchronization logic in the SWT design, recognition of the service sequence or configuration changes may require up to three system plus seven counter clock cycles.

46.4.1.1 SWR_CR register

The SWT_CR register includes bits to enable the timer, set configuration options and lock configuration of the module. The watchdog is enabled by setting the SWT_CR[WEN] bit. The reset value of the SWT_CR[WEN] bit is device specific[19]. If the reset value of this bit is 1, the watchdog starts operation automatically after reset is released. Some devices can be configured to clear this bit automatically during the boot process.

19. Refer to SWT section in device configuration chapter.

46.4.1.2 SWT_TO register

The SWT_TO register holds the watchdog time-out period in clock cycles unless the value is less than 0x100 in which case the time-out period is set to 0x100.When the SWT is enabled, this time-out period is loaded into an internal 32-bit down counter each time a valid service operation is performed.The source used to drive the down counter and the reset value of the SWT_TO register is device specific, refer to the device configuration chapter.

46.4.1.3 SWT_CO register

The SWT_CO register shows the value of the down counter when the watchdog is disabled. When the watchdog is enabled this register is cleared. The value shown in this register can lag behind the value in the internal counter for up to six system plus eight counter clock cycles.

The SWT_CO register can be used during a software self test of the SWT. For example, the SWT can be enabled and not serviced for a fixed period of time less than the time-out value. Then the SWT can be disabled (SWT_CR[WEN] cleared) and the value of the SWT_CO register read to determine if the internal down counter is working properly.

46.4.2 Configuration locking

The configuration of the SWT can be locked through use of either a soft lock or a hard lock. In either case, when locked, the SWT_CR, SWT_TO, SWT_WN and SWT_SK registers are read-only.

46.4.2.1 Hard lock

The hard lock is enabled by setting the SWT_CR[HLK] bit, which can only be cleared by a reset.

46.4.2.2 Soft lock

The soft lock is enabled by setting the SWT_CR[SLK] bit and is cleared by writing the unlock sequence to the service register.

46.4.3 Unlock sequence

The unlock sequence is a write of 0xC520 followed by a write of 0xD928 to the SWT_SR[WSC] field. There is no timing requirement between the two writes. The unlock sequence logic ignores service sequence writes and recognizes the 0xC520, 0xD928 sequence regardless of previous writes. The unlock sequence can be written at any time and does not require the SWT_CR[WEN] bit to be set.

46.4.4 Servicing operations

When enabled, the SWT requires periodic execution of a servicing operation which is determined by the SWT_CR[SMD] field. Properly servicing the watchdog loads the internal down counter with the time-out period. The servicing modes are:

- Fixed service sequence
- Keyed service sequence
- Fixed execution address
- Incremental execution address

46.4.4.1 Fixed service sequence mode

If the SWT_CR[SMD] field is 00b, the fixed service sequence mode is selected which requires writing 0xA602, 0xB480 to the SWT_SR[WSC] field to service the watchdog. There is no timing requirement between the two writes and the service sequence logic ignores unlock sequence writes.

46.4.4.2 Keyed service sequence mode

If the SWT_CR[SMD] field is 01b, then the keyed service sequence mode is selected which requires writing two pseudo-random keys to the SWT_SR[WSC] field to service the watchdog. The key values are determined by the pseudo-random key generator defined in *Equation 28*. This algorithm will generate a sequence of 2^{16} different key values before repeating. The state of the key generator is held in the SWT_SK register. For example, if SWT_SK[SK] is 0x0100, then the service sequence keys are 0x1103, 0x2136. In this mode, each time a valid key is written to the SWT_SR register, the SWT_SK register is updated. So, after servicing the watchdog by writing 0x1103 and then 0x2136 to the SWT_SR[WSC] field, SWT_SK[SK] is 0x2136 and the next key sequence is 0x3499, 0x7E2C.

Equation 28

$$SK_{n+1} = (17*SK_n+3) \bmod 2^{16}$$

46.4.4.3 Fixed execution address mode

If the SWT_CR[SMD] field is set to 10b, the fixed execution address mode is selected which requires executing code at the address loaded into the designated IAC register to service the watchdog. In this mode, the IAC register cannot be updated while the watchdog is enabled.

46.4.4.4 Incremental execution address mode

If the SWT_CR[SMD] field is set to 11b, the incremental execution address mode is selected in which requires executing code at the address loaded into the designated IAC register to service the watchdog. In this mode, the IAC register can be updated while the watchdog is enabled.

46.4.4.5 Window mode

If window mode is enabled (SWT_CR[WND] bit is set), the service sequence must be performed in the last part of the time-out period defined by the window register. The window is open when the down counter is less than the value in the SWT_WN register. Outside of this window, service sequence writes are invalid accesses and generate a bus error or reset depending on the value of the SWT_CR[RIA] bit. For example, if the SWT_TO register is set to 5000 and SWT_WN register is set to ‘1000’ then the service sequence must be performed in the last 20% of the time-out period. There is a short lag in the time it takes for the window to open due to synchronization logic in the watchdog design. This delay could be up to three system plus four counter clock cycles.

46.4.5 Time-out

The SWT_CR[ITR] (“Interrupt Then Reset” bit) controls the action taken when a time-out occurs. If the SWT_CR[ITR] bit is not set, a reset is generated immediately on a time-out. If

the SWT_CR[ITR] bit is set, an initial time-out causes the SWT to generate an interrupt and load the down counter with the time-out period. If the service sequence is not written before the second consecutive time-out, the SWT generates a system reset. The interrupt is indicated by the time-out interrupt flag (SWT_IR[TIF]). The interrupt request is cleared by writing a one to the SWT_IR[TIF] bit.

46.4.6 Initialization

All registers should be initialized before setting the SWT_CR[WEN] bit to enable the watchdog. Registers can be initialized in any sequence.

47 GTM343 Integration (GTMINT) Module

47.1 Introduction

This chapter specifies the functionality of the GTMINT module. The GTMINT is integrated IP with timer and debug functions and is comprised of:

- The GTM-IP_343 timer module (also called GTM-IP)
- The GTMDI debug interface IP
- 14 ECC error correction encoder / decoders for the RAMs required by the timer.

The timer GTM-IP_343 is a very complex block that can be configured to perform several timer functions. The debugger module is the GTMDI IP (GTM Development Interface IP) that is a Nexus based interface that is connected to the chip level NAR and used to monitor the software and hardware activities of GTM-IP. The ECC sub-blocks generate the parity bits on RAM writes and detect errors in the data read from the RAMs.

Note: This chapter does not intend to present details of GTM-IP nor GTMDI. For full information about the timer and the debugger, please refer to the GTMDI subsection in the 'Calibration and Debug' chapter of the Eiger Reference Manual and the GTM343 Reference Manual.

47.1.1 Features

The GTMINT module features include:

- GTM-IP timer features
  - Time base generators
  - Advanced data signal / time base router
  - Input signal processors (40 inputs)
  - Simple output signal generators (64 outputs)
  - Microprocessed output signal generators (40 outputs)
  - Digital PLL
  - BLDC support
  - Safety features - GTM is a non-safety relevant module

    Output compare unit

    Monitoring unit
  - CPU bus interface clock operates asynchronously to timer section clock allowing CPU low power operation.
- GTMDI debug block features
  - Full duplex pin interface for medium and high visibility throughput

    Pin interface shared among all Nexus-based development interfaces on the device

    Auxiliary Output Port

    DATA with MDO (Message Data Out) and MSEO (Message Start/End)

    2 $\overline{\text{EVTO}}$ (Event Out) pin for GTM real time signal monitoring

    Auxiliary Input Port

    1 $\overline{\text{EVTI}}$ (Event In) pin

    IEEE 1149.1 (JTAG) Test Access Port (TAP)

    Support for optional Multi-JTAG TAP Linking Module (TLM)

    4 pins (TDI, TDO, TMS, and TCK)

    Reset input $\overline{\text{TRST}}$ driven by either the Nexus Port Controller or an external pin
  - GTM Development Support

    Read/Write registers in debug mode from JTAG port

    Ability to enter debug mode at reset negation or during normal execution

    Breakpoint and Watchpoint Configuration

    Data Trace via Data Write Messaging (DWM) and Data Read Messaging (DRM). This allows the development tool to trace reads and writes to selected RAM address ranges

    Two data trace windows for each MCS core with programmable address ranges and access attributes are provided. Data trace windowing reduces the

requirement on the Aurora port bandwidth by constraining the number of trace locations.

Fetch trace capability controlled by two programmable Watchpoints.

Private messages are used to indicate special MCS core cases not covered by public messages. Up to two[20] MCS cores can be debugged at the same time.

Watchpoint Messaging (WPM)

– Trigger Control

The GTMDI provides selection of trigger signals that will generate debug actions at system level such as Halt GTM itself or cores in the same device.

Trace enable signals are provided to control the start/stop of trace messages for each one of the GTM Nexus message sources, such as TIM, TOM, ATOM, etc.

– GTMDI Clock operation

The module is able to operate in any frequency ratio considering GTMDI and JTAG frequency. The GTMDI should operate in the same clock source as the GTM IP module.

- ECC encoder / decoder for each of 14 RAM interfaces
  - Parity calculation using the whole word and the target memory address value.
  - Detection and correction of single error on the read data value
  - Detection of single bit error on address field but no need for correction
  - Double error detection on data or address field without error correction.
  - ECC report generation for each RAM containing:

    Address of the error in the register map (see *Section 47.2: Memory Map and Register Definition*)

    Non-correctable error indication (not correctable data + address ECC detected errors)

    Correctable error indication (correctable data error)
- Generation of 151 interrupt requests and or DMA requests.
- Debugger is allowed to read all GTM registers through the peripheral bus even when the timer is running. Read-access of the debugger is possible without destroying or changing the contents of these registers.
- Debugger is allowed to read all GTM addresses read-/writeable from the peripheral bus when GTM is running or halted.

47.1.2 Modes of Operation

47.1.2.1 Normal Mode

The normal mode is the mode in which the GTMINT module is after a reset is applied. In this mode the timer IP is functional and processes input timing signals and generates timed outputs based on the input timing signals or controlled by the internal channels software. The GTMDI development interface is also active.

20.Note that by enabling two MCS cores at the same time for data trace can exceed the capacity of the Nexus Aurora Router throughput thus information can be lost.

47.1.2.2 Stop Mode

This mode is set when the global stop input is received or when the bit field MDIS on the GTMMCR register is asserted. In this case functional clock is disabled (ipg_clk), but the slave bus clock is not disabled. Please, see more details on *Section 47.3.4: Stop Mode Description*.

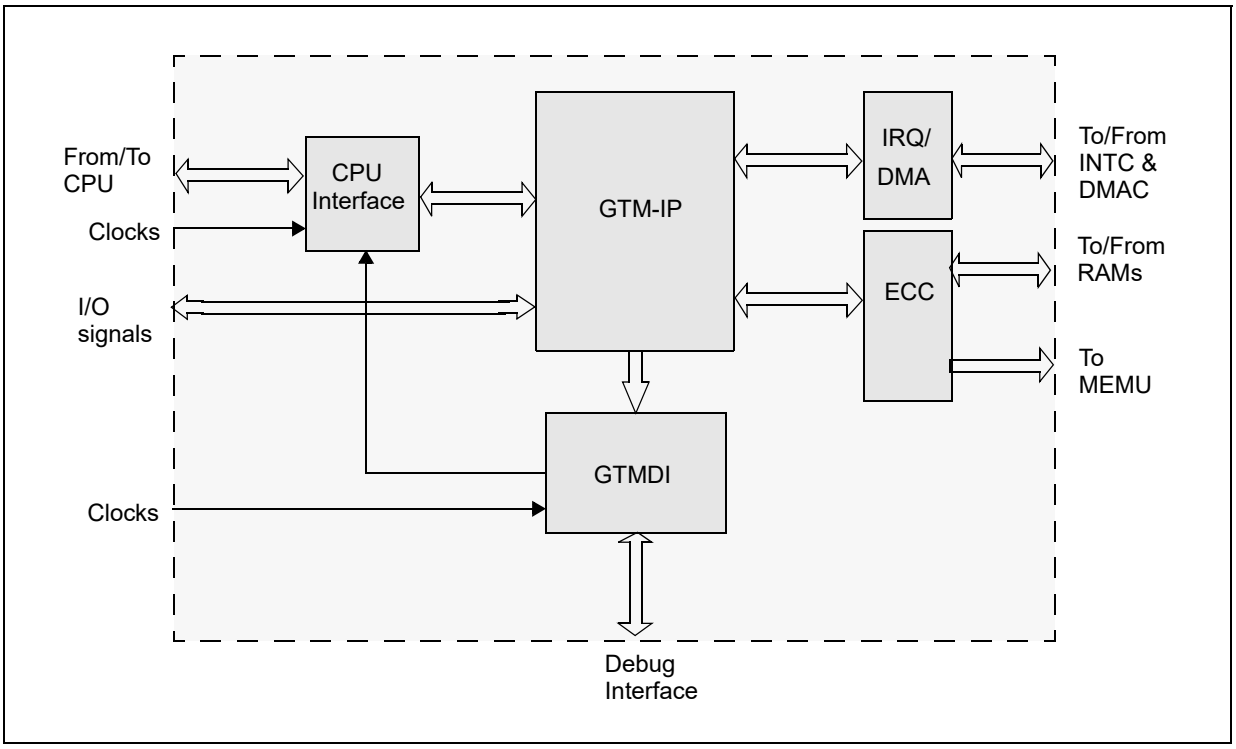
47.1.2.3 Debug Mode (or Halt Mode)

The debug mode or GTM-IP halt mode is externally requested through the GTMDI. In this mode, the GTM-IP internally switches off its main clock and the bus interface clocks remain running to access all registers and memories. It is possible to read the registers not interfering into the logic status, even for FIFOs, by configuring the debug access controls on GTMDI.

47.1.3 Block Diagram

Figure 563 shows the interconnections in the GTMINT Module.

Figure 563. GTMINT block diagram



47.2 Memory Map and Register Definition

This section provides a reference table to the GTM-IP sub-module registers. Details of each register are found in the *GTM-IP specification* document. Additional registers from those used in the GTM-IP are described in this section.

47.2.1 Module Memory Map

The GTMINT simplified memory map is shown in *Table 704*. Offsets are relative to the GTMINT base address.. See the device memory map for the module base address.

Note: The GTMINT module has only a few registers, detailed in Section 47.2.2: Register Descriptions. The module also acts as an interface to all implemented GTM-343 control registers. For details see the GTM-IP Specification document.

Table 704. High level memory map

Offset	Use
0x00000000	GTM-IP TOP-Level Configuration Registers
0x00000030	GTM-IP AEI Bridge Registers
0x000000C0	GTM-IP Integration Registers
0x00000100	TBU
0x00000180	MON
0x00000200	CMP
0x00000280	ARU
0x00000300	CMU
0x00000400	BRC
0x00000600	ICM
0x00000800	SPE0
0x00000880	SPE1
0x00000F00	MAP
0x00000F40	MCFG
0x00001000	TIM0
0x00001800	TIM1
0x00002000	TIM2
0x00002800	TIM3
0x00003000	TIM4
0x00008000	TOM0
0x00008800	TOM1
0x00009000	TOM2
0x00009800	TOM3
0x0000D000	LEG_ATOM0(1)
0x0000D800	LEG_ATOM1(1)
0x0000E000	LEG_ATOM2(1)
0x0000E800	LEG_ATOM3(1)
0x0000F000	LEG_ATOM4(1)
0x00013000	DTM0

Table 704. High level memory map (continued)

Offset	Use
0x00013040	DTM1
0x00013100	DTM4
0x00013140	DTM5
0x00013200	DTM8
0x00013240	DTM9
0x00013600	DTM24
0x00013640	DTM25
0x00013680	DTM26
0x000136C0	DTM27
0x00013700	DTM28
0x00013740	DTM29
0x00013780	DTM30
0x000137C0	DTM31
0x00013800	DTM32
0x00013840	DTM33
0x00018000	F2A0
0x00018080	AFD0
0x00018400	FIFO0
0x00019000	FIFO0_MEMORY
0x00028000	DPLL
0x00028200	DPLL_MEM R1A
0x00028400	DPLL_MEM R1B
0x00028600	DPLL_MEM_REGION R1C
0x00028E00	DPLL REGION EXT
0x0002C000	DPLL_MEM R2
0x00030000	LEG_MCS0(1)
0x00031000	LEG_MCS1(1)
0x00032000	LEG_MCS2(1)
0x00033000	LEG_MCS3(1)
0x00034000	LEG_MCS4(1)
0x00038000	MCS0_MEMORY
0x00040000	MCS1_MEMORY
0x00048000	MCS2_MEMORY
0x00050000	MCS3_MEMORY
0x00058000	MCS4_MEMORY

Table 704. High level memory map (continued)

Offset	Use
0x000E2000	CCM0
0x000E2200	CCM1
0x000E2400	CCM2
0x000E2600	CCM3
0x000E2800	CCM4
0x000E4000	CDTM0_DTM0
0x000E4040	CDTM0_DTM1
0x000E4100	CDTM0_DTM4
0x000E4140	CDTM0_DTM5
0x000E4400	CDTM1_DTM0
0x000E4440	CDTM1_DTM1
0x000E4500	CDTM1_DTM4
0x000E4540	CDTM1_DTM5
0x000E8000	CDTM2_DTM0
0x000E8040	CDTM2_DTM1
0x000E9000	CDTM2_DTM4
0x000E9040	CDTM2_DTM5
0x000E4D00	CDTM3_DTM4
0x000E4D40	CDTM3_DTM5
0x000E5100	CDTM4_DTM4
0x000E5140	CDTM4_DTM5
0x000E8000	ATOM0
0x000E8800	ATOM1
0x000E9000	ATOM2
0x000E9800	ATOM3
0x000EA000	ATOM4
0x000F0000	MCS0
0x000F1000	MCS1
0x000F2000	MCS2
0x000F3000	MCS3
0x000F4000	MCS4

1. In previous specification a smaller number of instances of the modules ATOM and MCS were included in the address list. The old address areas of modules are marked with LEG_ (legacy). These LEG_ areas are mapped to the normal areas.

This section presents the GTMINT Module Memory Map (that is basically the same of the timer GTM-IP) in *Table 705*.

Table 705. GTMINT Module Memory Map

Offset	Use	Access	Reset value(1)	Location
GTM-IP TOP-Level Configuration Registers				
0x00000	GTM-IP Version control register (GTM_REV)	R	0x343XXXX(6)	See GTM Spec.(2)
0x00004	GTM-IP Global reset register (GTM_RST)	R/W	0x0000_0000	See GTM Spec.(2)
0x00008	GTM-IP Global control register (GTM_CTRL)	R/W	0x0000_0001	See GTM Spec.(2)
0x0000C	GTM-IP Timeout Address Register (GTM_AEI_ADDR_XPT)	R	0x0000_0000	See GTM Spec.(2)
0x00010	GTM-IP Interrupt Notification Register (GTM_IRQ_NOTIFY)	R/W	0x0000_0000	See GTM Spec.(2)
0x00014	GTM-IP Interrupt Enable (GTM_IRQ_EN)	R/W	0x0000_0000	See GTM Spec.(2)
0x00018	GTM-IP Interrupt Force (GTM_IRQ_FORCINT)	R/W	0x0000_0000	See GTM Spec.(2)
0x0001C	GTM-IP Interrupt Mode (GTM_IRQ_MODE)	R/W	0x0000_0000	See GTM Spec.(2)
0x00020	GTM-IP (GTM_EIRQ_EN)	R/W	0x0000_0000	See GTM Spec.(2)
0x00024	GTM-IP Hardware configuration (GTM_HW_CONF)	R	0x3081_1333	See GTM Spec.(2)
0x00028	GTM-IP configuration (GTM-CFG)	R/W	0x0000_0000	See GTM Spec.(2)
0x0002C	GTM AEI Non zero status register (GTM_AEI_STA_XPT)	R/W	0x0000_0000	See GTM Spec.(2)
AEI Bridge (BRIDGE) Registers				
0x00030	GTM-IP AEI Bridge Mode (GTM_BRIDGE_MODE)	R/W	0x0400_0000 (3)	See GTM Spec.(2)
0x00034	GTM-IP AEI Bridge Pointer1 (GTM_BRIDGE_PTR1)	R	0x0040_0000 (3)	See GTM Spec.(2)
0x00038	GTM-IP AEI Bridge Pointer2 (GTM_BRIDGE_PTR2)	R	0x0000_0000	See GTM Spec.(2)
0x0003C	GTM-IP AEM Disable (GTM_MCS_AEM_DIS)	R	0x0000_0000	See GTM Spec.(2)
TIM AUX Registers				
0x00040+m*0x04	GTM-IP TIM[m] auxiliary source selection register (m = 0..4) (GTM_TIM[m]_AUX_IN_SRC)	R/W	0x0000_0000	See GTM Spec.(2)
Capture Trigger Registers				
0x0005C+m*0x04	GTM-IP external capture trigger enable (m=0..4) (GTM_EXT_CAP_EN_[m])	R/W	0x0000_0000	See GTM Spec.(2)

Table 705. GTMINT Module Memory Map (continued)

Offset	Use	Access	Reset value(1)	Location
TOM Output Registers				
0x00080+m*0x04	GTM-IP TOM[m] output level register (m=0..2) (GTM_TOM[m]_OUT)	R	0x00FF_FFFF	See GTM Spec.(2)
0x0008C	GTM-IP TOM3 output level register (GTM_TOM3_OUT)	R	0x0000_FFFF	See GTM Spec.(2)
0x00090 – 0x00094	Reserved			
ATOM Output Registers				
0x00098	GTM-IP ATOM0 output level register (GTM_ATOM0_OUT)	R	0xFFFF_FFFF	See GTM Spec.(2)
0x0009C	GTM-IP ATOM2 output level register (GTM_ATOM2_OUT)	R	0xFFFF_FFFF	See GTM Spec.(2)
0x000A0	GTM-IP ATOM4 output level register (GTM_ATOM4_OUT)	R	0x0000_FFFF	See GTM Spec.(2)
Cluster Configuration Registers				
0x000B0	GTM-IP cluster clock configuration register (GTM_CLS_CLK_CFG)	R/W	0x0000_0155	See GTM Spec.(2)
Top GTM Integration Registers				
0x000C0	GTMMCR – GTM Module Configuration Register	R/W	0xX000_X000(3)	*47.2.2.1 on page 1239*
0x000C4	Reserved			
0x000C8	GTMINTCLR – GTM Interrupts Clear Register	R/W	0x0000_0000	*47.2.2.2 on page 1240*
0x000CC	GTMMAEICR – GTM AEI Control Register	R/W	0x0000_0000	*47.2.2.3 on page 1242*
0x000D0 – 0x000FC	Reserved			
Time Base Unit (TBU) Configuration Registers				
0x00100	TBU global channel enable (TBU_CHEN)	R/W	0x0000_0000	See GTM Spec.(2)
0x00104+m*0x08	TBU channel m control (m = 0....2) (TBU_CH[m]_CTRL)	R/W	0x0000_0000	See GTM Spec.(2)
0x00108+n*0x08	TBU channel n base (n = 0...3) (TBU_CH[n]_BASE)	R/W	0x0000_0000	See GTM Spec.(2)
0x0011C	TBU channel 3 control (TBU_CH3_CTRL)	R/W	0x0000_0001	See GTM Spec.(2)
0x00124	TBU channel 3 base mark (TBU_CH3_BASE_MARK)	R/W	0x0000_0000	See GTM Spec.(2)
0x00128	TBU channel 3 base capture (TBU_CH3_BASE_CAPTURE)	R	0x0000_0000	See GTM Spec.(2)
0x0012C – 0x0017C	Reserved			

Table 705. GTMINT Module Memory Map (continued)

Offset	Use	Access	Reset value(1)	Location
Monitor Unit (MON) Configuration Registers				
0x00180	Monitor Status register (MON_STATUS)	R/W	0x0000_4000	See GTM Spec.(2)
0x00184+m*0x04	Monitor activity register m (m = 0,1) (MON_ACTIVITY_[m])	R/W	0x0000_0000	See GTM Spec.(2)
0x0018C+m*0x04	Monitor activity MCS m (m = 0...4) register (MON_ACTIVITY_MCS[m])	R/W	0x0000_0000	See GTM Spec.(2)
0x001A0 – 0x001FC	Reserved			
Output Compare Unit (CMP) Configuration Registers				
0x00200	Comparator enable register (CMP_EN)	R/W	0x0000_0000	See GTM Spec.(2)
0x00204	Event notification register (CMP_IRQ_NOTIFY)	R/W	0x0000_0000	See GTM Spec.(2)
0x00208	Interrupt enable register (CMP_IRQ_EN)	R/W	0x0000_0000	See GTM Spec.(2)
0x0020C	Interrupt force register (CMP_IRQ_FORCINT)	R/W	0x0000_0000	See GTM Spec.(2)
0x00210	IRQ mode configuration register (CMP_IRQ_MODE)	R/W	0x0000_0000	See GTM Spec.(2)
0x00214	IRQ mode configuration register (CMP_EIRQ_EN)	R/W	0x0000_0000	See GTM Spec.(2)
0x00218 – 0x00220	Reserved for CMP			
0x00224 – 0x0027C	Reserved			
Advanced Routing Unit (ARU) Configuration Registers				
0x00280	ARU access register (ARU_ACCESS)	R/W	0x0000_01FE	See GTM Spec.(2)
0x00284	ARU access register upper data word (ARU_DATA_H)	R/W	0x0000_0000	See GTM Spec.(2)
0x00288	ARU access register lower data word (ARU_DATA_L)	R/W	0x0000_0000	See GTM Spec.(2)
0x0028C	Debug access channel 0 (ARU_DBG_ACCESS0)	R/W	0x0000_01FE	See GTM Spec.(2)
0x00290	Debug access 0 transfer register upper data word (ARU_DBG_DATA0_H)	R	0x0000_0000	See GTM Spec.(2)
0x00294	Debug access 0 transfer register lower data word (ARU_DBG_DATA0_L)	R	0x0000_0000	See GTM Spec.(2)
0x00298	Debug access channel 1 (ARU_DBG_ACCESS1)	R/W	0x0000_01FE	See GTM Spec.(2)
0x0029C	Debug access 1 transfer register upper data word (ARU_DBG_DATA1_H)	R	0x0000_0000	See GTM Spec.(2)
0x002A0	Debug access 1 transfer register lower data word (ARU_DBG_DATA1_L)	R	0x0000_0000	See GTM Spec.(2)

Table 705. GTMINT Module Memory Map (continued)

Offset	Use	Access	Reset value(1)	Location
0x002A4	ARU Interrupt notification register (ARU_IRQ_NOTIFY)	R/W	0x0000_0000	See GTM Spec.(2)
0x002A8	ARU Interrupt enable register (ARU_IRQ_EN)	R/W	0x0000_0000	See GTM Spec.(2)
0x002AC	Register for forcing the ARU_NEW_DATA_IRQ interrupt (ARU_IRQ_FORCINT)	R/W	0x0000_0000	See GTM Spec.(2)
0x002B0	IRQ mode configuration register (ARU_IRQ_MODE)	R/W	0x0000_0000	See GTM Spec.(2)
0x002B4	End value of ARU CADDR counter (ARU_CADDR_END)	R/W	0x0000_00XX(6)	See GTM Spec.(2)
0x002BC	ARU control register (ARU_CTRL)	R/W	0x0000_0000	See GTM Spec.(2)
0x002C0	ARU0 dynamic route control register (ARU_0_DYN_CTRL)	R/W	0x0000_0000	See GTM Spec.(2)
0x002C4	ARU1 dynamic route control register (ARU_1_DYN_CTRL)	R/W	0x0000_0000	See GTM Spec.(2)
0x002C8	ARU0 read ID register low (ARU_0_DYN_ROUTE_LOW)	R/W	0x0000_0000	See GTM Spec.(2)
0x002CC	ARU1 read ID register low (ARU_1_DYN_ROUTE_LOW)	R/W	0x0000_0000	See GTM Spec.(2)
0x002D0	ARU0 read ID register high (ARU_0_DYN_ROUTE_HIGH)	R/W	0x0000_0000	See GTM Spec.(2)
0x002D4	RU1 read ID register high (ARU_1_DYN_ROUTE_HIGH)	R/W	0x0000_0000	See GTM Spec.(2)
0x002D8	Shadow register for ARU_0_DYN_ROUTE_LOW (ARU_0_DYN_ROUTE_SR_LOW)	R/W	0x0000_0000	See GTM Spec.(2)
0x002DC	Shadow register for ARU_1_DYN_ROUTE_LOW (ARU_1_DYN_ROUTE_SR_LOW)	R/W	0x0000_0000	See GTM Spec.(2)
0x002E0	Shadow register for ARU_0_DYN_ROUTE_HIGH (ARU_0_DYN_ROUTE_SR_HIGH)	R/W	0x0000_0000	See GTM Spec.(2)
0x002E4	Shadow register for ARU_1_DYN_ROUTE_HIGH (ARU_1_DYN_ROUTE_SR_HIGH)	R/W	0x0000_0000	See GTM Spec.(2)
0x002E8	ARU0 Read address for DYN_ROUTE (ARU_0_DYN_RDADDR)	R/W	0x0000_0000	See GTM Spec.(2)
0x002EC	ARU1 Read address for DYN_ROUTE (ARU_1_DYN_RDADDR)	R/W	0x0000_0000	See GTM Spec.(2)
0x002FC	Value of ARU CADDR counter (ARU_CADDR)	R	0x0000_0000	See GTM Spec.(2)

Table 705. GTMINT Module Memory Map (continued)

Offset	Use	Access	Reset value(1)	Location
Clock Management Unit (CMU) Configuration Registers				
0x00300	Clock enable (CMU_CLK_EN)	R/W	0x0000_0000	See GTM Spec.(2)
0x00304	Global clock control numerator (CMU_GCLK_NUM)	R/W	0x0000_0001	See GTM Spec.(2)
0x00308	Global clock control denominator (CMU_GCLK_DEN)	R/W	0x0000_0001	See GTM Spec.(2)
0x0030C+n*0x04	Control for clock source n (n = 0...7) (CMU_CLK_[n]_CTRL)	R/W	0x0000_0000	See GTM Spec.(2)
0x0032C+m*0x08	External clock m control numerator (m = 0...2) (CMU_ECLK_[m]_NUM)	R/W	0x0000_0001	See GTM Spec.(2)
0x00330+m*0x08	External clock m control denominator (m = 0...2) (CMU_ECLK_[m]_DEN)	R/W	0x0000_0001	See GTM Spec.(2)
0x00344	Global clock control denominator (CMU_FXCLK_CTRL)	R/W	0x0000_0000	See GTM Spec.(2)
0x00348	CMU global control (CMU_GLB_CTRL)	R/W	0x0000_0000	See GTM Spec.(2)
0x0034C	CMU clock control register (CMU_CLK_CTRL)	R/W	0x0000_0000	See GTM Spec.(2)
Broadcast Module (BRC) Configuration Registers (n = 0...11)				
0x00400+n*0x08	Read address for input channel n (n = 0..11) (BRC_SRC_[n]_ADDR)	R/W	0x0000_01FE	See GTM Spec.(2)
0x00404+n*0x08	Destination channels for input channel n (n = 0..11) (BRC_SRC_[n]_DEST)	R/W	0x0000_0000	See GTM Spec.(2)
0x00460	BRC Interrupt notification register (BRC_IRQ_NOTIFY)	R/W	0x0000_0000	See GTM Spec.(2)
0x00464	BRC Interrupt enable register (BRC_IRQ_EN)	R/W	0x0000_0000	See GTM Spec.(2)
0x00468	Register for forcing the BRC_DEST_ERR interrupt (BRC_IRQ_FORCINT)	R/W	0x0000_0000	See GTM Spec.(2)
0x0046C	IRQ mode configuration register (BRC_IRQ_MODE)	R/W	0x0000_0000	See GTM Spec.(2)
0x00470	Software reset (BRC_RST)	R/W	0x0000_0000	See GTM Spec.(2)
0x00474	Software reset (BRC_EIRQ_EN)	R/W	0x0000_0000	See GTM Spec.(2)
0x00478 – 0x005FC	Reserved			
Interrupt Concentrator Module (ICM) Configuration Registers				
0x00600	ICM Interrupt group register covering infrastructural and safety components (ARU, BRC, AEI, PSM, MAP, CMP, SPE) (ICM_IRQG_0)	R	0x0000_0000	See GTM Spec.(2)

Table 705. GTMINT Module Memory Map (continued)

Offset	Use	Access	Reset value(1)	Location
0x00604	ICM Interrupt group register covering DPLL (ICM_IRQG_1)	R	0x0000_0000	See GTM Spec.(2)
0x00608	ICM Interrupt group register covering TIM0, TIM1, TIM2, TIM3 (ICM_IRQG_2)	R	0x0000_0000	See GTM Spec.(2)
0x0060C	ICM Interrupt group register covering TIM4 (ICM_IRQG_3)	R	0x0000_0000	See GTM Spec.(2)
0x00610	ICM Interrupt group register covering MCS0 to MCS3 submodules (ICM_IRQG_4)	R	0x0000_0000	See GTM Spec.(2)
0x00614	ICM Interrupt group register covering MCS4 submodule (ICM_IRQG_5)	R	0x0000_0000	See GTM Spec.(2)
0x00618	ICM Interrupt group register covering GTM-IP output submodules TOM0 to TOM1 (ICM_IRQG_6)	R	0x0000_0000	See GTM Spec.(2)
0x0061C	ICM Interrupt group register covering GTM-IP output submodule TOM2 to TOM3 (ICM_IRQG_7)	R	0x0000_0000	See GTM Spec.(2)
0x00620	Reserved			
0x00624	ICM Interrupt group register covering GTM-IP output submodules ATOM0, ATOM1, ATOM2 and ATOM3 (ICM_IRQG_9)	R	0x0000_0000	See GTM Spec.(2)
0x00628	ICM Interrupt group register covering GTM-IP output submodule ATOM4 (ICM_IRQG_10)	R	0x0000_0000	See GTM Spec.(2)
0x0062C	Reserved for ICM			
0x00630	ICM Interrupt group register for module error interrupt information (ICM_IRQG_MEI)	R	0x0000_0000	See GTM Spec.(2)
0x00634	ICM Interrupt group register 0 for FIFO channel error interrupt information (ICM_IRQG_CEI0)	R	0x0000_0000	See GTM Spec.(2)
0x00638	ICM Interrupt group register 1 for TIM0..3 channel error interrupt information (ICM_IRQG_CEI1)	R	0x0000_0000	See GTM Spec.(2)
0x0063C	ICM Interrupt group register 2 for TIM0..3 channel error interrupt information (ICM_IRQG_CEI2)	R	0x0000_0000	See GTM Spec.(2)
0x00640	ICM Interrupt group register 3 for MCS0..3 channel error interrupt information (ICM_IRQG_CEI3)	R	0x0000_0000	See GTM Spec.(2)
0x00644	ICM Interrupt group register 2 for TIM0..3 channel error interrupt information (ICM_IRQG_CEI4)	R	0x0000_0000	See GTM Spec.(2)
0x00648 – 0x00660	Reserved			

Table 705. GTMINT Module Memory Map (continued)

Offset	Use	Access	Reset value(1)	Location
0x00664+m*0x04	MCS[m] channel error interrupt (m=0..4) (ICM_IRQG_MCS[m]_CEI)	R	0x0000_0000	See GTM Spec.(2)
0x006A4	PSM channel 0 error interrupt (ICM_IRQG_PSM_0_CEI)	R	0x0000_0000	See GTM Spec.(2)
0x006B4	SPE error interrupt (ICM_IRQG_SPE_CEI)	R	0x0000_0000	See GTM Spec.(2)
0x00710+m*0x04	Cluster [m] error interrupt (m=0,1) (ICM_IRQG_CLS_[m]_MEI)	R	0x0000_0000	See GTM Spec.(2)
0x00720+m*0x04	MCS channel [m] interrupt (m=0..4) (ICM_IRQG_MCS[m]_CI)	R	0x0000_0000	See GTM Spec.(2)
0x00760	PSM channel 0 interrupt (ICM_IRQG_PSM_0_CI)	R	0x0000_0000	See GTM Spec.(2)
0x00770	SPE channel interrupt (ICM_IRQG_SPE_CI)	R	0x0000_0000	See GTM Spec.(2)
0x00790	ATOM channel 0 interrupt (ICM_IRQG_ATOM_0_CI)	R	0x0000_0000	See GTM Spec.(2)
0x00794	ATOM channel 1 interrupt (ICM_IRQG_ATOM_1_CI)	R	0x0000_0000	See GTM Spec.(2)
0x007A0	TOM channel 0 interrupt (ICM_IRQG_TOM_0_CI)	R	0x0000_0000	See GTM Spec.(2)
0x007A4	TOM channel 1 interrupt (ICM_IRQG_TOM_1_CI)	R	0x0000_0000	See GTM Spec.(2)
Sensor Pattern Evaluation (SPE) Registers (AAi = i*0x80**, i = 0....1)**				
0x00800 + AAi	SPE Control status register (SPE[i]_CTRL_STAT)	R/W	0x0000_0000	See GTM Spec.(2)
0x00804 + AAi	SPE Input pattern definition register (SPE[i]_PAT)	R/W	0x0000_0000	See GTM Spec.(2)
0x00808+n*0x04 + AAi	SPE Output definition registers (SPE[i]_OUT_PAT[n]), n = 0...7	R/W	0x0000_0000	See GTM Spec.(2)
0x00828 + AAi	SPE output control register (SPE[i]_OUT_CTRL)	R/W	0x0000_0000	See GTM Spec.(2)
0x0082C + AAi	SPE Interrupt notification register (SPE[i]_IRQ_NOTIFY)	R/W	0x0000_0000	See GTM Spec.(2)
0x00830 + AAi	SPE Interrupt enable register (SPE[i]_IRQ_EN)	R/W	0x0000_0000	See GTM Spec.(2)
0x00834 + AAi	SPE Interrupt generation by software (SPE[i]_IRQ_FORCINT)	R/W	0x0000_0000	See GTM Spec.(2)
0x00838 + AAi	IRQ mode configuration register (SPE[i]_IRQ_MODE)	R/W	0x0000_0000	See GTM Spec.(2)
0x0083C + AAi	SPE Interrupt enable register (SPE[i]_EIRQ_EN)	R/W	0x0000_0000	See GTM Spec.(2)

Table 705. GTMINT Module Memory Map (continued)

Offset	Use	Access	Reset value(1)	Location
0x00840 + AAi	SPE Interrupt generation by software (SPE[i]_REV_CNT)	R/W	0x0000_0000	See GTM Spec.(2)
0x00844 + AAi	IRQ mode configuration register (SPE[i]_REV_CMP)	R/W	0x0000_0000	See GTM Spec.(2)
0x00848 + AAi	SPE control status 2 (SPE[i]_CTRL_STAT2)	R/W	0x0000_0000	See GTM Spec.(2)
0x0084C + AAi	SPE control command (SPE[i]_CMD)	R/W	0x0000_0000	See GTM Spec.(2)
0x00900 – 0x009C0	Reserved for SPE2/3			
0x00A00 – 0x00EFC	Reserved			
TIM0 Input Mapping Module (MAP) Registers				
0x00F00	MAP Control register (MAP_CTRL)	R/W	0x0000_0000	See GTM Spec.(2)
0x00F04 – 0x00F3C	Reserved for MAP			
Memory Configuration (MCFG) Configuration Registers				
0x00F40	Memory layout configuration (MCFG_CTRL)	R/W	0x0000_0000	See GTM Spec.(2)
0x00F44 – 0x00FA0	Reserved for MCFG			
0x00FA4 – 0x00FFC	Reserved			
Timer Input Module (TIM) Configuration Registers (BBi = i*0x800, n = 0...7, i = 0..4)				
0x01000+n*0x80 + BBi	Channel n general purpose 0 (TIM[i]_CH[n]_GPR0)	R	0x0X00_0000	See GTM Spec.(2)
0x01004+n*0x80 + BBi	Channel n general purpose 1 (TIM[i]_CH[n]_GPR1)	R	0x0X00_0000	See GTM Spec.(2)
0x01008+n*0x80 + BBi	Channel n SMU counter (TIM[i]_CH[n]_CNT)	R	0x0000_0000	See GTM Spec.(2)
0x0100C+n*0x80 + BBi	Channel n SMU counter (TIM[i]_CH[n]_ECNT)	R	0x0000_0000	See GTM Spec.(2)
0x01010+n*0x80 + BBi	Channel n SMU shadow counter (TIM[i]_CH[n]_CNTS)	R/W	0x0X00_0000	See GTM Spec.(2)
0x01014+n*0x80 + BBi	Channel n TDU control (TIM[i]_CH[n]_TDUC)	R/W	0x0000_0000	See GTM Spec.(2)
0x01018+n*0x80 + BBi	Channel n TDU control (TIM[i]_CH[n]_TDUV)	R/W	0x0000_0000	See GTM Spec.(2)
0x0101C+n*0x80 + BBi	Channel n filter parameter 0 (TIM[i]_CH[n]_FLT_RE)	R/W	0x0000_0000	See GTM Spec.(2)
0x01020+n*0x80 + BBi	Channel n filter parameter 1 (TIM[i]_CH[n]_FLT_FE)	R/W	0x0000_0000	See GTM Spec.(2)
0x01024+n*0x80 + BBi	Channel n control (TIM[i]_CH[n]_CTRL)	R/W	0x0000_0000	See GTM Spec.(2)

Table 705. GTMINT Module Memory Map (continued)

Offset	Use	Access	Reset value(1)	Location
0x01028+n*0x80 + BBi	Channel n control (TIM[i]_CH[n]_ECTRL)	R/W	0x0000_0000	See GTM Spec.(2)
0x0102C+n*0x80 + BBi	Channel n interrupt notification (TIM[i]_CH[n]_IRQ_NOTIFY)	R/W	0x0000_0000	See GTM Spec.(2)
0x01030+n*0x80 + BBi	Channel n interrupt enable (TIM[i]_CH[n]_IRQ_EN)	R/W	0x0000_0000	See GTM Spec.(2)
0x01034+n*0x80 + BBi	Channel n software interrupt force (TIM[i]_CH[n]_IRQ_FORCINT)	R/W	0x0000_0000	See GTM Spec.(2)
0x01038+n*0x80 + BBi	Channel n IRQ mode configuration register (TIM[i]_CH[n]_IRQ_MODE)	R/W	0x0000_0000	See GTM Spec.(2)
0x0103C+n*0x80 + BBi	Channel n software interrupt force (TIM[i]_CH[n]_EIRQ_EN)	R/W	0x0000_0000	See GTM Spec.(2)
(0x01040+n*0x80 – 0x01070+n*0x80) + BBi	Reserved for TIM[i]_CH[n]			
0x01074 + BBi	TIM i input value observation register (TIM[i]_INP_VAL)	R	0x0000_0000	See GTM Spec.(2)
0x01078 + BBi	Channel n IRQ mode configuration register (TIM[i]_IN_SRC)	R/W	0x0000_0000	See GTM Spec.(2)
0x107C + BBi	TIM Global software reset (TIM[i]_RST)	R/W	0x0000_0000	See GTM Spec.(2)
(0x01400 – 0x017FC) + BBi	Reserved			
0x033C0 – 0x04400	Reserved for TIM5 to TIM6			
0x04404 – 0x07FFC	Reserved			
Timer Output Module (TOM) Configuration Registers (CCi = i*0x800, **i = 0...3, n = 0...15)**				
0x08000 + n*0x0040 + CCi	TOM Channel m control register (TOM[i]_CH[n]_CTRL), i = 0...2, n = 0...15	R/W	0x0000_X000	See GTM Spec.(2)
0x08004 + n*0x0040 + CCi	TOM Channel n CCU0 compare shadow register (TOM[i]_CH[n]_SR0)	R/W	0x0000_0000	See GTM Spec.(2)
0x08008 + n*0x0040 + CCi	TOM Channel n CCU1 compare shadow register (TOM[i]_CH[n]_SR1)	R/W	0x0000_0000	See GTM Spec.(2)
0x0800C + n*0x0040 + CCi	TOM Channel n CCU0 compare register (TOM[i]_CH[n]_CM0)	R/W	0x0000_0000	See GTM Spec.(2)
0x08010 + n*0x0040 + CCi	TOM Channel n CCU1 compare register (TOM[i]_CH[n]_CM1)	R/W	0x0000_0000	See GTM Spec.(2)
0x08014 + n*0x0040 + CCi	TOM Channel n CCU0 counter register (TOM[i]_CH[n]_CN0)	R/W	0x0000_0000	See GTM Spec.(2)
0x08018 + n*0x0040 + CCi	TOM channel n status (TOM[i]_CH[n]_STAT)	R	0x0000_0001	See GTM Spec.(2)

Table 705. GTMINT Module Memory Map (continued)

Offset	Use	Access	Reset value(1)	Location
0x0801C + n*0x0040 + CCi	TOM channel n interrupt notification register (TOM[i]_CH[n]_IRQ_NOTIFY)	R/W	0x0000_0000	See GTM Spec.(2)
0x08020 + n*0x0040 + CCi	TOM channel n interrupt enable register (TOM[i]_CH[n]_IRQ_EN)	R/W	0x0000_0000	See GTM Spec.(2)
0x08024 + n*0x0040 + CCi	TOM channel n software interrupt generation (TOM[i]_CH[n]_IRQ_FORCINT)	–	0x0000_0000	See GTM Spec.(2)
0x08028 + n*0x0040 + CCi	Channel n IRQ mode configuration register (TOM[i]_CH[n]_IRQ_MODE)	R/W	0x0000_0000	See GTM Spec.(2)
0x0802C + n*0x0040 + CCi	Reserved			
0x08030 + CCi	TGC0 global control reg (TOM[i]_TGC0_GLB_CTRL)	R/W	0x0000_0000	See GTM Spec.(2)
0x08034 + CCi	TGC0 action time base register (TOM[i]_TGC0_ACT_TB)	R/W	0x0000_0000	See GTM Spec.(2)
0x08038 + CCi	TGC0 force update control reg (TOM[i]_TGC0_FUPD_CTRL)	R/W	0x0000_0000	See GTM Spec.(2)
0x0803C + CCi	TGC0 internal trigger control (TOM[i]_TGC0_INT_TRIG)	R/W	0x0000_0000	See GTM Spec.(2)
0x08070 + CCi	TGC0 enable/disable control reg (TOM[i]_TGC0_ENDIS_CTRL)	R/W	0x0000_0000	See GTM Spec.(2)
0x08074 + CCi	TGC0 enable/disable status reg (TOM[i]_TGC0_ENDIS_STAT)	R/W	0x0000_0000	See GTM Spec.(2)
0x08078 + CCi	TGC0 output enable control reg (TOM[i]_TGC0_OUTEN_CTRL)	R/W	0x0000_0000	See GTM Spec.(2)
0x0807C + CCi	TGC0 output enable status reg (TOM[i]_TGC0_OUTEN_STAT)	R/W	0x0000_0000	See GTM Spec.(2)
x08230 + CCi	TGC1 global control register (TOM[i]_TGC1_GLB_CTRL)	R/W	0x0000_0000	See GTM Spec.(2)
0x08234 + CCi	TGC1 action time base register (TOM[i]_TGC1_ACT_TB)	R/W	0x0000_0000	See GTM Spec.(2)
0x08238 + CCi	TGC1 force update control reg (TOM[i]_TGC1_FUPD_CTRL)	R/W	0x0000_0000	See GTM Spec.(2)
0x0823C + CCi	TGC1 internal trigger control (TOM[i]_TGC1_INT_TRIG)	R/W	0x0000_0000	See GTM Spec.(2)
0x08270 + CCi	TGC1 enable/disable control reg (TOM[i]_TGC1_ENDIS_CTRL)	R/W	0x0000_0000	See GTM Spec.(2)
0x08274 + CCi	TGC1 enable/disable status reg (TOM[i]_TGC1_ENDIS_STAT)	R/W	0x0000_0000	See GTM Spec.(2)
0x08278 + CCi	TGC1 output enable control reg (TOM[i]_TGC1_OUTEN_CTRL)	R/W	0x0000_0000	See GTM Spec.(2)
0x0827C + CCi	TGC1 output enable status reg (TOM[i]_TGC1_OUTEN_STAT)	R/W	0x0000_0000	See GTM Spec.(2)

Table 705. GTMINT Module Memory Map (continued)

Offset	Use	Access	Reset value[1]	Location
(0x08030 – 0x0803C) + m*0x0040 + CCi	Reserved (m= 2....7, 10....15)			
(0x08400 – 0x087FC) + CCi	Reserved			
0x09BEC – 0x0ABFC	Reserved for TOM4 to TOM5			
0x0AC00 – 0x0CFFC	Reserved			
ARU-connected Timer Output Module (ATOM) Registers (DDi = i*0x800, i = 0..4, m = 0..7)[4]				
0x0D000 + m*0x0080 + DDi	LEG_ATOM Channel m ARU read address register (LEG_ATOM[i]_CH[m]_RDADDR)	R/W	0x01FE_01FE	See GTM Spec.[2]
0xE8000 + m*0x0080 + DDi	ATOM Channel m ARU read address register (ATOM[i]_CH[m]_RDADDR)	R/W	0x01FE_01FE	See GTM Spec.[2]
0x0D004 + m*0x0080 + DDi	LEG_ATOM Channel m control register (LEG_ATOM[i]_CH[m]_CTRL)	R/W	0x0000_X000	See GTM Spec.[2]
0xE8004 + m*0x0080 + DDi	ATOM Channel m control register (ATOM[i]_CH[m]_CTRL)	R/W	0x0000_X000	See GTM Spec.[2]
0x0D008 + m*0x0080 + DDi	LEG_ATOM Channel m CCU0 compare shadow register (LEG_ATOM[i]_CH[m]_SR0)	R/W	0x0000_0000	See GTM Spec.[2]
0xE8008 + m*0x0080 + DDi	ATOM Channel m CCU0 compare shadow register (ATOM[i]_CH[m]_SR0)	R/W	0x0000_0000	See GTM Spec.[2]
0x0D00C + m*0x0080 + DDi	LEG_ATOM Channel m CCU1 compare shadow register (LEG_ATOM[i]_CH[m]_SR1)	R/W	0x0000_0000	See GTM Spec.[2]
0xE800C + m*0x0080 + DDi	ATOM Channel m CCU1 compare shadow register (ATOM[i]_CH[m]_SR1)	R/W	0x0000_0000	See GTM Spec.[2]
0x0D010 + m*0x0080 + DDi	LEG_ATOM Channel m CCU0 compare register (LEG_ATOM[i]_CH[m]_CM0)	R/W	0x0000_0000	See GTM Spec.[2]
0xE8010 + m*0x0080 + DDi	ATOM Channel m CCU0 compare register (ATOM[i]_CH[m]_CM0)	R/W	0x0000_0000	See GTM Spec.[2]
0x0D014 + m*0x0080 + DDi	LEG_ATOM Channel m CCU1 compare register (LEG_ATOM[i]_CH[m]_CM1)	R/W	0x0000_0000	See GTM Spec.[2]
0xE8014 + m*0x0080 + DDi	ATOM Channel m CCU1 compare register (ATOM[i]_CH[m]_CM1)	R/W	0x0000_0000	See GTM Spec.[2]
0x0D018 + m*0x0080 + DDi	LEG_ATOM Channel m CCU0 counter register (LEG_ATOM[i]_CH[m]_CN0)	R/W	0x0000_0000	See GTM Spec.[2]
0xE8018 + m*0x0080 + DDi	ATOM Channel m CCU0 counter register (ATOM[i]_CH[m]_CN0)	R/W	0x0000_0000	See GTM Spec.[2]
0x0D01C + m*0x0080 + DDi	LEG_ATOM Channel m status register (LEG_ATOM[i]_CH[m]_STAT)	R/W	0x0000_0001	See GTM Spec.[2]
0xE801C + m*0x0080 + DDi	ATOM Channel m status register (ATOM[i]_CH[m]_STAT)	R/W	0x0000_0001	See GTM Spec.[2]

Table 705. GTMINT Module Memory Map (continued)

Offset	Use	Access	Reset value[1]	Location
0x0D020 + m*0x0080 + DDi	LEG_ATOM channel m interrupt notification register (LEG_ATOM[i]_CH[m]_IRQ_NOTIFY)	R/W	0x0000_0000	See GTM Spec.[2]
0xE8020 + m*0x0080 + DDi	ATOM channel m interrupt notification register (ATOM[i]_CH[m]_IRQ_NOTIFY)	R/W	0x0000_0000	See GTM Spec.[2]
0x0D024 + m*0x0080 + DDi	LEG_ATOM channel m interrupt enable register (LEG_ATOM[i]_CH[m]_IRQ_EN)	R/W	0x0000_0000	See GTM Spec.[2]
0xE8024 + m*0x0080 + DDi	ATOM channel m interrupt enable register (ATOM[i]_CH[m]_IRQ_EN)	R/W	0x0000_0000	See GTM Spec.[2]
0x0D028 + m*0x0080 + DDi	LEG_ATOM channel m software interrupt generation (LEG_ATOM[i]_CH[m]_IRQ_FORCINT)	R/W	0x0000_0000	See GTM Spec.[2]
0xE8028 + m*0x0080 + DDi	ATOM channel m software interrupt generation (ATOM[i]_CH[m]_IRQ_FORCINT)	R/W	0x0000_0000	See GTM Spec.[2]
0x0D02C + m*0x0080 + DDi	Legacy channel m IRQ mode configuration register (LEG_ATOM[i]_CH[m]_IRQ_MODE)	R/W	0x0000_0000	See GTM Spec.[2]
0xE802C + m*0x0080 + DDi	Channel m IRQ mode configuration register (ATOM[i]_CH[m]_IRQ_MODE)	R/W	0x0000_0000	See GTM Spec.[2]
(0x0D030 – 0x0D03C) + m*0x0080 + DDi	Reserved for LEG_ATOM[i]_CH[m]			
(0xE8030 – 0xE803C) + m*0x0080 + DDi	Reserved for ATOM[i]_CH[m]			
0x0D040 + DDi	Legacy AGC Global control register (LEG_ATOM[i]_AGC_GLB_CTRL)	R/W	0x0000_0000	See GTM Spec.[2]
0xE8040 + DDi	AGC Global control register (ATOM[i]_AGC_GLB_CTRL)	R/W	0x0000_0000	See GTM Spec.[2]
0x0D044 + DDi	Legacy AGC0 Enable/disable control register (LEG_ATOM[i]_AGC_ENDIS_CTRL)	R/W	0x0000_0000	See GTM Spec.[2]
0xE8044 + DDi	AGC0 Enable/disable control register (ATOM[i]_AGC_ENDIS_CTRL)	R/W	0x0000_0000	See GTM Spec.[2]
0x0D048 + DDi	Legacy AGC Enable/disable status register (LEG_ATOM[i]_AGC_ENDIS_STAT)	R/W	0x0000_0000	See GTM Spec.[2]
0xE8048 + DDi	AGC Enable/disable status register (ATOM[i]_AGC_ENDIS_STAT)	R/W	0x0000_0000	See GTM Spec.[2]
0x0D04C + DDi	Legacy AGC Action time base register (LEG_ATOM[i]_AGC_ACT_TB)	R/W	0x0000_0000	See GTM Spec.[2]
0xE804C + DDi	AGC Action time base register (ATOM[i]_AGC_ACT_TB)	R/W	0x0000_0000	See GTM Spec.[2]

Table 705. GTMINT Module Memory Map (continued)

Offset	Use	Access	Reset value(1)	Location
0x0D050 + DDi	Legacy AGC Output enable control register (LEG_ATOM[i]_AGC_OUTEN_CTRL)	R/W	0x0000_0000	See GTM Spec.(2)
0xE8050 + DDi	AGC Output enable control register (ATOM[i]_AGC_OUTEN_CTRL)	R/W	0x0000_0000	See GTM Spec.(2)
0x0D054 + DDi	Legacy AGC Output enable status register (LEG_ATOM[i]_AGC_OUTEN_STAT)	R/W	0x0000_0000	See GTM Spec.(2)
0xE8054 + DDi	AGC Output enable status register (ATOM[i]_AGC_OUTEN_STAT)	R/W	0x0000_0000	See GTM Spec.(2)
0x0D058 + DDi	Legacy AGC Force update control register (LEG_ATOM[i]_AGC_FUPD_CTRL)	R/W	0x0000_0000	See GTM Spec.(2)
0xE8058 + DDi	AGC Force update control register (ATOM[i]_AGC_FUPD_CTRL)	R/W	0x0000_0000	See GTM Spec.(2)
0x0D05C + DDi	Legacy AGC Internal trigger control register (LEG_ATOM[i]_AGC_INT_TRIG)	R/W	0x0000_0000	See GTM Spec.(2)
0xE805C + DDi	AGC Internal trigger control register (ATOM[i]_AGC_INT_TRIG)	R/W	0x0000_0000	See GTM Spec.(2)
(0xE8060 – 0xE807C) + m*0x0080 + DDi	Reserved for ATOM[i]_CH[m]			
(0xE83B0 – 0xE87C0) + DDi	Reserved for ATOM[i]			
0xEA3B0 – 0xEFFFC	Reserved for ATOM5 to ATOM11			
Dead Time Module (DTM) Configuration Registers (BBi = i*0x40, i = 0,1,4,8,9,24..33)				
0x013000 + BBi	Global Configuration and Control Register (DTM[i]_CTRL)	R/W	0x0000_0000	See GTM Spec.(2)
0x13004 + BBi	Channel Control Register 1 (DTM[i]_CH_CTRL1)	R/W	0x0000_0000	See GTM Spec.(2)
0x13008 + BBi	Channel Control Register 2 (DTM[i]_CH_CTRL2)	R/W	0x0000_0000	See GTM Spec.(2)
0x1300C + BBi	Channel Control Register 2 Shadow (DTM[i]_CH_CTRL2_SR)	R/W	0x0000_0000	See GTM Spec.(2)
0x13010 + BBi	Channel Control Register 3 (DTM[i]_CH_CTRL3)	R/W	0x0000_0000	See GTM Spec.(2)
0x13014 + BBi	Phase Shift Unit Configuration and Control Register (DTM[i]_PS_CTRL)	R/W	0x0000_0000	See GTM Spec.(2)
0x13018 + BBi	Dead Time Reload Values; x=0..3 (DTM[i]_CH[x]_DTV)	R/W	0x0000_0000	See GTM Spec.(2)
0x13024 + BBi	Channel Shadow Register (DTM[i]_CH_SR)	R/W	0x0000_0000	See GTM Spec.(2)
0x1386C – 0x17FFC	Reserved			

Table 705. GTMINT Module Memory Map (continued)

Offset	Use	Access	Reset value[1]	Location
F2A[i] Configuration Registers (m = 0...7, i = 0, EEi = 0x4000)				
0x18000+m*0x04 + EEi	F2A read channel address register (F2A[i]_CH[m]_ARU_RD_FIFO)	R/W	0x0000_01FE	See GTM Spec.[2]
0x18020+m*0x04 + EEi	F2A stream x configuration register (F2A[i]_CH[m]_STR_CFG)	R/W	0x0000_0000	See GTM Spec.[2]
0x18040 + EEi	F2A stream activation register (F2A[i]_ENABLE)	R/W	0x0000_0000	See GTM Spec.[2]
0x18044 + EEi	F2A stream control register (F2A[i]_CTRL)	R/W	0x0000_0000	See GTM Spec.[2]
(0x18048 - 0x1807C) + EEi	Reserved			
AFD[i] Registers (m = 0...7, i = 0, EEi = 0x4000)				
0x18080+m*0x10 + EEi	AFD FIFO m buffer access register (AFD[i]_CH[m]_BUFF_ACC)	R/W	0x0000_0000	See GTM Spec.[2]
(0x18084+m*0x10 - 0x1808C+m*0x10) + EEi	Reserved for AFD1			
(0x18100 - 0x183FC) + EEi	Reserved			
FIFO[i] Configuration Registers (m = 0...7, i = 0, EEi = 0x4000)				
0x18400+m*0x40 + EEi	FIFO Channel m control register (FIFO[i]_CH[m]_CTRL)	R/W	0x0000_0000	See GTM Spec.[2]
0x18404+m*0x40 + EEi	FIFO Channel m end address register (FIFO[i]_CH[m]_END_ADDR)	R/W	0x0000_0XXX[3]	See GTM Spec.[2]
0x18408+m*0x40 + EEi	FIFO Channel m start address register (FIFO[i]_CH[m]_START_ADDR)	R/W	0x0000_0XXX[3]	See GTM Spec.[2]
0x1840C+m*0x40 + EEi	FIFO Channel m upper watermark register (FIFO[i]_CH[m]_UPPER_WM)	R/W	0x0000_0060	See GTM Spec.[2]
0x18410+m*0x40 + EEi	FIFO Channel m lower watermark register (FIFO[i]_CH[m]_LOWER_WM)	R/W	0x0000_0020	See GTM Spec.[2]
0x18414+m*0x40 + EEi	FIFO Channel x status register (FIFO[i]_CH[m]_STATUS)	R	0x0000_0005	See GTM Spec.[2]
0x18418+m*0x40 + EEi	FIFO Channel m fill level register (FIFO[i]_CH[m]_FILL_LEVEL)	R	0x0000_0000	See GTM Spec.[2]
0x1841C+m*0x40 + EEi	FIFO Channel m write pointer register (FIFO[i]_CH[m]_WR_PTR)	R	0x0000_0XXX[3]	See GTM Spec.[2]
0x18420+m*0x40 + EEi	FIFO Channel m read pointer register (FIFO[i]_CH[m]_RD_PTR)	R	0x0000_0XXX[3]	See GTM Spec.[2]
0x18424+m*0x40 + EEi	FIFO m interrupt notification register (FIFO[i]_CH[m]_IRQ_NOTIFY)	R/W	0x0000_0005	See GTM Spec.[2]

Table 705. GTMINT Module Memory Map (continued)

Offset	Use	Access	Reset value(1)	Location
0x18428+m*0x40 + EEi	FIFO m interrupt enable register (FIFO[i]_CH[m]_IRQ_EN)	R/W	0x0000_0000	See GTM Spec.(2)
0x1842C+m*0x40 + EEi	FIFO m register to force interrupt by software (FIFO[i]_CH[m]_IRQ_FORCINT)	R/W	0x0000_0000	See GTM Spec.(2)
0x18430+m*0x40 + EEi	FIFO m IRQ mode control register (FIFO[i]_CH[m]_IRQ_MODE)	R/W	0x0000_0000	See GTM Spec.(2)
0x18434+m*0x40 + EEi	FIFO m error interrupt enable register (FIFO[i]_CH[m]_EIRQ_EN)	R/W	0x0000_0000	See GTM Spec.(2)
(0x18438+m*0x40 – 0x1843C+m*0x40) + EEi	Reserved			
(0x18600 – 0x18FFC) + EEi	Reserved for FIFO[i]			
(0x19000 – 0x19FFC) + EEi	FIFO[i] Memory			See GTM Spec.(2)
(0x1A000 – 0x1BFFC) + EEi	Reserved			
0x1C000 – 0x1DFFC	Reserved for F2A1, AFD1, FIFO1, and FIFO1 MEM			
0x1E000 – 0x27FFC	Reserved			
Digital PLL Module (DPLL)				
0x28000	Control Register 0 (DPLL_CTRL_0	R/W	0x003B_BA57	See GTM Spec.(2)
0x28004	Control Register 1 (DPLL_CTRL_1)	R/W	0xB000_0000	See GTM Spec.(2)
0x28008	Control Register 2 (actions 0–7 enable) (DPLL_CTRL_2)	R/W	x0000_0000	See GTM Spec.(2)
0x2800C	Control Register 3 (actions 8–15 enable) (DPLL_CTRL_3)	R/W	x0000_0000	See GTM Spec.(2)
0x28010	Control Register 4 (actions 16–24 enable) (DPLL_CTRL_4)	R/W	x0000_0000	See GTM Spec.(2)
0x28014	Reserved			
0x28018	ACTION Status Register with connected shadow register (DPLL_ACT_STA)	R/W	x0000_0000	See GTM Spec.(2)
0x2801C	Offset and switch old/new address register (DPLL_OSW)	R/W	0x0000_0200	See GTM Spec.(2)
0x28020	Address offset register for APT in RAM region 2 (DPLL_AOSV_2)	R/W	0x1810_0800	See GTM Spec.(2)
0x28024	Actual RAM pointer to RAM regions 2a, b and d (DPLL_APT)	R/W	0x0000_0000	See GTM Spec.(2)

Table 705. GTMINT Module Memory Map (continued)

Offset	Use	Access	Reset value(1)	Location
0x28028	Actual RAM pointer to regions 1c1, 1c2 and 1c4 (DPLL_APS)	R/W	0x0000_0000	See GTM Spec.(2)
0x2802C	Actual RAM pointer to RAM region 2c (DPLL_APT_2c)	R/W	0x0000_0000	See GTM Spec.(2)
0x28030	Actual RAM pointer to RAM region 1c3 (DPLL_APS_1c3)	R/W	0x0000_0000	See GTM Spec.(2)
0x28034	Number of recent TRIGGER events used for calculations (mod 2*(TNU +1-SYN_NT)) (DPLL_NUTC)	R/W	0x0001_2001	See GTM Spec.(2)
0x28038	Number of recent STATE events used for calculations (mod 2*(SNU +1-SYN_NS) (DPLL_NUSC)	R/W	0x0000_2081	See GTM Spec.(2)
0x2803C	Number of active TRIGGER events to interrupt (DPLL_NTI_CNT)	R/W	0x0000_0000	See GTM Spec.(2)
0x28040	Interrupt notification register (DPLL_IRQ_NOTIFY)	R/W	0x0000_0000	See GTM Spec.(2)
0x28044	Interrupt enable register (DPLL_IRQ_EN)	R/W	0x0000_0000	See GTM Spec.(2)
0x28048	Interrupt force register (DPLL_IRQ_FORCINT)	R/W	0x0000_0000	See GTM Spec.(2)
0x2804C	Interrupt mode register (DPLL_IRQ_MODE)	R/W	0x0000_0000	See GTM Spec.(2)
0x28050	Error Interrupt enable register (DPLL_EIRQ_EN)	R/W	0x0000_0000	See GTM Spec.(2)
0x28054 – 0x280AC	Reserved			
0x280B0	Counter for pulses for TBU_TS1 to be sent in automatic end mode (DPLL_INC_CNT1)	R/W	0x0000_0000	See GTM Spec.(2)
0x280B4	Counter for pulses for TBU_TS2 to be sent in automatic end mode when SMC = RMO = 1w (DPLL_INC_CNT2)	R/W	0x0000_0000	See GTM Spec.(2)
0x280B8	Old RAM pointer and offset value for TRIGGER (DPLL_APT_SYNC)	R/W	0x0000_0000	See GTM Spec.(2)
0x280BC	Old RAM pointer and offset value for STATE (DPLL_APS_SYNC)	R/W	0x0000_0000	See GTM Spec.(2)
0x280C0	TBU_TS0 value at last TRIGGER event (DPLL_TBU_TS0_T)	R/W	0x0000_0000	See GTM Spec.(2)
0x280C4	TBU_TS0 value at last STATE event (DPLL_TBU_TS0_S)	R/W	0x0000_0000	See GTM Spec.(2)
0x280C8	Direct load input value for SUB_INC1 (DPLL_ADD_IN_LD1)	R/W	0x0000_0000	See GTM Spec.(2)
0x280CC	Direct load input value for SUB_INC2 (DPLL_ADD_IN_LD2)	R/W	0x0000_0000	See GTM Spec.(2)
0x280D0 – 0x280F8	Reserved for DPLL			

Table 705. GTMINT Module Memory Map (continued)

Offset	Use	Access	Reset value(1)	Location
0x280FC	Status Register (DPLL_STATUS)	R/W	x0000_0000	See GTM Spec.(2)
0x28100 + n*0x0004	9 bit ID information for input signal PMTR_n (8:0), n = 0...23 (DPLL_ID_PMTR_n)	R/W	0x0000_01FE	See GTM Spec.(2)
0x28160 – 0x281DC	Reserved for DPLL			
0x281E0	shadow register of DPLL_CTRL_0 (DPLL_CTRL_0_SHADOW_TRIGGER)	R/W	0x0000_0257	See GTM Spec.(2)
0x281E4	shadow register of DPLL_CTRL_0 (DPLL_CTRL_0_SHADOW_STATE)	R/W	0x0000_0000	See GTM Spec.(2)
0x281E8	shadow register of DPLL_CTRL_1 (DPLL_CTRL_1_SHADOW_TRIGGER)	R/W	0x0000_0000	See GTM Spec.(2)
0x281EC	shadow register of DPLL_CTRL_1 (DPLL_CTRL_1_SHADOW_STATE)	R/W	0x0000_0000	See GTM Spec.(2)
0x281F0 – 0x281F8	Reserved			
0x281FC	Initialization Control and Status for RAMs (DPLL_RAM_INI)	R/W	0x0000_0000	See GTM Spec.(2)
DPLL RAM 1A Map				
	RAM Region 1a (0.288 KB for 96 words of 24 bits)			
0x28200 + n*0x0004	ACTION_n Position/Value action request Register (PSAn), n = 0...23	R/W	0x0000_0000	See GTM Spec.(2)
0x28260 + m*0x0004	ACTION_m time to react before PSAm (DLAm), m = 0...23	R/W	0x0000_0000	See GTM Spec.(2)
0x282C0 + n*0x0004	Number of TRIGGER/STATE increments to ACTION_n (NAn), n = 0...23	R/W	0x0000_0000	See GTM Spec.(2)
0x28320 + m*0x0004	Calculated relative time to ACTION_m (DTAm), m = 0...23	R/W	0x0000_0000	See GTM Spec.(2)
0x28380 – 0x283FC	Reserved	R/W		
DPLL RAM 1B Map (Regions 1b and 1c)				
	DPLL RAM Region 1b (0.375 KB for 128 words of 24 bits) TRIGGER signal information stored Note: The switch of the LSB address bits is performed using the SWON register at 0x0020.			
0x28400/ 0x0404	Actual signal TRIGGER time stamp register TRIGGER_TS (TS_T)	R/W	0x0000_0000	See GTM Spec.(2)
0x28404/ 0x0400	Previous signal TRIGGER time stamp register TRIGGER_TS_old (TS_T_old)	R/W	0x0000_0000	See GTM Spec.(2)
0x28408	Actual signal TRIGGER filter and signal value (FTV_T)	R/W	0x0000_0000	See GTM Spec.(2)
0x2840C	Reserved(5)			

Table 705. GTMINT Module Memory Map (continued)

Offset	Use	Access	Reset value(1)	Location
	STATE signal information stored Note: The switch of the LSB address bits is performed using the SWON register at 0x0020.	R/W	0x0000_0000	See GTM Spec.(2)
0x28410/ 0x0414	Actual signal STATE time stamp register STATE_TS (TS_S)	R/W	0x0000_0000	See GTM Spec.(2)
0x28414/ 0x0410	Previous signal STATE time stamp register STATE_TS_old (TS_S_old)	R/W	0x0000_0000	See GTM Spec.(2)
0x28418	Actual signal STATE filter and signal value (FTV_S)	R/W	0x0000_0000	See GTM Spec.(2)
0x2841C	Reserved(5)			
0x28420	TRIGGER hold time min value (THMI)	R/W	0x0000_0000	See GTM Spec.(2)
0x28424	TRIGGER hold time max value (THMA)	R/W	0x0000_0000	See GTM Spec.(2)
0x28428	Measured last pulse time from valid to invalid TRIGGER slope (THVAL)	R/W	0x0000_0000	See GTM Spec.(2)
0x2842C	Reserved(5)			
0x28430	Time out value of active TRIGGER slope (TOV)	R/W	0x0000_0000	See GTM Spec.(2)
0x28434	Time out value of active STATE slope (TOV_S)	R/W	0x0000_0000	See GTM Spec.(2)
0x28438	Calculated ADD_IN value for normal SUB_INC1 generation (ADD_IN_CAL1)	R/W	0x0000_0000	See GTM Spec.(2)
0x2843C	Calculated ADD_IN value for normal SUB_INC2 generation (ADD_IN_CAL2)	R/W	0x0000_0000	See GTM Spec.(2)
0x28440	missing pulses to be added/subtracted directly to SUB_INC1 and INC_CNT1 once (MPVAL1)	R/W	0x0000_0000	See GTM Spec.(2)
0x28444	missing pulses to be added/subtracted directly to SUB_INC2 and INC_CNT2 once (MPVAL2)	R/W	0x0000_0000	See GTM Spec.(2)
0x28448	target number of TRIGGER pulses (NMB_T_TAR)	R/W	0x0000_0000	See GTM Spec.(2)
0x2844C	target number of TRIGGER pulses (NMB_T_TAR_old)	R/W	0x0000_0000	See GTM Spec.(2)
0x28450	target number of STATE pulses (NMB_S_TAR)	R/W	0x0000_0000	See GTM Spec.(2)
0x28454	target number of STATE pulses (NMB_S_TAR_old)	R/W	0x0000_0000	See GTM Spec.(2)
0x28458 – 0x2845C	Reserved(5)			
0x28460	reciprocal value of expected increment duration (T) (RCDT_TX)	R/W	0x0000_0000	See GTM Spec.(2)

Table 705. GTMINT Module Memory Map (continued)

Offset	Use	Access	Reset value[1]	Location
0x28464	reciprocal value of expected increment duration (S) (RCDT_SX)	R/W	0x0000_0000	See GTM Spec.[2]
0x28468	reciprocal value of the expected nominal increment duration (T) (RCDT_TX_nom)	R/W	0x0000_0000	See GTM Spec.[2]
0x2846C	reciprocal value of the expected nominal increment duration (S) (RCDT_SX_nom)	R/W	0x0000_0000	See GTM Spec.[2]
0x28470	actual reciprocal value of TRIGGER (RDT_T_actual)	R/W	0x0000_0000	See GTM Spec.[2]
0x28474	actual reciprocal value of STATE (RDT_S_actual)	R/W	0x0000_0000	See GTM Spec.[2]
0x28478	Duration of last TRIGGER increment (DT_T_actual)	R/W	0x0000_0000	See GTM Spec.[2]
0x2847C	Duration of last STATE increment (DT_S_actual)	R/W	0x0000_0000	See GTM Spec.[2]
0x28480	Absolute error of prediction for last TRIGGER increment (EDT_T)	R	0x0000_0000	See GTM Spec.[2]
0x28484	Average absolute error of prediction up to the last TRIGGER increment (MEDT_T)	R	0x0000_0000	See GTM Spec.[2]
0x28488	absolute error of prediction for last STATE increment (EDT_S)	R	0x0000_0000	See GTM Spec.[2]
0x2848C	Average absolute error of prediction up to the last STATE increment (MEDT_S)	R	0x0000_0000	See GTM Spec.[2]
0x28490	Expected duration of current TRIGGER increment (CDT_TX)	R	0x0000_0000	See GTM Spec.[2]
0x28494	Expected duration of current STATE increment (CDT_SX)	R	0x0000_0000	See GTM Spec.[2]
0x28498	Expected nominal duration of current TRIGGER increment (without consideration of missing events) (CDT_TX_nom)	R	0x0000_0000	See GTM Spec.[2]
0x2849C	Expected nominal duration of current STATE increment (without consideration of missing events) (CDT_SX_nom)	R	0x0000_0000	See GTM Spec.[2]
0x284A0	TRIGGER Locking Range (TLR)	R/W	0x0000_0000	See GTM Spec.[2]
0x284A4	STATE Locking Range (SLR)	R/W	0x0000_0000	See GTM Spec.[2]
0x28500 + n*0x0004	For ACTION_n in normal mode (PDT_n), n = 0...23	R/W	0x0000_0000	See GTM Spec.[2]
0x028560 – 0x285BC	Reserved[5]			
0x285C0	Calculated number of sub-pulses between two STATE events (to be set by CPU) (MLS1)	R/W	0x0000_0000	See GTM Spec.[2]

Table 705. GTMINT Module Memory Map (continued)

Offset	Use	Access	Reset value(1)	Location
0x285C4	Calculated number of sub-pulses between two STATE events (to be set by CPU) for the use when SMC = RMO = 1B (MLS2)	R/W	0x0000_0000	See GTM Spec.(2)
0x285C8	number of sub-pulses of SUB_INC1 in continuous mode, updated by the host only (CNT_NUM1)	R/W	0x0000_0000	See GTM Spec.(2)
0x285CC	number of sub-pulses of SUB_INC2 in continuous mode, updated by the host only (CNT_NUM2)	R/W	0x0000_0000	See GTM Spec.(2)
0x285D0	Plausibility value of next active TRIGGER slope (PVT)	R/W	0x0000_0000	See GTM Spec.(2)
0x285E0	Accurate calculated position stamp of last TRIGGER input (PSTC)	R/W	0x0000_0000	See GTM Spec.(2)
0x285E4	Accurate calculated position stamp of last STATE input (PSSC)	R/W	0x0000_0000	See GTM Spec.(2)
0x285E8/ 0x285EC	Measured position stamp of last TRIGGER input (PSTM)	R/W	0x0000_0000	See GTM Spec.(2)
0x285EC/ 0x285E8	Measured position stamp of last but one TRIGGER input (PSTM_old)	R/W	0x0000_0000	See GTM Spec.(2)
0x285F0/ 0x285F4	Measured position stamp of last STATE input (PSSM)	R/W	0x0000_0000	See GTM Spec.(2)
0x285F4/ 0x285F0	Measured position stamp of last but one STATE input (PSSM_old)	R/W	0x0000_0000	See GTM Spec.(2)
0x285F8	Number of pulses of current increment in normal mode for SUB_INC1 (NMB_T)	R/W	0x0000_0000	See GTM Spec.(2)
0x285FC	Number of pulses of current increment in emergency mode for SUB_INC1 (NMB_S)	R/W	0x0000_0000	See GTM Spec.(2)
	DPLL RAM Region 1c (0.75 KB for 256 words of 24 bits) Note: The following registers for the signal STATE are located in RAM Region 1c, read access by AEI via bus interface possible, writing results in an interrupt; data width of 3 bytes used for 24-bit values.			
	1c1			
0x28600 + n*0x0004	Nominal STATE reciprocal values in FULL_SCALE (RDT_Sn), n = 0...63 (2*(SNU+1-SYN_NS) valid entries)	R/W	0x0000_0000	See GTM Spec.(2)
	1c2			
0x28700 + m*0x0004	Time stamp field of STATE Events (TSF_Sm), m = 0...63 (2*(SNU+1) valid entries)	R/W	0x0000_0000	See GTM Spec.(2)

Table 705. GTMINT Module Memory Map (continued)

Offset	Use	Access	Reset value(1)	Location
	1c3			
0x28800 + n*0x0004	Adapt Values for all STATE Increments (ADT_Sn), n = 0...63 (2*(SNU+1-SYN_NS) valid entries)	R/W	0x0000_0000	See GTM Spec.(2)
	1c4			
0x28900 + m*0x0004	Nominal STATE Increment Values for FULL_SCALE (DT_Sm), m = 0...63 (2*(SNU+1-SYN_NS) valid entries)	R	0x0000_0000	See GTM Spec.(2)
0x28A00 – 0x28DFC	Reserved			
0x28E00 + n*0x0004	Calculated ACTION_n TIME STAMP (DPLL_TSACn), n = 0...23	R/W	0x0000_0000	See GTM Spec.(2)
0x28E60 - 0x28E7C	Reserved			
0x28E80 + n*0x0004	Calculated position value for action n (DPLL_PSACn), n = 0...23	R/W	0x0000_0000	See GTM Spec.(2)
0x28EE0 - 0x28EFC	Reserved			
0x28F00	control bits for actions 0...3 (DPLL_ACB_0)	R/W	0x0000_0000	See GTM Spec.(2)
0x28F04	control bits for actions 4...7 (DPLL_ACB_1)	R/W	0x0000_0000	See GTM Spec.(2)
0x28F08	control bits for actions 8....11 (DPLL_ACB_2)	R/W	0x0000_0000	See GTM Spec.(2)
0x28F0C	control bits for actions 12...15 (DPLL_ACB_3)	R/W	0x0000_0000	See GTM Spec.(2)
0x28F10	control bits for actions 16...19 (DPLL_ACB_4)	R/W	0x0000_0000	See GTM Spec.(2)
0x28F14	control bits for actions 20...23 (DPLL_ACB_5)	R/W	0x0000_0000	See GTM Spec.(2)
0x28F18 - 0x28F1C	Reserved			
0x28F20	DPLL control register 11 (DPLL_CTRL_11)	R/W	0x0000_0000	See GTM Spec.(2)
0x28F24	DPLL immediate THVAL value register (DPLL_THVAL2)	R	0x0000_0000	See GTM Spec.(2)
0x28F28	DPLL trigger input delay register (DPLL_TIDEL)	R/W	0x0000_0000	See GTM Spec.(2)
0x28F2C	DPLL STATE input delay (DPLL_SIDEL)	R/W	0x0000_0000	See GTM Spec.(2)
0x28F30	DPLL STATE time stamp field offset at synchronization time (DPLL_APS_SYNC_EXT)	R/W	0x0000_0000	See GTM Spec.(2)
0x28F34	DPLL STATE time stamp field offset at synchronization time(DPLL_CTRL_EXT)	R/W	0x0000_0017	See GTM Spec.(2)
0x28F38	Actual RAM Pointer Address for STATE (DPLL_APS_EXT)	R/W	0x0000_0000	See GTM Spec.(2)

Table 705. GTMINT Module Memory Map (continued)

Offset	Use	Access	Reset value(1)	Location
0x28F3C	Address pointer STATE for RAM region 1c3 (DPLL_APS_1C3_EXT)	R/W	0x0000_0000	See GTM Spec.(2)
0x28F40	DPLL state machine status information (DPLL_STA)	R	0x0000_0000	See GTM Spec.(2)
0x28F44	DPLL ADD_IN_ADDER1 offset for fast pulse generation (DPLL_INCF1_OFFSET)	R/W	0x0000_0000	See GTM Spec.(2)
0x28F48	DPLL ADD_IN_ADDER2 offset for fast pulse generation (DPLL_INCF2_OFFSET)	R/W	0x0000_0000	See GTM Spec.(2)
0x28F4C	DPLL first value of DPLL_DT_T_ACT for the first increment after setting SIP1 from 0 to 1 (DPLL_DT_T_START)	R/W	0x0000_0101	See GTM Spec.(2)
0x28F50	DPLL first value of DPLL_DT_S_ACT for the first increment after setting SIP2 from 0 to 1 (DPLL_DT_S_START)	R/W	0x0000_0101	See GTM Spec.(2)
0x28F54	DPLL trigger masks for signals DPLL_STA_T and DPLL_STA_S (DPLL_STA_MASK)	R/W	0x0000_0000	See GTM Spec.(2)
0x28F58	DPLL STA_T/S and INC_CNT1/2 flags (DPLL_STA_FLAG)	R/W	0x0000_0000	See GTM Spec.(2)
0x28F5C	DPLL INC_CNT1 trigger mask (DPLL_INC_CNT1_MASK)	R/W	0x0000_0000	See GTM Spec.(2)
0x28F60	DPLL INC_CNT2 trigger mask (DPLL_INC_CNT2_MASK)	R/W	0x0000_0000	See GTM Spec.(2)
0x28F64	Extension register number 1 for DPLL_NUSC (DPLL_NUSC_EXT1)	R/W	0x0001_0001	See GTM Spec.(2)
0x28F68	Extension register number 2 for DPLL_NUSC (DPLL_NUSC_EXT2)	R/W	0x0000_0001	See GTM Spec.(2)
0x28F6C	Extension register for DPLL_APS (DPLL_APS_EXT)	R/W	0x0000_0000	See GTM Spec.(2)
0x28F70	Extension register for DPLL_APS_1C3 (DPLL_APS_1C3_EXT)	R/W	0x00FF_FFFF	See GTM Spec.(2)
0x28F74	Extension register for DPLL_APS_SYNC (DPLL_APS_SYNC_EXT)	R/W	0x0000_0000	See GTM Spec.(2)
0x28F78	Extension register for DPLL_CTRL (DPLL_CTRL_EXT)	R/W	0x00FF_FFFF	See GTM Spec.(2)

Table 705. GTMINT Module Memory Map (continued)

Offset	Use	Access	Reset value[1]	Location
DPLL RAM 2 map				
	Region 2a			
0x2C000 + n*0x0004	TRIGGER nominal increment reciprocals in FULL_SCALE (RDT_Tn), n = 0 ... N-1, with N = 2*(TNU+1-SYN_NT) valid entries, and N = 128...1024 OBS: final address 2a = 0x2C000 + (N-1)*0x4	R/W	0x0000_0000	See GTM Spec.[2]
	Region 2b			
final address 2a + 0x4 + n*0x0004	Time Stamp Field of TRIGGER events (TSF_Tn), n = 0...N-1, with N = 2*(TNU+1) valid entries, and N = 128...1024 OBS: the final address 2b depends on N	R/W	0x0000_0000	See GTM Spec.[2]
	Region 2c			
final address 2b + 0x4 + m*0x0004	Adapt Values for all Increments (ADT_Tm), m = 0...M-1, with M = 2*(TNU+1-SYN_NT) valid entries, and M = 128...1024 OBS: the final address 2c depends on M	R/W	0x0000_0000	See GTM Spec.[2]
	Region 2d			
final address 2c + 0x4 + m*0x0004	nominal TRIGGER increment values for FULL_SCALE (DT_Tm), m = 0...M-1, with M = 2*(TNU+1-SYN_NT) valid entries, and M = 128...1024 OBS: the final address 2d depends on M	R/W	0x0000_0000	See GTM Spec.[2]
Multi Channel Sequencer (MCS) Configuration Registers (FFi = i*0x1000, i = 0.....4, n = 0.....7)[4]				
0x30000+n*0x80+ y*0x04 + FFi	LEG_MCS Channel n GPR registers (LEG_MCS[i]_CH[n]_R[y]), y = 0...7	R/W	0x0000_0000	See GTM Spec.[2]
0xF0000+n*0x80+ y*0x04 + FFi	MCS Channel n GPR registers (MCS[i]_CH[n]_R[y]), y = 0...7	R/W	0x0000_0000	See GTM Spec.[2]
0x30020+n*0x80 + FFi	LEG_MCS Channel n control register (LEG_MCS[i]_CH[n]_CTRL)	R/W	0x0000_0000	See GTM Spec.[2]
0xF0020+n*0x80 + FFi	MCS Channel n control register (MCS[i]_CH[n]_CTRL)	R/W	0x0000_0000	See GTM Spec.[2]
0x30024+n*0x80 + FFi	LEG_MCS Channel n ACB register (LEG_MCS[i]_CHn]_ACB)	R	0x0000_0000	See GTM Spec.[2]
0xF0024+n*0x80 + FFi	MCS Channel n ACB register (MCS[i]_CHn]_ACB)	R	0x0000_0000	See GTM Spec.[2]
0x30028 + FFi	LEG_MCS Clear trigger control register (LEG_MCS[i]_CTRG)	R/W	0x0000_0000	See GTM Spec.[2]

Table 705. GTMINT Module Memory Map (continued)

Offset	Use	Access	Reset value(1)	Location
0xF0028 + FFi	MCS Clear trigger control register (MCS[i]_CTRG)	R/W	0x0000_0000	See GTM Spec.(2)
0x3002C + FFi	LEG_MCS Set trigger control register (LEG_MCS[i]_STRG)	R/W	0x0000_0000	See GTM Spec.(2)
0xF002C + FFi	MCS Set trigger control register (MCS[i]_STRG)	R/W	0x0000_0000	See GTM Spec.(2)
(0xF0028 – 0xF002C) + n*0x80 + FFi	Reserved (n > 0)			
(0xF0030 – 0xF0038) + n*0x80 + FFi	Reserved			
0xF003C+n*0x80 + FFi	MCS high byte of memory transfer (MCS[i]_CH[n]_MHB)	R/W	0x0000_0000	See GTM Spec.(2)
0x30040+n*0x80 + FFi	LEG_MCS Channel n Program counter register (LEG_MCS[i]_CH[n]_PC)	R/W	0x0000_0000 + 4*n	See GTM Spec.(2)
0xF0040+n*0x80 + FFi	MCS Channel n Program counter register (MCS[i]_CH[n]_PC)	R/W	0x0000_0000 + 4*n	See GTM Spec.(2)
0x30044+x*0x80 + FFi	LEG_MCS Channel n interrupt notification register (LEG_MCS[i]_CH[n]_IRQ_NOTIFY)	R/W	0x0000_0000	See GTM Spec.(2)
0xF0044+x*0x80 + FFi	MCS Channel n interrupt notification register (MCS[i]_CH[n]_IRQ_NOTIFY)	R/W	0x0000_0000	See GTM Spec.(2)
0x30048+n*0x80 + FFi	LEG_MCS Channel n interrupt enable register (LEG_MCS[i]_CH[n]_IRQ_EN)	R/W	0x0000_0000	See GTM Spec.(2)
0xF0048+n*0x80 + FFi	MCS Channel n interrupt enable register (MCS[i]_CH[n]_IRQ_EN)	R/W	0x0000_0000	See GTM Spec.(2)
0x3004C+n*0x80 + FFi	LEG_MCS Channel n software interrupt generation register (LEG_MCS[i]_CH[n]_IRQ_FORCINT)	R/W	0x0000_0000	See GTM Spec.(2)
0xF004C+n*0x80 + FFi	MCS Channel n software interrupt generation register (MCS[i]_CH[n]_IRQ_FORCINT)	R/W	0x0000_0000	See GTM Spec.(2)
0x30050+n*0x80 + FFi	Legacy IRQ mode configuration register (LEG_MCS[i]_CH[n]_IRQ_MODE)	R/W	0x0000_0000	See GTM Spec.(2)
0xF0050+n*0x80 + FFi	IRQ mode configuration register (MCS[i]_CH[n]_IRQ_MODE)	R/W	0x0000_0000	See GTM Spec.(2)
0x30054+n*0x80 + FFi	LEG_MCS Channel n error interrupt enable register (LEG_MCS[i]_CH[n]_EIRQ_EN)	R/W	0x0000_0000	See GTM Spec.(2)
0xF0054+n*0x80 + FFi	MCS Channel n error interrupt enable register (MCS[i]_CH[n]_EIRQ_EN)	R/W	0x0000_0000	See GTM Spec.(2)
(0xF0058 – 0xF005C) + n*0x80 + FFi	Reserved			
0xF0060 + FFi	MCSi write protect register (MCS[i]_REG_PROT)	R/W	0x0000_0000	See GTM Spec.(2)

Table 705. GTMINT Module Memory Map (continued)

Offset	Use	Access	Reset value(1)	Location
0x30074 + FFi	LEG_MCS Control register (LEG_MCS[i]_CTRL)	R/W	0x0001_0000	See GTM Spec.(2)
0xF0064 + FFi	MCS Control register (MCS[i]_CTRL_STAT)	R/W	0x0001_0000	See GTM Spec.(2)
0x30078 + FFi	LEG_MCS Channel reset register (LEG_MCS[i]_RST)	R/W	0x0000_0000	See GTM Spec.(2)
0xF0068 + FFi	MCS Channel reset register (MCS[i]_RESET)	R/W	0x0000_0000	See GTM Spec.(2)
0xF006C + FFi	MCS cancel ARU transfer (MCS[i]_CAT)	R/W	0x0000_0000	See GTM Spec.(2)
0xF0070 + FFi	MCS cancel WURM instruction (MCS[i]_CWT)	R/W	0x0000_0000	See GTM Spec.(2)
0x3007C + FFi	LEG_MCS Error register (LEG_MCS[i]_ERR)	R/W	0x0000_0000	See GTM Spec.(2)
0xF007C + FFi	MCS Error register (MCS[i]_ERR)	R/W	0x0000_0000	See GTM Spec.(2)
(0xF0074 – 0xF007C) + n*0x80 + FFi	Reserved (n > 0)			
(0xF03D8 – 0xF0FFC) + FFi	Reserved			
0xF4000 – 0xF6FFC	Reserved for MCS4 / 5 / 6 registers			
0xF7000 – 0xF7FFC	Reserved			
MCS Memories (GGi = i*0x8000, i = 0.....4)				
(0x38000 – 0x3AFFC) + GGi	MCS Memory (MCS[i]_MEM)			See GTM Spec.(2)
(0x3B000 – 0x3FFFC) + GGi	Reserved			
0x5B000 – 0x6BFFC	Reserved for MCS 5 / 6 Memories			
0x6C000 – 0xE2000	Reserved			
Cluster Configuration Module (CCM) registers (ZZi = i*0x200, i = 0....4, n = 0.....9)				
0xE2000 + n*0x08 + ZZi	CCMi Address Range Protector n Control Register (CCM[i]_ARP[n]_CTRL)	R/W	0x0000_3000	See GTM Spec.(2)
0xE2004 + n*0x08 + ZZi	CCMi Address Range Protector n Protection Register (CCM[i]_ARP[n]_PROT)	R/W	0x0000_0000	See GTM Spec.(2)
0xE21D8 + ZZi	CCMi MCS Bus Master Status Register (CCM[i]_AEIM_STA)	R/W	0x0000_0000	See GTM Spec.(2)
0xE21DC + ZZi	CCM hardware configuration register (CCM[i]_HW_CONF) NOTE: The content of this register mirrors the content of register GTM_HW_CONF	R/W	0x284F_1339	See GTM Spec.(2)

Table 705. GTMINT Module Memory Map (continued)

Offset	Use	Access	Reset value(1)	Location
0xE21E0 + ZZi	CCMi TIM AUX input source Register (CCM[i]_TIM_AUX_IN_SRC)	R/W	0x0000_0000	See GTM Spec.(2)
0xE21E4 + ZZi	CCMi External Capture Enable Register (CCM[i]_EXT_CAP_EN)	R/W	0x0000_0000	See GTM Spec.(2)
0xE21E8 + ZZi	CCMi TOM output Register (CCM[i]_TOM_OUT) Note: content of this register mirrors the content of register GTM_TOM[i]_OUT	R/W	0x00FF_FFFF	See GTM Spec.(2)
0xE21EC + ZZi	CCMi ATOM output Register (CCM[i]_ATOM_OUT) Note:The content of this register mirrors the content of register GTM_ATOM[i]_OUT	R/W	0xFFFF_FFFF	See GTM Spec.(2)
0xE21F0 + ZZi	CCMi CMU Clock Configuration Register (CCM[i]_CMU_CLK_CFG)	R/W	0x0000_0000	See GTM Spec.(2)
0xE21F4 + ZZi	CCMi CMU Fixed Clock Configuration Register (CCM[i]_CMU_FXCLK_CFG)	R/W	0x0000_0000	See GTM Spec.(2)
0xE21F8 + ZZi	CCMi configuration Register (CCM[i]_CFG)	R/W	0x0001_00FF	See GTM Spec.(2)
0xE21FC + ZZi	CCMi Protection Register (CCM[i]_PROT)	R/W	0x0000_0001	See GTM Spec.(2)
DTM Configuration (CDTM) registers (RRi = i*0x400, i = 0..4) (j = 0,1,4,5 for i = 0,1,2 and j = 4,5 for i = 3,4)				
0xE4000 + j*0x40 + RRi	CDTMi DTMj global configuration and control register (CDTM[i]_DTM[j]_CTRL)	R/W	0x0000_0000	See GTM Spec.(2)
0xE4004 + j*0x40 + RRi	CDTMi DTMj channel control register 1 (CDTM[i]_DTM[j]_CH_CTRL1)	R/W	0x0000_0000	See GTM Spec.(2)
0xE4008 + j*0x40 + RRi	CDTMi DTMj channel control register 2 (CDTM[i]_DTM[j]_CH_CTRL2)	R/W	0x0000_0000	See GTM Spec.(2)
0xE400C + j*0x40 + RRi	CDTMi DTMj channel control register 2 shadow (CDTM[i]_DTM[j]_CH_CTRL2_SR)	R/W	0x0000_0000	See GTM Spec.(2)
0xE4010 + j*0x40 + RRi	CDTMi DTMj phase shift unit configuration and control register (CDTM[i]_DTM[j]_PS_CTRL)	R/W	0x0000_0000	See GTM Spec.(2)
0xE4014 + j*0x40 + RRi + z*0x04	CDTMi DTMj dead time reload values (z=0..3) (CDTM[i]_DTM[j]_CH[z]_DTV)	R/W	0x0000_0000	See GTM Spec.(2)
0xE4024 + j*0x40 + RRi	CDTM DTMj channel shadow register (CDTM[i]_DTM[j]_CH_SR)	R/W	0x0000_0000	See GTM Spec.(2)
0xE4028 + j*0x40 + RRi	CDTMi DTMj channel control register 3 (CDTM[i]_DTM[j]_CH_CTRL3)	R/W	0x0000_0000	See GTM Spec.(2)
(0xE402C - 0xE403C) + RRi	Reserved			
(0xE406C - 0xE40FC) + RRi	Reserved			

Table 705. GTMINT Module Memory Map (continued)

Offset	Use	Access	Reset value(1)	Location
(0xE412C - 0xE403C) + RRi	Reserved			
(0xE416C - 0xE43FC) + RRi	Reserved			
0xE496C - 0xE4CFC	Reserved			
0xE4D6C - 0xE50FC	Reserved			
0xE516C - 0xE7FFC	Reserved			

1. Default value when in reset state.
2. For detailed information see the *GTM-IP Specification* document.
3. Reset value depends on configuration parameters.
4. The old address areas of modules are marked with LEG_ (Legacy). These legacy areas are mapped to the normal areas.
5. RAM reserved position for GTM-IP internal use. No error occurs when accessed.
6. These values are device specific and will change with every device. For GTM IP 343 the value of GTM_REV for final release is 0x343314A0 and the value of ARU_CADDR_END is 0x0000_0048.

47.2.2 Register Descriptions

This section contains the description of additional registers on the top integration module. Description of GTM-IP registers, or GTMDI registers are found in their respective specification document.

47.2.2.1 GTM Module Configuration Register (GTMMCR)

The GTMMCR contains the control bits to configure the general operation of the GTMINT and GTM-IP Module.

Offset 0x000C0 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	MDIS	0	0	0	0	0	0	0	0	0	0	0	0	0	AEISREN
W																
Reset	0	—(1)	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	STPS	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	—(1)	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 564. GTM Module Configuration Register (GTMMCR)

1. Reset value is SoC dependent.

GTMMCR register functions are as shown in *Table 706*.

Table 706. GTMMCR field descriptions

Field	Description
1 MDIS	MDIS Module Disable. The MDIS bit puts the GTMINT Module in stop/disabled mode. Communication through the slave-bus Interface is ignored in this mode except reads/writes to the GTMMCR register and reads from other integration registers which are allowed. 1 Module disable request 0 Normal Mode
15 AEISREN	AEISREN AEI interface soft-reset control enable. The AEISREN bit enables the soft-reset of the AEI interface that is controlled by AEISRST bit on GTMMAEICR register.. Please, refer to *Section 47.3.7: AEI Interface Software Reset Description* for more details. 1 AEI soft-reset control is enabled 0 AEI soft-reset control is disabled
17 STPS	STPS Stop Mode Status. The STPS bit indicates the GTM-IP Module is in stop mode, or by MDIS bit field setting or by stop line request. Please, refer to *Section 47.3.4: Stop Mode Description* for more details. 1 GTM is in Stop Mode 0 Normal Mode

47.2.2.2 GTM Interrupts Clear Register (GTMINTCLR)

The GTMINTCLR contains the control bits to clear some specific interrupt request output.

Offset 0x000C8 Access: User read/write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W							INTCLR_PTR									
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 565. GTM Interrupts Clear Register (GTMINTCLR)

GTMINTCLR register functions are as shown in *Table 707*.

Table 707. GTMINTCLR field description

Field	Description
22:31 INTCLR_PTR	INTCLR_PTR Interrupt Clear Pointer. This field value indicates / selects which interrupt will be cleared. The correspondence between the number and the interrupt is given in *Table 708*. Please, refer to *Section 47.3.8: Interrupt and DMA Support Logic* for more details.

Table 708. Correspondence between the INTCLR_PTR and the interrupts outputs

INTCLR_PTR		Interrupt Output Signal to be cleared
HEX	DEC	
0x000	0	No interrupt is selected
0x001 to 0x2C1	1 to 705	Reserved
0x2C2	706	ipi_int_gtm_aei
0x2C3 to 0x2C5	707 to 709	ipi_int_gtm_aru[0] - [2]
0x2C6	710	ipi_int_gtm_brc
0x2C7	711	ipi_int_gtm_cmp
0x2C8	712	ipi_int_gtm_spe0
0x2C9	713	ipi_int_gtm_spe1
0x2CA to 0x2D1	714 to 721	ipi_int_gtm_psm0[0] - [7]
0x2D2 to 0x2EC	722 to 748	ipi_int_gtm_dpll[0] - [26]
0x2ED	749	ipi_int_gtm_err
0x2EE to 0x2F5	750 to 757	ipi_int_gtm_mcs0[0] - [7]
0x2F6 to 0x2FD	758 to 765	ipi_int_gtm_mcs1[0] - [7]
0x2FE to 0x305	766 to 773	ipi_int_gtm_mcs2[0] - [7]
0x306 to 0x30D	774 to 781	ipi_int_gtm_mcs3[0] - [7]
0x30E to 0x315	782 to 789	ipi_int_gtm_mcs4[0] - [7]
0x316 to 0x31D	790 to 797	ipi_int_gtm_tim0[0] - [7]
0x31E to 0x325	798 to 805	ipi_int_gtm_tim1[0] - [7]
0x326 to 0x32D	806 to 813	ipi_int_gtm_tim2[0] - [7]
0x32E to 0x335	814 to 821	ipi_int_gtm_tim3[0] - [7]
0x336 to 0x33D	822 to 829	ipi_int_gtm_tim4[0] - [7]
0x33E to 0x345	830 to 837	ipi_int_gtm_tom0[0] - [7]
0x346 to 0x34D	838 to 845	ipi_int_gtm_tom1[0] - [7]
0x34E to 0x355	846 to 853	ipi_int_gtm_tom2[0] - [7]
0x356 to 0x35D	854 to 861	ipi_int_gtm_tom3[0] - [7]
0x35E to 0x361	862 to 865	ipi_int_gtm_atom0[0] - [3]
0x362 to 0x365	866 to 869	ipi_int_gtm_atom1[0] - [3]
0x366 to 0x369	870 to 873	ipi_int_gtm_atom2[0] - [3]
0x36A to 0x36D	874 to 877	ipi_int_gtm_atom3[0] - [3]
0x36E to 0x371	878 to 881	ipi_int_gtm_atom4[0] - [3]
0x359 to 0x3FF	857 to 1023	Reserved

47.2.2.3 GTM AEI Control Register (GTMMAEICR)

The GTMMAEICR contains the control bits to the AEI interface in GTM-IP. GTMMAEICR register functions are as shown in *Table 709*.

Offset 0x000CC Access: User read/write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																AEISRST
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 566. GTM AEI Control Register (GTMMAEICR)

Table 709. GTMMAEICR field descriptions

Field	Description
31 AEISRST	AEISRST AEI interface soft-reset control. The AEISRST bit when set applies a reset on the GTM-IP AEI interface. The set is enabled by the AEISREN bit on GTMMCR register. This is a self clear bit and is always read as 0.. Please, refer to *Section 47.3.7: AEI Interface Software Reset Description* for more details.

47.3 Functional Description

The following sections describe the GTMINT Module functionality.

The GTMINT Module is designed to accommodate the multipurpose timer (GTM-IP) and the debugger block (GTMDI). The GTM-IP can generate timed signals and also can handle external signals to obtain their timing information. The GTMDI provides several features to help the development of programs to be run on GTM-IP. Please refer to *Figure 563* for the integration architecture.

The GTMINT Module integrates the following internal sub-modules into the SPC58xEx/SPC58xGx platform:

- GTM-IP timer
- GTMDI debug interface
- ECC error correction sub-blocks
- Interface match logic

The following sections describe each one of these sub-modules and their combined operation.

47.3.1 GTM-IP Timer

The Generic Timer Module (GTM) is composed of several submodules of different functionality. These submodules can be combined in a configurable manner to form a complex timer module that serves different application domains and different classes within one application domain. Because of this scalability and configurability the timer is called generic. The scalability and configurability is reached with an architecture philosophy where dedicated hardware submodules are located around a central routing unit called Advanced Routing Unit (ARU). The ARU can connect the submodules in a flexible manner. The connectivity is software programmable and can be configured during runtime.

Nevertheless, the GTM-IP is designed to unload the CPU or a peripheral core from a high interrupt load. Most of the tasks inside the GTM-IP can run (once set up by an external CPU) independent and in parallel to the software. There may be special situations, where the CPU has to take action but the goal of the GTM design was to reduce these situations to a minimum.. In addition, the timer section of GTM-IP runs at a fixed clock frequency (100MHz) to allow real time period counting, but the interface to CPU can work with a frequency variable clock.

47.3.1.1 GTM-IP Timer Architecture

The hardware submodules have dedicated functions, e.g., there are timer input modules where incoming signals can be captured and characterized together with a notion of time. By combination of several submodules through the ARU complex functions can be established. For example, the signals characterized at an input module can be routed to a signal processing unit where an intermediate value about the incoming signal frequency can be calculated.

The modules that help to implement such complex functions are called infrastructural components further on. These components are present in all GTM variants. However, the number of these components may vary from device to device. Other submodules have a more general architecture and can fulfill typical timer functions, e.g., there are PWM generation units. The third class of submodules are those fulfilling a dedicated functionality for a certain application domain, e.g., the DPLL serves engine management applications. A fourth group of submodules is responsible for supporting the implementation of safety functions to fulfill a defined safety level.

Each GTM-IP is build up therefore with submodules coming from those four groups. The application class is defined by the amount of components of those submodules integrated into the implemented GTM-IP.

A list of submodules is given in *Table 710* below.

Table 710. Submodules of GTM-IP

Item	Submodule	Group
1	Advanced Routing Unit (ARU)	Infrastructural components
2	Broadcast Module (BRC)	
3	First In First Out Module (FIFO)	
4	AEI-to-FIFO Data Interface (AFD)	
5	FIFO-to-ARU Interface (F2A)	
6	Clock Management Unit (CMU)	
7	Time Base Unit (TBU)	

Table 710. Submodules of GTM-IP (continued)

Item	Submodule	Group
8	Timer Input Module (TIM)	IO Modules
9	Timer Output Module (TOM)	
9	Dead Time Module (DTM)	
10	ARU-connected Timer Output Module (ATOM)	
11	Multi Channel Sequencer (MCS)	Signal generation and processing
12	Memory Configuration Module (MCFG)	Infrastructural component for MCS
13	TIM0 Input Mapping Module (MAP)	Dedicated
14	Digital PLL (DPLL)	
15	Sensor Pattern Evaluation Module (SPE)	BLDC support
16	Interrupt Concentrator Module (ICM)	Interrupt services
17	Output Compare Unit (CMP)	Safety features
18	Monitoring Unit (MON)	

For more details, please refer to the *GTM-IP Specification* document.

47.3.2 GTMDI Debug Interface

The GTM Development Interface (GTMDI) provides real-time development capabilities for the GTM system including MCS cores, ARU data trace, Input Channel Filters trace, Output Timer channels, DPLL and SPE sub-modules. The GTMDI interfaces to the GTM IP using dedicated signals. The GTMDI operates in conjunction with on-chip debug modules in order to implement complex debug features. Trace capability is also provided along with a global time-stamp which allows real time trace for several GTM sub-module events.

The GTMDI implements a Nexus Client standard interface and a JTAG standard Port Controller to provide communication capability with the JTAG and Nexus Aurora router on-chip modules. Please, note that the Nexus standard is only used for the physical interfaces. In order to implement the debug features such as breakpoint and messages required for the GTM debug, the Nexus standard could not be followed.

For detailed architecture and features description, please refer to the GTMDI chapter of this reference manual.

47.3.2.1 Peripheral Slave Bus Read Debug Access

Debugger is able to read all GTM registers, which are visible from the peripheral bus even when the timer is running. Read-access of the debugger is possible without destroying or changing the contents of these registers. A read debug access using the slave bus has no side effects on the timer. The main example is a read access to the FIFO that does not update the FIFO pointers.

47.3.3 Error Detection on GTM Memories Accesses

The GTM-IP architecture requires 14 RAM units that should be placed on the SoC. Each RAM is connected with separated input and output ports to the GTM-IP for safety purposes.

Memories of this block are equipped with error detection features for safety improvement . The following procedures detect errors on memory accesses:

- Memory write data and its address are considered in the parity bits calculation. These parity bits are stored in that memory address as part of the written data.
- Memory read data and its address are considered in the error detection. The parity bits are part of the data read from the specified memory address.
- Error Correcting Code (ECC), that corrects single errors and detects double errors on each read access

47.3.3.1 ECC Description

The Error Correcting Code (ECC) for GTM RAMs provides guaranteed single bit error correction and double bit error detection (without correction) per accessed word. On write accesses, the parity bits are calculated and appended to the original data bits to be written into the specified memory address. And error detection and correction are performed on read accesses.

In case of error detection (with correction or not), no write operation is done here to fix possible errors in the memory content.

Note: All memories must be initialized with write operations in all addresses because the ECC parity bits need to be stored in each memory address before any read operation.

Note: GTM-IP sub-block does this initialization automatically for all memories, except for FIFO0 RAM. In this case, the user must be careful to execute writes before reads.

Table 711 presents a summary of GTM memories and the necessary ECCs for single error correction / double error detection.

Table 711. ECC Characteristics for GTM Memories

Memory	MEM Number of Words	MEM Address Width (bits)	Uncoded Data Width (bits)	ECC (bits)	MEM Data In/Out (Data + ECC) (bits)	Matrix Size (N.o Synd.)
FIFO0 RAM	1024	10	29	7	36	46
DPLL RAM1A	96	7	24	7	31	38
DPLL RAM1B	384	9	24	7	31	40
DPLL RAM2	2048	11	24	7	31	42
MCS0 RAM0	2048	11	32	7	39	50
MCS1 RAM0	2048	11	32	7	39	50
MCS2 RAM0	2048	11	32	7	39	50
MCS3 RAM0	2048	11	32	7	39	50
MCS4 RAM0	2048	11	32	7	39	50
MCS0 RAM1	1024	10	32	7	39	49
MCS1 RAM1	1024	10	32	7	39	49
MCS2 RAM1	1024	10	32	7	39	49

Table 711. ECC Characteristics for GTM Memories (continued)

Memory	MEM Number of Words	MEM Address Width (bits)	Uncoded Data Width (bits)	ECC (bits)	MEM Data In/Out (Data + ECC) (bits)	Matrix Size (N.o Synd.)
MCS3 RAM1	1024	10	32	7	39	49
MCS4 RAM1	1024	10	32	7	39	49

47.3.3.1.1 Error Correcting Code (ECC) and Syndrome Definition

This ECC is based on the Hsiao code with 7 parity bits for each data word.

Table 712 shows the definition of the syndrome field for each bit position. Note that the ECC data vector width can be different for each memory, so we consider only the necessary positions for each one, not all the table positions.

Table 712. 42 bits Word Based Syndrome Definition

SYND[6:0] / Element Value (hex)	Error Result (for SMATRIX)	Matrix Element (for MATRIX)	SYND[6:0] / Element Value (hex)	Error Result (for SMATRIX)	Matrix Element (for MATRIX)
40	error on **code** bit 6	-	54	error on data bit 23	element 23
20	error on **code** bit 5	-	15	error on data bit 22	element 22
10	error on **code** bit 4	-	16	error on data bit 21	element 21
08	error on **code** bit 3	-	34	error on data bit 20	element 20
04	error on **code** bit 2	-	25	error on data bit 19	element 19
02	error on **code** bit 1	-	26	error on data bit 18	element 18
01	error on **code** bit 0	-	-	-	-
3D	error on data(1) bit 43	element 43	-	-	-
6E	error on data(1) bit 42	element 42	64	error on data bit 17	element 17
4F-> 79	error on data(1) bit 41	element 41	2C	error on data bit 16	element 16
3E-> 37	error on data bit 40	element 40	1A	error on data bit 15	element 15
3D-> 5B	error on data bit 39	element 39	23	error on data bit 14	element 14
3B-> 76	error on data bit 38	element 38	2A	error on data bit 13	element 13

Table 712. 42 bits Word Based Syndrome Definition (continued)

SYND[6:0] / Element Value (hex)	Error Result (for SMATRIX)	Matrix Element (for MATRIX)	SYND[6:0] / Element Value (hex)	Error Result (for SMATRIX)	Matrix Element (for MATRIX)
37-> 5D	error on data bit 37	element 37	32	error on data bit 12	element 12
2F	error on data bit 36	element 36	46	error on data bit 11	element 11
1F	error on data bit 35	element 35	4A	error on data bit 10	element 10
70	error on data bit 34	element 34	52	error on data bit 9	element 9
68	error on data bit 33	element 33	62	error on data bit 8	element 8
07	error on data bit 32	element 32	13	error on data bit 7	element 7
49	error on data bit 31	element 31	29	error on data bit 6	element 6
0D	error on data bit 30	element 30	31	error on data bit 5	element 5
0E	error on data bit 29	element 29	43	error on data bit 4	element 4
38	error on data bit 28	element 28	45	error on data bit 3	element 3
4C	error on data bit 27	element 27	19	error on data bit 2	element 2
1C	error on data bit 26	element 26	51	error on data bit 1	element 1
58	error on data bit 25	element 25	61	error on data bit 0	element 0
0B	error on data bit 24	element 24	-	-	-
-	-	-	-	-	-
00	no error	-	any other(2)	non-correctable error	-

1. Actually the data can be data or address bit.
2. For the FIFO0 RAM with 29 data bits and 10 address bits, the valid error positions are from bit 0 to bit 38, only. The syndromes with values for bit 39 to 41 of the table are considered also for non-correctable errors.

47.3.3.2 Error Report Description

Each RAM generates a defined number of information to report an error on an access. Each of them is described below.

- Error Address - this is the mapped address of the RAM as defined in *Section 47.2.1: Module Memory Map*. This field has only meaning when an error occurs and is flagged by the two error signals below. It is reported on read or write operations.
- Non-correctable Error - this one bit width error indicates that:
  - The ECC detected errors in the read access and it is not possible to correct them because the ECC can only fix single bit error. The error can occur in the RAM read data bus (parity bits + data) or in the address value that the RAM has received.
- Correctable Error - this one bit width error indicates that:
  - The ECC detected and corrected a single bit error on the RAM read data bus (parity bits + data). The parity bits are discarded after the ECC decoder but this error report includes these bits to obtain the performance of the whole RAM data storage sector..

This Error Report remains active for only one RAM clock cycle and should be captured by the external RAM Error Monitor module.These error indications are also available for the GTM-IP timer module that is the block that requests the RAM accesses. Uncorrectable ECC error indication are provided to GTM-IP but the reaction to the error depends on this block itself. Please refer to its specification for more details.

47.3.4 Stop Mode Description

The stop mode is characterized by the stopping of the functional clock and the timer configuration registers clock. Therefore, the timers become stopped and almost all registers access are not allowed. This mode can be requested by two ways:

- By asserting the bit field MDIS in the GTMMCR configuration register.
- By receiving the SoC stop request signal ipg_stop.
  - the minimum waiting time to a new stop request is about 20 bus clock cycles.

However, it is important to note that to confirm the stop request was granted, it is necessary to read the stop status bit field STPS in the GTMMCR register.

47.3.4.1 Stop Mode by MDIS

The stop mode is requested by asserting this bit field with a write operation on the GTMMCR mapped register. The module only requests to stop the clocks when no slave bus accesses are pending in the GTM-IP. When in stop mode:

- The clocks are disabled to feed the whole GTM-IP and peripheral functions like the ECC
- The stop status bit field STPS in GTMMCR register is set.
- Read or write accesses to all mapped registers are not allowed and generates an access error indication, except accesses for the GTMMCR register.
- The slave bus clock is not affected by stop mode.

This stop mode ends by clearing the MDIS bit field.. The STPS stop status bit is automatically cleared after a while (it is considered that the SoC stop request is not set).

47.3.5 Debug / Halt Mode Description

The debug / halt mode is requested via GTMDI and is used to debug the GTM-IP sub-block. In this mode it is required that the core of GTM-IP becomes halted by stopping its clock. Therefore all time bases and counters are freezed. In earlier version of GTM IP, its core clock (ipg_clk_gtm) was provided separately from its system clock (ipg_clk). GTM IP 343 takes only ipg_clk as input and generates its core clock internally from this clock. It also handles the gating of core clock based on GTM IP Halt request input.

Debug or halt mode requests should be enabled by setting the bit field DBE on the GTMDI GTMDI_DC register that is accessible by the JTAG interface (refer to GTMDI block guide for more details). GTM halt is only activated when no read- or write-access to GTM over AEI are taking place in parallel. Only in this condition the GTM is released to change to debug / halt mode read.

It is important to notice that any peripheral slave bus access is active in this mode . Debugger is able to read all GTM addresses read-/writeable from the peripheral bus when GTM is running or halted. All registers in the register map can be accessed by the peripheral bus, even the RAMs because they are in the register map. Of course, the clocks for the RAMs are not stopped in debug mode. GTM IP also provides an additional output signal (gtm_restore) which indicates restore phase of functional memory accesses after any GTM Halt phase.

For more details about the debug mode, please refer to GTMDI documentation.

Note: If this module is in disable mode by setting the MDIS bit at the GTMMCR register, the debug request is not considered until the MDIS bit is cleared.

47.3.6 GTM-IP Timer Software Reset Description

The GTM-IP timer portion has the software reset possibility by writing a 1 in the RST bit field on the GTM_RST register.. This bit is self cleared and is always read as zero. In addition, this bit is write protected by the bit field RF_PROT in the GTM_CTRL register.

This is a global reset to GTM-IP block only. The GTMDI is not affected by this soft-reset.

47.3.7 AEI Interface Software Reset Description

The GTM-IP has software reset control for the AEI interface sub-block only. This block is the slave bus interface of GTM-IP and is not reset by the GTM-IP timer soft-reset. The AEI soft-reset occurs by writing a 1 in the AEISRST bit field on the GTMMAEICR register. This bit is self cleared and is always read as zero. In addition, this bit is write protected by the bit field AEISREN in the GTMMCR register.

A soft-reset in the AEI interface is equivalent to the hard-reset. It occurs in the next clock cycle of the request and takes one bus clock cycle. The AEI buffer and configuration registers are cleared, and some slave bus access can be aborted. Therefore, access commands in the buffer memory are lost and the configuration registers of the AEI interface change to their reset values.

47.3.8 Interrupt and DMA Support Logic

The GTM-IP provides several interrupts / DMA request signals to signalize internal events to other external modules. The list of all signals are provided in the GTM-IP documentation and all have the same type of controls and configuration options.

Each interrupt / DMA request line represents a set of internal interrupt / DMA request signals, and each of this internal signal has its own set of status / control / configuration registers: IRQ_EN, IRQ_NOTIFY, IRQ_FORCINT, and IRQ_MODE registers. Please, refer to *GTM-IP specification* for more details.

It is required to maintain the configuration register IRQ_MODE=00 that is the Level Interrupt Mode, the default mode to be compatible with the chip standard. the IRQ_NOTIFY register is the internal flag and can be enabled to set an external signal by the IRQ_EN register. For more details, please refer to the GTM-IP documentation.

The GTM integration module provides the same GTM-IP set of signals as interrupt requests and as DMA requests and the real functionality on the chip can be configured later. However, if a signal is configured to start a DMA request, the request can be cleared by the respective DMA answer, but it can be also cleared by software as occurs for interrupt requests.

47.3.8.1 Interrupt Request Clear Controls

Each interrupt request output can be a composition of several internal events flags that are enabled by configuration registers. After an interrupt request is recognized by the chip interrupt controller and it is serviced, the interrupt routine must clear the flags that originated that service request. The GTM integration module provides two ways to clear those flags, but each method has its limitations that are described below.

The first way to clear the events flags that originated the interrupt request is by a slave bus write operation in the interrupt mapped registers named *_IRQ_NOTIFY in *Table 705*. The main characteristics are listed below:

- For example, writing a 1 in the bit field DEST_ERR on the register BRC_IRQ_NOTIFY is going to clear this bit to clear the ipi_int_gtm_brc interrupt output signal
- Several flags in this register could have generated the interrupt request, therefore all of them can be cleared with this write operation.
- There is a delay between the reception of the clear command and the actual clear of the flags. This time interval can vary and depends on the number of commands inside the AEI buffer. Therefore it is not possible to know easily when the flags were cleared.
- This method of interrupt clear can generate a new service request if we are using write masked response mode for AEI interface.

The second way to clear the events flags of the interrupt requests is by using the mapped register GTMINTCLR that is described in *Section 47.2.2.2: GTM Interrupts Clear Register (GTMINTCLR)*. The main characteristics of this method are listed below:

- For example, to clear the ipi_int_gtm_brc interrupt output signal, we execute a write operation in the GTMINTCLR register with data value 0x05 in the INTCLR_PTR bit field. This value selects to execute a clear in all flags that are used to generate that interrupt request.
- There is a delay between the reception of the clear command and the actual clear of the flags. This time interval can vary and depends on the number of commands inside the AEI buffer. However, the flags clear instant is well defined and can be seem by the integration module.
- As this method clear all flags for an interrupt request output, some flag that is set during the interrupt service is also cleared, and then some event can be lost.

47.3.9 Clock Enables

GTM IP 343 provides additional set of clock enable outputs related to each cluster clock (gtm_clsx_clken) and for FIFO, DPLL and MCS RAM clocks.

For current version 343 of GTM IP, all the clusters (0 to 4) are working at same clock frequency. But in future version, all clusters can operate at different clock frequencies. So these clock enables are provided using which it can be checked if a cluster is operating at ipg_clk , ipg_clk/2 or its clock is off. For GTM version 343 these clock enables will only be used for indicating if ipg_clk is on or off.

GTM IP also provides clock enable outputs for all the RAMs (FIFO, DPLL or MCS). These RAMs are incorporated at SOC level outside GTM wrapper. Currently these clock enables are also not used for GTM IP 343.

47.3.10 ADC Interface

ADC read interface signals are provided from GTM IP to SOC through GTM wrapper. For detailed information on this interface, please refer to GTM IP Specifications document.

47.4 Initialization/Application Information

The GTM integration module receives two active low reset signals, one on the slave bus clock domain, and the other in the peripheral clock domain. The last one resets all the timer logic, and registers inside the GTM-IP block. Just after this reset signal is negated, the GTM-IP is in the following condition:

- The GTM bridge is in asynchronous mode.
- after reset, write operations waits the response status (write response is not masked).
- If MDIS_DEFAULT is 1, a write access to GTMMCR is necessary to clear the MDIS bit register and have the GTM-IP clocks toggling.
- DPLL RAMs and MCSs RAMs starts the write initialization cycles. This is necessary for ECC parity bits storage on each RAM address.
- FIFO0 RAM is not automatically initialized. In normal operation mode, the initialization is not required because the FIFO memory always have to written first to be read later.

Refer to GTM-IP and GTMDI documentation to initialize the GTMINT module.

48 Generic Timer Module (GTM343)

48.1 About this module

GTM343 is a programmable timer that can perform complex timing and I/O management independently of the chip's processor cores.

48.2 For more information

For more information on GTM343, see the Generic Timer Module 343 (GTM343) documentation set.

49 GTM Development Interface (GTMDI)

49.1 Introduction

The GTMDI debug module unit is a hardware module that:

- Monitors the software and hardware activities of GTM-IP
- Read the registers without interfering with the logic status, even for FIFOs, by configuring the debug access controls

49.2 Overview

Figure 567 shows a block diagram of the Generic Timer Module (GTM) Development Interface.

The GTM Development Interface (GTMDI) provides real-time development capabilities for the GTM system including Multi-Channel Sequencer (MCS) cores, Advanced Routing Unit (ARU) data trace, input channel filters trace, output timer channels, DPLL and Sensor Pattern Evaluation (SPE) sub-modules. The GTMDI interfaces to the GTM IP using dedicated signals. The GTMDI operates in conjunction with on-chip debug modules to implement complex debug features. Trace capability is also provided along with a global time-stamp which allows real time trace for several GTM sub-module events.

The GTMDI implements a Nexus Client standard interface and a JTAG standard Port Controller to provide communication capability with the JTAG and NAR on-chip modules. The Nexus standard is only used for the physical interfaces. Due to specific requirements for the GTM debug, the Nexus standard is not followed.

Figure 567. GTMDI block diagram

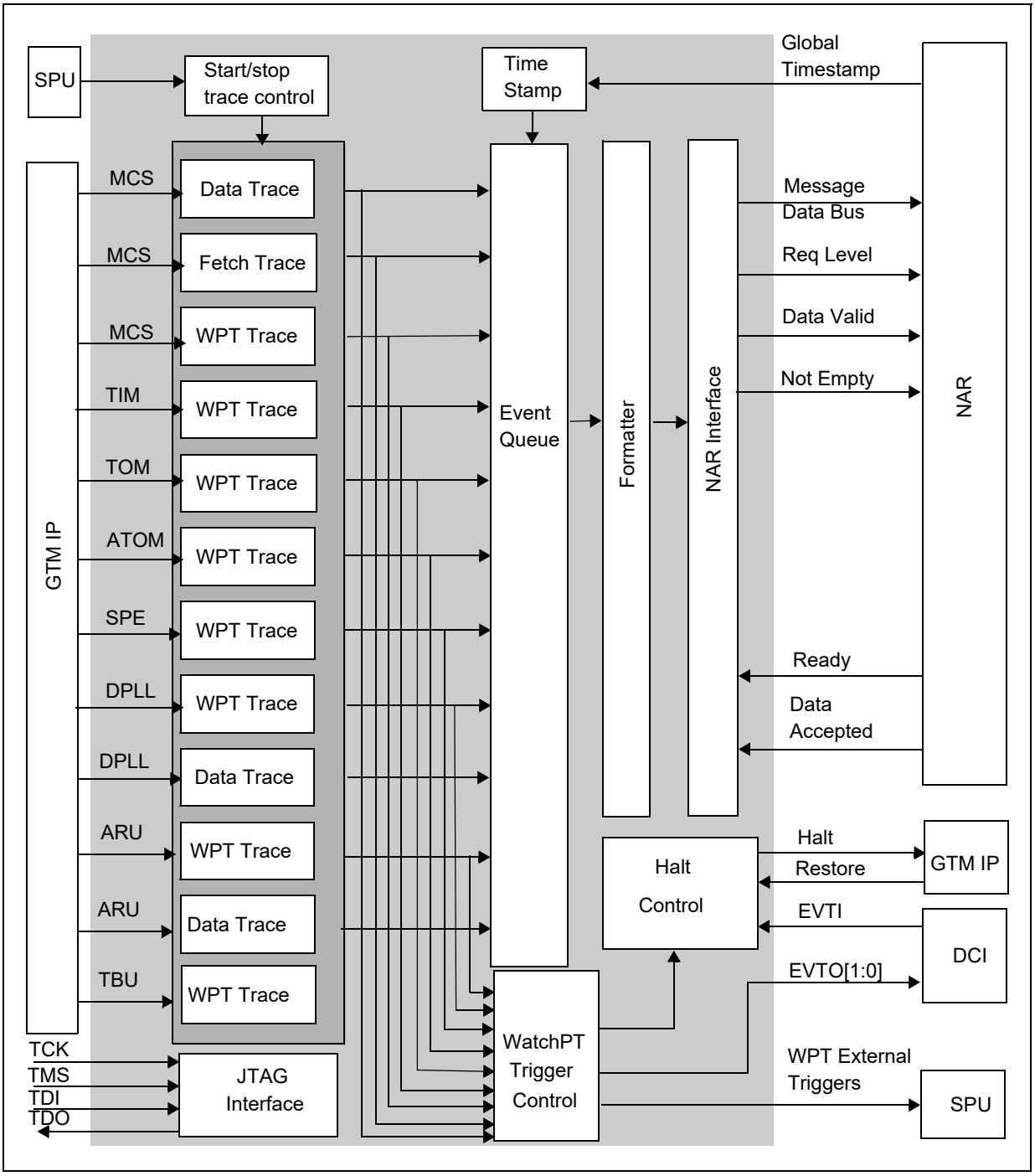
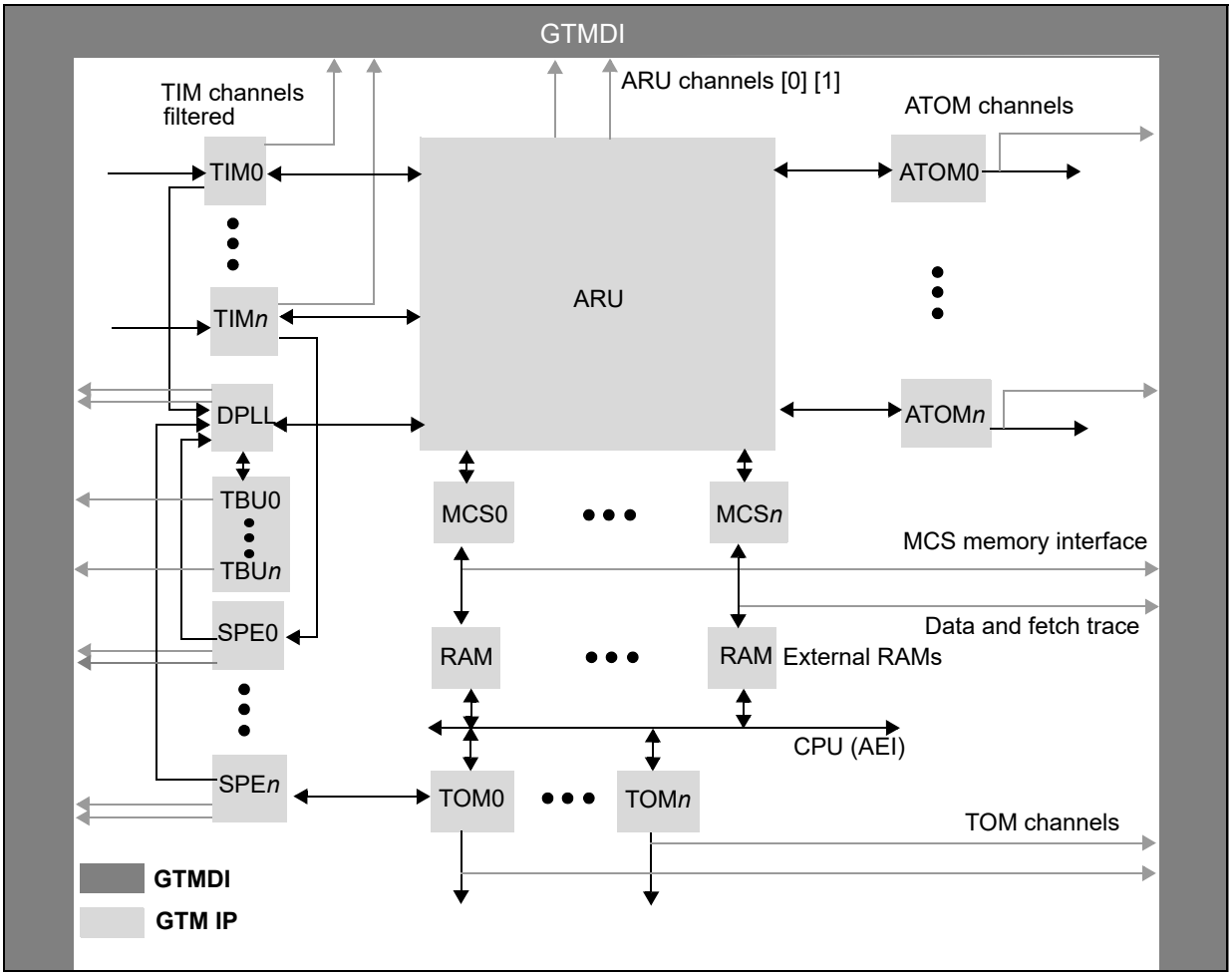


Figure 568 describes the interconnection between the GTMDI and GTM modules. The GTMDI selects which signal or bus to monitor since it does not have the capability to monitor all signals in parallel. The multiplexing architecture is described in the following sections.

Figure 568. GTMDI and GTM interface



Note: Figure 568 does not show all of the GTM internal modules. It shows only the modules that interface to the GTMDI debug module.

The number of ATOM, TIM, TOM, MCS and SPE modules in *Figure 568* is shown in *Table 713*.

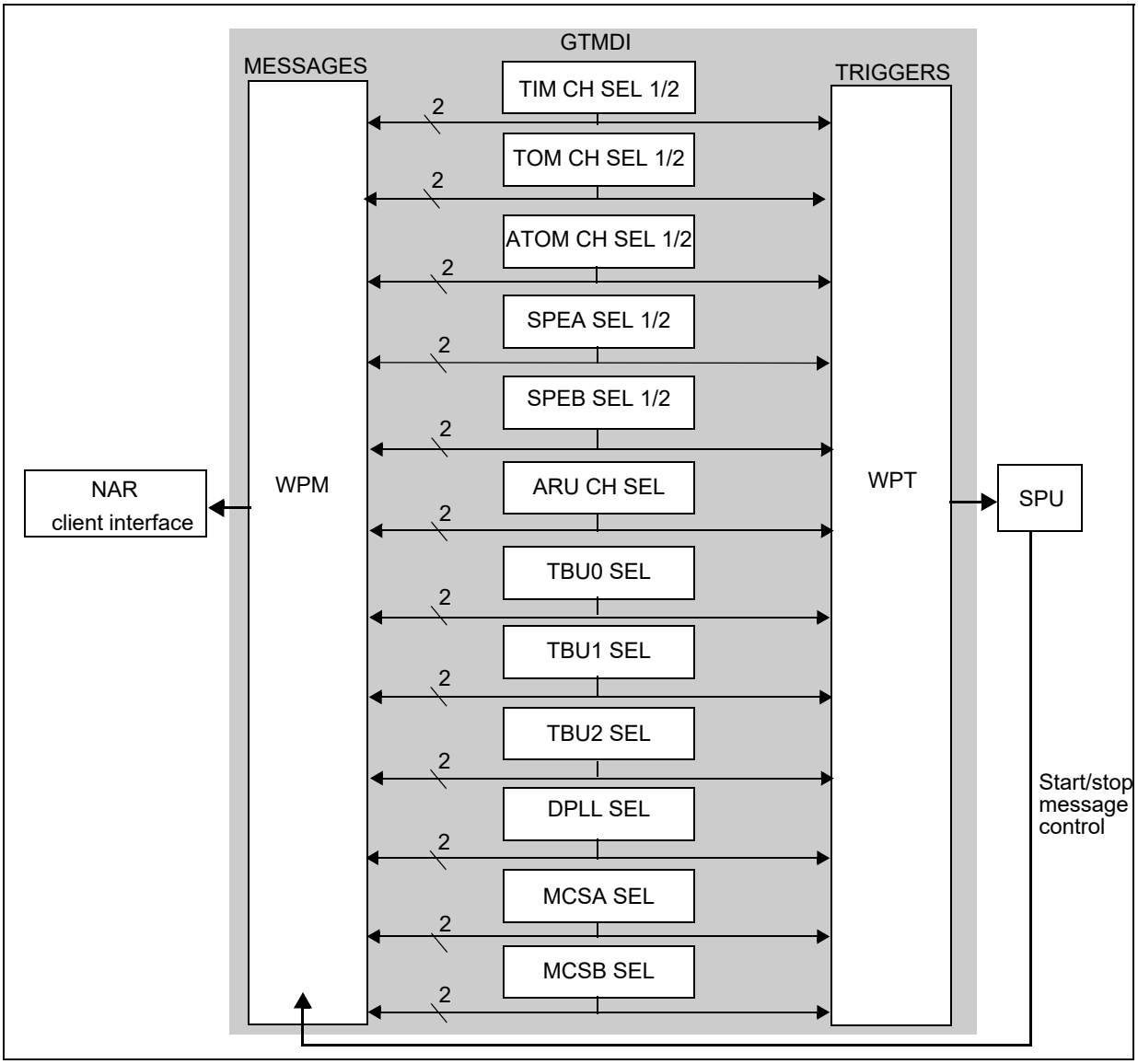
Table 713. GTM Sub-module instances

Module	Number of instances
ATOM	5
TIM	5
TOM	4
MCS	5
SPE	2

Figure 569 shows the Watchpoint Triggers (WPT) and Watchpoint Messages (WPM). Each GTM sub-module generates two watchpoints which can be routed internally to the GTMDI and generate watchpoint triggers or routed to the message formatter and generate watchpoint messages. The selection between watchpoint triggers and messages is

implemented per sub-module. Messages and watchpoint triggers can also be generated at the same time based on the same event within a sub-module, thus a TIM channel transition can generate a watchpoint trigger and a watchpoint message. The watchpoint triggers may be sent to the Sequence Processing Unit (SPU) outside the GTMDI if enabled by appropriate control register bits for each sub-module, such as Timer Input Module (TIM), Timer Output Module (TOM) and ARU-connected Timer Output Module (ATOM). The watchpoint messages, if enabled, are sent to the NAR through the Nexus Client interface.

Figure 569. Watchpoint triggers and watchpoint messages



The SPU provides external control of the watchpoint messages generated by the GTMDI thus controlling the bandwidth over the Nexus Client interface. This functionality is implemented through Start/Stop control signals from the SPU. These signals are connected to the sub-module register bit that enables the message transmission implementing a dual control over these registers, either by JTAG interface programing or by SPU Start/Stop

signals. See *Section 49.5: Functional description* for more details about register control and general architecture.

49.2.1 Features

The GTMDI block implements the following features:

- Full duplex pin interface for medium and high visibility throughput
  - 2 $\overline{\text{EVTO}}$ (Event Out) for GTM real time signal monitoring
  - 1 $\overline{\text{EVTI}}$ (Event In)
  - IEEE 1149.1 (JTAG) Test Access Port (TAP)

    4 pins (TDI, TDO, TMS, and TCK)

    Reset input $\overline{\text{TRST}}$ driven by either the Nexus Port Controller or an external pin
- GTM Development Support
  - Read/Write registers in debug mode from JTAG port
  - Ability to enter debug mode at reset negation or during normal execution

Note: Enter debug out-of-reset is provided through the SoC system level debug control signal. In order to allow this functionality the GDE and SDBE control bits in the GTMDI_DC register should be set.

  - Halt and Watchpoint Configuration
  - MCS Data Trace Messaging (DTM) and Watchpoint Messaging (WPM) allows the development tool to trace reads and writes to selected MCS RAM address ranges

    Two data trace windows for each MCS core with programmable address ranges and access attributes are provided. Data trace windowing reduces the requirement on the Aurora port bandwidth by constraining the number of trace locations.

    Fetch trace capability controlled by two programmable Watchpoints.

    Private messages are used to indicate special MCS core cases not covered by public messages. Two MCS cores can be debugged at the same time.

Note: Enabling two MCS cores at the same time for data trace can exceed the capacity of the NAR Client interface throughput thus information can be lost.

  - DPLL Data Trace Messaging and Watchpoint Trace Messaging allows the development tool to trace reads and writes to selected DPLL RAM addresses and selected data value.
  - ARU Data Trace Messaging and Watchpoint Trace Messaging allows the development tool to trace selected ARU channels.
  - Watchpoint Messaging (WPM)
- Trigger Control
  - The GTMDI provides selection of trigger signals that generate debug actions at system level such as Halt GTM itself or cores in the same device.
  - Trace enable signals are provided to control the start/stop of trace messages for each one of the GTM Nexus message sources, such as Timer Input Module (TIM), Timer Output Module (TOM), ARU-connected Timer Output Module (ATOM).
- GTMDI clock operation
  - The module is able to operate in any frequency ratio considering GTMDI and JTAG frequency. The GTMDI operates from the same clock source as the GTM IP module.

49.2.2 Modes of operation

The GTMDI block is placed in reset when the Nexus reset signal is asserted. The JTAG registers controlled by the TCK clock are also put into the reset state if the TAPcontroller is in the TEST-LOGIC-RESET state. Holding TMS high for five consecutive rising edges of TCK guarantees entry into the TEST-LOGIC-RESET state regardless of the initial TAP controller state. Asserting the Nexus reset signal *nex_reset_b* results in asynchronous entry into the reset state not only of the TCK controlled registers but all registers in the GTMDI. The GTMDI is unaffected by other sources of reset. While in reset, the following actions occur:

- The TAP controller is forced into the TEST-LOGIC-RESET state
- The auxiliary output port pins are negated
- The auxiliary output port enable outputs are negated
- The TDI, TMS, and TCK TAP inputs are ignored (The TMS and TCK signal are ignored only while the Nexus reset signal *nex_reset_b* is asserted.)
- Registers default back to their reset values

Note: The GTMDI is not reset by the system reset signal. It is reset during device Power On reset or by a dedicated Nexus reset thus preserving internal registers state and programming during system reset.

Debug Mode can be controlled by on-chip logic through the system debug signal. The level on this signal defines if the GTM module is in debug mode, meaning that it is in Halt state and accepts write and read accesses to internal registers that do not modify the state of those registers. If asserted, the GTM enters Halt state. If negated, the GTM debug mode is controlled by other sources, internal to the GTMDI. The GTM may enter debug out of reset by asserting the system debug signal while the device is being reset. By detecting this condition, the GTMDI puts the GTM into the Halt state. The GDE and SDBE bits described in *Figure 571* must be set during reset allowing the GTM to enter debug state.

49.2.3 DATA port

GTMDI messages are sent through the Message Data Bus. The Message Data Bus has a fixed width of 32 bits which defines one BEAT of data. There are three BEATS per NAR clock thus 96 bits(MDO[89:0] + $\overline{\text{MSEO}}$[5:0]) are transmitted in parallel in the DATA port.

49.3 External signal description

The GTMDI pinout provides interface for the transmission of messages and for accessing Nexus registers. The GTMDI pin definition is outlined in *Table 714*.

Table 714. GTMDI signal properties

Name	Port	Function	Reset State
EVTI	Auxiliary	Event In pin	—
$\overline{\text{EVTO}}$[1:0]	Auxiliary	Event Out pin	[11]
TCK	TAP	Test Clock Input	—
TDI	TAP	Test Data Input	—
TDO	TAP	Test Data Output	0
TMS	TAP	Test Mode Select Input	—
TRST	TAP	Test Reset Input	—

Each of the signals listed in *Table 714* is described in more detail in the following sections.

49.3.1 Event in ($\overline{\text{EVTI}}$)

Event In ($\overline{\text{EVTI}}$) is used to generate a Halt if enabled by the corresponding control bit. $\overline{\text{EVTI}}$ is an edge-sensitive signal. It is synchronized at the GTMDI input and a falling edge is detected.

49.3.2 Event out $\overline{\text{EVTO}}$[1:0]

Event Out ($\overline{\text{EVTO}}$) are two outputs from GTMDI for development tools with exact timing for a two watchpoints or Halt occurrence. $\overline{\text{EVTO}}$ is asserted for one system clock cycle when Halt or watchpoints are monitored. When internal GTM signals are monitored, such as TIM filtered signals or TOM/ATOM outputs, the $\overline{\text{EVTO}}$ represent the exact signal shape with a certain delay. In this monitoring mode the signal is not inverted through the $\overline{\text{EVTO}}$ output. The real time monitoring mode is intended for oscilloscope type of monitoring thus allowing the user to evaluate the actual shape of the signals. The source selection for the $\overline{\text{EVTO}}$ is centralized in the Development Control register GTMDI_DC.

49.3.3 Test clock input (TCK)

Test Clock Input (TCK) is an input pin used to synchronize the test logic and control register access through the JTAG port.

49.3.4 Test data input (TDI)

Test Data Input (TDI) is an input pin that receives serial test instructions and data. TDI is sampled on the rising edge of TCK.

49.3.5 Test data output (TDO)

Test Data Output (TDO) is an output pin that transmits serial output for test instructions and data. TDO is actively driven in the SHIFT-IR and SHIFT-DR states of the JTAG state machine. TDO changes on the falling edge of TCK and is sampled on the rising edge of TCK.

49.3.6 Test Mode Select (TMS)

Test Mode Select (TMS) is an input pin used to sequence the JTAG state machine. TMS is sampled on the rising edge of TCK.

49.4 Register definition

This section provides a detailed description of all GTMDI registers accessible to the development tool. Individual bit-level descriptions and reset states of each register are included. *Table 715* shows the GTMDI registers and index values. These registers are not memory-mapped and can only be accessed via the JTAG interface.

Table 715. GTMDI registers

Index	Register	Location
Register list		
0	Device Identity (DID)	*Section 49.4.1.1*
1–4	Reserved	
5	GTMDI Development Control (GTMDI_DC)	*Section 49.4.1.2*
6	GTMDI Development Status (GTMDI_DS)	*Section 49.4.1.3*
7–10	Reserved	
11	TIM Watchpoint Control 1 (GTMDI_TIM_WPC1)	*Section 49.4.1.4*
12	TIM Watchpoint Control 2 (GTMDI_TIM_WPC2)	*Section 49.4.1.5*
13–16	Reserved	
17	TOM Watchpoint Control 1 (GTMDI_TOM_WPC1)	*Section 49.4.1.6*
18	TOM Watchpoint Control 2 (GTMDI_TOM_WPC2)	*Section 49.4.1.7*
19–22	Reserved	
23	ATOM Watchpoint Control 1 (GTMDI_ATOM_WPC1)	*Section 49.4.1.8*
24	ATOM Watchpoint Control 2 (GTMDI_ATOM_WPC2)	*Section 49.4.1.9*
25–28	Reserved	
29	SPEA Watchpoint Control 1 (GTMDI_SPEA_WPC1)	*Section 49.4.1.10*
30	SPEA Watchpoint Control 2 (GTMDI_SPEA_WPC2)	*Section 49.4.1.11*
31–34	Reserved	
35	SPEB Watchpoint Control 1 (GTMDI_SPEB_WPC1)	*Section 49.4.1.12*
36	SPEB Watchpoint Control 2 (GTMDI_SPEB_WPC2)	*Section 49.4.1.13*
37–40	Reserved	
41	DPLL Development Control (GTMDI_DPLL_WPC1)	*Section 49.4.1.14*
42	DPLL Watchpoint Control 1 (GTMDI_DPLL_WPC2)	*Section 49.4.1.15*
43	DPLL Watchpoint Control 2 (GTMDI_DPLL_WPC3)	*Section 49.4.1.16*
44	DPLL Watchpoint Control 3 (GTMDI_DPLL_WPC4)	*Section 49.4.1.17*
45	DPLL Watchpoint Control 4 (GTMDI_DPLL_WPC5)	*Section 49.4.1.18*

Table 715. GTMDI registers (continued)

Index	Register	Location
46	DPLL Data Trace Control (GTMDI_DPLL_DTC)	*Section 49.4.1.19*
47–49	Reserved	
50	ARU Watchpoint Control (GTMDI_ARU_WPC1)	*Section 49.4.1.20*
51	ARU Watchpoint Control (GTMDI_ARU_WPC2)	*Section 49.4.1.21*
52	ARU DATA0H (GTMDI_ARU_DATA0H)	*Section 49.4.1.22*
53	ARU DATA0L (GTMDI_ARU_DATA0L)	*Section 49.4.1.23*
54	ARU DATA1H (GTMDI_ARU_DATA1H)	*Section 49.4.1.24*
55	ARU DATA1L (GTMDI_ARU_DATA1L)	*Section 49.4.1.25*
56	ARU Data Trace Control (GTMDI_ARU_DTC)	*Section 49.4.1.26*
57–59	Reserved	
60	MCSA Development Control (GTMDI_MCSA_DC)	*Section 49.4.1.27*
61	Reserved	
62	MCSA Watchpoint Control (GTMDI_MCSA_WPC)	*Section 49.4.1.28*
63	MCSA Program Fetch Trace Control (GTMDI_MCSA_PTC)	*Section 49.4.1.29*
64	MCSA Data Trace Control (GTMDI_MCSA_DTC)	*Section 49.4.1.30*
65	MCSA Watchpoint Address 1 (GTMDI_MCSA_WPA1)	*Section 49.4.1.31*
66	MCSA Watchpoint Address 2 (GTMDI_MCSA_WPA2)	*Section 49.4.1.32*
67	MCSA Watchpoint Data 1 (GTMDI_MCSA_WPD1)	*Section 49.4.1.33*
68	MCSA Watchpoint Data 2 (GTMDI_MCSA_WPD2)	*Section 49.4.1.34*
69	MCSA Program Trace Channel Enable (GTMDI_MCSA_CE)	*Section 49.4.1.35*
70	Reserved	
71	MCSA Data Trace Address Range 1 (GTMDI_MCSA_DTAR1)	*Section 49.4.1.36*
72	MCSA Data Trace Address Range 2 (GTMDI_MCSA_DTAR2)	*Section 49.4.1.37*
73–76	Reserved	
77	MCSB Development Control (GTMDI_MCSB_DC)	*Section 49.4.1.38*
78	Reserved	
79	MCSB Watchpoint Control (GTMDI_MCSB_WPC)	*Section 49.4.1.39*
80	MCSB Program Fetch Control (GTMDI_MCSB_PTC)	*Section 49.4.1.40*
81	MCSB Data Trace Control (GTMDI_MCSB_DTC)	*Section 49.4.1.41*
82	MCSB Watchpoint Address 1 (GTMDI_MCSB_WPA1)	*Section 49.4.1.42*
83	MCSB Watchpoint Address 2 (GTMDI_MCSB_WPA2)	*Section 49.4.1.43*
84	MCSB Watchpoint Data 1 (GTMDI_MCSB_WPD1)	*Section 49.4.1.44*
85	MCSB Watchpoint Data 2 (GTMDI_MCSB_WPD2)	*Section 49.4.1.45*
86	MCSB Program Trace Channel Enable (GTMDI_MCSB_CE)	*Section 49.4.1.46*
87	Reserved	

Table 715. GTMDI registers (continued)

Index	Register	Location
88	MCSB Data Trace Address Range 1 (GTMDI_MCSB_DTAR1)	*Section 49.4.1.47*
89	MCSB Data Trace Address Range 2 (GTMDI_MCSB_DTAR2)	*Section 49.4.1.48*
90–93	Reserved	
94	TBU0 Watchpoint Control 1 (GTMDI_TBU0_WPC1)	*Section 49.4.1.49*
95	TBU0 Watchpoint Control 2 (GTMDI_TBU0_WPC2)	*Section 49.4.1.50*
96	TBU0 DATA (GTMDI_TBU0_DATA)	*Section 49.4.1.51*
97–100	Reserved	
101	TBU1 Watchpoint Control 1 (GTMDI_TBU1_WPC1)	*Section 49.4.1.52*
102	TBU1 Watchpoint Control 2 (GTMDI_TBU1_WPC2)	*Section 49.4.1.53*
103	TBU1 DATA (GTMDI_TBU1_DATA)	*Section 49.4.1.54*
104–107	Reserved	
108	TBU2 Watchpoint Control 1 (GTMDI_TBU2_WPC1)	*Section 49.4.1.55*
109	TBU2 Watchpoint Control 2 (GTMDI_TBU2_WPC2)	*Section 49.4.1.56*
110	TBU2 DATA (GTMDI_TBU2_DATA)	*Section 49.4.1.57*

49.4.1 Register descriptions

This section consists of GTMDI register descriptions. The registers are shown in *Table 715*. The MCS cores selected for debugging are named MCSA and MCSB thus two MCS cores can be debugged at the same time. See registers GTMDI_MCSA_DC and GTMDI_MCSB_DC which specify the real number of the MCS associated with MCSA and MCSB.

49.4.1.1 Device identity register (DID)

The DID register contains the unique identification information as shown in *Figure 570*.

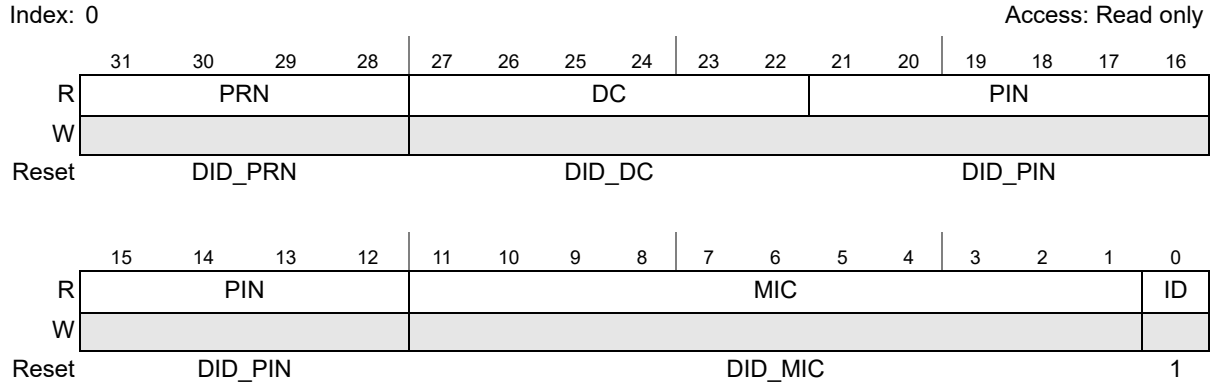


Figure 570. Device identity register (DID)

Table 716 describes the device identification register functions.

Table 716. DID field descriptions

Field	Description
31:28 PRN	Part Revision Number. Contains the revision number of the part.
27:22 DC	Design Center. Indicates the design center.
21:12 PIN	Part Identification Number. Contains the part number of the device. Value is 0x3E3.
11:1 MIC	Manufacturer Identity Code. Contains the reduced Joint Electron Device Engineering Council (JEDEC) ID. Value is 0x020.
0 ID	IDCODE Register ID. Identifies this register as the device identification register and not the bypass register. Always set to 1.

49.4.1.2 GTMDI development control register (GTMDI_DC)

The GTMDI_DC register controls various trace and debug features of the GTM modules as described in *Figure 571*.

Index: 5 Access: Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	GDE	0	0	0	EOS0				0	0	0	0	EOS1			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	EIDR	0	DBE	DBR	SDBE	0	TSFT
W								CHR								
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 571. GTMDI development control register (GTMDI_DC)

Table 717 describes the GTMDI_DC register functions.

Table 717. GTMDI_DC field descriptions

Field	Description
31 GDE	Global Debug Enable. This control bit disables all debug features of the GTMDI module including ability to put GTM IP in Halt state, modify EVTO output, send messages to the DATA bus and set triggers to the SPU module. Before changing the GTMDI configuration it is recommended to clear GDE bit to prevent spurious behavior of the module outputs. See *Figure 627* for details about debug enable logic. 0 GTMDI Global debug is disabled 1 GTMDI Global debug is enabled
27:24 EOS0	$\overline{\text{EVTO0}}$ Output Selection. This signal is dedicated to sel 1 which is the selection 1 from the indicated GTM modules. See the control registers for each module for a description of this selection. For TIM, TOM and ATOM selections the input filtered signal or output channel signal are used instead of the triggers. 0000 No event on output $\overline{\text{EVTO0}}$ remains at one 0001 Halt Indication, If $\overline{\text{EVTO0}}$ is zero indicates that GTM is halted 0010 TIM, Map EVTO0 from TIM channel filter output using ssel 1 selection control 0011 TOM, Map EVTO0 from TOM channel output using ssel1 selection control 0100 ATOM, Map EVTO0 from ATOM channel output using ssel 1 selection control 0101 SPEA, Map EVTO0 from SPEA, controlled by TSS1 described in *Figure 579* 0110 SPEB, Map EVTO0 from SPEB, controlled by TSS1 described in *Figure 581* 0111 ARU CH0, Map EVTO0 from ARU CH0I 1000 DPLL, Map EVTO0 from DPLL according to TSEL1 in *Figure 583* 1001 MCSA, Map EVTO0 from MCSA MCSA_sel selection control 1010 MCSB, Map EVTO0 from MCSB MCSB_sel selection control 1011 TBU0 Watchpoint 1, Watchpoint controlled by TSS1 in GTMDI_TBU0_WPC1 register 1100 TBU1 watchpoint 1, Watchpoint controlled by TSS1 in GTMDI_TBU1_WPC1 register 1101 TBU2 watchpoint 1, Watchpoint controlled by TSS1 in GTMDI_TBU2_WPC1 register 1110 Reserved 1111 Reserved
19:16 EOS1	$\overline{\text{EVTO1}}$ Output Selection. This signal is dedicated to sel 2 which is the selection 2 from the indicated GTM modules. See the control registers for each module for a description of this selection. For TIM, TOM and ATOM selections the input filtered signal or output channel signal are used instead of the triggers. 0000 No event on output $\overline{\text{EVTO1}}$ remains at one 0001 Halt Indication, If $\overline{\text{EVTO1}}$ is zero indicates that GTM is halted 0010 TIM, Map EVTO1 from TIM channel filter output using ssel 1 selection control 0011 TOM, Map EVTO1 from TOM channel output using ssel1 selection control 0100 ATOM, Map EVTO1 from ATOM channel output using ssel 1 selection control 0101 SPEA, Map EVTO1 from SPEA, controlled by TSS2 described in *Figure 579* 0110 SPEB, Map EVTO1 from SPEB, controlled by TSS2 described in *Figure 581* 0111 ARU CH1, Map EVTO1 from ARU CH1 1000 DPLL, Map EVTO1 from DPLL according to TSEL2A/B in *Figure 583* 1001 MCSA, Map EVTO1 from MCSA MCSA_sel selection control 1010 MCSB, Map EVTO1 from MCSB MCSB_sel selection control 1011 TBU0 Watchpoint 2, Watchpoint controlled by TSS2 in GTMDI_TBU0_WPC1 register 1100 TBU1 watchpoint 2, Watchpoint controlled by TSS2 in GTMDI_TBU1_WPC1 register 1101 TBU2 watchpoint 2, Watchpoint controlled by TSS2 in GTMDI_TBU2_WPC1 register 1110 Reserved 1111 Reserved

Table 717. GTMDI_DC field descriptions (continued)

Field	Description
8 CHR	Clear Halt Request. Writing this bit to one clears halt requests., which means GTM resumes normal operation out of halt state. The status bits of the GTMDI_DS register are cleared. This bit is self-cleared thus read always as zero. 0 No action 1 Clear halt requests
6 EIDR	EVTI Debug Request enable. The EIDR bit enables EVTI input to request the GTM to enter Halt state. After the GTM enters Halt state due to a pulse in EVTI signal, the EIDR bit state is not meaningful, thus it may be cleared and the GTM continues in Halt state. See *Figure 627* for details about debug enable logic. 0 Disable EVTI to control GTM to enter Halt state 1 Enable EVTI to control GTM to enter Halt state
4 DBE	Debug Enable. DBE enables debug mode thus allowing the GTM to enter debug mode which means halt regular operation. The debug events controlled by this register are the ones generated internally to the GTMDI, excluding ipg_debug and EVTI which have their dedicated enable control. See *Figure 627* for details about debug enable logic. 0 Debug mode disabled 1 Debug mode enabled
3 DBR	Debug Request. Mechanism to allow development tool to request modules to enter debug mode directly. See *Figure 627* for details about debug enable logic. 0 Does not modify GTM IP debug state 1 Request GTM IP to enter debug mode
2 SDBE	System Debug Enable. The SDBE bit enables the GTM to enter halt state if system level debug enable signal is asserted. This bit allows the debug controller to select specific sub-groups of cores or modules to halt, while others continue to operate. GDE bit must also be set to allow GTM to halt. See *Figure 627* for details about debug enable logic. 0 GTM does not enter halt state due to system debug enable signal assertion 1 GTM is enabled to enter halt state due to system debug enable signal assertion
0 TSFT	Timestamp Force Transmission Forces the transmission of the Timestamp value along with the Nexus message data for all messages transmitted by the GTMDI Client module. When Timestamp inclusion is not forced into the client message, its inclusion depends upon the NAR settings. 0 Does not force transmission of Timestamp value 1 Forces transmission of Timestamp along with Nexus message

49.4.1.3 GTMDI development status register (GTMDI_DS)

The GTMDI_DS register described in *Figure 572* shows the status of various conditions that impact GTM development support. All status bits are dynamic and do not require clearing. Debug Status Event is queued due to a change in the Halt or Stop modes and a Debug Status Message is generated as soon as there is an opportunity in the NAR interface. If any status bit changes before the Status message is sent, this change is reflected in the message. The GTMDI_DS register dynamically tracks the debug condition status even when several of them are simultaneously set.

Index: 6 Access: Read only

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	HLT	0	0	HS1												
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	STP	0	0	HS2												
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 572. GTMDI development status register (GTMDI_DS)

Table 718 describes the GTMDI_DS register functions.

Note: The HS1 and HS2 bit fields are cleared as soon as the GTM resumes normal operation which is controlled by the CHR bit in the GTMDI_DC register. One exception is HS1[0] that indicates system debug request, which is not cleared by CHR. This bit is cleared when system debug request is negated.

Table 718. GTMDI_DS field descriptions

Field	Description
31 HLT	Halt Indication. The HLT bit indicates if the GTM IP is in Halt state, ready to receive debug accesses. 0 GTM is running 1 GTM is in Halt state (GTM Halt or Restore is high)
28:16 HS1	Halt Status1. The HS1 field shows which source generated the halt condition. More than one source can generate the halt condition at the same time. This register is only meaningful if the GTM is halted. Its content indicates the halt sources that occurred immediately when the halt mode is entered. 0000000000000 No Trigger generated 1xxxxxxxxxxxx TIM x1xxxxxxxxxxx TOM xx1xxxxxxxxxx ATOM xxx1xxxxxxxxx SPEA xxxx1xxxxxxxx SPEB xxxxx1xxxxxxx ARU xxxxxx1xxxxxx DPLL xxxxxxx1xxxxx MCSA xxxxxxxx1xxxx MCSB xxxxxxxxx1xxx TBU0 Watchpoint 1 xxxxxxxxxx1xx TBU1 Watchpoint 1 xxxxxxxxxxx1x TBU2 Watchpoint 1 xxxxxxxxxxxx1 External through system debug enable signal

Table 718. GTMDI_DS field descriptions (continued)

Field	Description
15 STP	Stop Status. Indicates if the GTM is in stop mode with no active clocks. Besides the system level stop mode mechanism, the GTM may independently be programmed for STOP mode by setting the Module Disable Control bit. Thus, the STP field indicates if the GTM is in stop mode no matter if due to the system stop mode assertion or due to the MDIS bit in the GTM module control register. 0 GTM not stopped 1 GTM is stopped in low power mode
12:0 HS2	Halt Status2. Similarly to HS1 this field indicates the source for the halt condition selection 2. See *Figure 628* for reference about source selection 1 and 2. The HS2 field shows which source generated the halt condition 2. More than one source can generate the halt condition at the same time. This register is only meaningful if the GTM is halted. Its content indicates the halt condition source that occurred when the halt mode is entered. 0000000000000 No Trigger generated 1xxxxxxxxxxxTIM x1xxxxxxxxxxTOM xx1xxxxxxxxxATOM xxx1xxxxxxxxSPEA xxxx1xxxxxxxSPEB xxxxx1xxxxxxARU xxxxxx1xxxxxDPLL xxxxxxx1xxxxx MCSA xxxxxxxx1xxxxMCSB xxxxxxxxx1xxx TBU0 Watchpoint 2 xxxxxxxxxx1xx TBU1 Watchpoint 2 xxxxxxxxxxx1x TBU2 Watchpoint 2 xxxxxxxxxxxx1 $\overline{\text{EVTI}}$

49.4.1.4 TIM watchpoint control register 1 (GTMDI_TIM_WPC1)

The GTMDI_TIM_WPC1 register shown in *Figure 573* controls the Watchpoint Triggers WPT issued by TIM channels. These triggers are used by the GTM internal modules in order to control GTM Halt mode entering and also at SoC level. See *Figure 628* which describes the use of this register.

Note: Start/Stop signal inputs to the GTMDI can overwrite the state of the WMC1 and WMC2 control bits thus controlling the Watchpoint messages behavior. In case of JTAG modifies the WMC1 or WMC2 bits and Start/Stop signals occur at the same time also modifying the same bits, Start/Stop signals have precedence over JTAG. The SEN1 and SEN2 control bits from the GTMDI_TIM_WPC2 register need to be set in order to allow external Start/Stop signals to control the WMC1/2 bits.

Index: 11 Access: Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	TSS1		0	SSEL1			HEN1	WMC1	0	0	0	CHSEL1		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	TSS2		0	SSEL2			HEN2	WMC2	0	0	0	CHSEL2		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 573. TIM watchpoint control 1 register (GTMDI_TIM_WPC1)

Table 719 describes the GTMDI_TIM_WPC1 register functions.

Table 719. GTMDI_TIM_WPC1 field descriptions

Field	Description
29:28 TSS1	TIM Channel Slope Selection 1. The TSS1 field selects the slope for the TIM selected channel. 00 Any transition (both) 01 Transition from 0 to 1 (positive slope) 10 Transition from 1 to zero (negative slope) 11 Reserved
26:24 SSEL1	TIM sub-module Source Selection 1. Selects which TIM sub-module is the source for watchpoint generation. 000 TIM0 001 TIM1 010 TIM2 011 TIM3 100 TIM4 101 Reserved 110 Reserved 111 Reserved
23 HEN1	Halt Enable 1. The HEN1 Halt Enable 1 bit controls if the selected TIM Channel is used to halt the GTM module. If this bit is set and an event occurs on the selected TIM Channel filter output, a Halt signal is generated to the GTM which enters debug state. 0 Disables Halt GTM 1 Enables Halt GTM
22 WMC1	Watchpoint Message Control 1. The WMC1 Watchpoint Message Control 1 bit controls the watchpoint message sent through the Message Data bus. If this bit is set it enables valid transitions on the selected TIM Filter output to generate a Watchpoint message. This bit is written by the JTAG interface but can also be controlled at SoC level with dedicated Start/Stop signals. 0 Watchpoint Messages disabled 1 Watchpoint Messages enabled

Table 719. GTMDI_TIM_WPC1 field descriptions (continued)

Field	Description
18:16 CHSEL1	TIM channel selection 1. Selects which channel within a selected TIM sub-module generates watchpoints. The selection is actually on the TIM channel input filter. 000 TIM CH0 001 TIM CH1 010 TIM CH2 011 TIM CH3 100 TIM CH4 101 TIM CH5 110 TIM CH6 111 TIM CH7
13:12 TSS2	TIM Channel Slope Selection 2. The TSS2 field selects the slope for the TIM selected channel. 00 Any transition (both) 01 Transition from 0 to 1 (positive slope) 10 Transition from 1 to zero (negative slope) 11 Reserved
10:8 SSEL2	TIM sub-module Source Selection 2. Selects which TIM sub-module is the source for watchpoint generation. 000 TIM0 001 TIM1 010 TIM2 011 TIM3 100 TIM4 101 Reserved 110 Reserved 111 Reserved
7 HEN2	Halt Enable 2. The HEN1 Halt Enable 2 bit controls if the selected TIM Channel is used to halt the GTM module. If this bit is set and an event occurs on the selected TIM Channel filter output, a Halt signal is generated to the GTM which enters debug state. 0 Disables Halt GTM 1 Enables Halt GTM
6 WMC2	Watchpoint Message Control 2. The WMC1 Watchpoint Message Control 2 bit controls the watchpoint message sent through the Message Data bus. If this bit is set it enables valid transitions on the selected TIM Filter output to generate a Watchpoint message. This bit is written by the JTAG interface but can also be controlled at SoC level with dedicated Start/Stop signals. 0 Watchpoint Messages disabled 1 Watchpoint Messages enabled
2:0 CHSEL2	TIM channel selection 2. Selects which channel within a selected TIM sub-module generates watchpoints. The selection is actually on the TIM channel input filter. 000 TIM CH0 001 TIM CH1 010 TIM CH2 011 TIM CH3 100 TIM CH4 101 TIM CH5 110 TIM CH6 111 TIM CH7

49.4.1.5 TIM watchpoint control register 2 (GTMDI_TIM_WPC2)

The GTMDI_TIM_WPC2 register shown in *Figure 574* controls the Watchpoint Triggers WPT issued by TIM channels. These watchpoint triggers are used to control GTM Halt state entering, watchpoint messages and control external logic to the GTMDI. See *Figure 628* for details of the logic that uses the bits in this register.

Index: 12 Access: Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	WTSEL1			0	0	0	0	0	0	STSEL1		0	0	TEN1	SEN1
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	WTSEL2			0	0	0	0	0	0	STSEL2		0	0	TEN2	SEN2
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 574. TIM watchpoint control 2 register (GTMDI_TIM_WPC2)

Table 720 describes the GTMDI_TIM_WPC2 register functions.

Note: The SHARED_WPT and SHARED_WPT_ST signals mentioned in Table 720 and in other tables in this document, refer to the GTMDI top level connections. They correspond to Watchpoint trigger outputs of GTMDI and Stop/Start input signals respectively.

Table 720. GTMDI_TIM_WPC2 field descriptions

Field	Description
30:28 WTSEL1	Watchpoint Trigger Output Selection 1. The WTSEL1[2:0] bit field selects among 8 available output Watchpoint triggers which one is connected to the selected TIM Channel. Other modules can select the same WTSEL1 value, in this case the result watchpoint signal implements an OR of all selected sources. 000 SHARED_WPT[0] 001 SHARED_WPT[1] 010 SHARED_WPT[2] 011 SHARED_WPT[3] 100 SHARED_WPT[4] 101 SHARED_WPT[5] 110 SHARED_WPT[6] 111 SHARED_WPT[7]
21:20 STSEL1	Start/Stop input signal selection 1. Selects the Start/Stop input that is used to control the watchpoint trace messages. These signals are actually a pair of signals. 00 SHARED_WPT_ST[0] 01 SHARED_WPT_ST[1] 10 SHARED_WPT_ST[2] 11 SHARED_WPT_ST[3]
17 TEN1	Watchpoint Trigger Enable 1. The TEN1 field controls if a Watchpoint external trigger is issued. This enable refers to the output selected by the WTSEL1 field. 0 Disable Watchpoint Trigger Generation 1 Enable Watchpoint Trigger Generation

Table 720. GTMDI_TIM_WPC2 field descriptions (continued)

Field	Description
16 SEN1	Start/Stop Enable 1. The SEN1 bit field enables the selected TIM channel to consider the Start/Stop inputs for watchpoint trace messages control. If this bit is cleared, the messages are controlled only by JTAG writes to the WMC1 field in the GTMDI_TIM_WPC1 register. 0 Disable trace control from Start/Stop inputs 1 Enable trace control from Start/Stop inputs
14:12 WTSEL2	Watchpoint Trigger Output Selection 2. The WTSEL1[2:0] bit field selects among 8 available output Watchpoint triggers which one is connected to the selected TIM Channel. Other modules can select the same watchpoint trigger, in this case the result signal implements an OR of all selected sources. 000 SHARED_WPT[0] 001 SHARED_WPT[1] 010 SHARED_WPT[2] 011 SHARED_WPT[3] 100 SHARED_WPT[4] 101 SHARED_WPT[5] 110 SHARED_WPT[6] 111 SHARED_WPT[7]
5:4 STSEL2	Start/Stop input signal selection 2. Selects the Start/Stop input that is used to control the watchpoint trace messages for the selected TIM channel. These signals are actually a pair of signals. 00 SHARED_WPT_ST[0] 01 SHARED_WPT_ST[1] 10 SHARED_WPT_ST[2] 11 SHARED_WPT_ST[3]
1 TEN2	Watchpoint Trigger Enable 2. The TEN2 field controls if a Watchpoint external trigger is issued by the selected TIM channel. This enable refers to the output selected by the WTSEL2 field. 0 Disable Trigger Generation 1 Enable Trigger Generation
0 SEN2	Start/Stop Enable 2. The SEN2 bit field enables the Start/Stop inputs for watchpoint trace messages control. If this bit is cleared, the messages are controlled only by JTAG writes to the WMC2 field in the GTMDI_TIM_WPC1 register. 0 Disable trace control from Start/Stop inputs 1 Enable trace control from Start/Stop inputs

49.4.1.6 TOM watchpoint control register 1 (GTMDI_TOM_WPC1)

The GTMDI_TOM_WPC1 register shown in *Figure 575* controls the Watchpoint Triggers issued by TOM channels. These Watchpoint triggers are used to control the GTM requests to enter Halt, issue watchpoint messages and also by logic external to the GTMDI at SoC level.

Note: Start/Stop signal inputs to the GTMDI can overwrite the state of the WMC1 and WMC2 control bits thus controlling the Watchpoint messages behavior. In case of JTAG modifies the WMC1 or WMC2 bits and Start/Stop signals occur at the same time also modifying the same bits, Start/Stop signals have precedence over JTAGJTAG writes. The SEN1 and SEN2 control bits from the GTMDI_TOM_WPC2 register need to be set in order to allow external Start/Stop signals to control the WMC1/2 bits.

Index: 17 Access: Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	TSS1		0	SSEL1			HEN1	WMC1	0	0	CHSEL1			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	TSS2		0	SSEL2			HEN2	WMC2	0	0	CHSEL2			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 575. TOM watchpoint control 1 register (GTMDI_TOM_WPC1)

Table 721 describes the GTMDI_TOM_WPC1 register functions.

Table 721. GTMDI_TOM_WPC1 field descriptions

Field	Description
29:28 TSS1	TOM Channel Slope Selection 1. The TSS1 field selects the slope for the TOM selected channel. 00 Any transition (both) 01 Transition from 0 to 1 (positive slope) 10 Transition from 1 to zero (negative slope) 11 Reserved
26:24 SSEL1	TOM sub-module Source Selection 1. Selects which TOM sub-module is the source for watchpoint generation. 000 TOM0 001 TOM1 010 TOM2 011 TOM3 100 Reserved 101 Reserved 110 Reserved 111 Reserved
23 HEN1	Halt Enable 1. The HEN1 Halt Enable 1 bit controls if the selected TOM Channel is used to halt the GTM module. If this bit is set and an event occurs on the selected TOM Channel filter output, a Halt signal is generated to the GTM which enters debug state. 0 Disables Halt GTM 1 Enables Halt GTM
22 WMC1	Watchpoint Message Control 1. The WMC1 Watchpoint Message Control 1 bit controls the watchpoint message sent through the Message Data bus. If this bit is set it enables valid transitions on the selected TOM Channel output to generate a watchpoint message. This bit is written by the JTAG interface but can also be controlled at SoC level with dedicated Start/Stop signal inputs to GTMDI. 0 Watchpoint Messages disabled 1 Watchpoint Messages enabled

Table 721. GTMDI_TOM_WPC1 field descriptions (continued)

Field	Description
19:16 CHSEL1	TOM channel selection 1. Selects which channel within a selected TOM sub-module generates watchpoints. 0000 TOM CH0 0001 TOM CH1 0010 TOM CH2 0011 TOM CH3 0100 TOM CH4 0101 TOM CH5 0110 TOM CH6 0111 TOM CH7 1000 TOM CH8 1001 TOM CH9 1010 TOM CH10 1011 TOM CH11 1100 TOM CH12 1101 TOM CH13 1110 TOM CH14 1111 TOM CH15
13:12 TSS2	TOM Channel Slope Selection 2. The TSS2 field selects the slope for the TOM selected channel. 00 Any transition (both) 01 Transition from 0 to 1 (positive slope) 10 Transition from 1 to zero (negative slope) 11 Reserved
10:8 SSEL2	TOM sub-module Source Selection 2. Selects which TOM sub-module is the source for watchpoint generation. 000 TOM0 001 TOM1 010 TOM2 011 TOM3 100 Reserved 101 Reserved 110 Reserved 111 Reserved
7 HEN2	Halt Enable 2. The HEN1 Halt Enable 2 bit controls if the selected TOM Channel is used to halt the GTM module. If this bit is set and an event occurs on the selected TOM Channel filter output, a Halt signal is generated to the GTM which enters debug state. 0 Disables Halt GTM 1 Enables Halt GTM

Table 721. GTMDI_TOM_WPC1 field descriptions (continued)

Field	Description
6 WMC2	Watchpoint Message Control 2. The WMC2 Watchpoint Message Control 2 bit controls the watchpoint message sent through the Message Data bus. If this bit is set it enables valid transitions on the selected TOM Channel output to generate a watchpoint message. This bit is written by the JTAG interface but can also be controlled at SoC level with dedicated Start and Stop signals inputs to the GTMDI. 0 Watchpoint Messages disabled 1 Watchpoint Messages enabled
3:0 CHSEL2	TOM channel selection 2. The CHSEL2[2:0] field selects a channel within the TOM selected sub-module to generate watchpoints. 0000 TOM CH0 0001 TOM CH1 0010 TOM CH2 0011 TOM CH3 0100 TOM CH4 0101 TOM CH5 0110 TOM CH6 0111 TOM CH7 1000 TOM CH8 1001 TOM CH9 1010 TOM CH10 1011 TOM CH11 1100 TOM CH12 1101 TOM CH13 1110 TOM CH14 1111 TOM CH15

49.4.1.7 TOM watchpoint control 2 register (GTMDI_TOM_WPC2)

The GTMDI_TOM_WPC2 register shown in *Figure 576* controls the Watchpoint Triggers issued by TOM channels. These watchpoints are used to force GTM into Halt state, generate watchpoint messages and control logic external to the GTMDI.

Index: 18 Access: Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	WTSEL1			0	0	0	0	0	0	STSEL1		0	0	TEN1	SEN1
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	WTSEL2			0	0	0	0	0	0	STSEL2		0	0	TEN2	SEN2
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 576. TOM watchpoint control 2 register (GTMDI_TOM_WPC2)

Table 722 describes the GTMDI_TOM_WPC2 register functions.

Table 722. GTMDI_TOM_WPC2 field descriptions

Field	Description
30:28 WTSEL1	Watchpoint Trigger Output Selection 1. The WTSEL1[2:0] bit field selects among 8 available output Watchpoint triggers which one is connected to the selected TOM Channel. Other modules can select the same WTSEL1 value, in this case the result watchpoint signal implements an OR of all selected sources. 000 SHARED_WPT[0] 001 SHARED_WPT[1] 010 SHARED_WPT[2] 011 SHARED_WPT[3] 100 SHARED_WPT[4] 101 SHARED_WPT[5] 110 SHARED_WPT[6] 111 SHARED_WPT[7]
21:20 STSEL1	Start/Stop input signal selection 1. Selects the Start/Stop input that is used to control the watchpoint trace messages. These signals are actually a pair of signals. 00 SHARED_WPT_ST[0] 01 SHARED_WPT_ST[1] 10 SHARED_WPT_ST[2] 11 SHARED_WPT_ST[3]
17 TEN1	Watchpoint Trigger Enable 1. The TEN1 field controls if a Watchpoint external trigger is issued by the TOM Channel output selected by WTSEL1. 0 Disable Trigger Generation 1 Enable Trigger Generation
16 SEN1	Start/Stop Enable 1. The SEN1 bit field enables the selected TOM channel to consider the Start/Stop inputs for watchpoint trace messages control. If this bit is cleared, the messages are controlled only by JTAG writes to the WMC1 field in the GTMDI_TOM_WPC1 register. 0 Disable trace control from Start/Stop inputs 1 Enable trace control from Start/Stop inputs
14:12 WTSEL2	Watchpoint Trigger Output Selection 2. The WTSEL2[2:0] field selects among 8 available Watchpoint trigger outputs which one is connected to the TOM selected channel. Other modules can select the same WTSEL2 value, in this case the result signal is an OR of all selected sources. 000 SHARED_WPT[0] 001 SHARED_WPT[1] 010 SHARED_WPT[2] 011 SHARED_WPT[3] 100 SHARED_WPT[4] 101 SHARED_WPT[5] 110 SHARED_WPT[6] 111 SHARED_WPT[7]
5:4 STSEL2	Start/Stop input signal selection 2. Selects the Start/Stop input that is used for the watchpoint message control. These signals are actually a pair of signals. 00 SHARED_WPT_ST[0] 01 SHARED_WPT_ST[1] 10 SHARED_WPT_ST[2] 11 SHARED_WPT_ST[3]

Table 722. GTMDI_TOM_WPC2 field descriptions (continued)

Field	Description
1 TEN2	Watchpoint Trigger Enable 2. The TEN2 field controls if a Watchpoint external trigger is issued by the selected TOM channel. 0 Disable Trigger Generation 1 Enable Trigger Generation
0 SEN2	Start/Stop Enable 2. The SEN2 bit field enables the Start/Stop inputs of the GTMDI to control Watchpoint trace messages generated by the TOM selected channel. If this bit is cleared, the messages are controlled only by JTAG writes to the WMC2 field in the GTMDI_TOM_WPC1 register. 0 Disable trace control from Start/Stop inputs 1 Enable trace control from Start/Stop inputs

49.4.1.8 ATOM watchpoint control 1 register (GTMDI_ATOM_WPC1)

The GTMDI_ATOM_WPC1 register shown in *Figure 577* controls the Watchpoint Triggers issued by ATOM channels. These watchpoint triggers are used to control the requests for GTM to enter Halt state, issue watchpoint messages and also to control logic external to the GTMDI at SoC level such as in the SPU interface.

Note: Start/Stop signal inputs to the GTMDI can overwrite the state of the WMC1 and WMC2 control bits thus controlling the Watchpoint messages behavior. In case of JTAG modifies the WMC1 or WMC2 bits and Start/Stop signals occur at the same time also modifying the same bits, Start/Stop signals have precedence over JTAG writes. The SEN1 and SEN2 control bits from the GTMDI_ATOM_WPC2 register need to be set in order to allow external Start/Stop signals to control the WMC1/2 bits.

Index: 23 Access: Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	TSS1		SSEL1				HEN1	WMC1	0	0	0	CHSEL1		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	TSS2		SSEL2				HEN2	WMC2	0	0	0	CHSEL2		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 577. ATOM watchpoint control 1 register (GTMDI_ATOM_WPC1)

Table 723 describes the GTMDI_ATOM_WPC1 register functions.

Table 723. GTMDI_ATOM_WPC1 field descriptions

Field	Description
29:28 TSS1	ATOM Channel Slope Selection 1. The TSS1 field selects the slope for the ATOM selected channel. 00 Any transition (both) 01 Transition from 0 to 1 (positive slope) 10 Transition from 1 to zero (negative slope) 11 Reserved
27:24 SSEL1	ATOM sub-module Source Selection 1. Selects which ATOM sub-module is the source for watchpoint generation. 0000 ATOM0 0001 ATOM1 0010 ATOM2 0011 ATOM3 0100 ATOM4 0101 Reserved 0110 Reserved 0111 Reserved 1000 Reserved 1001 Reserved 1010 Reserved 1011 Reserved 1100 Reserved 1101 Reserved 1110 Reserved 1111 Reserved
23 HEN1	Halt Enable 1. The HEN1 Halt Enable 1 bit controls if the selected ATOM Channel is used to halt the GTM module. If this bit is set and an event occurs on the selected ATOM Channel filter output, a Halt signal is generated to the GTM which enters debug state. 0 Disables Halt GTM 1 Enables Halt GTM
22 WMC1	Watchpoint Message Control 1. The WMC1 Watchpoint Message Control 1 bit controls the watchpoint message sent through the Message Data bus. If this bit is set it enables valid transitions on the selected ATOM Channel output to generate a watchpoint message. This bit is written by the JTAG interface but can also be controlled at SoC level with dedicated Start/Stop signal inputs to GTMDI. 0 Watchpoint Messages disabled 1 Watchpoint Messages enabled
18:16 CHSEL1	ATOM channel selection 1. Selects which channel in an ATOM module generates watchpoints. 000 ATOM CH0 001 ATOM CH1 010 ATOM CH2 011 ATOM CH3 100 ATOM CH4 101 ATOM CH5 110 ATOM CH6 111 ATOM CH7
13:12 TSS2	ATOM Channel Slope Selection 2. The TSS2 field selects the slope for the ATOM selected channel. 00 Any transition (both) 01 Transition from 0 to 1 (positive slope) 10 Transition from 1 to zero (negative slope) 11 Reserved

Table 723. GTMDI_ATOM_WPC1 field descriptions (continued)

Field	Description
11:8 SSEL2	ATOM sub-module Source Selection 2. Selects which ATOM module is the source for watchpoint generation. 0000 ATOM0 0001 ATOM1 0010 ATOM2 0011 ATOM3 0100 ATOM4 0101 Reserved 0110 Reserved 0111 Reserved 1000 Reserved 1001 Reserved 1010 Reserved 1011 Reserved 1100 Reserved 1101 Reserved 1110 Reserved 1111 Reserved
7 HEN2	Halt Enable 2. The HEN2 Halt Enable 2 bit controls if the ATOM selected channel is enabled to HALT the GTM module. If this bit is set and an event occurs on the ATOM selected channel, a Halt signal is generated to the GTM which enters Halt state. 0 Disables Halt GTM 1 Enables Halt GTM
6 WMC2	Watchpoint Message Control 2. The WMC2 Watchpoint Message Control 2 bit controls the watchpoint message sent through the Message Data bus. If this bit is set it enables valid transitions on the selected ATOM Channel output to generate a watchpoint message. This bit is written by the JTAG interface but can also be controlled at SoC level with dedicated Start/Stop signal inputs to GTMDI. 0 Watchpoint Messages disabled 1 Watchpoint Messages enabled
2:0 CHSEL2	ATOM channel selection 2. The CHSEL2[2:0] field selects a channel within the TOM selected sub-module to generate watchpoints. 000 ATOM CH0 001 ATOM CH1 010 ATOM CH2 011 ATOM CH3 100 ATOM CH4 101 ATOM CH5 110 ATOM CH6 111 ATOM CH7

49.4.1.9 ATOM watchpoint control 2 register (GTMDI_ATOM_WPC2)

The GTMDI_ATOM_WPC2 register shown in *Figure 578* controls the Watchpoint Triggers issued by ATOM channels. These triggers are used for GTM to enter Halt state, generate watchpoint messages and control logic external to GTMDI, such as in the SPU interface.

Index: 24 Access: Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	WTSEL1			0	0	0	0	0	0	STSEL1		0	0	TEN1	SEN1
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	WTSEL2			0	0	0	0	0	0	STSEL2		0	0	TEN2	SEN2
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 578. ATOM watchpoint control 2 register (GTMDI_ATOM_WPC2)

Table 724 describes the GTMDI_ATOM_WPC2 register functions.

Table 724. GTMDI_ATOM_WPC2 field descriptions

Field	Description
30:28 WTSEL1	Watchpoint Trigger Output Selection 1. The WTSEL1[2:0] bit field selects among 8 available output Watchpoint triggers which one is connected to the selected ATOM Channel. Other modules can select the same WTSEL1 value, in this case the result watchpoint signal implements an OR of all selected sources. 000 SHARED_WPT[0] 001 SHARED_WPT[1] 010 SHARED_WPT[2] 011 SHARED_WPT[3] 100 SHARED_WPT[4] 101 SHARED_WPT[5] 110 SHARED_WPT[6] 111 SHARED_WPT[7]
21:20 STSEL1	Start/Stop input signal selection 1. Selects the Start/Stop input that is used to control the watchpoint trace messages. These signals are actually a pair of signals. 00 SHARED_WPT_ST[0] 01 SHARED_WPT_ST[1] 10 SHARED_WPT_ST[2] 11 SHARED_WPT_ST[3]
17 TEN1	Watchpoint Trigger Enable 1. The TEN1 field controls if a Watchpoint external trigger is issued by the ATOM Channel output selected by WTSEL1. 0 Disable Trigger Generation 1 Enable Trigger Generation
16 SEN1	Start/Stop Enable 1. The SEN1 bit field enables the selected ATOM channel to consider the Start/Stop inputs for watchpoint trace messages control. If this bit is cleared, the messages are controlled only by JTAG writes to the WMC1 field in the GTMDI_ATOM_WPC1 register. 0 Disable trace control from Start/Stop inputs 1 Enable trace control from Start/Stop inputs

Table 724. GTMDI_ATOM_WPC2 field descriptions (continued)

Field	Description
14:12 WTSEL2	Watchpoint Trigger Output Selection 2. The WTSEL2[2:0] field selects among 8 available Watchpoint trigger outputs which one is connected to the ATOM selected channel. Other modules can selected the same WTSEL2 value, in this case the result signal is an OR of all selected sources. 000 SHARED_WPT[0] 001 SHARED_WPT[1] 010 SHARED_WPT[2] 011 SHARED_WPT[3] 100 SHARED_WPT[4] 101 SHARED_WPT[5] 110 SHARED_WPT[6] 111 SHARED_WPT[7]
5:4 STSEL2	Start/Stop input signal selection 2. Selects the Start/Stop input that is used for the watchpoint message control. These signals are actually a pair of signals. 00 SHARED_WPT_ST[0] 01 SHARED_WPT_ST[1] 10 SHARED_WPT_ST[2] 11 SHARED_WPT_ST[3]
1 TEN2	Watchpoint Trigger Enable 2. The TEN2 field controls if a Watchpoint external trigger is issued by the selected ATOM channel. 0 Disable Trigger Generation 1 Enable Trigger Generation
0 SEN2	Start/Stop Enable 2. The SEN2 bit field enables the Start/Stop inputs of the GTMDI to control Watchpoint trace messages generated by the ATOM selected channel. If this bit is cleared, the messages are controlled only by JTAG writes to the WMC2 field in the GTMDI_ATOM_WPC1 register. 0 Disable trace control from Start/Stop inputs 1 Enable trace control from Start/Stop inputs

49.4.1.10 SPEA watchpoint control 1 register 1 (GTMDI_SPEA_WPC1)

The GTMDI_SPEA_WPC1 register shown in *Figure 579* controls the Watchpoint Triggers issued by SPEA channels. These triggers are used for GTM to enter Halt state, generate watchpoint messages and to control logic outside of the GTMDI module, such as SPU interface.

Note: Start/Stop signal inputs to the GTMDI can overwrite the state of the WMC1 and WMC2 control bits thus controlling the Watchpoint messages behavior. In case of JTAG modifies the WMC1 or WMC2 bits and Start/Stop signals occur at the same time also modifying the same bits, Start/Stop signals have precedence over JTAG writes. The SEN1 and SEN2 control bits from the GTMDI_SEP0_WPC2 register need to be set in order to allow external Start/Stop signals to control the WMC1/2 bits.

Index: 29 Access: Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	SSEL			0	WMC1	TSS1		0	0	HEN1	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	WMC2	TSS2		0	0	HEN2	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 579. SPEA watchpoint control 1 register (GTMDI_SPEA_WPC1)

Table 725 describes the GTMDI_SPEA_WPC1 register functions.

Table 725. GTMDI_SPEA_WPC1 field descriptions

Field	Description
26:24 SSEL	SPEA sub-module Source Selection Selects which SPE sub-module is the source for watchpoint generation. 000 SPE0 001 SPE1 010 Reserved 011 Reserved 100 Reserved 101 Reserved 110 Reserved 111 Reserved
22 WMC1	Watchpoint Message Control 1. The WMC1 Watchpoint Message Control 1 controls the watchpoint message sent through the Message Data bus. If this bit is set it enables valid transitions on SPEA NIPD to generate a watchpoint message. This bit is written by the JTAG interface but can also be controlled at SoC level with dedicated Start/Stop signal inputs to GTMDI. 0 Watchpoint Messages disabled 1 Watchpoint Messages enabled
21:20 TSS1	SPEA NIPD Slope Selection. The TSS1 field selects the slope for the NIPD signal. 00 Any transition (both) 01 Transition from 0 to 1 (positive slope) 10 Transition from 1 to zero (negative slope) 11 Reserved
17 HEN1	Halt Enable 1. The HEN1 Halt Enable 1bit controls if the SPEA NIPD active slope event is enabled to HALT the GTM module. 0 Disables Halt GTM 1 Enables Halt GTM

Table 725. GTMDI_SPEA_WPC1 field descriptions (continued)

Field	Description
6 WMC2	Watchpoint Message Control 2. The WMC2 Watchpoint Message Control 2 controls the watchpoint message sent through the Message Data bus. If this bit is set it enables valid transitions on SPEA DIR output to generate a watchpoint message. This bit is written by the JTAG interface but can also be controlled at SoC level with dedicated Start/Stop signal inputs to GTMDI. 0 Watchpoint Messages disabled 1 Watchpoint Messages enabled
5:4 TSS2	SPEA DIR Slope Selection. The TSS2 field selects the slope for the DIR signal. 00 Any transition (both) 01 Transition from 0 to 1 (positive slope) 10 Transition from 1 to zero (negative slope) 11 Reserved
1 HEN2	Halt Enable 2. The HEN2 Halt Enable 2 bit controls if the SPEA DIR active slope event is enabled to HALT the GTM module. 0 Disables Halt GTM 1 Enables Halt GTM

49.4.1.11 SPEA watchpoint control 2 register (GTMDI_SPEA_WPC2)

The GTMDI_SPEA_WPC2 register shown in *Figure 580* controls the Watchpoint Triggers issued by SPEA. These watchpoints are used for GTM to enter Halt state, generate watchpoint trace messages and to control logic outside the GTMDI module, such as SPU interface.

Index: 30 Access: Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	WTSEL1			0	0	0	0	0	0	STSEL1		0	0	TEN1	SEN1
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	WTSEL2			0	0	0	0	0	0	STSEL2		0	0	TEN2	SEN2
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 580. SPEA watchpoint control 2 register (GTMDI_SPEA_WPC2)

Table 726 describes the GTMDI_SPEA_WPC2 register functions.

Table 726. GTMDI_SPEA_WPC2 field descriptions

Field	Description
30:28 WTSEL1	Watchpoint Trigger Output Selection 1. The WTSEL1[2:0] bit field selects among 8 available watchpoint triggers which one is connected to the NIPD SPEA signal. Other modules can select the same WTSEL1 value, in this case the result watchpoint signal implements an OR of all selected sources. 000 SHARED_WPT[0] 001 SHARED_WPT[1] 010 SHARED_WPT[2] 011 SHARED_WPT[3] 100 SHARED_WPT[4] 101 SHARED_WPT[5] 110 SHARED_WPT[6] 111 SHARED_WPT[7]
21:20 STSEL1	Start/Stop input signal selection 1. Selects the Start/Stop input that is used to control the watchpoint trace messages. These signals are actually a pair of signals. 00 SHARED_WPT_ST[0] 01 SHARED_WPT_ST[1] 10 SHARED_WPT_ST[2] 11 SHARED_WPT_ST[3]
17 TEN1	Watchpoint Trigger Enable 1. The TEN1 field controls if a watchpoint external trigger is issued by the SPEA NIPD selected event. 0 Disable External Watchpoint Trigger Generation 1 Enable External Watchpoint Trigger Generation
16 SEN1	Start/Stop Enable 1. The SEN1 bit field enable the SPEA NIPD selected event to consider the Start/Stop inputs for watchpoint trace message generation. If this bit is cleared, the messages are controlled only by JTAG writes to the WMC1 field in the GTMDI_SPEA_WPC1 register. 0 Disable trace control from Start/Stop inputs 1 Enable trace control from Start/Stop inputs
14:12 WTSEL2	Watchpoint Trigger Output Selection 2. The WTSEL2[2:0] field selects among 8 available watchpoint triggers which one is connected to the DIR SPEA signal. Other modules can selected the same WTSEL2 value, in this case the result signal is an OR of all selected sources. 000 SHARED_WPT[0] 001 SHARED_WPT[1] 010 SHARED_WPT[2] 011 SHARED_WPT[3] 100 SHARED_WPT[4] 101 SHARED_WPT[5] 110 SHARED_WPT[6] 111 SHARED_WPT[7]
5:4 STSEL2	Start/Stop input signal selection 2. Selects the Start/Stop input that is used for the watchpoint message control. These signals are actually a pair of signals. 00 SHARED_WPT_ST[0] 01 SHARED_WPT_ST[1] 10 SHARED_WPT_ST[2] 11 SHARED_WPT_ST[3]

Table 726. GTMDI_SPEA_WPC2 field descriptions (continued)

Field	Description
1 TEN2	Watchpoint Trigger Enable 2. The TEN2 field controls if a watchpoint external trigger is issued by the SPEA DIR selected event. 0 Disable External Watchpoint Trigger Generation 1 Enable External Watchpoint Trigger Generation
0 SEN2	Start/Stop Enable 2. The SEN2 bit field enable the SPEA DIR selected event to consider the Start/Stop inputs for watchpoint trace message generation. If this bit is cleared, the messages are controlled only by JTAG writes to the WMC2 field in the GTMDI_SPEA_WPC1 register. 0 Disable trace control from Start/Stop inputs 1 Enable trace control from Start/Stop inputs

49.4.1.12 SPEB watchpoint control 1 register 1 (GTMDI_SPEB_WPC1)

The GTMDI_SPEB_WPC1 register shown in *Figure 581* controls the Watchpoint Triggers issued by SPEB channels. These triggers are used for GTM to enter Halt state, generate watchpoint messages and to control logic outside of the GTMDI module, such as SPU interface.

Note: Start/Stop signal inputs to the GTMDI can overwrite the state of the WMC1 and WMC2 control bits thus controlling the Watchpoint messages behavior. In case of JTAG modifies the WMC1 or WMC2 bits and Start/Stop signals occur at the same time also modifying the same bits, Start/Stop signals have precedence over JTAG writes. The SEN1 and SEN2 control bits from the GTMDI_SPEB_WPC2 register need to be set in order to allow external Start/Stop signals to control the WMC1/2 bits.

Index: 35 Access: Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	SSEL			0	WMC1	TSS1		0	0	HEN1	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	WMC2	TSS2		0	0	HEN2	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 581. SPEB watchpoint control 1 register (GTMDI_SPEB_WPC1)

Table 727 describes the GTMDI_SPEB_WPC1 register functions.

Table 727. GTMDI_SPEB_WPC1 field descriptions

Field	Description
26:24 SSEL	SPEB sub-module Source Selection(1) Selects which SPE sub-module is the source for watchpoint generation. 000 SPE1 001 Reserved 010 Reserved 011 Reserved 100 Reserved 101 Reserved 110 Reserved 111 SPE0
22 WMC1	Watchpoint Message Control 1. The WMC1 Watchpoint Message Control 1 controls the watchpoint message sent through the Message Data bus. If this bit is set it enables valid transitions on SPEB NIPD to generate a watchpoint message. This bit is written by the JTAG interface but can also be controlled at SoC level with dedicated Start/Stop signal inputs to GTMDI. 0 Watchpoint Messages disabled 1 Watchpoint Messages enabled
21:20 TSS1	SPEB NIPD Slope Selection. The TSS1 field selects the slope for the NIPD signal. 00 Any transition (both) 01 Transition from 0 to 1 (positive slope) 10 Transition from 1 to zero (negative slope) 11 Reserved
17 HEN1	Halt Enable 1. The HEN1 Halt Enable 1bit controls if the SPEB NIPD active slope event is enabled to HALT the GTM module. 0 Disables Halt GTM 1 Enables Halt GTM
6 WMC2	Watchpoint Message Control 2. The WMC2 Watchpoint Message Control 2 controls the watchpoint message sent through the Message Data bus. If this bit is set it enables valid transitions on SPEB DIR output to generate a watchpoint message. This bit is written by the JTAG interface but can also be controlled at SoC level with dedicated Start/Stop signal inputs to GTMDI. 0Watchpoint Messages disabled 1 Watchpoint Messages enabled
5:4 TSS2	SPEB DIR Slope Selection. The TSS2 field selects the slope for the DIR signal. 00 Any transition (both) 01 Transition from 0 to 1 (positive slope) 10 Transition from 1 to zero (negative slope) 11 Reserved
1 HEN2	Halt Enable 2. The HEN2 Halt Enable 2 bit controls if the SPEB DIR active slope event is enabled to HALT the GTM module. 0 Disables Halt GTM 1 Enables Halt GTM

1. The SSEL field for SPEB starts from SPE1 selection, differently from the equivalent field for SPEA which selects SPE0 instead.

49.4.1.13 SPEB watchpoint control 2 register (GTMDI_SPEB_WPC2)

The GTMDI_SPEB_WPC2 register shown in *Figure 582* controls the Watchpoint Triggers issued by SPEB. These watchpoints are used for GTM to enter Halt state, generate

watchpoint trace messages and to control logic outside the GTMDI module, such as SPU interface.

Index: 36 Access: Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	WTSEL1			0	0	0	0	0	0	STSEL1		0	0	TEN1	SEN1
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	WTSEL2			0	0	0	0	0	0	STSEL2		0	0	TEN2	SEN2
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 582. SPEB watchpoint control 2 register (GTMDI_SPEB_WPC2)

Table 728 describes the GTMDI_SPEB_WPC2 register functions.

Table 728. GTMDI_SPEB_WPC2 field descriptions

Field	Description
30:28 WTSEL1	Watchpoint Trigger Output Selection 1. The WTSEL1[2:0] bit field selects among 8 available watchpoint triggers which one is connected to the NIPD SPEB signal. Other modules can select the same WTSEL1 value, in this case the result watchpoint signal implements an OR of all selected sources. 000 SHARED_WPT[0] 001 SHARED_WPT[1] 010 SHARED_WPT[2] 011 SHARED_WPT[3] 100 SHARED_WPT[4] 101 SHARED_WPT[5] 110 SHARED_WPT[6] 111 SHARED_WPT[7]
21:20 STSEL1	Start/Stop input signal selection 1. Selects the Start/Stop input that is used to control the watchpoint trace messages. These signals are actually a pair of signals. 00 SHARED_WPT_ST[0] 01 SHARED_WPT_ST[1] 10 SHARED_WPT_ST[2] 11 SHARED_WPT_ST[3]
17 TEN1	Watchpoint Trigger Enable 1. The TEN1 field controls if a watchpoint external trigger is issued by the SPEB NIPD selected event. 0 Disable External Watchpoint Trigger Generation 1 Enable External Watchpoint Trigger Generation
16 SEN1	Start/Stop Enable 1. The SEN1 bit field enable the SPEB NIPD selected event to consider the Start/Stop inputs for watchpoint trace message generation. If this bit is cleared, the messages are controlled only by JTAG writes to the WMC1 field in the GTMDI_SPEB_WPC1 register. 0 Disable trace control from Start/Stop inputs 1 Enable trace control from Start/Stop inputs

Table 728. GTMDI_SPEB_WPC2 field descriptions (continued)

Field	Description
14:12 WTSEL2	Watchpoint Trigger Output Selection 2. The WTSEL2[2:0] field selects among 8 available watchpoint triggers which one is connected to the DIR SPEB signal. Other modules can selected the same WTSEL2 value, in this case the result signal is an OR of all selected sources. 000 SHARED_WPT[0] 001 SHARED_WPT[1] 010 SHARED_WPT[2] 011 SHARED_WPT[3] 100 SHARED_WPT[4] 101 SHARED_WPT[5] 110 SHARED_WPT[6] 111 SHARED_WPT[7]
5:4 STSEL2	Start/Stop input signal selection 2. Selects the Start/Stop input that is used for the watchpoint message control. These signals are actually a pair of signals. 00 SHARED_WPT_ST[0] 01 SHARED_WPT_ST[1] 10 SHARED_WPT_ST[2] 11 SHARED_WPT_ST[3]
1 TEN2	Watchpoint Trigger Enable 2. The TEN2 field controls if a watchpoint external trigger is issued by the SPEB DIR selected event. 0 Disable External Watchpoint Trigger Generation 1 Enable External Watchpoint Trigger Generation
0 SEN2	Start/Stop Enable 2. The SEN2 bit field enable the SPEB DIR selected event to consider the Start/Stop inputs for watchpoint trace message generation. If this bit is cleared, the messages are controlled only by JTAG writes to the WMC2 field in the GTMDI_SPEB_WPC1 register. 0 Disable trace control from Start/Stop inputs 1 Enable trace control from Start/Stop inputs

49.4.1.14 DPLL watchpoint control 1 register (GTMDI_DPLL_WPC1)

The GTMDI_DPLL_WPC1 register shown in *Figure 583* controls various trace and debug features of the DPLL modules as described below.

Note: Start/Stop signal inputs to the GTMDI can overwrite the state of the WMC1 and WMC2 control bits thus controlling the Watchpoint messages behavior. In case of JTAG modifies the WMC1 or WMC2 bits and Start/Stop signals occur at the same time also modifying the same bits, Start/Stop signals have precedence over JTAG writes. The SEN1 and SEN2 control bits from the GTMDI_DPLL_WPC2 register need to be set in order to allow external Start/Stop signals to control the WMC1/2 bits.

Index: 41 Access: Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	TSEL1		0	WMC1	TSS1		0	0	HEN1	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	TSEL2B		0	0	TSEL2A		0	WMC2	0	0	0	0	HEN2	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 583. DPLL watchpoint control 1 register (GTMDI_DPLL_WPC1)

Table 729 describes the GTMDI_DPLL_WPC1 register functions.

Table 729. GTMDI_DPLL_WPC1 field descriptions

Field	Description
25:24 TSEL1	Watchpoint Selection 1. The TSEL1[1:0] field selects if a watchpoint trigger is issued based on TASI or SASI events. 00 Watchpoint on TASI event 01 Watchpoint on SASI event 10 Reserved 11 Reserved
22 WMC1	Watchpoint Message Control 1. The WMC1 Watchpoint Message Control 1 controls the watchpoint message sent through the Message Data bus. If this bit is set it enables valid transitions on DPLL SASI or TASI to generate a watchpoint message. This bit is written by the JTAG interface but can also be controlled at SoC level with dedicated Start/Stop signal inputs to GTMDI. 0 Watchpoint Messages disabled 1 Watchpoint Messages enabled
21:20 TSS1	TASI/SASI Slope Selection. The TSS1 field selects the slope for TASI or SASI depending on TSEL1 selection. The selected slope is monitored by watchpoints. 00 Any transition (both) 01 Transition from 0 to 1 (positive slope) 10 Transition from 1 to zero (negative slope) 11 Reserved
17 HEN1	The HEN1 Halt Enable 1 bit controls if the DPLL SASI or TASI selected signal is enabled to HALT the GTM module. If this bit is set and an event occurs on the DPLL selected signal, a Halt signal is generated to the GTM which enters Halt state. 0 Disables Halt GTM 1 Enables Halt GTM
13:12 TSEL2B	Watchpoint Selection 2B. The TSEL2B field selects the RAM access type to mach on 00 Watchpoint is issued on any access (read or write) 01 Watchpoint is issued based on write accesses 10 Watchpoint is issued based on read accesses 11 Reserved

Table 729. GTMDI_DPLL_WPC1 field descriptions (continued)

Field	Description
9:8 TSEL2A	Watchpoint Selection 2A. The TSEL2A field selects if a watchpoint is issued based on the RAM1a, RAM1bc or RAM2 access. 00 Watchpoint is issued based on RAM1a access 01 Watchpoint is issued based on RAM1b access 10 Watchpoint is issued based on RAM2 access 11 Reserved
6 WMC2	Watchpoint Message Control 2. The WMC2 Watchpoint Message Control 2 controls the watchpoint message sent through the Message Data bus. If this bit is set it enables selected accesses on DPLL RAMs to generate a watchpoint message. This bit is written by the JTAG interface but can also be controlled at SoC level with dedicated Start/Stop signal inputs to GTMDI. 0 Watchpoint Messages disabled 1 Watchpoint Messages enabled
1 HEN2	Halt Enable 2. The HEN Halt Enable bit controls if watchpoint from TSEL2 selection is enabled to HALT the GTM module. If this bit is set and an event occurs on an DPLL selected RAM, a Halt signal is generated to the GTM which enters Halt state. 0 Disables Halt GTM 1 Enables Halt GTM

49.4.1.15 DPLL watchpoint control 2 register (GTMDI_DPLL_WPC2)

The GTMDI_DPLL_WPC2 register shown in *Figure 584* controls the Watchpoint Triggers issued by DPLL. These triggers are used for GTM to enter Halt state and by GTMDI external logic at SoC level. This register also controls the Watchpoint Trace messages WPM.

Index: 42 Access: Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	WTSEL1			0	0	0	0	0	0	STSEL1		0	0	TEN1	SEN 1
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	WTSEL2			0	0	0	0	0	0	STSEL2		0	0	TEN2	SEN 2
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 584. DPLL watchpoint control 2 register (GTMDI_DPLL_WPC2)

Table 730 describes the GTMDI_DPLL_WPC2 register functions.

Table 730. GTMDI_DPLL_WPC2 field descriptions

Field	Description
30:28 WTSEL1	Watchpoint Trigger Output Selection 1. The WTSEL1[2:0] field selects among 8 available watchpoint triggers which one is connected to the DPLL SASI or TASI signal. Other modules can select this same WTSEL1 value, in this case the result trigger signal is an OR of all selected sources. 000 SHARED_WPT[0] 001 SHARED_WPT[1] 010 SHARED_WPT[2] 011 SHARED_WPT[3] 100 SHARED_WPT[4] 101 SHARED_WPT[5] 110 SHARED_WPT[6] 111 SHARED_WPT[7]
21:20 STSEL1	Start/Stop input signal selection 1. Selects the Start/Stop input that is used to control the watchpoint trace messages related to DPLL SASI or TASI signals. These signals are actually a pair of signals. 00 SHARED_WPT_ST[0] 01 SHARED_WPT_ST[1] 10 SHARED_WPT_ST[2] 11 SHARED_WPT_ST[3]
17 TEN1	Watchpoint Trigger Enable 1. The TEN1 field controls if a watchpoint external trigger is issued by the DPLL TASI or SASI selected event. 0 Disable External Watchpoint Trigger Generation 1 Enable External Watchpoint Trigger Generation
16 SEN1	Start/Stop Enable 1. The SEN1 bit field enable the DPLL TASI or SASI selected event to consider the Start/Stop inputs for watchpoint trace message generation. If this bit is cleared, the messages are controlled only by JTAG writes to the WMC1 field in the GTMDI_DPLL_WPC1 register. 0 Disable trace control from Start/Stop inputs 1 Enable trace control from Start/Stop inputs
14:12 WTSEL2	Watchpoint Trigger Output Selection 2. The WTSEL2[2:0] field selects among 8 available watchpoint triggers which one is connected to the DPLL RAM access signal. Other modules can select this same WTSEL2 value, in this case the result trigger signal is an OR of all selected sources. 000 SHARED_WPT[0] 001 SHARED_WPT[1] 010 SHARED_WPT[2] 011 SHARED_WPT[3] 100 SHARED_WPT[4] 101 SHARED_WPT[5] 110 SHARED_WPT[6] 111 SHARED_WPT[7]
5:4 STSEL2	Start/Stop input signal selection 2. Selects the Start/Stop input that is used by the watchpoint trace messages related to DPLL RAM access events for RAM1A, RAM1B and RAM2. These signals are actually a pair of signals. 00 SHARED_WPT_ST[0] 01 SHARED_WPT_ST[1] 10 SHARED_WPT_ST[2] 11 SHARED_WPT_ST[3]

Table 730. GTMDI_DPLL_WPC2 field descriptions (continued)

Field	Description
1 TEN2	Watchpoint Trigger Enable 2. The TEN2 field controls if a watchpoint external trigger is issued by the DPLL RAM selected access. 0 Disable External Watchpoint Trigger Generation 1 Enable External Watchpoint Trigger Generation
0 SEN2	Start/Stop Enable 2. The SEN2 bit field enable the DPLL RAM selected access event to consider the Start/Stop inputs for watchpoint trace message generation. If this bit is cleared, the messages are controlled only by JTAG writes to the WMC1 field in the GTMDI_DPLL_WPC1 register. 0 Disable trace control from Start/Stop inputs 1 Enable trace control from Start/Stop inputs

49.4.1.16 DPLL watchpoint control 3 register (GTMDI_DPLL_WPC3)

The GTMDI_DPLL_WPC3 register shown in *Figure 585* defines the data value observed in the selected DPLL RAM access. A match to this value causes a Watchpoint.

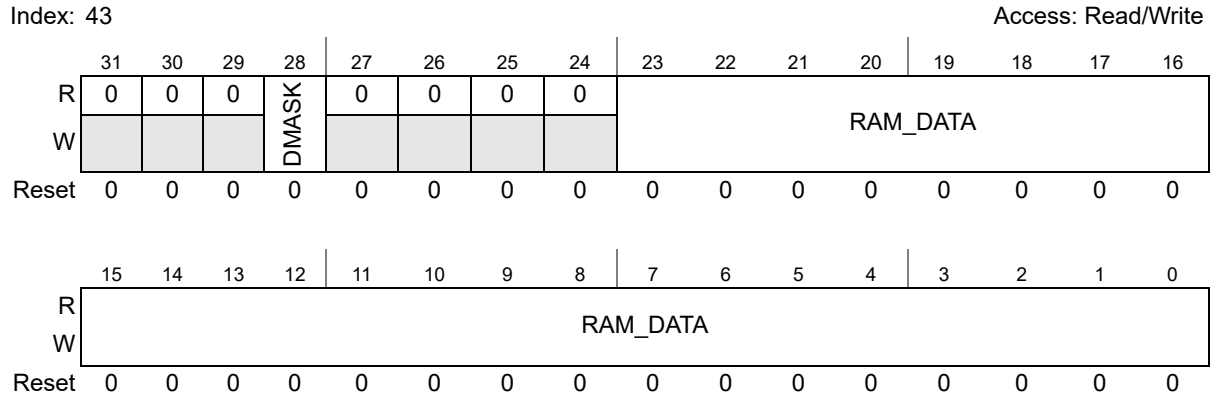


Figure 585. DPLL watchpoint control 3 register (GTMDI_DPLL_WPC3)

Table 731 describes the GTMDI_DPLL_WPC3 register functions.

Table 731. GTMDI_DPLL_WPC3 field descriptions

Field	Description
28 DMASK	Data Mask. The DMASK field is a mask for the data comparator. If set, the data read or written to the memory should not be considered for watchpoint generation. 0 RAM_DATA should be compared with data from RAM access 1 RAM_DATA is masked and should not be compared
23:0 RAM_DATA	Data value. The RAM_DATA[23:0] field defines the data value that is compared against the data accessed from the selected DPLL RAM.

49.4.1.17 DPLL watchpoint control 4 register (GTMDI_DPLL_WPC4)

The GTMDI_DPLL_WPC4 register shown in *Figure 586* defines the RAM address mask for the DPLL selected RAM address comparisson.

Index: 44 Access: Read/Write

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

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	RAM_ADDR_MASK											
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 586. DPLL watchpoint control 4 register (GTMDI_DPLL_WPC4)

Table 732 describes the GTMDI_DPLL_WPC4 register functions.

Table 732. GTMDI_DPLL_WPC4 field descriptions

Field	Description
11:0 RAM_ADDR_MASK	RAM Address mask. The RAM_ADDR_MASK[11:0] field is a mask applied bit by bit to the RAM_ADDR defined in the GTMDI_DPLL_WPC5 register. If mask is 1, the corresponding RAM_ADDR bit is not valid for comparison, thus it always returns a match when compared to the selected RAM address.

49.4.1.18 DPLL watchpoint control 5 register (GTMDI_DPLL_WPC5)

The GTMDI_DPLL_WPC5 register shown in *Figure 587* defines the address value that is monitored during DPLL selected RAM access. Once this address value matches with the DPLL RAM address a Watchpoint is issued.

Index: 45 Access: Read/Write

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

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	RAM_ADDR											
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 587. DPLL watchpoint control 5 register (GTMDI_DPLL_WPC5)

Table 733 describes the GTMDI_DPLL_WPC5 register functions.

Table 733. GTMDI_DPLL_WPC5 field descriptions

Field	Description
11:0 RAM_ADDR	RAM Address Compare. The RAM_ADDR[11:0] field defines the address value that is compared against the address for write and read DPLL RAM accesses. Before the comparison is executed the RAM_ADDR_MASK should be considered thus masking corresponding bits. A masked bit is always considered a match. **Note:** DPLL module has three RAM modules with different address widths. The address value in the RAM_ADDR field and the RAM address should be right aligned in order to be compared.

49.4.1.19 DPLL data trace control register (GTMDI_DPLL_DTC)

The GTMDI_DPLL_DTC register shown in *Figure 588* controls data trace message transmission.

Index: 46 Access: Read/Write

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

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	RWC		0	0	RAMSEL		0	DMC	STSEL		0	0	0	SEN
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 588. DPLL data trace control register (GTMDI_DPLL_DTC)

Note: Start/Stop signal inputs to the GTMDI can overwrite the state of the DMC control bit thus controlling the Data Trace messages behavior. In case of JTAG modifies the DMC bit and Start/Stop signals occur at the same time also modifying the same bit, Start/Stop signals have precedence over JTAG writes. The SEN control bit from the GTMDI_DPLL_DTC register need to be set in order to allow external Start/Stop signals to control the DMC bit.

Table 734 describes the GTMDI_DPLL_DTC register functions.

Table 734. GTMDI_DPLL_DTC field descriptions

Field	Description
13:12 RWC	Read/Write Control. This bit controls DPLL data trace message transmission. 00 Transmits only memory reads 01 Transmits only memory writes 10 Transmits all memory accesses 11 Reserved
9:8 RAMSEL	RAM Module Selection. This field selects a DPLL memory module for data tracing. 00 RAM1a selected for data tracing 01 RAM1b selected for data tracing 10 RAM2 selected for data tracing 11 Reserved

Table 734. GTMDI_DPLL_DTC field descriptions (continued)

Field	Description
6 DMC	Data Trace Message Control. This bit is written by JTAG interface but can also be controlled at SoC level with dedicated Start/Stop signal inputs to GTMDI. 0 Data Trace Messages disabled 1 Data Trace Messages enabled
5:4 STSEL	Start/Stop input signal selection 1. Selects the Start/Stop input that is used to control the watchpoint trace messages. These signals are actually a pair of signals. 00 SHARED_WPT_ST[0] 01 SHARED_WPT_ST[1] 10 SHARED_WPT_ST[2] 11 SHARED_WPT_ST[3]
0 SEN	Start/Stop Enable. The SEN bit field enables the DPLL data trace message generator to consider the Start/Stop inputs for data trace message transmission. If this bit is cleared, the messages are controlled only by JTAG writes to the DMC Data Trace Message Control bit in this register. 0 Disable trace control from Start/Stop inputs 1 Enable trace control from Start/Stop inputs

49.4.1.20 ARU watchpoint control 1 register (GTMDI_ARU_WPC1)

The GTMDI_ARU_WPC1 register shown in *Figure 589* controls the Watchpoint Triggers issued by ARU Debugging Channels. These watchpoint triggers are used to control the requests for GTM to enter Halt state, issue watchpoint messages and also to control logic external to the GTMDI at SoC level such as in the SPU interface.

Note: Start/Stop signal inputs to the GTMDI can overwrite the state of the WMC1 and WMC2 control bits thus controlling the Watchpoint messages behavior. In case of JTAG modifies the WMC1 or WMC2 bits and Start/Stop signals occur at the same time also modifying the same bits, Start/Stop signals have precedence over JTAG writes. The SEN1 and SEN2 control bits from the GTMDI_ARU_WPC2 register need to be set in order to allow external Start/Stop signals to control the WMC1/2 bits.

Index: 50 Access: Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	TSS1	0	0	0	0	HEN1	WMC1	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	TSS2	0	0	0	0	HEN2	WMC2	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 589. ARU watchpoint control 1 register (GTMDI_ARU_WPC1)

Table 735 describes the GTMDI_ARU_WPC1 register functions.

Table 735. GTMDI_ARU_WPC1 field descriptions

Field	Description
28 TSS1	ARU Debugging Channel 0 Data Activity Selection 1. The TSS1 bit selects the type of data activity to monitor on ARU Debugging Channel 0. 0 Any Data (do not compare data). Watchpoint generation is based on a valid data indication on ARU Debugging Channel 0 1 Compare data with expected value
23 HEN1	Halt Enable 1. The HEN1 Halt Enable 1 bit controls if the ARU Debugging Channel 0 is enabled to HALT the GTM module. If this bit is set and an event occurs on the ARU Debugging Channel 0, a Halt signal is generated to the GTM which enters debug state. 0 Disables Halt GTM 1 Enables Halt GTM
22 WMC1	Watchpoint Message Control 1. The WMC1 Watchpoint Message Control 1 bit controls the watchpoint message sent through the Message Data bus. If this bit is set it enables valid matches on the ARU Debugging Channel 0 to generate a watchpoint message. This bit is written by the JTAG interface but can also be controlled at SoC level with dedicated Start/Stop signal inputs to GTMDI. 0 Watchpoint Messages disabled 1 Watchpoint Messages enabled
12 TSS2	ARU Debugging Channel 1 Data Activity Selection. The TSS2 bit selects the type of data activity to monitor on the ARU Debugging Channel 1. 0 Any Data (do not compare data). Watchpoint generation is based on a valid data indication on ARU Debugging Channel Monitor 1 1 Compare data with expected value
7 HEN2	Halt Enable 2. The HEN2 Halt Enable 2 bit controls if the ARU debugging Channel 1 is enabled to HALT the GTM module. If this bit is set and an event occurs on the ARU Debugging Channel 1, a Halt signal is generated to the GTM which enters Halt state. 0 Disables Halt GTM 1 Enables Halt GTM
6 WMC2	Watchpoint Message Control 2. The WMC2 Watchpoint Message Control 2 bit controls the watchpoint message sent through the Message Data bus. If this bit is set it enables match on the ARU Debugging Channel 1 to generate a watchpoint message. This bit is written by the JTAG interface but can also be controlled at SoC level with dedicated Start/Stop signal inputs to GTMDI. 0 Watchpoint Messages disabled 1 Watchpoint Messages enabled

49.4.1.21 ARU watchpoint control 2 register (GTMDI_ARU_WPC2)

The GTMDI_ARU_WPC2 register shown in *Figure 589* controls the Watchpoint Triggers issued by ARU Debugging Channels. These triggers are used for GTM to enter Halt state, generate watchpoint messages and control logic external to GTMDI, such as in the SPU interface.

Index: 51 Access: Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	WTSEL1			0	0	0	0	0	0	STSEL1		0	0	TEN1	SEN1
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	WTSEL2			0	0	0	0	0	0	STSEL2		0	0	TEN2	SEN2
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 590. ARU watchpoint control 2 register (GTMDI_ARU_WPC2)

Table 736 describes the GTMDI_ARU_WPC2 register functions.

Table 736. GTMDI_ARU_WPC2 field descriptions

Field	Description
30:28 WTSEL1	Watchpoint Trigger Output Selection 1. The WTSEL1[2:0] field selects among 8 available watchpoint triggers which one is connected to the ARU Debugging Channel 0. Other modules can selected the same WTSEL1 value, in this case the result selection is an OR of all selected sources. 000 SHARED_WPT[0] 001 SHARED_WPT[1] 010 SHARED_WPT[2] 011 SHARED_WPT[3] 100 SHARED_WPT[4] 101 SHARED_WPT[5] 110 SHARED_WPT[6] 111 SHARED_WPT[7]
21:20 STSEL1	Start/Stop input signal selection 1. Selects the Start/Stop input that is used to control the watchpoint trace messages for ARU Debugging Channel 0. These signals are actually a pair of signals. 00 SHARED_WPT_ST[0] 01 SHARED_WPT_ST[1] 10 SHARED_WPT_ST[2] 11 SHARED_WPT_ST[3]
17 TEN1	Watchpoint Trigger Enable 1. The TEN1 field controls if a watchpoint external trigger is issued by the ARU debugging Channel 0. 0 Disable Trigger Generation 1 Enable Trigger Generation
16 SEN1	Start/Stop Enable 1. The SEN1 bit field enable the ARU Debugging Channel 0 to consider the Start/Stop inputs for watchpoint trace message generation. If this bit is cleared, the messages are controlled only by JTAG writes to the WMC1 field in the GTMDI_ARU_WPC1 register. 0 Disable trace control from Start/Stop inputs 1 Enable trace control from Start/Stop inputs

Table 736. GTMDI_ARU_WPC2 field descriptions (continued)

Field	Description
14:12 WTSEL2	Watchpoint Trigger Output Selection 2. The WTSEL2[2:0] field selects among 8 available watchpoint triggers which one is connected to the ARU Debugging Channel 1. Other modules can selected the same WTSEL2 value, in this case the result signal is an OR of all selected sources. 000 SHARED_WPT[0] 001 SHARED_WPT[1] 010 SHARED_WPT[2] 011 SHARED_WPT[3] 100 SHARED_WPT[4] 101 SHARED_WPT[5] 110 SHARED_WPT[6] 111 SHARED_WPT[7]
5:4 STSEL2	Start/Stop input signal selection 2. Selects the Start/Stop input that is used for the watchpoint message control for ARU Debugging Channel 1. These signals are actually a pair of signals. 00 SHARED_WPT_ST[0] 01 SHARED_WPT_ST[1] 10 SHARED_WPT_ST[2] 11 SHARED_WPT_ST[3]
1 TEN2	Watchpoint Trigger Enable 2. The TEN2 field controls if a watchpoint external trigger is issued by the ARU Debugging Channel 1 0 Disable Trigger Generation 1 Enable Trigger Generation
0 SEN2	Start/Stop Enable 2. The SEN2 bit field enable the ARU Debugging Channel 1 to consider the Start/Stop inputs for watchpoint trace message generation. If this bit is cleared, the messages are controlled only by JTAG writes to the WMC2 field in the GTMDI_ARU_WPC1 register. 0 Disable trace control from Start/Stop inputs 1 Enable trace control from Start/Stop inputs

49.4.1.22 ARU watchpoint DATA0H register (GTMDI_ARU_DATA0H)

The ARU Watchpoint DATA0H high word shown in *Figure 591* is used for comparison against the high word of the ARU Debugging Channel 0. The address of the channel to be monitored is defined by the Host CPU in the GTM IP ARU_DBG_ACCESS0 register.

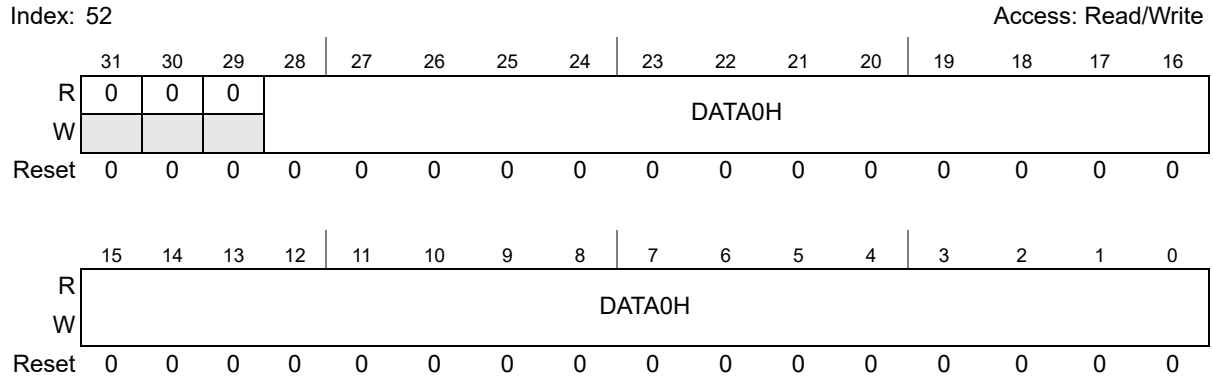


Figure 591. ARU watchpoint DATA0H register (GTMDI_ARU_DATA0H)

Table 737 describes the GTMDI_ARU_DATA0H register functions.

Table 737. GTMDI_ARU_DATA0H field descriptions

Field	Description
28:0 DATA0H	Watchpoint DATA0H. This field is compared against the high word of the ARU Debugging Channel 0.

49.4.1.23 ARU watchpoint DATA0L register (GTMDI_ARU_DATA0L)

The ARU Watchpoint DATA0H low word shown in *Figure 592* is used for comparison against the low word of the ARU Debugging Channel 0. The address for the channel to be monitored is defined by the Host CPU in the GTM IP ARU_DBG_ACCESS0 register.

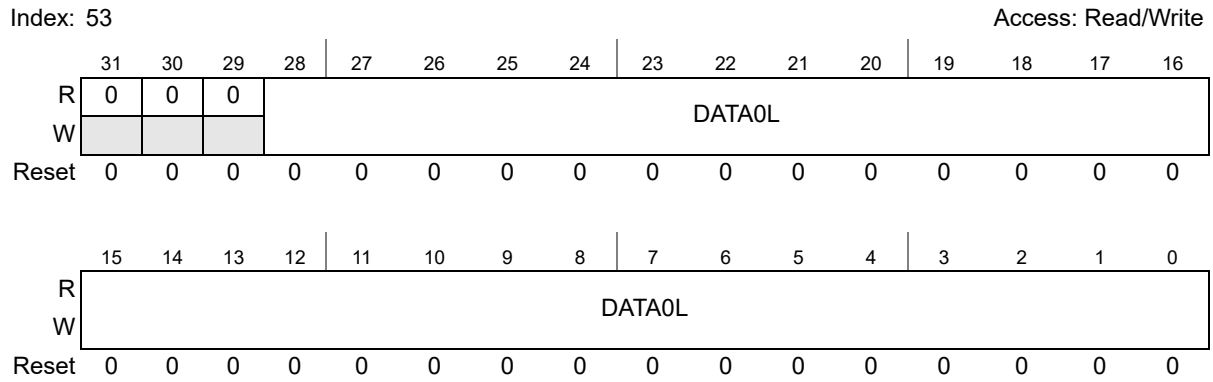


Figure 592. ARU watchpoint DATA0L register (GTMDI_ARU_DATA0L)

Table 738 describes the GTMDI_ARU_DATA0L register functions.

Table 738. GTMDI_ARU_DATA0L field descriptions

Field	Description
28:0 DATA0L	Watchpoint DATA0L. This field is compared against the low word of the ARU Debugging Channel 0.

49.4.1.24 ARU watchpoint DATA1H register (GTMDI_ARU_DATA1H)

The ARU Watchpoint DATA1H high word shown in *Figure 593*. is used for comparison against the high word of the ARU Debugging Channel 1. The address for the channel to be monitored is defined by the Host CPU in the GTM IP ARU_DBG_ACCESS1 register.

Index: 54 Access: Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	DATA1H												
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	DATA1H															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 593. ARU watchpoint DATA1H register (GTMDI_ARU_DATA1H)

Table 739 describes the GTMDI_ARU_DATA1H register functions.

Table 739. GTMDI_ARU_DATA1H field descriptions

Field	Description
28:0 DATA1H	Watchpoint DATA1H. This field is compared against the high word of the ARU Debugging Channel 1.

49.4.1.25 ARU watchpoint DATA1L register (GTMDI_ARU_DATA1L)

The ARU Watchpoint DATA1L low word shown in *Figure 594* is used to be compared against the low word of the ARU Debugging Channel 1. The address for the channel to be monitored is defined by the Host CPU in the GTM IP ARU_DBG_ACCESS1 register.

Index: 55 Access: Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	DATA1L												
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	DATA1L															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 594. ARU Watchpoint DATA1L (GTMDI_ARU_DATA1L)

Table 740 describes the GTMDI_ARU_DATA1L register functions.

Table 740. GTMDI_ARU_DATA1L field descriptions

Field	Description
28:0 DATA1L	Watchpoint DATA1L. This field is compared against the low word of the ARU Debugging Channel 1.

49.4.1.26 ARU data trace control register (GTMDI_ARU_DTC)

The GTMDI_ARU_DTC register shown in *Figure 595* controls the ARU data trace message transmission. The GTM Module selects two ARU addresses to be monitored, using the ARU Debugging Channels 0 and 1. The data trace for these two channels is controlled by DMC1/2 bits in this register. Once enabled, messages are generated as soon as a valid data is identified in one of the ARU Debugging Channels, 0 or 1.

Note: Start/Stop signal inputs to the GTMDI can overwrite the state of the DMC1 and DMC2 control bits thus controlling the Data Trace messages behavior. In case of JTAG modifies the DMC1 or DMC2 bits and Start/Stop signals occur at the same time also modifying the same bits, Start/Stop signals have precedence over JTAG writes. The SEN1 and SEN2 control bits need to be set in order to allow external Start/Stop signals to control the DMC1/2 bits.

Index: 56 Access: Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	STSEL1		0	0	DMC1	SEN1
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	STSEL2		0	0	DMC2	SEN2
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 595. ARU data trace control register (GTMDI_ARU_DTC)

Table 741 describes the GTMDI_ARU_DTC register functions.

Table 741. GTMDI_ARU_DTC field descriptions

Field	Description
21:20 STSEL1	Start/Stop input signal selection 1. Selects the Start/Stop input that is used by the ARU Debugging Channel 0 data trace messages. These signals are actually a pair of signals. 00 SHARED_WPT_ST[0] 01 SHARED_WPT_ST[1] 10 SHARED_WPT_ST[2] 11 SHARED_WPT_ST[3]
17 DMC1	Data Trace Message Control. This bit controls ARU data trace messages transmission for ARU Debugging Channel 0. This bit is written by the JTAG interface but can also be controlled at SoC level by dedicated Start/Stop signal inputs to GTMDI. 0 Data Trace Messages disabled 1 Data Trace Messages enabled
16 SEN1	Start/Stop Enable 1. The SEN 1 bit field enables the ARU data trace message generator for ARU Debugging Channel 0 to consider the Start/Stop inputs for data trace message transmission. If this bit is cleared, the messages are controlled only by JTAG writes to the DMC1 Data Trace Message Control bit in this register. 0 Disable trace control from Start/Stop inputs 1 Enable trace control from Start/Stop inputs

Table 741. GTMDI_ARU_DTC field descriptions (continued)

Field	Description
5:4 STSEL2	Start/Stop input signal selection 2. Selects the Start/Stop input that is used by the ARU Debugging Channel 1 data trace messages. These signals are actually a pair of signals. 00 SHARED_WPT_ST[0] 01 SHARED_WPT_ST[1] 10 SHARED_WPT_ST[2] 11 SHARED_WPT_ST[3]
1 DMC2	Data Trace Message Control. This bit controls ARU data trace message transmission for ARU Debugging Channel 1. This bit is written by the JTAG interface but can also be controlled at SoC level by dedicated Start/Stop signal inputs to GTMDI. 0 Data Trace Messages disabled 1 Data Trace Messages enabled
0 SEN2	The SEN2 bit field enables the ARU data trace message generator for ARU Debugging Channel 1 to consider the Start/Stop inputs for data trace message transmission. If this bit is cleared, the messages are controlled only by JTAG writes to DMC2 Data Trace Message Control bit in this register. 0 Disable trace control from Start/Stop inputs 1 Enable trace control from Start/Stop inputs

49.4.1.27 MCSA development control register (GTMDI_MCSA_DC)

The GTMDI_MCSA_DC register shown in *Figure 596* controls various trace and debug features.

Index: 60 Access: Read/Write

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

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	MCSA_SEL				0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 596. MCSA/B development control register (GTMDI_MCSA_DC)

Table 742 describes the GTMDI_MCSA_DC register functions.

Table 742. GTMDI_MCSA_DC field descriptions

Field	Description
11:8 MCSA_SEL	MCSA Selection field. The MCSA_SEL[3:0] selection field selects which one of the MCS cores in the GTM module are assigned as MCSA. 0000 MCS0 0001 MCS1 0010 MCS2 0011 MCS3 0100 MCS4 0101 Reserved 0110 Reserved 0111 Reserved 1000 Reserved 1001 Reserved 1010 Reserved 1011 Reserved 1100 Reserved 1101 Reserved 1110 Reserved 1111 Reserved

49.4.1.28 MCSA watchpoint control register (GTMDI_MCSA_WPC)

The MCSA Watchpoint Control Register shown in *Figure 597* is used to configure MCSA watchpoints. This register can be configured for halt/watchpoint based on instruction fetch, data read or data write, with address and data masking options.

Note: Start/Stop signal inputs to the GTMDI can overwrite the state of the WMC control bit thus controlling the Watchpoint messages behavior. In case of JTAG modifies the WMC bit and Start/Stop signals occur at the same time also modifying the same bit, Start/Stop signals have precedence over JTAG writes. The SEN control bit needs to be asserted in order to allow external Start/Stop signals to control the WMC bit.

Note: The watchpoint channel selection is made through the GTMDI_MCSA_CE by the WPCE field.

Index: 62 Access: Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	HACH		0	WMC	0	0	HME			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	HWO		0	0	0	0	0	EOC	HEN	0	STSEL		0	0	TEN	SEN
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 597. MCSA watchpoint control register (GTMDI_MCSA_WPC)

Table 743 describes the GTMDI_MCSA_WPC register functions.

Table 743. GTMDI_MCSA_WPC field descriptions

Field	Description
25:24 HACH	Halt/Watchpoint on Active Channel. The HACH field selects which type of MCS RAM access generates a watchpoint or halt. Combining this field with HWO gives GTMDI the capability of monitoring channel activity as well as accesses to specific addresses or data. 00 Match on Instruction Fetch 01 Match on Data Read 10 Match on Data Write 11 Match on Data Read or Write
22 WMC	Watchpoint Message Control. The WMC Watchpoint Message Control bit controls the watchpoint message sent through the Message Data bus. If this bit is asserted it enables MCSA Watchpoints to generate a watchpoint message. This bit is written by the JTAG interface but can also be controlled at SoC level with dedicated Start/Stop signal inputs to GTMDI. 0 Watchpoint Messages disabled 1 Watchpoint Messages enabled
19:16 HME	Halt/Watchpoint Data Mask Enable. The HME field selects which data bytes are used for hardware halt/watchpoint generation when comparing to HWD register. This register field is only meaningful if HWO bit field is [x1], otherwise the data comparison is disabled independent of HME value. 0000 All bytes of data compared xxx1 Least significant byte of data masked out xx1x Second least significant byte of data masked out x1xx Second most significant byte of data masked out 1xxx Most significant byte of data masked out
15:14 HWO	Halt/Watchpoint Operand. The HWO field selects if address or data matching , or both, is done. Registers WPA and WPD are used for address and data comparison, respectively. If address is not compared WPA is not use. If data is not compared then WPD is not used. 00 Comparison Disabled for address or data x1 Compare data with HWD field 1x Compare address with HWA field
8 EOC	$\overline{\text{EVTO}}$ Control. The EOC bit controls the indication on the $\overline{\text{EVTO}}$ output. See *Figure 600*, *Figure 601*, *Figure 602* and *Figure 603* for definition about watchpoint 1 and 2. 0 Indicates Watchpoint 2 occurrence 1 Indicates Watchpoint 1 occurrence
7 HEN	Halt Enable. The HEN Halt Enable bit controls if the MCSA Watchpoint is enabled to HALT the GTM IP module. If this bit is set and watchpoint occurs on the MCSA a Halt signal is generated to the GTM IP which enters debug state. 0 Disables Halt GTM 1 Enables Halt GTM
5:4 STSEL	Start/Stop input signal selection. Selects the Start/Stop input that is used to control the watchpoint messages. These signals are actually a pair of signals. 00 SHARED_WPT_ST[0] 01 SHARED_WPT_ST[1] 10 SHARED_WPT_ST[2] 11 SHARED_WPT_ST[3]

Table 743. GTMDI_MCSA_WPC field descriptions (continued)

Field	Description
1 TEN	Watchpoint Trigger Enable.The TEN field controls if a external trigger is issued by the MCSA when a watchpoint occurs. 0 Disables Trigger Generation 1 Enables Trigger Generation
0 SEN	Start/Stop Enable. The SEN bit field enables the MCSA Watchpoint message to consider the Start/Stop inputs for watchpoint message transmission. If this bit is cleared, the messages are controlled only by JTAG writes to the WMC Watchpoint Trace Message Control bit. 0 Disables trace control from Start/Stop inputs 1 Enables trace control from Start/Stop inputs

49.4.1.29 MCSA program fetch trace control register (GTMDI_MCSA_PTC)

The GTMDI_MCSA_PTC register shown in *Figure 598* allows program fetch traces to be enabled or disabled on the occurrence of a watchpoint. If the same watchpoint is programed to enable and disable a trace, the occurrence of the watchpoint toggles the current value of the trace flags. If one watchpoint occurs to enable trace and a different watchpoint occurs to disable trace at the same time the current value of the trace flags also toggle.

Index: 63 Access: Read/Write

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

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	PFTS		0	0	PFTE		0	FTC	0	0	0	0	0	SEN
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 598. MCSA program fetch trace control register (GTMDI_MCSA_PTC)

Table 744 describes the GTMDI_MCSA_PTC register functions.

Table 744. GTMDI_MCSA_PTC field descriptions

Field	Description
13:12 PFTS	Program Fetch Trace Start. This bit field allows Program fetch trace to be enabled at a watchpoint occurrence. See *Figure 600*, *Figure 601*, *Figure 602* and *Figure 603* for definition about watchpoint 1 and 2. 00 No watchpoint selected (do not start Program Fetch Trace) 01 Program Fetch Trace enabled on MCSA Program Watchpoint 1 occurrence 10 Program Fetch Trace enabled on MCSA Program Watchpoint 2 occurrence 11 Reserved
9:8 PFTE	Program Fetch Trace End. This bit field allows program trace to be disabled at a watchpoint occurrence. 00 No watchpoint selected 01 Program Fetch Trace disabled on MCSA Program Watchpoint 1 occurrence 10 Program Fetch Trace disabled on MCSA Program Watchpoint 2 occurrence 11 Reserved
6 FTC	Fetch Trace Control. This bit controls the Fetch Trace message sent through the Message Data bus. If this bit is asserted it enables MCSA Fetch access to generate a trace message. This bit is written by the JTAG interface but can also be controlled at SoC level with dedicated Start/Stop signal inputs to GTMDI 0 Fetch Messages disabled 1 Fetch Messages enabled
0 SEN	Start/Stop Enable. The SEN bit field enables the MCSA Fetch message to consider the Start/Stop inputs for message transmission. If this bit is cleared, the messages are controlled only by JTAG writes to the FTC Fetch Trace Message Control bit. 0 Disable fetch trace control from Start/Stop inputs 1 Enable fetch trace control from Start/Stop inputs

49.4.1.30 MCSA data trace control register (GTMDI_MCSA_DTC)

The GTMDI_MCSA_DTC register shown in *Figure 599* controls which address ranges are enabled for data trace, and if reads and writes are traced in that range.

Index: 64 Access: Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	RC0	RC1
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	RWT0		0	0	RWT1		0	DTC	DTS		0	DTE		SEN
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 599. MCSA data trace control register (GTMDI_MCSA_DTC)

Table 745 describes the GTMDI_MCSA_DTC register functions.

Table 745. GTMDI_MCSA_DTC field descriptions

Field	Description
17 RC0	Range Control 0. Controls the range on data trace window 0. The window 0 is defined by DTAR1 register. See *Figure 605*. 0 Trace addresses outside (exclusive) of data trace window 0 1 Trace address inside (inclusive) of data trace window 0
16 RC1	Range Control 1. Controls the range on data trace window 1. The window 1 is defined by DTAR2 register. See *Figure 606*. 0 Trace addresses outside (exclusive) of data trace window 1 1 Trace address inside (inclusive) of data trace window 1
13:12 RWT0	MCSA Read/Write Trace Control 0. RWT0 controls whether data trace messages are generated for MCSA RAM read or write accesses. 00 No MCSA data trace messages generated for data trace window 0 01 Enable MCSA data read trace for MCS window 0 10 Enable MCSA data write trace for MCS window 0 11 Enable MCSA data read and write trace for data trace window 0
9:8 RWT1	MCSA Read/Write Trace Control 1. RWT1 controls whether data trace messages are generated for MCSA accesses inside MCS data trace window, and if so, whether reads, writes, or both generate the data trace messages. 00 No MCSA data trace messages generated for data trace window 1 01 Enable MCSA data read trace for MCS window 1 10 Enable MCSA data write trace for MCS window 1 11 Enable MCSA data read and write trace for data trace window 1
6 DTC	Data Trace Control. This bit controls the Data Trace message sent through the Message Data bus. If this bit is asserted it enables MCSA Data access to generate a data trace message. This bit is written by the JTAG interface but can also be controlled at SoC level with dedicated Start/Stop signal inputs to GTMDI 0 Data Messages disabled 1 Data Messages enabled
5:4 DTS	Data Trace Start. This bit field allows data trace to be enabled at a watchpoint occurrence. 00 No watchpoint selected 01 Data Trace enabled on MCSA Watchpoint 1 occurrence 10 Data Trace enabled on MCSA/B Watchpoint 2 occurrence 11 Reserved
2:1 DTE	Data Trace End. This bit field allows data trace to be disabled at a watchpoint occurrence. 00 No watchpoint selected 01 Data Trace disabled on MCSA Watchpoint 1 occurrence 10 Data Trace disabled on MCSA Watchpoint 2 occurrence 11 Reserved
0 SEN	Start/Stop Enable. The SEN bit field enables the MCSA Data message to consider the Start/Stop inputs for message transmission. If this bit is cleared, the messages are controlled only by JTAG writes to the DTC Data Trace Message Control bit. 0 Disable data trace control from Start/Stop inputs 1 Enable data trace control from Start/Stop inputs

49.4.1.31 MCSA watchpoint address 1 register (GTMDI_MCSA_WPA1)

The MCSA Watchpoint Address 1 Register shown in *Figure 600* is used to configure the MCSA halt/watchpoint hardware. It isused for data and instruction fetch accesses.

Index: 65 Access: Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	HWAM													
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	HWA													
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 600. MCSA watchpoint address 1 register (GTMDI_MCSA_WPA1)

Note: The HWA field represents word addresses relative to the MCS address map, not the CPU address map. This applies to the GTMDI_MCSA_WPA1 and GTMDI_MCSA_WPA2 registers.

Table 746 describes the GTMDI_MCSA_WPA1 register functions.

Table 746. GTMDI_MCSA_WPA1 field descriptions

Field	Description
29:16 HWAM	Halt/Watchpoint Address Mask. The HWAM field is used to select which address bits are compared for halt and watchpoint generation. Address match = (HWA&HWAM) == (Address & HWAM);
13:0 HWA	Halt/Watchpoint Address. The HWA field is used for comparison with instruction or data ddresses. The HWA field represents a word address.

49.4.1.32 MCSA watchpoint address 2 register (GTMDI_MCSA_WPA2)

The MCSA Watchpoint Address 2 Register shown in *Figure 601* is used to configure the MCSA halt/watchpoint hardware. It is used for data and instruction fetch accesses.

Index: 66 Access: Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	HWAM													
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	HWA													
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 601. MCSA watchpoint address 2 register (GTMDI_MCSA_WPA2)

Table 747 describes the GTMDI_MCSA_WPA2 register functions.

Table 747. GTMDI_MCSA_WPA2 field descriptions

Field	Description
29:16 HWAM	Halt/Watchpoint Address Mask. The HWAM field is used to select which address bits are compared for halt and watchpoint generation. Address match = (HWA&HWAM) == (Address & HWAM);
13:0 HWA	Halt/Watchpoint Address. The HWA field is used for comparison with instruction or data addresses. The HWA field represents a word address.

49.4.1.33 MCSA watchpoint data 1 register (GTMDI_MCSA_WPD1)

The MCSA Halt/Watchpoint Data 1 Register shown in *Figure 602* is used to configure the MCSA halt/watchpoint hardware. It is used for data accesses selection only.

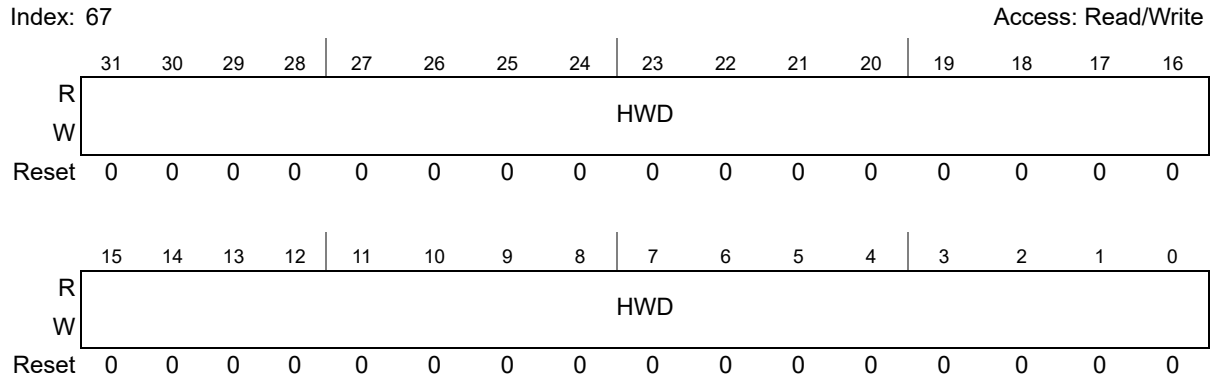


Figure 602. MCSA watchpoint data 1 register (GTMDI_MCSA_WPD1)

Table 748 describes the GTMDI_MCSA_WPD1 register functions.

Table 748. GTMDI_MCSA_WPD1 field descriptions

Field	Description
31:0 HWD	Halt/Watchpoint Data. The HWD field is used to compare data operands in a RAM read or write access. Depending upon the size of the access, only certain bits of HWD are compared against the the RAM data. If the operation size is 8-bits, the 8 most significant bits of HWD are compared against the RAM data. If the operation size is 24-bits, the 24 least significant bits of HWD are compared against the RAM data. If the operation size is 32-bits, all bits from HWD are compared against the RAM data.

49.4.1.34 MCSA watchpoint data 2 register (GTMDI_MCSA_WPD2)

The MCSA Watchpoint Data 2 Register shown in *Figure 603* is used to configure the MCSA halt/watchpoint hardware. It is used for data accesses selection only.

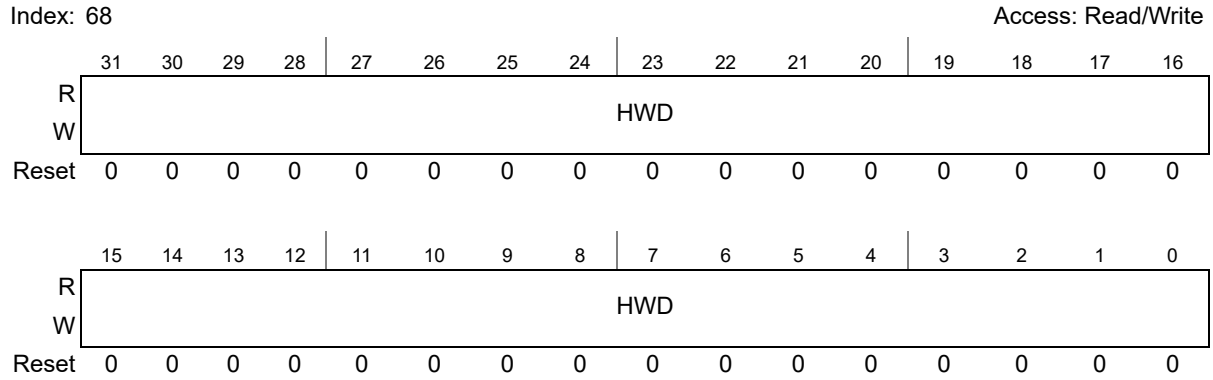


Figure 603. MCSA watchpoint data 2 register (GTMDI_MCSA_WPD2)

Table 749 describes the GTMDI_MCSA_WPD2 register functions.

Table 749. GTMDI_MCSA_WPD2 field descriptions

Field	Description
31:0 HWD	Halt/Watchpoint Data. The HWD field is used to compare data operands in a RAM read or write access. Depending upon the size of the access, only certain bits of HWD are compared against the RAM data. If the operation size is 8-bits, the 8 most significant bits of HWD are compared against the RAM data. If the operation size is 24-bits, the 24 least significant bits of HWD are compared against the RAM data. If the operation size is 32-bits, all bits from HWD are compared against the RAM data.

49.4.1.35 MCSA channel enable register (GTMDI_MCSA_CE)

The GTMDI_MCSA_CE register shown in *Figure 604* enables the monitoring for 8 MCSA channels independently.

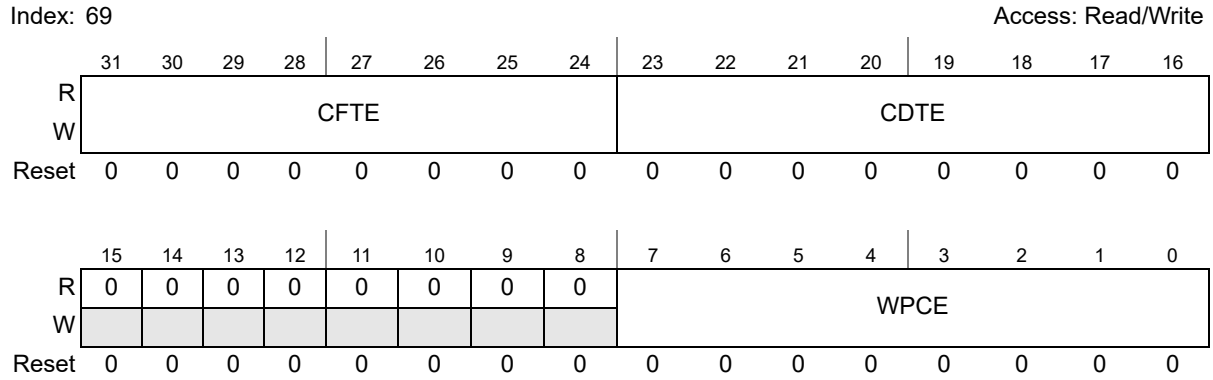


Figure 604. MCSA channel enable register (GTMDI_MCSA_CE)

Table 750 describes the GTMDI_MCSA_CE register functions.

Table 750. GTMDI_MCSA_CE field descriptions

Field	Description
31:24 CFTE	MCSA Channel Fetch Trace Enable. The CFTE[7:0] is an one hot bit field which defines the channel or channels to be monitored for fetch traces. If enabled, the corresponding channel Fetches are monitored by Fetch Trace messages. Start and Stop message controls are controlled by the FTC bit in the GTMDI_MCSA_PTC registers (*Section 49.4.1.29: MCSA program fetch trace control register (GTMDI_MCSA_PTC)*. 00000000No channel selected for trace xxxxxxx1Channel [0] selected xxxxxx1xChannel [1] selected xxxxx1xxChannel [2] selected xxxx1xxxChannel [3] selected xxx1xxxxChannel [4] selected xx1xxxxxChannel [5] selected x1xxxxxxChannel [6] selected 1xxxxxxxChannel [7] selected 11111111All Channels selected
23:16 CDTE	MCSA Channel Data Trace Enable. The CDTE[7:0] is an one hot bit field which defines the channel or channels to be monitored for data trace messages. If enabled, the corresponding channel Data Accesses are monitored by Data Trace messages. Start and Stop message are controlled by the DTC bit in the GTMDI_MCSA_DTC registers (*Section 49.4.1.30: MCSA data trace control register (GTMDI_MCSA_DTC)* and by the Data trace window. 00000000No channel selected for trace xxxxxxx1Channel [0] selected xxxxxx1xChannel [1] selected xxxxx1xxChannel [2] selected xxxx1xxxChannel [3] selected xxx1xxxxChannel [4] selected xx1xxxxxChannel [5] selected x1xxxxxxChannel [6] selected 1xxxxxxxChannel [7] selected 11111111 All Channels selected
7:0 WPCE	MCSA Channel Enable. The WPCE[7:0] is a one hot bit field which defines the channel or channels to be monitored for watchpoint, data and fetch traces. The messages controlled by this bit must be also enabled and disabled according to occurrence of Watchpoints and Address Ranges. 00000000No channel selected for trace xxxxxxx1Channel [0] selected xxxxxx1xChannel [1] selected xxxxx1xxChannel [2] selected xxxx1xxxChannel [3] selected xxx1xxxxChannel [4] selected xx1xxxxxChannel [5] selected x1xxxxxxChannel [6] selected 1xxxxxxxChannel [7] selected 11111111 All Channels selected

49.4.1.36 MCSA data trace address range 1 register (GTMDI_MCSA_DTAR1)

The GTMDI_MCSA_DTAR1 register shown in *Figure 605* defines a data trace address range. The address range is used by the MCSA and enables the range for reads and writes access to data only. If the start address value is greater than the end address value, all accesses are considered to have the address value outside to the data trace window 0.

Index: 71 Access: Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	DTSA [13:2]												0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	DTEA [13:2]												0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 605. MCSA Data trace address range 1 register (GTMDI_MCSA_DTAR1)

Note: The DTSA and DTEA fields represent word addresses relative to the MCS address map, not the CPU address.

Table 751 describes the GTMDI_MCSA_DTAR1 register functions.

Table 751. GTMDI_MCSA_DTAR1 field descriptions

Field	Description
29:18 DTSA	Data Trace Start Address. The DTSA field is the start address for MCSA data trace window.
13:2 DTEA	Data Trace End Address. The DTEA field is the end address for MCSA data trace window.

49.4.1.37 MCSA data trace address range 2 register (GTMDI_MCSA_DTAR2)

The GTMDI_MCSA_DTAR2 register shown in *Figure 606* defines a data trace address range. The address range is used by the MCSA and enables the range for reads and writes access to data only. If the start address value is greater than the end address value, all accesses are considered to have the address value outside to the data trace window 1.

Index: 72 Access: Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	DTSA [13:2]												0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	DTEA [13:2]												0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 606. MCSA data trace address range 2 register (GTMDI_MCSA_DTAR2)

Note: DTSA and DTEA fields represent word addresses which are included in the address range.

Table 752 describes the GTMDI_MCSA_DTAR2 register functions.

Table 752. GTMDI_MCSA_DTAR2 field descriptions

Field	Description
29:18 DTSA	Data Trace Start Address. The DTSA field is the start address for MCSA data trace window.
13:2 DTEA	Data Trace End Address. The DTEA field is the end address for MCSA data trace window.

49.4.1.38 MCSB development control register (GTMDI_MCSB_DC)

The GTMDI_MCSB_DC register shown in *Figure 607* controls various trace and debug features.

Index: 77 Access: Read/Write

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

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	MCSB_SEL				0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 607. MCSB development control register (GTMDI_MCSB_DC)

Table 753 describes the GTMDI_MCSB_DC register functions.

Table 753. GTMDI_MCSB_DC field descriptions

Field	Description
11:8 MCSB_SEL	MCSB Selection field. The MCSB_SEL[3:0] selection field selects which one of the MCS cores in the GTM module are assigned as MCSB. 0000 MCS0 0001 MCS1 0010 MCS2 0011 MCS3 0100 MCS4 0101 Reserved 0110 Reserved 0111 Reserved 1000 Reserved 1001 Reserved 1010 Reserved 1011 Reserved 1100 Reserved 1101 Reserved 1110 Reserved 1111 Reserved

49.4.1.39 MCSB watchpoint control register (GTMDI_MCSB_WPC)

The MCSB Watchpoint Control Register shown in *Figure 608* is used to configure MCSB watchpoints. This register can be configured for halt/watchpoint based on instruction fetch, data read or data write, with address and data masking options.

Note: Start/Stop signal inputs to the GTMDI can overwrite the state of the WMC control bit thus controlling the Watchpoint messages behavior. In case of JTAG modifies the WMC bit and Start/Stop signals occur at the same time also modifying the same bit, Start/Stop signals have precedence over JTAG writes. The SEN control bit needs to be asserted in order to allow external Start/Stop signals to control the WMC bit.

Note: The watchpoint channel selection is made through the GTMDI_MCSB_CE by the WPCE field.

Index: 79 Access: Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	HACH		0	WMC	0	0	HME			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	HWO		0	0	0	0	0	EOC	HEN	0	STSEL		0	0	TEN	SEN
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 608. MCSB watchpoint control register (GTMDI_MCSB_WPC)

Table 754 describes the GTMDI_MCSB_WPC register functions.

Table 754. GTMDI_MCSB_WPC field descriptions

Field	Description
25:24 HACH	Halt/Watchpoint on Active Channel. The HACH field selects which type of MCS RAM access generates a watchpoint or halt. Combining this field with HWO gives GTMDI the capability of monitoring channel activity as well as accesses to specific addresses or data. 00 Match on Instruction Fetch 01 Match on Data Read 10 Match on Data Write 11 Match on Data Read or Write
22 WMC	Watchpoint Message Control. The WMC Watchpoint Message Control bit controls the watchpoint message sent through the Message Data bus. If this bit is asserted it enables MCSA/B Watchpoints to generate a watchpoint message. This bit is written by the JTAG interface but can also be controlled at SoC level with dedicated Start/Stop signal inputs to GTMDI. 0 Watchpoint Messages disabled 1 Watchpoint Messages enabled

Table 754. GTMDI_MCSB_WPC field descriptions (continued)

Field	Description
19:16 HME	Halt/Watchpoint Data Mask Enable. The HME field selects which data bytes are used for hardware halt/watchpoint generation when comparing to HWD register. This register field is only meaningful if HWO bit field is [x1], otherwise the data comparison is disabled independent of HME value. 0000 All bytes of data compared xxx1 Least significant byte of data masked out xx1x Second least significant byte of data masked out x1xx Second most significant byte of data masked out 1xxx Most significant byte of data masked out
15:14 HWO	Halt/Watchpoint Operand. The HWO field selects if address or data matching (or both) is done. Registers WPA and WPD are used for address and data comparison, respectively. If address is not compared WPA is not use. If data is not compared then WPD is not used. 00 Comparison Disabled for address or data x1 Compare data with HWD field 1x Compare address with HWA field
8 EOC	$\overline{\text{EVTO}}$ Control. The EOC bit controls the indication on the $\overline{\text{EVTO}}$ output. See *Figure 611*, *Figure 612*, *Figure 613* and *Figure 614* for definition about watchpoint 1 and 2. 0 Indicates Watchpoint 2 occurrence 1 Indicates Watchpoint 1 occurrence
7 HEN	Halt Enable. The HEN Halt Enable bit controls if the MCSA/B Watchpoint is enabled to HALT the GTM IP module. If this bit is set and watchpoint occurs on the MCSA/B a Halt signal is generated to the GTM IP which enters debug state. 0 Disables Halt GTM 1 Enables Halt GTM
5:4 STSEL	Start/Stop input signal selection. Selects the Start/Stop input that is used to control the watchpoint messages. These signals are actually a pair of signals. 00 SHARED_WPT_ST[0] 01 SHARED_WPT_ST[1] 10 SHARED_WPT_ST[2] 11 SHARED_WPT_ST[3]
1 TEN	Watchpoint Trigger Enable.The TEN field controls if a external trigger is issued by the MCSB when a watchpoint occurs. 0 Disables Trigger Generation 1 Enables Trigger Generation
0 SEN	Start/Stop Enable. The SEN bit field enables the MCSB Watchpoint message to consider the Start/Stop inputs for watchpoint message transmission. If this bit is cleared, the messages are controlled only by JTAG writes to the WMC Watchpoint Trace Message Control bit. 0 Disables trace control from Start/Stop inputs 1 Enables trace control from Start/Stop inputs

49.4.1.40 MCSB program fetch trace control register (GTMDI_MCSB_PTC)

The GTMDI_MCSB_PTC register shown in *Figure 609* allows program fetch traces to be enabled or disabled on the occurrence of a watchpoint. If the same watchpoint is programed to enable and disable a trace, the occurrence of the watchpoint toggles the current value of the trace flags. If one watchpoint occurs to enable trace and a different watchpoint occurs to disable trace at the same time the current value of the trace flags also toggle.

Index: 80 Access: Read/Write

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

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	PFTS		0	0	PFTE		0	FTC	0	0	0	0	0	SEN
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 609. MCSB program fetch trace control register (GTMDI_MCSB_PTC)

Table 755 describes the GTMDI_MCSB_PTC register functions.

Table 755. GTMDI_MCSB_PTC field descriptions

Field	Description
13:12 PFTS	Program Fetch Trace Start. This bit field allows Program fetch trace to be enabled at a watchpoint occurrence. See *Figure 611*, *Figure 612*, *Figure 613* and *Figure 614* for definition about watchpoint 1 and 2. 00 No watchpoint selected (do not start Program Fetch Trace) 01 Program Fetch Trace enabled on MCSB Program Watchpoint 1 occurrence 10 Program Fetch Trace enabled on MCSB Program Watchpoint 2 occurrence 11 Reserved
9:8 PFTE	Program Fetch Trace End. This bit field allows program trace to be disabled at a watchpoint occurrence. 00 No watchpoint selected 01 Program Fetch Trace disabled on MCSB Program Watchpoint 1 occurrence 10 Program Fetch Trace disabled on MCSB Program Watchpoint 2 occurrence 11 Reserved
6 FTC	Fetch Trace Control. This bit controls the Fetch Trace message sent through the Message Data bus. If this bit is asserted it enables MCSB Fetch access to generate a trace message. This bit is written by the JTAG interface but can also be controlled at SoC level with dedicated Start/Stop signal inputs to GTMDI 0 Fetch Messages disabled 1 Fetch Messages enabled
0 SEN	Start/Stop Enable. The SEN bit field enables the MCSB Fetch message to consider the Start/Stop inputs for message transmission. If this bit is cleared, the messages are controlled only by JTAG writes to the FTC Fetch Trace Message Control bit. 0 Disable fetch trace control from Start/Stop inputs 1 Enable fetch trace control from Start/Stop inputs

49.4.1.41 MCSB data trace control register (GTMDI_MCSB_DTC)

The GTMDI_MCSB_DTC register shown in *Figure 610* controls which address ranges are enabled for data trace, and if reads and writes are traced in that range.

Index: 81 Access: Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	RC0	RC1
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	RWT0		0	0	RWT1		0	DTC	DTS		0	DTE		SEN
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 610. MCSB data trace control register (GTMDI_MCSB_DTC)

Table 756 describes the GTMDI_MCSB_DTC register functions.

Table 756. GTMDI_MCSB_DTC field descriptions

Field	Description
17 RC0	Range Control 0. Controls the range on data trace window 0. The window 0 is defined by DTAR1 register. See *Figure 616*. 0 Trace addresses outside (exclusive) of data trace window 0 1 Trace address inside (inclusive) of data trace window 0
16 RC1	Range Control 1. Controls the range on data trace window 1. The window 1 is defined by DTAR2 register. See *Figure 617*. 0 Trace addresses outside (exclusive) of data trace window 1 1 Trace address inside (inclusive) of data trace window 1
13:12 RWT0	MCSB Read/Write Trace Control 0. RWT0 controls whether data trace messages are generated for MCSB RAM read or write accesses. 00 No MCSB data trace messages generated for data trace window 0 01 Enable MCSB data read trace for MCS window 0 10 Enable MCSB data write trace for MCS window 0 11 Enable MCSB data read and write trace for data trace window 0
9:8 RWT1	MCSB Read/Write Trace Control 1. RWT1 controls whether data trace messages are generated for MCSB accesses inside MCS data trace window, and if so, whether reads, writes, or both generate the data trace messages. 00 No MCSB data trace messages generated for data trace window 1 01 Enable MCSB data read trace for MCS window 1 10 Enable MCSB data write trace for MCS window 1 11 Enable MCSB data read and write trace for data trace window 1
6 DTC	Data Trace Control. This bit controls the Data Trace message sent through the Message Data bus. If this bit is asserted it enables MCSB Data access to generate a data trace message. This bit is written by the JTAG interface but can also be controlled at SoC level with dedicated Start/Stop signal inputs to GTMDI 0 Data Messages disabled 1 Data Messages enabled

Table 756. GTMDI_MCSB_DTC field descriptions (continued)

Field	Description
5:4 DTS	Data Trace Start. This bit field allows data trace to be enabled at a watchpoint occurrence. 00 No watchpoint selected 01 Data Trace enabled on MCSB Watchpoint 1 occurrence 10 Data Trace enabled on MCSB Watchpoint 2 occurrence 11 Reserved
2:1 DTE	Data Trace End. This bit field allows data trace to be disabled at a watchpoint occurrence. 00 No watchpoint selected 01 Data Trace disabled on MCSB Watchpoint 1 occurrence 10 Data Trace disabled on MCSB Watchpoint 2 occurrence 11 Reserved
0 SEN	Start/Stop Enable. The SEN bit field enables the MCSA/B Data message to consider the Start/Stop inputs for message transmission. If this bit is cleared, the messages are controlled only by JTAG writes to the DTC Data Trace Message Control bit. 0 Disable data trace control from Start/Stop inputs 1 Enable data trace control from Start/Stop inputs

49.4.1.42 MCSB watchpoint address 1 register (GTMDI_MCSB_WPA1)

The MCSB Watchpoint Address Registers 1 shown in *Figure 611* are used to configure the MCSB halt/watchpoint hardware. They are used for data and instruction fetch accesses.

Index: 82 Access: Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	HWAM													
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	HWA													
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 611. MCSB watchpoint address 1 register (GTMDI_MCSB_WPA1)

Note: The HWA field represents word addresses relative to the MCS address map, not the CPU address map. This applies to GTMDI_MCSB_WPA1 and GTMDI_MCSB_WPA2 registers.

Table 757 describes the GTMDI_MCSB_WPA1 register functions.

Table 757. GTMDI_MCSB_WPA1 field descriptions

Field	Description
29:16 HWAM	Halt/Watchpoint Address Mask. The HWAM field is used to select which address bits are compared for halt and watchpoint generation. Address match = (HWA&HWAM) == (Address & HWAM);
13:0 HWA	Halt/Watchpoint Address. The HWA field is used for comparison with instruction or data ddresses. The HWA field represents a word address.

49.4.1.43 MCSB watchpoint address 2 register (GTMDI_MCSB_WPA2)

The MCSB Watchpoint Address 2 Registers shown in *Figure 612* are used to configure the MCSB halt/watchpoint hardware. They are used for data and instruction fetch accesses.

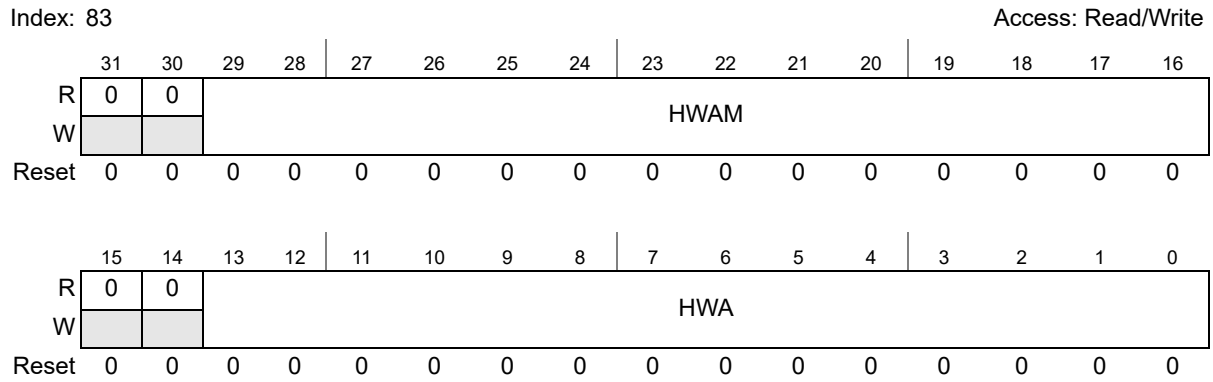


Figure 612. MCSB watchpoint address 2 register (GTMDI_MCSB_WPA2)

Table 758 describes the GTMDI_MCSB_WPA2 register functions.

Table 758. GTMDI_MCSB_WPA2 field descriptions

Field	Description
29:16 HWAM	Halt/Watchpoint Address Mask. The HWAM field is used to select which address bits are compared for halt and watchpoint generation. Address match = (HWA&HWAM) == (Address & HWAM);
13:0 HWA	Halt/Watchpoint Address. The HWA field is used for comparison with instruction or data addresses. The HWA field represents a word address.

49.4.1.44 MCSB watchpoint data 1 register (GTMDI_MCSB_WPD1)

The MCSB Halt/Watchpoint Data 1 Register shown in *Figure 613* is used to configure the MCSB halt/watchpoint hardware. It is used for data accesses selection only.

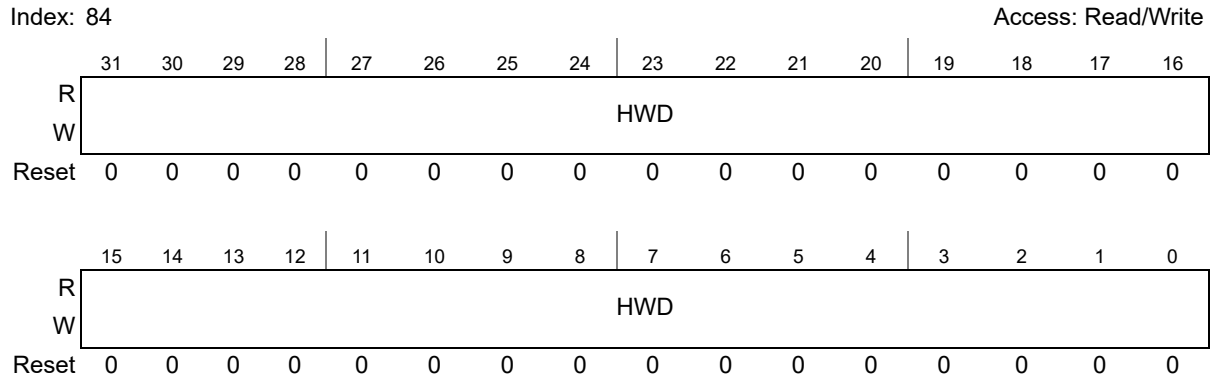


Figure 613. MCSB watchpoint data 1 register (GTMDI_MCSB_WPD1)

Table 759 describes the GTMDI_MCSB_WPD1 register functions.

Table 759. GTMDI_MCSB_WPD1 field descriptions

Field	Description
31:0 HWD	Halt/Watchpoint Data. The HWD field is used to compare data operands in a RAM read or write access. Depending upon the size of the access, only certain bits of HWD are compared against the the RAM data. If the operation size is 8-bits, the 8 most significant bits of HWD are compared against the RAM data. If the operation size is 24-bits, the 24 least significant bits of HWD are compared against the RAM data. If the operation size is 32-bits, all bits from HWD are compared against the RAM data.

49.4.1.45 MCSB watchpoint data 2 register (GTMDI_MCSB_WPD2)

The MCSB Watchpoint Data 2 Register shown in *Figure 614* is used to configure the MCSB halt/watchpoint hardware. It is used for data accesses selection only.

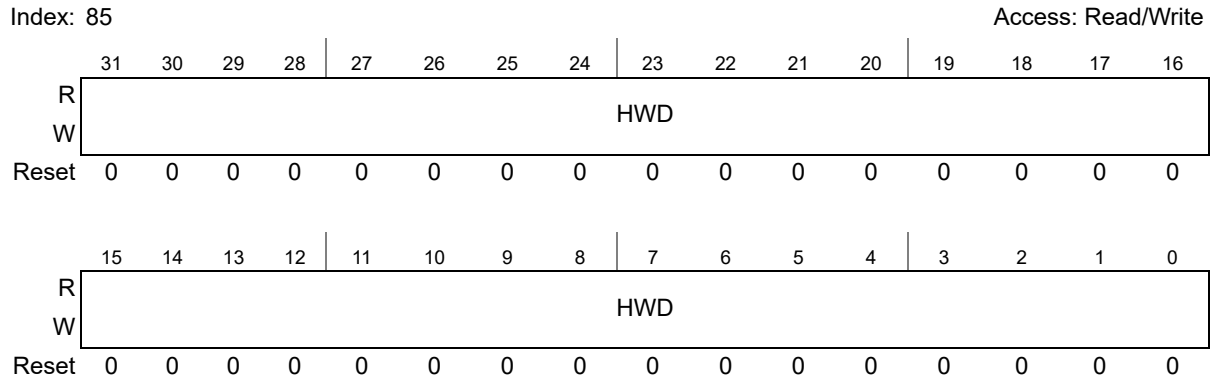


Figure 614. MCSB watchpoint data 2 register (GTMDI_MCSB_WPD2)

Table 760 describes the GTMDI_MCSB_WPD2 register functions.

Table 760. GTMDI_MCSB_WPD2 field descriptions

Field	Description
31:0 HWD	Halt/Watchpoint Data. The HWD field is used to compare data operands in a RAM read or write access. Depending upon the size of the access, only certain bits of HWD are compared against the RAM data. If the operation size is 8-bits, the 8 most significant bits of HWD are compared against the RAM data. If the operation size is 24-bits, the 24 least significant bits of HWD are compared against the RAM data. If the operation size is 32-bits, all bits from HWD are compared against the RAM data.

49.4.1.46 MCSB channel enable register (GTMDI_MCSB_CE)

The GTMDI_MCSB_CE register shown in *Figure 615* enables the monitoring for 8 MCSB channels independently.

Index: 86 Access: Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	CFTE								CDTE							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	WPCE							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 615. MCSB channel enable register (GTMDI_MCSB_CE)

Table 761 describes the GTMDI_MCSB_CE register functions.

Table 761. GTMDI_MCSB_CE field descriptions

Field	Description
31:24 CFTE	MCSB Channel Fetch Trace Enable. The CFTE[7:0] is an one hot bit field which defines the channel or channels to be monitored for fetch traces. If enabled, the corresponding channel Fetches are monitored by Fetch Trace messages. Start and Stop message controls are controlled by the FTC bit in the GTMDI_MCSB_PTC registers (*Section 49.4.1.40: MCSB program fetch trace control register (GTMDI_MCSB_PTC)*. 00000000No channel selected for trace xxxxxxx1Channel [0] selected xxxxxx1xChannel [1] selected xxxxx1xxChannel [2] selected xxxx1xxxChannel [3] selected xxx1xxxxChannel [4] selected xx1xxxxxChannel [5] selected x1xxxxxxChannel [6] selected 1xxxxxxxChannel [7] selected 11111111All Channels selected
23:16 CDTE	MCSB Channel Data Trace Enable. The CDTE[7:0] is an one hot bit field which defines the channel or channels to be monitored for data trace messages. If enabled, the corresponding channel Data Accesses are monitored by Data Trace messages. Start and Stop message are controlled by the DTC bit in the GTMDI_MCSB_DTC registers (*Section 49.4.1.41: MCSB data trace control register (GTMDI_MCSB_DTC)* and by the Data trace window. 00000000No channel selected for trace xxxxxxx1Channel [0] selected xxxxxx1xChannel [1] selected xxxxx1xxChannel [2] selected xxxx1xxxChannel [3] selected xxx1xxxxChannel [4] selected xx1xxxxxChannel [5] selected x1xxxxxxChannel [6] selected 1xxxxxxxChannel [7] selected 11111111 All Channels selected

Table 761. GTMDI_MCSB_CE field descriptions (continued)

Field	Description
7:0 WPCE	MCSB Channel Enable. The WPCE[7:0] is a one hot bit field which defines the channel or channels to be monitored for watchpoint, data and fetch traces. The messages controlled by this bit must be also enabled and disabled according to occurrence of Watchpoints and Address Ranges. 00000000No channel selected for trace xxxxxxx1Channel [0] selected xxxxxx1xChannel [1] selected xxxxx1xxChannel [2] selected xxxx1xxxChannel [3] selected xxx1xxxxChannel [4] selected xx1xxxxxChannel [5] selected x1xxxxxxChannel [6] selected 1xxxxxxxChannel [7] selected 11111111 All Channels selected

49.4.1.47 MCSB data trace address range 1 register (GTMDI_MCSB_DTAR1)

The GTMDI_MCSB_DTAR1 register shown in *Figure 616* defines a data trace address range. The address range is used by the MCSB and enables the range for reads and writes access to data only. If the start address value is greater than the end address value, all accesses are considered to have the address value outside to the data trace window 0.

Index: 88 Access: Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	DTSA [13:2]												0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	DTEA [13:2]												0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 616. MCSB data trace address range 1 register (GTMDI_MCSB_DTAR1)

Note: The DTSA and DTEA fields represent word addresses relative to the MCS address map, not the CPU address.

Table 762 describes the GTMDI_MCSB_DTAR1 register functions.

Table 762. GTMDI_MCSB_DTAR1 field descriptions

Field	Description
29:18 DTSA	Data Trace Start Address. The DTSA field is the start address for MCSB data trace window.
13:2 DTEA	Data Trace End Address. The DTEA field is the end address for MCSB data trace window.

49.4.1.48 MCSB data trace address range 2 register (GTMDI_MCSB_DTAR2)

The GTMDI_MCSA_DTAR2 register shown in *Figure 617* defines a data trace address range. The address range is used by the MCSB and enables the range for reads and writes access to data only. If the start address value is greater than the end address value, all accesses are considered to have the address value outside to the data trace window 1.

Index: 89 Access: Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	DTSA [13:2]												0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	DTEA [13:2]												0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 617. MCSB data trace address range 2 register (GTMDI_MCSB_DTAR2)

Note: DTSA and DTEA fields represent word addresses which are included in the address range.

Table 763 describes the GTMDI_MCSB_DTAR2 register functions.

Table 763. GTMDI_MCSB_DTAR2 field descriptions

Field	Description
29:18 DTSA	Data Trace Start Address. The DTSA field is the start address for MCSB data trace window.
13:2 DTEA	Data Trace End Address. The DTEA field is the end address for MCSB data trace window.

49.4.1.49 TBU0 watchpoint control 1 register (GTMDI_TBU0_WPC1)

The GTMDI_TBU0_WPC1 register shown in *Figure 618* controls the Watchpoint issued by TBU0 GTM Time Base Units (TBU). These watchpoints are used to control the requests for GTM to enter Halt state, issue watchpoint messages and also to control logic external to the GTMDI at SoC level such as in the SPU interface.

Note: Start/Stop signal inputs to the GTMDI can overwrite the state of the WMC1 and WMC2 control bits thus controlling the Watchpoint messages behavior. In case of JTAG modifies the WMC1 or WMC2 bits and Start/Stop signals occur at the same time also modifying the same bits, Start/Stop signals have precedence over JTAG writes. The SEN1 and SEN2 control bits from the GTMDI_TBU0_WPC2 register need to be set in order to allow external Start/Stop signals to control the WMC1/2 bits.

Index: 94 Access: Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	TSS1			0	0	0	0	HEN1	WMC1	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	TSS2			0	0	0	0	HEN2	WMC2	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 618. TBU0 watchpoint control 1 register (GTMDI_TBU0_WPC1)

Table 764 describes the GTMDI_TBU0_WPC1 register functions.

Table 764. GTMDI_TBU0_WPC1 field descriptions

Field	Description
30:28 TSS1	TBU0 Watchpoint 1 Compare Selection. The TSS1[2:0] field controls the type of comparison used for TBU0 watchpoint generation. 000 TBU0 24-bit update (do not compare data) 001 TBU 27-bit update (do not compare data) 010 Compare TBU0 value in 24-bit mode with expected data(1) 011 Compare TBU0 value in 27-bit mode with expected data 100 Reserved 101 Reserved 110 Reserved 111 Reserved
23 HEN1	Halt Enable 1. The HEN1 Halt Enable 1 bit controls if the TBU0 event is enabled to HALT the GTM module. If this bit is set and an event occurs on the TBU0 a Halt signal is generated to the GTM which enters debug state. 0 Disables Halt GTM 1 Enables Halt GTM
22 WMC1	Watchpoint Message Control 1. The WMC1 Watchpoint Message Control 1 bit controls the watchpoint message sent through the Message Data bus. If this bit is set it enables valid matches or increment on selected TBU0 to generate a watchpoint message. This bit is written by the JTAG interface but can also be controlled at SoC level with dedicated Start/Stop signal inputs to GTMDI. 0 Watchpoint Messages disabled 1 Watchpoint Messages enabled
14:12 TSS2	TBU0 Watchpoint 2 Compare Selection 2. The TSS2[2:0] field controls the type of comparison used for TBU0 watchpoint generation. 000 TBU0 24-bit update (do not compare data) 001 TBU 27-bit update (do not compare data) 010 Compare TBU0 value in 24-bit mode with expected data1 011 Compare TBU0 value in 27-bit mode with expected data 100 Reserved 101 Reserved 110 Reserved 111 Reserved

Table 764. GTMDI_TBU0_WPC1 field descriptions (continued)

Field	Description
7 HEN2	Halt Enable 2. The HEN2 Halt Enable 2 bit controls if the TBU0 valid event, which is a match or increment, is enabled to HALT the GTM module. If this bit is set and an event occurs on the TBU0, a Halt signal is generated to the GTM which enters Halt state. 0 Disables Halt GTM 1 Enables Halt GTM
6 WMC2	Watchpoint Message Control 2. The WMC2 Watchpoint Message Control 2 bit controls the watchpoint message sent through the Message Data bus. If this bit is set it enables match or increment on TBU0 to generate a watchpoint message. This bit is written by the JTAG interface but can also be controlled at SoC level with dedicated Start/Stop signal inputs to GTMDI. 0Watchpoint Messages disabled 1 Watchpoint Messages enabled

1. Only bits from 26 to 3 from register GTMDI_TBU0_DATA are used for comparison

49.4.1.50 TBU0 watchpoint control 2 register (GTMDI_TBU0_WPC2)

The GTMDI_TBU0_WPC2 shown in *Figure 619* register controls the Watchpoint Triggers issued by TBU0. These triggers are used for GTM to enter Halt state, generate watchpoint messages and control logic external to GTMDI, such as in the SPU interface.

Index: 95 Access: Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	WTSEL1			0	0	0	0	0	0	STSEL1		0	0	TEN1	SEN1
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	WTSEL2			0	0	0	0	0	0	STSEL2		0	0	TEN2	SEN2
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 619. TBU0 watchpoint control 2 register (GTMDI_TBU0_WPC2)

Table 765 describes the GTMDI_TBU0_WPC2 register functions.

Table 765. GTMDI_TBU0_WPC2 field descriptions

Field	Description
30:28 WTSEL1	Watchpoint Trigger Selection 1. The WTSEL1[2:0] field selects among 8 available watchpoint triggers which one is connected to the TBU0 selected event, match or increment. Other modules can selected the same WTSEL1 value, in this case the result selection is an OR of all selected sources. 000 SHARED_WPT[0] 001 SHARED_WPT[1] 010 SHARED_WPT[2] 011 SHARED_WPT[3] 100 SHARED_WPT[4] 101 SHARED_WPT[5] 110 SHARED_WPT[6] 111 SHARED_WPT[7]
21:20 STSEL1	Start/Stop input signal selection 1. Selects the Start/Stop input that is used by the watchpoint trace messages. These signals are actually a pair of signals. 00 SHARED_WPT_ST[0] 01 SHARED_WPT_ST[1] 10 SHARED_WPT_ST[2] 11 SHARED_WPT_ST[3]
17 TEN1	Watchpoint Trigger Enable 1.The TEN1 field controls if a watchpoint external trigger is issued by the TBU0 selected event, match or increment. 0 Disable Trigger Generation 1 Enable Trigger Generation
16 SEN1	Start/Stop Enable 1. The SEN1 bit field enables the TBU0 selected event to consider the Start/Stop inputs for watchpoint trace message generation. If this bit is cleared, the messages are controlled only by JTAG writes to the WMC1 field in the GTMDI_TBU0_WPC1 register. 0 Disable trace control from Start/Stop inputs 1 Enable trace control from Start/Stop inputs
14:12 WTSEL2	Watchpoint Trigger Output Selection 2. The WTSEL2[2:0] field selects among 8 available watchpoint triggers which one is connected to the TBU0 selected event. Other modules can selected the same WTSEL2 value, in this case the result signal is an OR of all selected sources. 000 SHARED_WPT[0] 001 SHARED_WPT[1] 010 SHARED_WPT[2] 011 SHARED_WPT[3] 100 SHARED_WPT[4] 101 SHARED_WPT[5] 110 SHARED_WPT[6] 111 SHARED_WPT[7]
5:4 STSEL2	Start/Stop input signal selection 2. Selects the Start/Stop input that is used the watchpoint trace messages. These signals are actually a pair of signals. 00 SHARED_WPT_ST[0] 01 SHARED_WPT_ST[1] 10 SHARED_WPT_ST[2] 11 SHARED_WPT_ST[3]

Table 765. GTMDI_TBU0_WPC2 field descriptions (continued)

Field	Description
1 TEN2	Watchpoint Trigger Enable 2. The TEN1 field controls if a watchpoint external trigger is issued by the TBU0 selected event. 0 Disable Trigger Generation 1 Enable Trigger Generation
0 SEN2	Start/Stop Enable 2. The SEN2 bit field enable the TBU0 selected event, match or increment to consider the Start/Stop inputs for watchpoint trace message generation. If this bit is cleared, the messages are controlled only by JTAG writes to the WMC2 field in the GTMDI_TBU0_WPC1 register. 0 Disable trace control from Start/Stop inputs 1 Enable trace control from Start/Stop inputs

49.4.1.51 TBU0 watchpoint DATA register (GTMDI_TBU0_DATA)

The TBU0 Watchpoint DATA shown in *Figure 620* is used for watchpoints generation for matching with the TBU0 value.

Index: 96 Access: Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	TBU0_DATA										
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	TBU0_DATA															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 620. TBU0 watchpoint DATA register (GTMDI_TBU0_DATA)

Table 766 describes the GTMDI_TBU0_DATA register functions.

Table 766. GTMDI_TBU0_DATA field descriptions

Field	Description
26:0 TBU0_DATA	TBU0 Watchpoint DATA. The field TBU0_DATA is used to compare against the TBU0 data.

49.4.1.52 TBU1 watchpoint control 1 register (GTMDI_TBU1_WPC1)

The GTMDI_TBU1_WPC1 register shown in *Figure 621* controls the Watchpoint issued by TBU1 GTM Time Base Units. These watchpoints are used to control the requests for GTM to enter Halt state, issue watchpoint messages and also to control logic external to the GTMDI at SoC level such as in the SPU interface.

Note: Start/Stop signal inputs to the GTMDI can overwrite the state of the WMC1 and WMC2 control bits thus controlling the Watchpoint messages behavior. In case of JTAG modifies the WMC1 or WMC2 bits and Start/Stop signals occur at the same time also modifying the

same bits, Start/Stop signals have precedence over JTAG writes. The SEN1 and SEN2 control bits from the GTMDI_TBU1_WPC2 register need to be set in order to allow external Start/Stop signals to control the WMC1/2 bits.

Index: 101 Access: Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	TSS1		0	0	0	0	HEN1	WMC1	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	TSS2		0	0	0	0	HEN2	WMC2	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 621. TBU1 watchpoint control 1 register (GTMDI_TBU1_WPC1)

Table 767 describes the GTMDI_TBU1_WPC1 register functions.

Table 767. GTMDI_TBU1_WPC1 field descriptions

Field	Description
29:28 TSS1	TBU1 Watchpoint 1 Compare Selection. The TSS1[1:0] field controls the type of comparison used for TBU1 watchpoint generation. 00 TBU1 update (do not compare data) 01 Compare TBU1 value with expected data 10 Reserved 11 Reserved
23 HEN1	Halt Enable 1. The HEN1 Halt Enable 1 bit controls if the TBU1 event is enabled to HALT the GTM module. If this bit is set and an event occurs on the TBU1 a Halt signal is generated to the GTM which enters debug state. 0 Disables Halt GTM 1 Enables Halt GTM
22 WMC1	Watchpoint Message Control 1. The WMC1 Watchpoint Message Control 1 bit controls the watchpoint message sent through the Message Data bus. If this bit is set it enables valid matches or increment on selected TBU1 to generate a watchpoint message. This bit is written by the JTAG interface but can also be controlled at SoC level with dedicated Start/Stop signal inputs to GTMDI. 0 Watchpoint Messages disabled 1 Watchpoint Messages enabled
13:12 TSS2	TBU1 Watchpoint 2 Compare Selection 2. The TSS2[1:0] field controls the type of comparison used for TBU1 watchpoint generation. 00 TBU1 update (do not compare data) 01 Compare TBU1 value with expected data 10 Reserved 11 Reserved

Table 767. GTMDI_TBU1_WPC1 field descriptions (continued)

Field	Description
7 HEN2	Halt Enable 2. The HEN2 Halt Enable 2 bit controls if the TBU1 valid event, which is a match or increment, is enabled to HALT the GTM module. If this bit is set and an event occurs on the TBU1, a Halt signal is generated to the GTM which enters Halt state. 0 Disables Halt GTM 1 Enables Halt GTM
6 WMC2	Watchpoint Message Control 2. The WMC2 Watchpoint Message Control 2 bit controls the watchpoint message sent through the Message Data bus. If this bit is set it enables match or increment on TBU1 to generate a watchpoint message. This bit is written by the JTAG interface but can also be controlled at SoC level with dedicated Start/Stop signal inputs to GTMDI. 0Watchpoint Messages disabled 1 Watchpoint Messages enabled

49.4.1.53 TBU1 watchpoint control 2 register (GTMDI_TBU1_WPC2)

The GTMDI_TBU1_WPC2 register shown in *Figure 622* controls the Watchpoint Triggers issued by TBU1. These triggers are used for GTM to enter Halt state, generate watchpoint messages and control logic external to GTMDI, such as in the SPU interface.

Index: 102 Access: Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	WTSEL1			0	0	0	0	0	0	STSEL1		0	0	TEN1	SEN1
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	WTSEL2			0	0	0	0	0	0	STSEL2		0	0	TEN2	SEN2
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 622. TBU1 watchpoint control 2 register (GTMDI_TBU1_WPC2)

Table 768 describes the GTMDI_TBU1_WPC2 register functions.

Table 768. GTMDI_TBU1_WPC2 field descriptions

Field	Description
30:28 WTSEL1	Watchpoint Trigger Selection 1. The WTSEL1[2:0] field selects among 8 available watchpoint triggers which one is connected to the TBU1 selected event, match or increment. Other modules can selected the same WTSEL1 value, in this case the result selection is an OR of all selected sources. 000 SHARED_WPT[0] 001 SHARED_WPT[1] 010 SHARED_WPT[2] 011 SHARED_WPT[3] 100 SHARED_WPT[4] 101 SHARED_WPT[5] 110 SHARED_WPT[6] 111 SHARED_WPT[7]
21:20 STSEL1	Start/Stop input signal selection 1. Selects the Start/Stop input that is used by the watchpoint trace messages. These signals are actually a pair of signals. 00 SHARED_WPT_ST[0] 01 SHARED_WPT_ST[1] 10 SHARED_WPT_ST[2] 11 SHARED_WPT_ST[3]
17 TEN1	Watchpoint Trigger Enable 1.The TEN1 field controls if a watchpoint external trigger is issued by the TBU1 selected event, match or increment. 0 Disable Trigger Generation 1 Enable Trigger Generation
16 SEN1	Start/Stop Enable 1. The SEN1 bit field enables the TBU1 selected event to consider the Start/Stop inputs for watchpoint trace message generation. If this bit is cleared, the messages are controlled only by JTAG writes to the WMC1 field in the GTMDI_TBU1_WPC1 register. 0 Disable trace control from Start/Stop inputs 1 Enable trace control from Start/Stop inputs
14:12 WTSEL2	Watchpoint Trigger Output Selection 2. The WTSEL2[2:0] field selects among 8 available watchpoint triggers which one is connected to the TBU1 selected event. Other modules can selected the same WTSEL2 value, in this case the result signal is an OR of all selected sources. 000 SHARED_WPT[0] 001 SHARED_WPT[1] 010 SHARED_WPT[2] 011 SHARED_WPT[3] 100 SHARED_WPT[4] 101 SHARED_WPT[5] 110 SHARED_WPT[6] 111 SHARED_WPT[7]
5:4 STSEL2	Start/Stop input signal selection 2. Selects the Start/Stop input that is used the watchpoint trace messages. These signals are actually a pair of signals. 00 SHARED_WPT_ST[0] 01 SHARED_WPT_ST[1] 10 SHARED_WPT_ST[2] 11 SHARED_WPT_ST[3]

Table 768. GTMDI_TBU1_WPC2 field descriptions (continued)

Field	Description
1 TEN2	Watchpoint Trigger Enable 2. The TEN1 field controls if a watchpoint external trigger is issued by the TBU1 selected event. 0 Disable Trigger Generation 1 Enable Trigger Generation
0 SEN2	Start/Stop Enable 2. The SEN2 bit field enable the TBU1 selected event, match or increment to consider the Start/Stop inputs for watchpoint trace message generation. If this bit is cleared, the messages are controlled only by JTAG writes to the WMC2 field in the GTMDI_TBU1_WPC1 register. 0 Disable trace control from Start/Stop inputs 1 Enable trace control from Start/Stop inputs

49.4.1.54 TBU1 watchpoint DATA register (GTMDI_TBU1_DATA)

The TBU1 Watchpoint DATA shown in *Figure 623* is used for watchpoints generation for matching with the TBU1 value.

Index: 103 Access: Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	TBU1_DATA							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	TBU1_DATA															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 623. TBU1 watchpoint DATA register (GTMDI_TBU1_DATA)

Table 769 describes the GTMDI_TBU1_DATA register functions.

Table 769. GTMDI_TBU1_DATA field descriptions

Field	Description
23:0 TBU1_DATA	TBU1 Watchpoint DATA. The field TBU1_DATA[23:0] is used to compare against the TBU1 data.

49.4.1.55 TBU2 watchpoint control 1 register (GTMDI_TBU2_WPC1)

The GTMDI_TBU2_WPC1 register shown in *Figure 624* controls the Watchpoint issued by TBU2 GTM Time Base Units. These watchpoints are used to control the requests for GTM to enter Halt state, issue watchpoint messages and also to control logic external to the GTMDI at SoC level such as in the SPU interface.

Note: Start/Stop signal inputs to the GTMDI can overwrite the state of the WMC1 and WMC2 control bits thus controlling the Watchpoint messages behavior. In case of JTAG modifies the WMC1 or WMC2 bits and Start/Stop signals occur at the same time also modifying the

same bits, Start/Stop signals have precedence over JTAG writes. The SEN1 and SEN2 control bits from the GTMDI_TBU2_WPC2 register need to be set in order to allow external Start/Stop signals to control the WMC1/2 bits.

Index: 108 Access: Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	TSS1		0	0	0	0	HEN1	WMC1	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	TSS2		0	0	0	0	HEN2	WMC2	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 624. TBU2 watchpoint control 1 register (GTMDI_TBU2_WPC1)

Table 770 describes the GTMDI_TBU2_WPC1 register functions.

Table 770. GTMDI_TBU2_WPC1 field descriptions

Field	Description
29:28 TSS1	TBU2 Watchpoint 2 Compare Selection. The TSS1[1:0] field controls the type of comparison used for TBU2 watchpoint generation. 00 TBU2 update (do not compare data) 01 Compare TBU2 value with expected data 10 Reserved 11 Reserved
23 HEN1	Halt Enable 1. The HEN1 Halt Enable 1 bit controls if the TBU2 event is enabled to HALT the GTM module. If this bit is set and an event occurs on the TBU2 a Halt signal is generated to the GTM which enters debug state. 0 Disables Halt GTM 1 Enables Halt GTM
22 WMC1	Watchpoint Message Control 1. The WMC1 Watchpoint Message Control 1 bit controls the watchpoint message sent through the Message Data bus. If this bit is set it enables valid matches or increment on selected TBU2 to generate a watchpoint message. This bit is written by the JTAG interface but can also be controlled at SoC level with dedicated Start/Stop signal inputs to GTMDI. 0 Watchpoint Messages disabled 1 Watchpoint Messages enabled
13:12 TSS2	TBU2 Watchpoint 2 Compare Selection 2. The TSS2[1:0] field controls the type of comparison used for TBU2 watchpoint generation. 00 TBU2 update (do not compare data) 01 Compare TBU2 value with expected data 10 Reserved 11 Reserved

Table 770. GTMDI_TBU2_WPC1 field descriptions (continued)

Field	Description
7 HEN2	Halt Enable 2. The HEN2 Halt Enable 2 bit controls if the TBU2 valid event, which is a match or increment, is enabled to HALT the GTM module. If this bit is set and an event occurs on the TBU2, a Halt signal is generated to the GTM which enters Halt state. 0 Disables Halt GTM 1 Enables Halt GTM
6 WMC2	Watchpoint Message Control 2. The WMC2 Watchpoint Message Control 2 bit controls the watchpoint message sent through the Message Data bus. If this bit is set it enables match or increment on TBU2 to generate a watchpoint message. This bit is written by the JTAG interface but can also be controlled at SoC level with dedicated Start/Stop signal inputs to GTMDI. 0 Watchpoint Messages disabled 1 Watchpoint Messages enabled

49.4.1.56 TBU2 watchpoint control 2 register (GTMDI_TBU2_WPC2)

The GTMDI_TBU2_WPC2 register shown in *Figure 625* controls the Watchpoint Triggers issued by TBU2. These triggers are used for GTM to enter Halt state, generate watchpoint messages and control logic external to GTMDI, such as in the SPU interface.

Index: 109 Access: Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	WTSEL1			0	0	0	0	0	0	STSEL1		0	0	TEN1	SEN1
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	WTSEL2			0	0	0	0	0	0	STSEL2		0	0	TEN2	SEN2
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 625. TBU2 watchpoint control 2 register (GTMDI_TBU2_WPC2)

Table 771 describes the GTMDI_TBU2_WPC2 register functions.

Table 771. GTMDI_TBU2_WPC2 field descriptions

Field	Description
30:28 WTSEL1	Watchpoint Trigger Selection 1. The WTSEL1[2:0] field selects among 8 available watchpoint triggers which one is connected to the TBU2 selected event, match or increment. Other modules can selected the same WTSEL1 value, in this case the result selection is an OR of all selected sources. 000 SHARED_WPT[0] 001 SHARED_WPT[1] 010 SHARED_WPT[2] 011 SHARED_WPT[3] 100 SHARED_WPT[4] 101 SHARED_WPT[5] 110 SHARED_WPT[6] 111 SHARED_WPT[7]
21:20 STSEL1	Start/Stop input signal selection 1. Selects the Start/Stop input that is used by the watchpoint trace messages. These signals are actually a pair of signals. 00 SHARED_WPT_ST[0] 01 SHARED_WPT_ST[1] 10 SHARED_WPT_ST[2] 11 SHARED_WPT_ST[3]
17 TEN1	Watchpoint Trigger Enable 1.The TEN1 field controls if a watchpoint external trigger is issued by the TBU2 selected event, match or increment. 0 Disable Trigger Generation 1 Enable Trigger Generation
16 SEN1	Start/Stop Enable 1. The SEN1 bit field enables the TBU2 selected event to consider the Start/Stop inputs for watchpoint trace message generation. If this bit is cleared, the messages are controlled only by JTAG writes to the WMC1 field in the GTMDI_TBU2_WPC1 register. 0 Disable trace control from Start/Stop inputs 1 Enable trace control from Start/Stop inputs
14:12 WTSEL2	Watchpoint Trigger Output Selection 2. The WTSEL2[2:0] field selects among 8 available watchpoint triggers which one is connected to the TBU2 selected event. Other modules can selected the same WTSEL2 value, in this case the result signal is an OR of all selected sources. 000 SHARED_WPT[0] 001 SHARED_WPT[1] 010 SHARED_WPT[2] 011 SHARED_WPT[3] 100 SHARED_WPT[4] 101 SHARED_WPT[5] 110 SHARED_WPT[6] 111 SHARED_WPT[7]
5:4 STSEL2	Start/Stop input signal selection 2. Selects the Start/Stop input that is used the watchpoint trace messages. These signals are actually a pair of signals. 00 SHARED_WPT_ST[0] 01 SHARED_WPT_ST[1] 10 SHARED_WPT_ST[2] 11 SHARED_WPT_ST[3]

Table 771. GTMDI_TBU2_WPC2 field descriptions (continued)

Field	Description
1 TEN2	Watchpoint Trigger Enable 2. The TEN1 field controls if a watchpoint external trigger is issued by the TBU2 selected event. 0 Disable Trigger Generation 1 Enable Trigger Generation
0 SEN2	Start/Stop Enable 2. The SEN2 bit field enable the TBU2 selected event, match or increment to consider the Start/Stop inputs for watchpoint trace message generation. If this bit is cleared, the messages are controlled only by JTAG writes to the WMC2 field in the GTMDI_TBU2_WPC1 register. 0 Disable trace control from Start/Stop inputs 1 Enable trace control from Start/Stop inputs

49.4.1.57 TBU2 watchpoint DATA register (GTMDI_TBU2_DATA)

The TBU2 Watchpoint DATA shown in *Figure 626* is used for watchpoints generation for matching with the TBU2 value.

Index: 110 Access: Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	TBU2_DATA							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	TBU2_DATA															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 626. TBU2 watchpoint DATA register (GTMDI_TBU2_DATA)

Table 772 describes the GTMDI_TBU2_DATA register functions.

Table 772. GTMDI_TBU2_DATA field descriptions

Field	Description
23:0 TBU2_DATA	TBU2 Watchpoint DATA. The field TBU2_DATA[23:0] is used to compare against the TBU2 data.

49.5 Functional description

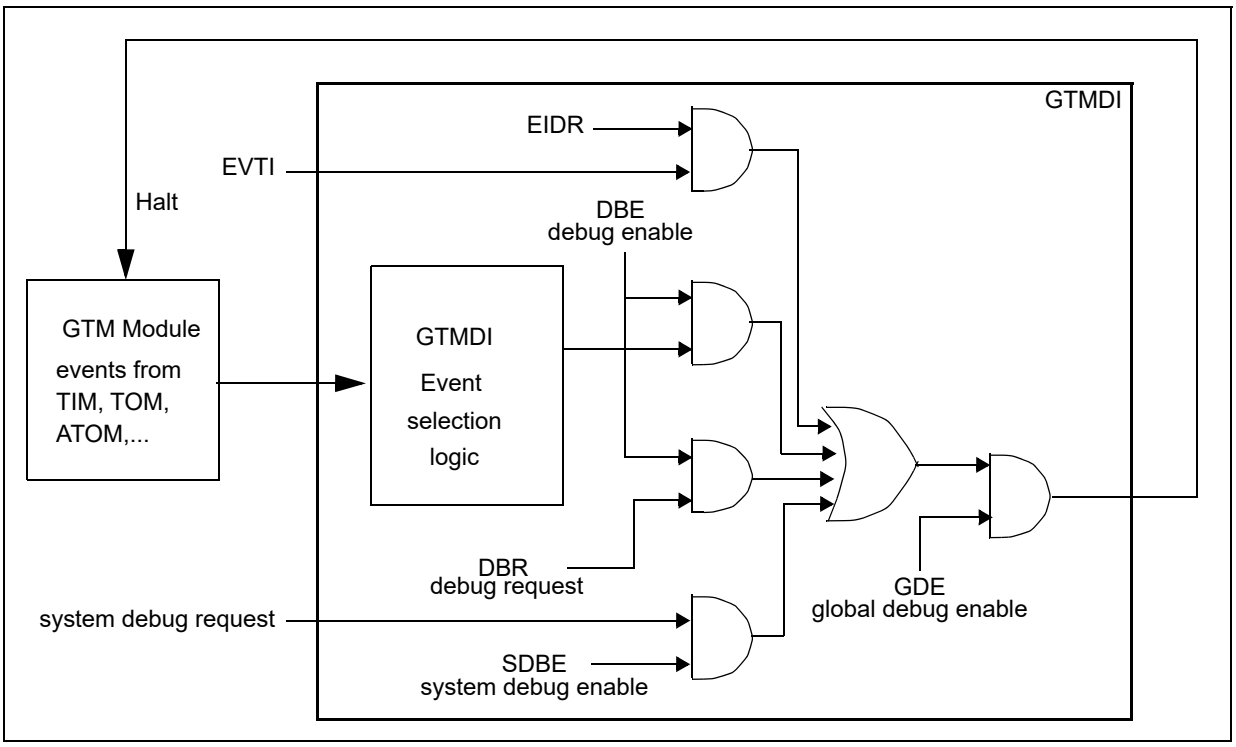
This section describes in detail the main functionality of the GTMDI module.

49.5.1 Debug enable logic

In order to enable the GTM to enter Halt state, which is the module debug state, there are some registers that need to be set. The sources that can put GTM in halt state are events

from TIM, TOM, ATOM, MCS and several other internal GTM modules that are selected and controlled by the GTMDI_DC register bits shown in *Figure 571*. The system debug request external signal is also a source of halts, as well as direct JTAG write to control register. *Figure 627* describes the logic behind the Halt signal generated by GTMDI. See *Figure 571* for a description of the register bits mentioned in *Figure 627*.

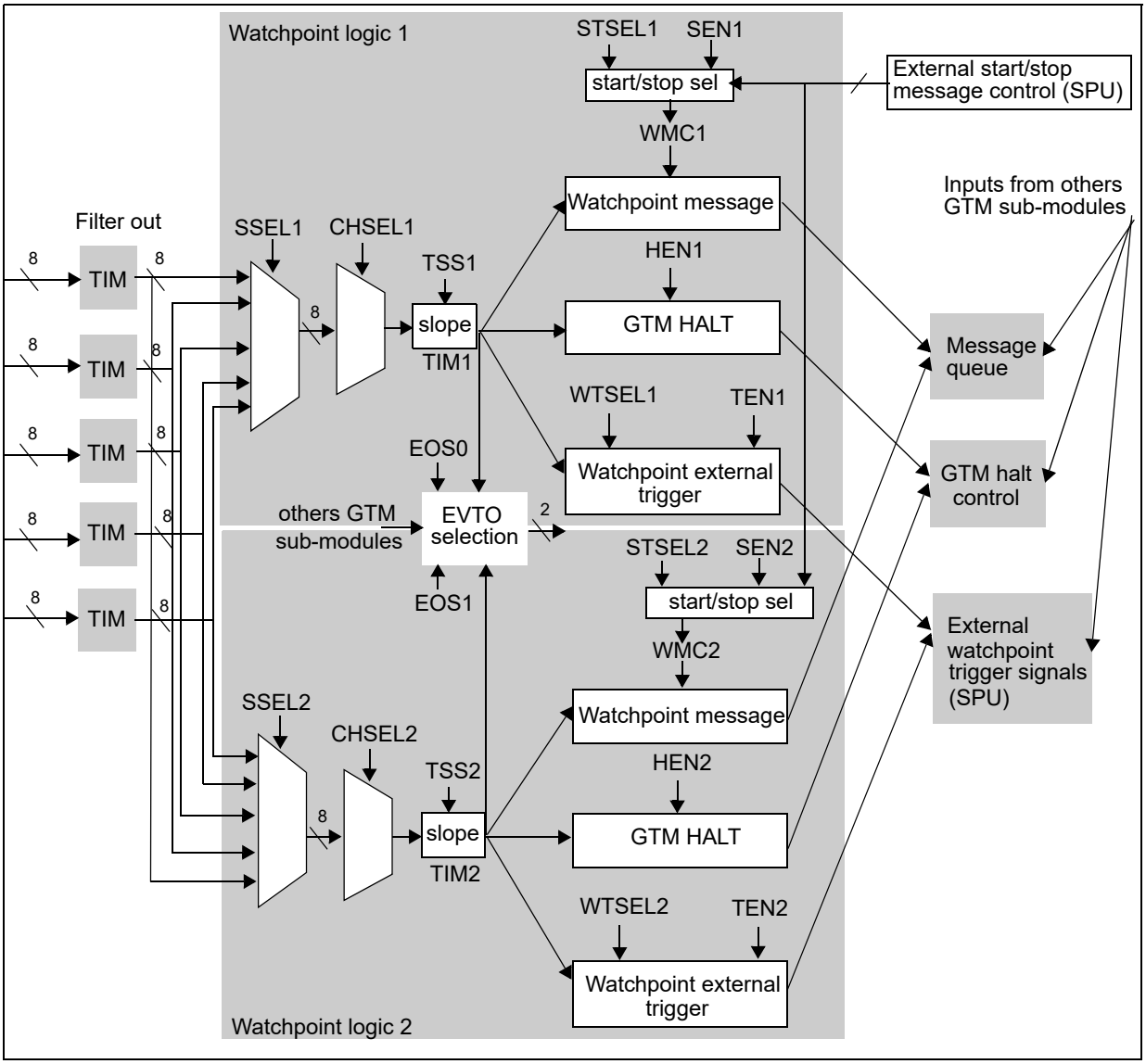
Figure 627. GTM Halt signal enable logic



49.5.2 TIM-TOM-ATOM selection and control logic

The TIM sub-modules can be selected to generate signals that control the watchpoint messages, Halt GTM and watchpoint outputs from the GTMDI module. *Figure 628* describes the TIM selection logic for generation of watchpoint messages, GTM halt and external watchpoint triggers. This logic is duplicated thus allowing independent selection of two TIM filtered inputs. *Figure 628* also shows the register bits that control the selection process. Watchpoint Logic 1 and Watchpoint Logic 2 are two implementations for the same logic.

Figure 628. TIM selection logic



Note: *Figure 628 shows five instances of the TIM sub-module as an example of a GTM configuration. Different configurations may have different number of TIM sub-modules, thus SSEL1/2 signals have to be able to select different number of TIM sub-modules.*

The selection logic for TOM and ATOM sub-modules is similar to the TIM logic. The register names are also kept the same for easy correlation of functionality.

49.5.3 GTMDI reset configuration

49.5.3.1 Enabling GTMDI class 1 operation

The GTMDI Class 1 features are always enabled after exiting the JTAG TEST-LOGIC-RESET state. But they are disabled while the JTAG is in this state. Thus, allowing low power mode. The registers are always reset while in the JTAG TEST-LOGIC-RESET state.

49.5.3.2 Enabling GTMDI class 3 operation

After exiting JTAG TEST-LOGIC-RESET state, the GTMDI classes 1 and 3 are enabled, otherwise the GTMDI class 3 features are disabled, entering in the Disable-Port Mode, thus no trace output is provided, and auxiliary port output pins are disabled.

49.5.4 Message Data Bus —interface with NAR module

49.5.4.1 Output messages

Table 773 describes the messages that the GTMDI can transmit on the Message Data Bus.

Table 773. GTMDI Nexus trace message formats

Message name	Min packet size (bits)	Max packet size (bits)	Packet type	Packet name	Packet description
Debug Status Message	6	6	fixed	TCODE	Value = 0
	4	4	fixed	SRC	GTMDI source ID. Value = 0x7
	32	32	fixed	STATUS	Value of the Development Status register GTMDI_DS, see *Table 774*
	0	30	variable	TSTAMP	Optional, globally synchronized timestamp
Error Message	6	6	fixed	TCODE	Value = 8
	4	4	fixed	SRC	GTMDI source ID. Value = 0x7
	4	4	fixed	ETYPE	Always 0 for GTMDI
	15	15	fixed	ECODE	Error Code, see *Table 775*
	0	30	variable	TSTAMP	Optional, globally synchronized timestamp
MCS Data Trace, Data Write with Sync Message	6	6	fixed	TCODE	Value = 56
	4	4	fixed	SRC	GTMDI source ID. Value = 0x7
	1	1	fixed	MCSN	MCS number (0 = MCSA; 1=MCSB)
	3	3	fixed	CHN	MCS Channel number
	14	14	fixed	ADDR	Full address of the memory location written
	32	32	fixed	DATA	Data value written
	0	30	variable	TSTAMP	Optional, globally synchronized timestamp
MCS Data Trace, Data Read with Sync Message	6	6	fixed	TCODE	Value = 57
	4	4	fixed	SRC	GTMDI source ID. Value = 0x7
	1	1	fixed	MCSN	MCS number (0 = MCSA; 1=MCSB)
	3	3	fixed	CHN	MCS Channel number
	14	14	fixed	ADDR	Full address of the memory location read
	32	32	fixed	DATA	Data value read
	0	30	variable	TSTAMP	Optional, globally synchronized timestamp

Table 773. GTMDI Nexus trace message formats (continued)

Message name	Min packet size (bits)	Max packet size (bits)	Packet type	Packet name	Packet description
Watchpoint Hit Message(1)	6	6	fixed	TCODE	Value = 15
	4	4	fixed	SRC	GTMDI source ID. Value = 0x7
	15	15	fixed	WPHIT	Number indicating watchpoint source (binary coded), see *Table 776*, *Table 777* and *Table 778*
	0	30	variable	TSTAMP	Optional, globally synchronized timestamp
Fetch Trace	6	6	fixed	TCODE	Value = 58
	4	4	fixed	SRC	GTMDI source ID. Value = 0x7
	1	1	fixed	MCSN	MCS number (0 = MCSA; 1= MCSB)
	3	3	fixed	CHN	MCS Channel number
	6	6	fixed	FSCNT(2)	Fetch Sequential Counter. Value 0 indicates there is no relationship between the last fetch trace message and the current message. The counter rolls back to one when a non sequential address is fetched. The counter increments every time a sequential address is fetched. When counter reaches 63 a message is sent and the counter rolls back to one. If an ERROR message is sent the counter rolls back to zero.
	14	14	fixed	FADDR	First element address of the next fetch sequence
	0	30	variable	TSTAMP	Optional, globally synchronized timestamp
Fetch Trace End (fetch trace was turned off, no faddr field is sent)	6	6	fixed	TCODE	Value = 59
	4	4	fixed	SRC	GTMDI source ID. Value = 0x7
	1	1	fixed	MCSN	MCS number (0 = MCSA; 1= MCSB)
	3	3	fixed	CHN	MCS Channel number
	6	6	fixed	FSCNT	The message is sent with the current value in the FSCNT counter and the counter rolls back to zero after the message is sent.
	0	30	variable	TSTAMP	Optional, globally synchronized timestamp
ARU Data Trace	6	6	fixed	TCODE	Value = 60
	4	4	fixed	SRC	GTMDI source ID. Value = 0x7
	1	1	fixed	CHN	ARU Debugging Channel number (0/1)
	58	58	fixed	DATA	Data value read {ARU_DATAH, ARU_DATAL}
	0	30	variable	TSTAMP	Optional, globally synchronized timestamp

Table 773. GTMDI Nexus trace message formats (continued)

Message name	Min packet size (bits)	Max packet size (bits)	Packet type	Packet name	Packet description
DPLL Data Trace, Data Read/Write	6	6	fixed	TCODE	Value = 61
	4	4	fixed	SRC	GTMDI source ID. Value = 0x7
	24	24	fixed	DATA	Data value read/written
	12	12	fixed	ADDR	Full address of the memory location written
	1	1	fixed	RW	Read 1, Write 0
	0	30	variable	TSTAMP	Optional, globally synchronized timestamp

1. The Watchpoint Hit Message conforms to the IEEE-ISTO 5001-2003 standard, however, other messages conform to the IEEE-ISTO 5001-2011 standard.
2. FSCNT is the sequential instruction fetch counter that indicates the number of sequential instruction fetches executed by the MCS core for a channel. See *Section 49.5.9.2: Fetch trace state machine* for more information.

Table 774. Debug status field format

STATUS bit	Selected source
12:0	HS2 field of GTMDI_DS register
14:13	Reserved
15	STP bit of GTMDI_DS register
28:16	HS1 field of GTMDI_DS register
30:29	Reserved
31	HLT bit of GTMDI_DS register

Table 775. Error field format

ECODE bit	Selected source
2:0	TIM, TOM, ATOM
5:3	ARU Watchpoint, DPLL Watchpoint, SPE Watchpoint
7:6	ARU Data Trace, DPLL Data Trace
9:8	MCSA Watchpoint, MCSB Watchpoint
11:10	MCSA Data Trace, MCSB Data Trace
13:12	MCSA Fetch Trace, MCSB Fetch Trace
14	TBU Watchpoint

Table 776. Watchpoint field WPHIT[11:0] format for WPHIT[14:12]= 0b000

WPHIT bit	Selected source
0	ATOM 1 Negedge
1	ATOM 1 Posedge

Table 776. Watchpoint field WPHIT[11:0] format for WPHIT[14:12]= 0b000 (continued)

WPHIT bit	Selected source
2	ATOM 0 Negedge
3	ATOM 0 Posedge
4	TOM 1 Negedge
5	TOM 1 Posedge
6	TOM 0 Negedge
7	TOM 0 Posedge
8	TIM 1 Negedge
9	TIM 1 Posedge
10	TIM 0 Negedge
11	TIM 0 Posedge

Table 777. Watchpoint field WPHIT[11:0] format for WPHIT[14:12] = 0b001

WPHIT bit	Selected source
0	Reserved
1	Reserved
2	Reserved
3	DPLL (memory watchpoint)
4	DPLL (SASI)
5	DPLL (TASI)
6	ARU ch1
7	ARU ch0
8	MCSB1
9	MCSB0
10	MCSA1
11	MCSA0

Table 778. Watchpoint field WPHIT[11:0] format for WPHIT[14:12] = 0b010

WPHIT bit	Selected source
0	SPEA (DIR NEGEDGE)
1	SPEA (DIR POSEDGE)
2	SPEA (NIPD NEGEDGE)
3	SPEA (NIPD POSEDGE)
4	SPEB (DIR NEGEDGE)
5	SPEB (DIR POSEDGE)

Table 778. Watchpoint field WPHIT[11:0] format for WPHIT[14:12] = 0b010 (continued)

WPHIT bit	Selected source
6	SPEB (NIPD NEGEDGE)
7	SPEB (NIPD POSEDGE)
11:8	Reserved

Table 779. Watchpoint field WPHIT[11:0] format for WPHIT[14:12] = 0b011

WPHIT - bit	Selected source
0	TBU0 Watchpoint 1
1	TBU0 Watchpoint 2
2	TBU1 Watchpoint 1
3	TBU1 Watchpoint 2
4	TBU2 Watchpoint 1
5	TBU2 Watchpoint 2
11:6	Reserved

49.5.4.1.1 Temporal ordering of transmitted messages

All messages are sent out in the sequence their related event actually occurred. The GTM interface with the GTMDI provides the GTM signals to be monitored and captured. At the occurrence of any event that would cause a message, a snapshot of all information needed by the message formatter to generate a message is queued. Doing so, several events from either MCS, TIM, ATOM, ARU, or others, can be queued at the same time.

49.5.4.1.2 Interleaved messages

Two MCS cores, TIM channels and other sources can be enabled for trace. The GTMDI formats and queues all messages in the order they are generated. Since there can be message inversion among sources during the send process, if enabled by the TSFT bit in *Figure 571* or by the NAR, the messages are sent along with a timestamp. This timestamp is globally synchronized among all Nexus clients at SoC level.

49.5.5 IEEE 1149.1 (JTAG) input port

The GTMDI block uses the IEEE 1149.1 TAP for accessing registers. This port is shared among all TAP controllers on the SoC. This sharing is controlled by the value in the instruction register and client select control register. The GTMDI implements a Nexus controller state machine that transitions based on the state of the IEEE 1149.1 state machine shown in *Figure 630*. The Nexus controller state machine, shown in *Figure 631*, is defined by the IEEE-ISTO 5001-2002 standard.

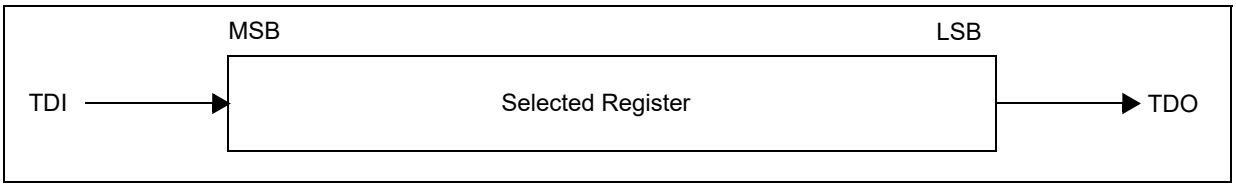
The instructions implemented by the GTMDI TAP controller are listed in *Table 780*. If any other instruction is received it is treated as a BYPASS.

Table 780. Implemented instructions

Instruction name	Private / public	Opcode	Description
NEXUS-ENABLE	Public	4b0000	Activate Nexus controller state machine to read and write GTMDI registers
BYPASS	Public	4b1111	Implements a single shift register stage providing a minimum serial length between TDI and TDO

Data is shifted between TDI and TDO starting with the least significant bit, as illustrated in *Figure 629*. This applies for the instruction register and all Nexus tool-mapped registers.

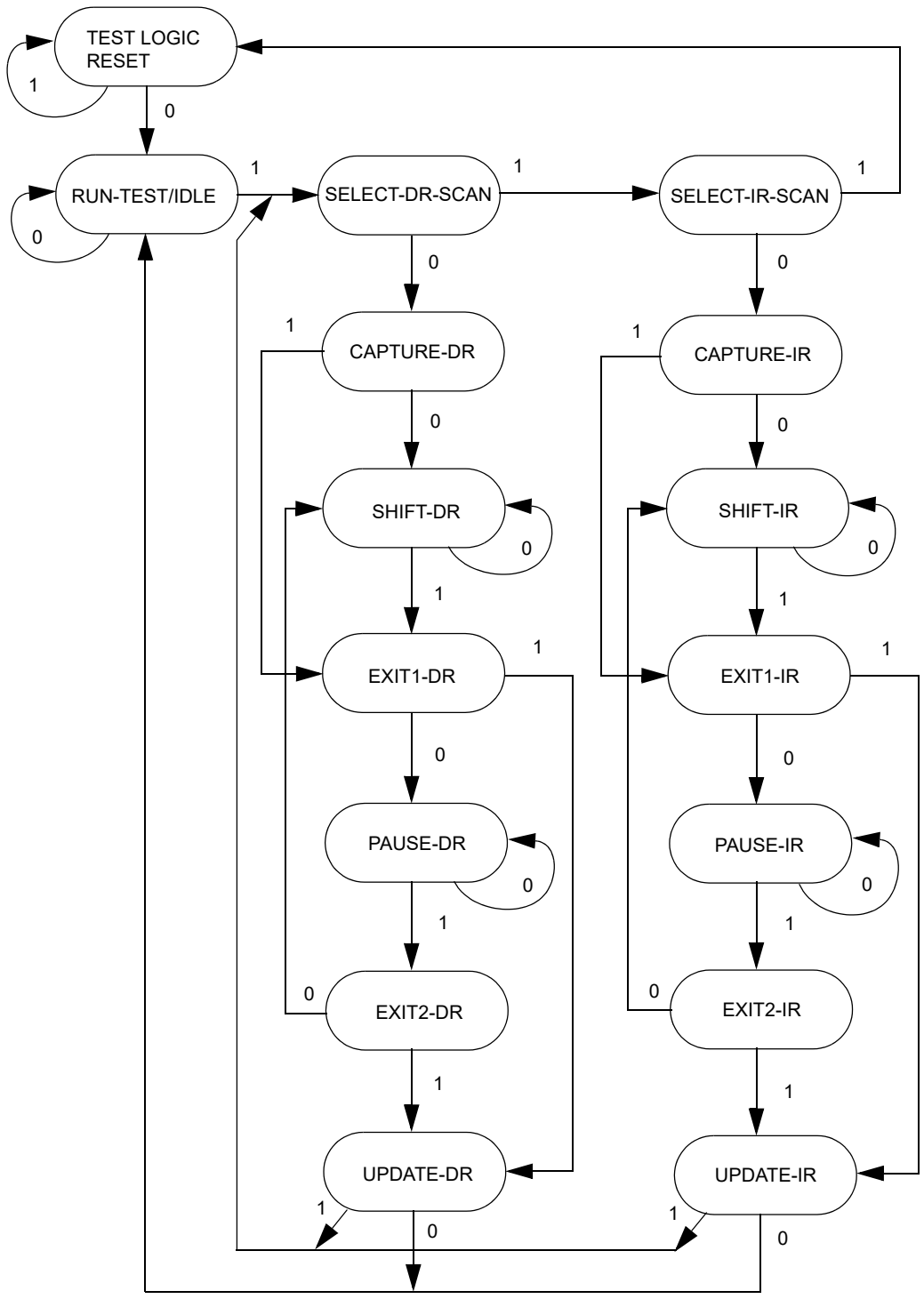
Figure 629. Shifting data into register



49.5.5.1 Enabling the GTMDI TAP controller

Assertion of the $\overline{\text{TRST}}$ input resets all TAP controllers on the SoC. The controllers are loaded with the BYPASS instruction upon exit of the TEST-LOGIC-RESET JTAG controller state. Loading the NEXUS-ENABLE instruction grants access to Nexus debug. GTMDI acts as BYPASS by loading any other instruction. The TDO bit is driven with the value of the instruction register while in SHIFT-IR state.

Figure 630. IEEE 1149.1 16-State finite state machine



NOTE: The value shown adjacent to each state transition in this figure represents the value of the TMS bit at the time of a rising edge of TCK.

49.5.5.2 Loading NEXUS-ENABLE instruction

Access to the GTMDI registers is enabled when the TAP controller instruction register is loaded with the NEXUS-ENABLE instruction. The current instruction value is loaded into the IEEE 1149.1 shifter in the CAPTURE-IR state. The instruction is shifted in via the SELECT-IR-SCAN path and loaded in the UPDATE-IR state. At this point, the Nexus controller state machine, shown in *Figure 631*, transitions to the REG_SELECT state. *Table 781* illustrates the IEEE 1149.1 sequence to load the NEXUS-ENABLE instruction.

Figure 631. Nexus controller state machine

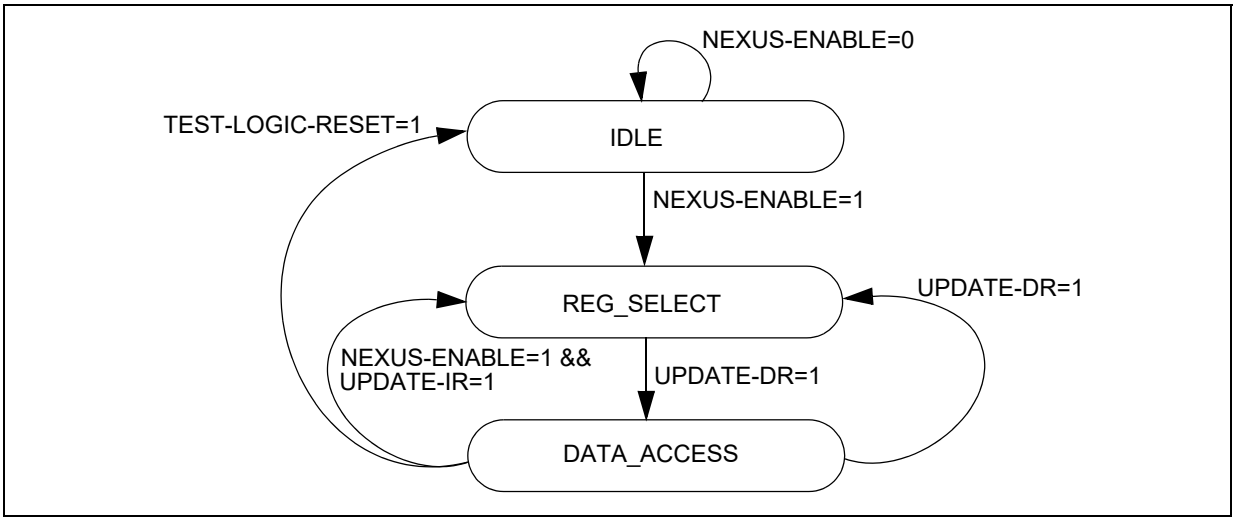


Table 781. Loading NEXUS-ENABLE instruction

Clock	TMS	IEEE 1149.1 State	Nexus state	Description
0	0	RUN-TEST/IDLE	IDLE	IEEE 1149.1 controller in idle state
1	1	SELECT-DR-SCAN	IDLE	Transitional state
2	1	SELECT-IR-SCAN	IDLE	Transitional state
3	0	CAPTURE-IR	IDLE	Internal shifter loaded with current instruction
4	0	SHIFT-IR	IDLE	TDO becomes active, and the IEEE 1149.1 shifter is ready. Shift in all but the last bit of the NEXUS_ENABLE instruction
4 TCKs				
12	1	EXIT1-IR	IDLE	Last bit of instruction shifted in
13	1	UPDATE-IR	IDLE	NEXUS-ENABLE loaded into instruction register
14	0	RUN-TEST/IDLE	REG_SELECT	NEXUS-ENABLE ready to be read

49.5.5.3 Selecting an IEEE 1149.1 register

When the NEXUS-ENABLE instruction is loaded in the GTMDI TAP controller instruction register, the input port allows development tool access to all GTMDI registers. Each register has a 7-bit address index. Some clients have registers with the same index as registers in other clients. Therefore, to access the registers of a particular client, the Client Select Control register must be set for that client.

All register access is performed via the SELECT-DR-SCAN path. The Nexus state machine defaults to the register select state when enabled. Accessing a register requires two passes through the SELECT-DR-SCAN path: one pass to select the register and the second pass to read/write the register.

The first pass through the SELECT-DR-SCAN path is used to enter an 8-bit Nexus command consisting of a read/write control bit in the LSB followed by a 7-bit register address index, as illustrated in *Figure 632*. The read/write control bit is set to 1 for writes and 0 for reads. The current value of the Controller Command Input is captured into the IEEE 1149.1 shifter during the CAPTURE-DR state while the Nexus controller state machine is in the REG_SELECT state.

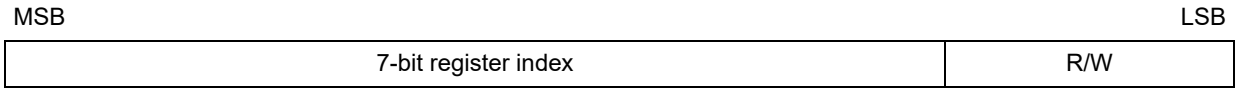


Figure 632. IEEE 1149.1 controller command input

The second pass through the SELECT-DR-SCAN path is used to read or write the register data by shifting in the data (LSB first) during the SHIFT-DR state. When reading a register, the current register value is loaded into the IEEE 1149.1 shifter during the CAPTURE-DR state. When writing a register, the value is loaded from the IEEE 1149.1 shifter to the register during the UPDATE-DR state. When reading a register, there is no requirement to shift out the entire register contents. Shifting may be terminated once the required number of bits have been acquired.

Table 782 illustrates a sequence which writes a 32-bit value to a register.

Table 782. Writing to a register

Clock	TMS	IEEE 1149.1 State	Nexus state	Description
0	0	RUN-TEST/IDLE	REG_SELECT	IEEE 1149.1 controller in idle state
1	1	SELECT-DR-SCAN	REG_SELECT	First pass through SELECT-DR-SCAN path
2	0	CAPTURE-DR	REG_SELECT	Internal shifter loaded with current value of controller command input
3	0	SHIFT-DR	REG_SELECT	TDO becomes active, and write bit and 6 bits of register index shifted in
7 TCKs				
12	1	EXIT1-DR	REG_SELECT	Last bit of register index shifted into TDI
13	1	UPDATE-DR	REG_SELECT	Controller decodes and selects register
14	1	SELECT-DR-SCAN	DATA_ACCESS	Second pass through SELECT-DR-SCAN path
15	0	CAPTURE-DR	DATA_ACCESS	Internal shifter loaded with current value of register
16	0	SHIFT-DR	DATA_ACCESS	TDO becomes active, and outputs current value of register while new value is shifted in through TDI
31 TCKs				
48	1	EXIT1-DR	DATA_ACCESS	Last bit of current value shifted out TDO. Last bit of new value shifted in TDI
49	1	UPDATE-DR	DATA_ACCESS	Value written to register
50	0	RUN-TEST/IDLE	REG_SELECT	Controller returned to idle state. It could also return to SELECT-DR-SCAN to write another register

49.5.6 Nexus Class 1 development support

The GTMDI contains a number of hardware hooks that aid in the development of software for the GTM MCS cores. This features described in this section are compliant with the following Nexus Class 1 features:

- Enter a debug mode at reset negation
- Enter a debug mode during normal execution
- Set halts or watchpoints

Refer to *Section 49.6: Initialization/application information* for how the following functions provide these features.

49.5.7 Debug status

The GTMDI module provides the Debug Status via the NAR SoC level modules.

49.5.7.1 Messaging

The GTMDI block provides debug status messaging using IEEE-ISTO 5001-2002 standard-defined public messages. If the message queue is full and a new message has to be sent, this message is lost and an ERROR message is queued instead, indicating the loss of trace messages. The debug status message format is shown in *Figure 633*.

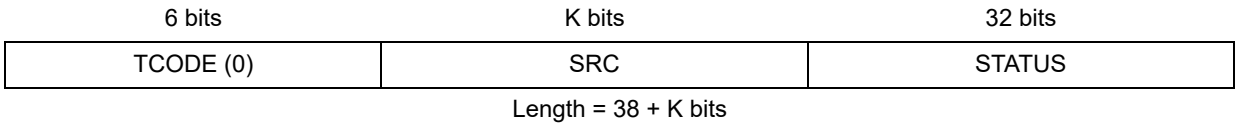


Figure 633. Debug status message format

49.5.7.2 Error messages

A debug status overrun error event is queued if a debug status event occurs while the Event Queue is full and is not able to store snapshots. The error message is stored to the event queue indicating a message was lost. The error message is sent as soon as the client is enabled for transmission. The error message format is shown in *Figure 634*.

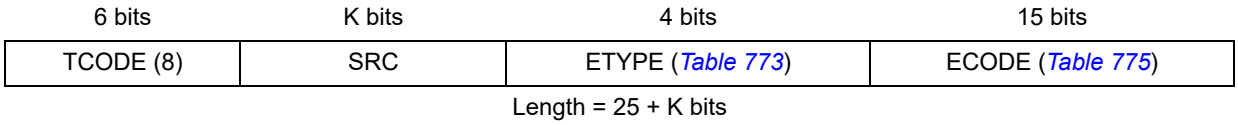


Figure 634. Debug status error message format

Note: The timestamp value in an Error Message does not contain valid information.

49.5.8 Data trace

MCS cores, ARU and DPLL perform loads and stores to RAM memory modules. The GTMDI block gathers information about these accesses and generate trace messages if enabled.

49.5.8.1 MCS data write message

The MCS data write message contains the data write value and the address of the RAM target location. The data write message format is shown in *Figure 635*.

6 bits	K bits	1 bit	3 bits	14 bits	32 bits
TCODE (56)	SRC	MCSN	CHN	ADDR	DATA

Length = 56+K bits

Figure 635. Data write message format

49.5.8.2 MCS data read message

The MCS data read message contains the data read value and the address of the RAM target location. The data read message format is shown in *Figure 636*.

6 bits	K bits	1 bit	3 bits	14 bits	32 bits
TCODE (57)	SRC	MCSN	CHN	ADDR	DATA

Length = 56+K bits

Figure 636. Data read message format

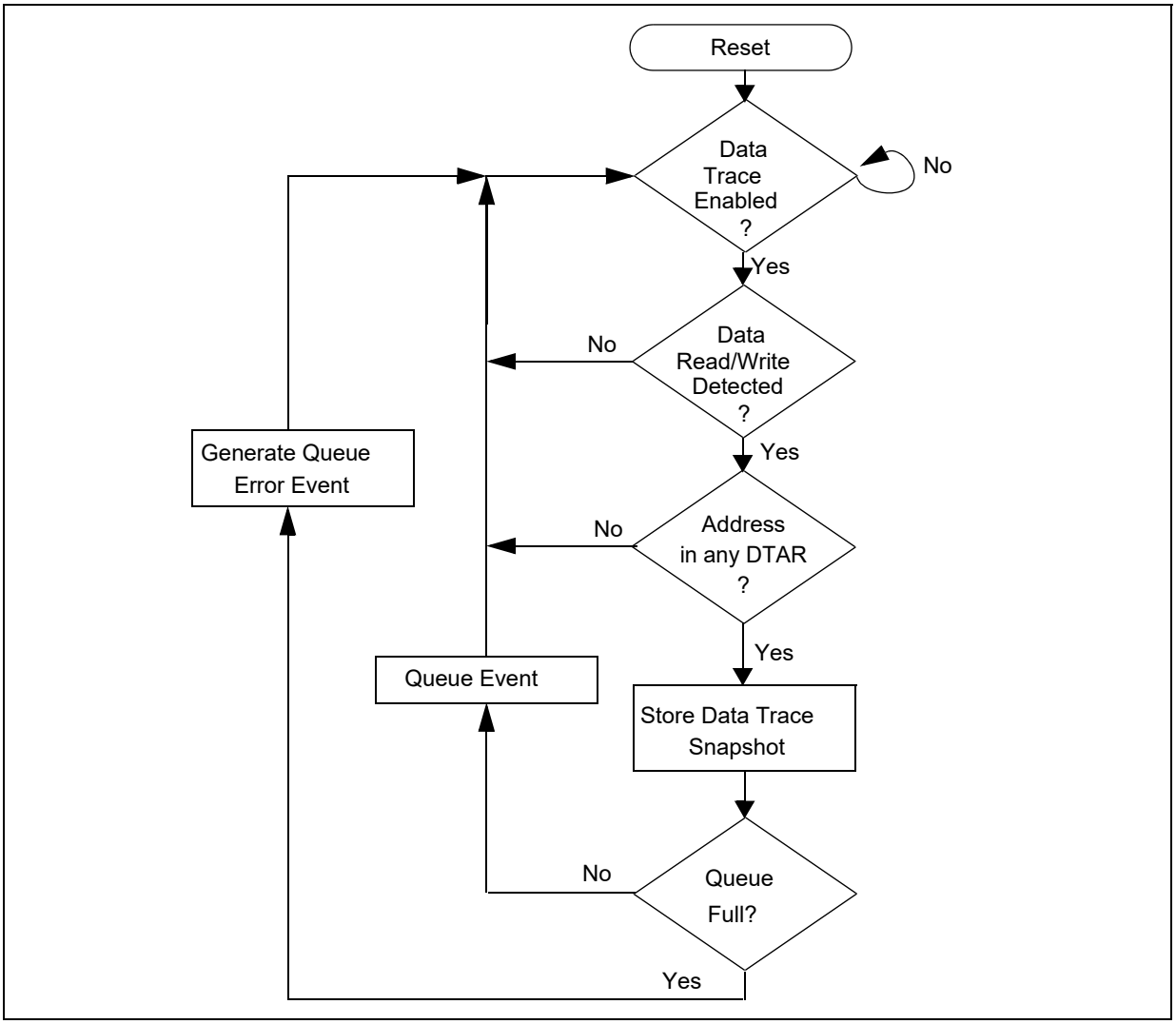
49.5.8.3 MCS data trace operation

Data tracing is performed by snooping a dedicated MCS/Nexus interface for RAM read and write cycles. Data trace functions are enabled by setting the appropriate fields in the following registers:

- MCSx Development Control (GTMDI_MCSx_DC) Register
- MCSx Data Trace Control (GTMDI_MCSx_DTC) Register
- MCSx Data Trace Address Range (GTMDI_MCSx_DTAR) Register

For details on register configuration, refer to *Section 49.4.1: Register descriptions*. The MCSx data tracing have independent Data Trace Address Range registers which allows more flexibility. An Error condition may occur in case of Queue full event. Data trace flow is depicted in *Figure 637*.

Figure 637. GTMDI generic data trace flow diagram



49.5.8.3.1 MCS data trace windowing

Data trace windowing is provided so that the development tool can decrease the auxiliary port usage by limiting the accesses that are traced. Data trace windowing is achieved via the address range defined by the DTSA and the DTEA fields of the DTAR register. RAM accesses are traced if their address fall in any of these ranges and the specific range is enabled in the data trace control register. Data read and data write trace may be enabled via the Read/Write Trace Control fields in the data trace control registers.

Data trace ranges are 32-bit aligned. This alignment is done by making the two least significant bits of the DTSA and DTEA fields read only. Since the two least significant bits of DTSA and DTEA are read only and default to different values, DTSA can never be equal to DTEA. *Figure 638* shows the relationship between the DTSA and DTEA fields.

Programmed Value	Range Selected
DTSA =< DTEA	[DTSA: DTEA]
DTSA > DTEA	All addresses are out of range(1)

Figure 638. Data trace address range options

1. Since all addresses are considered to be out of range, a Data Trace Event may be recognized if the DTC register is programmed so that addresses out of range are queued.

Note: Before sending through the NAR interface, the MCS, DPLL or ARU Data Trace message is buffered. Due to bandwidth constraints in the NAR interface and in the data trace buffering, this data trace message can be lost when superseded by another data trace message from the same source. In this case an error message is sent. See Table 773 for more details about GTMDI NAR messages.

49.5.8.4 DPLL data trace

DPLL Data Trace is a straight forward implementation in which the trace is enabled using the DMC bit in the *Figure 588*. It is possible to select individual DPLL RAM modules such as RAM1a, RAM1b or RAM2 for tracing using the RAMSEL bit in the same register. Only one memory module is selected for tracing at a given time. Start/Stop inputs from the SPU module can be used to control the DPLL Data Trace messages as well as JTAG interface. This is achieved by controlling the DMC bit which can be done from SPU or JTAG interface. See *Table 734* for more details about the bits that should be used in this process.

49.5.8.5 ARU data trace

The GTM module selects two ARU addresses to be monitored, using the ARU Debugging Channels 0 and 1, see the GTM documentation for details on how to program these channels. The data trace for these two channels is controlled by DMC1/2 bits in the GTMDI_ARU_DTC, see *Section 49.4.1.26: ARU data trace control register (GTMDI_ARU_DTC)*. Once enabled, messages are generated as soon as a valid data is identified in one of the ARU Debugging Channels, 0 or 1.

49.5.9 Fetch trace

Fetch trace tries to record all instruction fetch accesses done by an MCS channel which are executed in the MCS pipeline. Fetch trace is handled independently for each MCS channel, therefore what happens to a certain channel does not affect other channels since all channels run independently. Even error conditions are handled independently for each channel.

Fetch trace can be enabled/disabled by two ways:

- By watchpoint hit, if the watchpoint is configured to do so
- Directly by register

When a fetch access hits a watchpoint and the last enables fetch trace (having it disabled before the watchpoint hit), that access is ignored by the fetch trace logic.

49.5.9.1 Enabling MCS program fetch trace

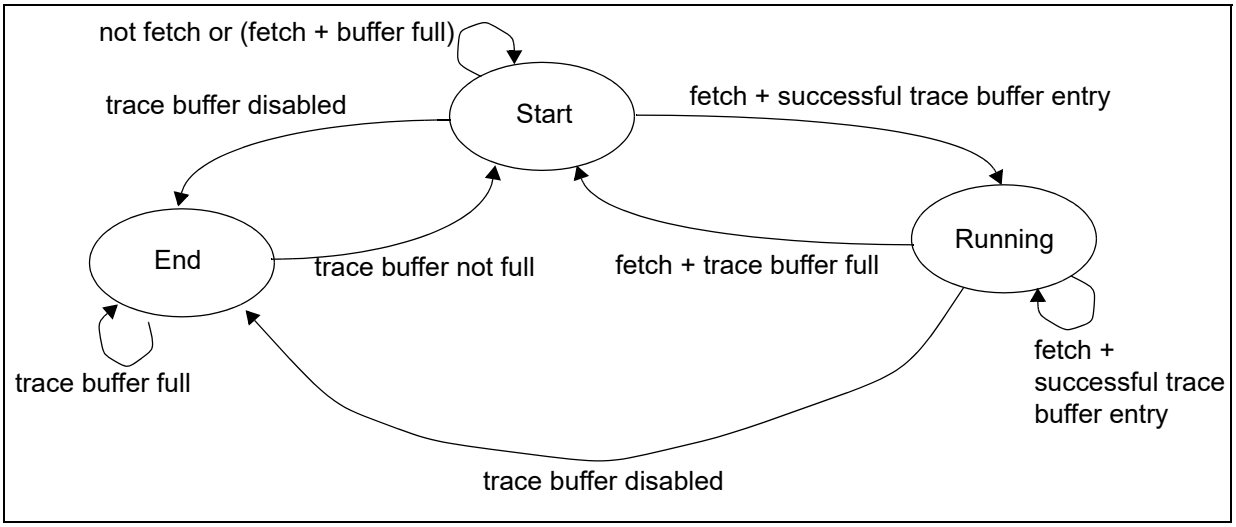
Program fetch trace for the MCS is enabled on a channel by channel basis. This selective enabling allows the debug tool to control the bandwidth for the transmitted messages. The

program fetch trace is enabled through a JTAG register, see *Figure 598*.Since the registers may be changed in the middle of a program execution, program fetch trace may be enabled or disabled during program execution as well.

49.5.9.2 Fetch trace state machine

The fetch trace state machine is composed basically by three states: *start*, *running* and *end*. When the fetch trace is enabled, fetch state machine goes to the *start* state. *Figure 639* describes the fetch trace machine state diagram.

Figure 639. Fetch trace state machine state diagram



49.5.9.2.1 Fetch trace start state

In this state, the fetch trace logic waits for a fetch access to occur. When an instruction is executed in the last stage of the MCS pipeline, it is considered as a valid fetch access. When a fetch access occurs, the fetch trace logic tries to insert (into the corresponding MCS trace buffer) a fetch trace message with FSCNT=0 and the FADDR field equal to the fetch access address. It is possible that the corresponding MCS trace message buffer is full when fetch trace logic tries to insert the message. In this case, the fetch trace logic ignores the current fetch access, does not generate any error, and tries again when another fetch access occurs.

After inserting a message into the buffer, the fetch trace state machine goes to the *running* state.

It is also possible that the fetch trace is disabled before a message is inserted into the MCS trace buffer. In this case state machine goes to the *end* state

49.5.9.2.2 Fetch trace running state

When the fetch trace state machine enters in the *running state* a counter, called CNT, is initialized to 1 and starts counting the number of sequential instruction fetch accesses.

When a fetch access with address ADDR is detected, the fetch trace logic tests if ADDR is equal to the previous fetch address, ADDR_PRV, +1 (ADDR = ADDR_PRV + 1). If so, no message is buffered for transmission and CNT increments by 1 (CNT = CNT + 1). Otherwise, if ADDR is not a sequential address, a fetch trace message is generated with the

FSCNT field equals to the CNT counter value (FSCNT = CNT) and FADDR equals to the current address ADDR. The CNT counter is then initialized to 1.

At the moment the previous message is generated, the corresponding MCS fetch trace buffer may or may not be full. If the buffer is not full, the message is inserted into the buffer and the fetch trace state machine continues in the *running* state, otherwise a fetch trace error message is generated for the corresponding MCS and the fetch trace state machine goes to the *start* state.

Note: The MCS is a multi-thread core which executes eight threads, also called channels, in fixed time-slots of one system clock for each thread. Fetch trace is a per channel resource. Thus if more than one channel is selected for trace, each one generates its corresponding fetch trace message. There is one independent fetch state machine and CNT counter for each selected channel.

49.5.9.2.3 Fetch trace end state

In this state, the fetch trace logic ignores all fetch accesses and waits for the corresponding MCS trace buffer to be not full. When this happens, a fetch trace end message is inserted into the buffer with FSCNT equal to CNT (FSCNT = CNT). The CNT value is always reset to 0 when previous state was *start* state.

If the buffer is always full, the fetch trace logic does not exit *end* state until a reset occurs.

Note: A Fetch trace end message is sent when a system reset is detected in the GTM module. There are no further indications about the GTM reset condition.

49.5.10 Watchpoint trace

The GTMDI module provides watchpoint messaging via the auxiliary port, as defined by the IEEE-ISTO 5001-2002 standard.

49.5.10.1 Messaging

The GTMDI block provides watchpoint messaging using IEEE-ISTO 5001-2002 standard-defined public messages. When a watchpoint occurs, a watchpoint event is sent to the Event Queue. If the watchpoint condition occurs while the Event Queue is full, an error message is generated. The watchpoint message format is shown in *Figure 640*.

6 bits	K bits	15 bits
TCODE (15)	SRC	WPHIT

Length = 21+Kbits

Figure 640. Watchpoint hit message format

The values for the WPHIT packet are described in *Table 776*, *Table 777* and *Table 778*. Notice that WPHIT is never 0 because the message is only generated when a watchpoint has occurred.

49.6 Initialization/application information

49.6.1 Accessing GTMDI tool-mapped registers

To initialize the JTAG port for register accesses, the following sequence is required:

1. Enable the Nexus TAP controller
2. Retrieve Device ID if needed
3. Load the Nexus TAP controller with the NEXUS-ENABLE instruction

To write control data to GTMDI tool-mapped registers, the following sequence is required:

1. Write the 7-bit register index and set the write bit to select register with a pass through the SELECT-DR-SCAN path in the JTAG state machine
2. Write the register value with a pass through the SELECT-DR-SCAN path. The prior value of this register is shifted out during the write

To read status and control data from GTMDI tool-mapped registers, the following sequence is required:

1. Write the 7-bit register index and clear write bit to select register with a pass through SELECT-DR-SCAN path
2. Read the register value with a pass through the SELECT-DR-SCAN path. Data shifted in is ignored

See *Section 49.5.5: IEEE 1149.1 (JTAG) input port* for more detail.

50 Periodic Interrupt Timer (PIT)

50.1 Introduction

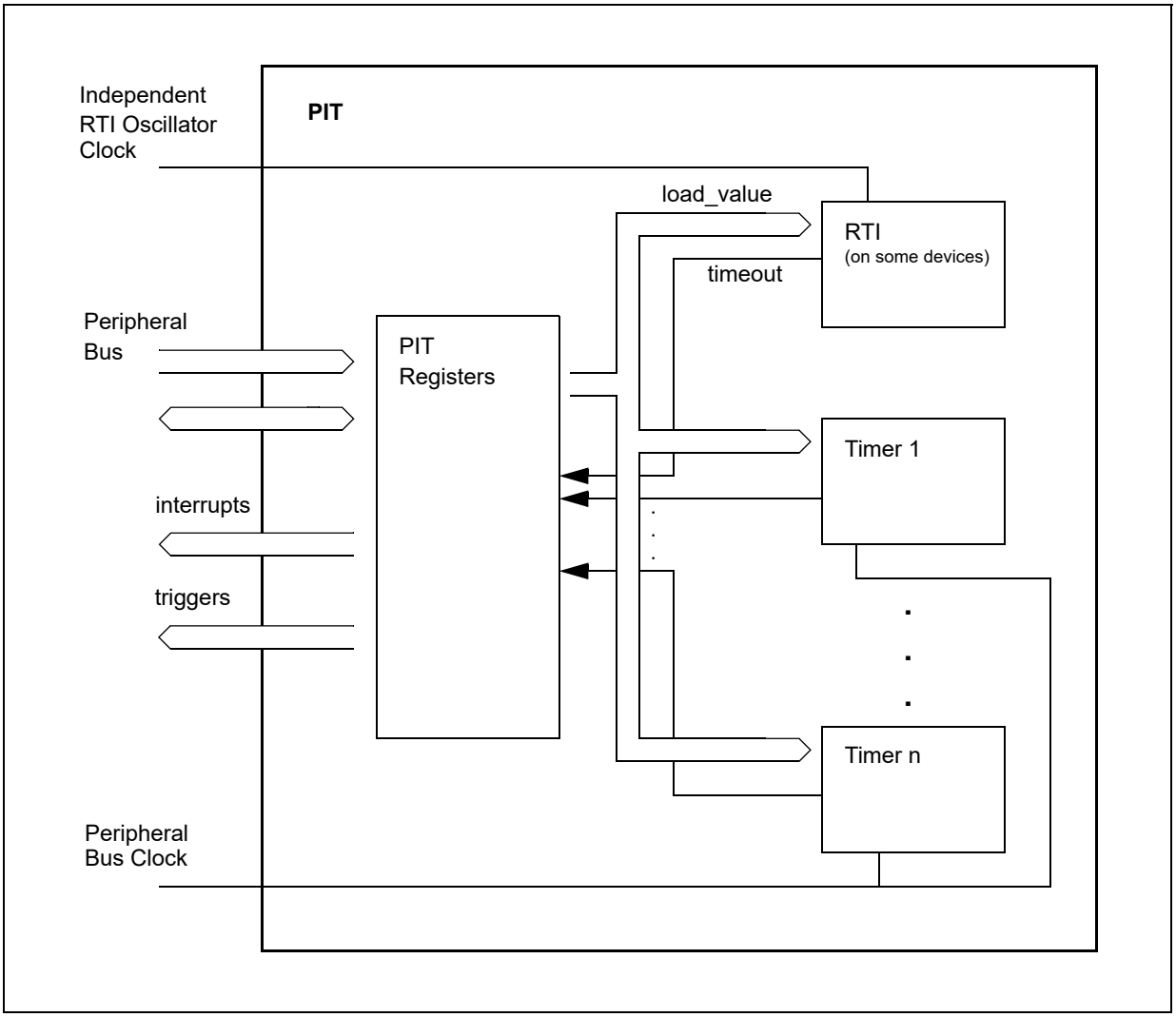
The PIT timer module is an array of timers that can be used to raise interrupts and trigger DMA channels. It includes a dedicated Real Time Interrupt Timer (RTI), which runs on a separate clock and can be used for system wakeup.

Note: For the chip-specific implementation details of this module's instances refer to the Device Configuration chapter.

50.1.1 Block diagram

Figure 641 shows the PIT block diagram.

Figure 641. Block diagram of the PIT



Note: Refer to the Chip Configuration information for the number of PIT channels used in this MCU.

50.1.2 Features

The main features of this block are:

- One RTI (Real-Time Interrupt) timer to wake up the CPU in Stop mode
- Timers can generate DMA trigger pulses
- Timers can generate interrupts
- All interrupts are maskable
- RTI interrupt can be raised, even when the bus clock is switched off
- Power saving with a separate input clock for the RTI timer (all other timers share one common core clock)
- Independent timeout periods for each timer

50.2 Signal description

This module has no external pins.

50.2.1 Memory map/register description

This section provides a detailed description of all registers accessible in this module.

Note: Reserved registers will read as 0, writes will have no effect.

Refer to the Chip Configuration information for the number of PIT channels used in this MCU.

The RTI registers should be programed only when the RTI clock is running.

Table 783. PIT memory map

Offset (hex)	Register name	Location
0x0000	PIT Control Register (MCR)	*Section 50.2.1.1*
0x0004–0x00DF	Reserved	
0x00E0	PIT Upper Lifetime Timer Register (LTMR64H)	*Section 50.2.1.2*
0x00E4	PIT Lower Lifetime Timer Register (LTMR64L)	*Section 50.2.1.3*
0x00E8–0x00EF	Reserved	
0x00F0	PIT RTI Timer Load Value Register (RTI_LDVAL)	*Section 50.2.1.4*
0x00F4	PIT RTI Current Timer Value Register (RTI_CVAL)	*Section 50.2.1.5*
0x00F8	PIT RTI Timer Control Register (RTI_TCTRL)	*Section 50.2.1.6*
0x00FC	PIT RTI Timer Flag Register (RTI_TFLG)	*Section 50.2.1.7*
PIT registers per channel (n = 0 to 7)		
0x0100 + *n**0x10	PIT Timer Load Value Register n (LDVAL*n*)	*Section 50.2.1.8*
0x0104 + *n**0x10	PIT Current Timer Value Register n (CVAL*n*)	*Section 50.2.1.9*
0x0108 + *n**0x10	PIT Timer Control Register n (TCTRL*n*)	*Section 50.2.1.10*

Table 783. PIT memory map (continued)

Offset (hex)	Register name	Location
0x010C + *n**0x10	PIT Timer Flag Register n (TFLG*n*)	*Section 50.2.1.11*
0x0180–0x3FFF	Reserved	

50.2.1.1 PIT Module Control Register (MCR)

The PIT Module Control Register (MCR) controls whether the timer clocks should be enabled and whether the timers should run in debug mode.

Offset: 0x0000 Access: User read/write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	MDIS_RTI	MDIS	FRZ
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0

Figure 642. PIT Module Control Register (MCR)

Table 784. MCR field descriptions

Field	Description
29 MDIS_RTI	Module Disable - RTI section This is used to disable the RTI timer. This bit must be enabled before any RTI setup is done. 0 Clock for RTI timers is enabled. 1 Clock for RTI timers is disabled.
30 MDIS	Module Disable This is used to disable the module clock. The RTI timer is not affected by this bit. This bit must be enabled before any other setup is done. 0 Clock for PIT timers is enabled. 1 Clock for PIT timers is disabled.
31 FRZ	Freeze Allows the timers to be stopped when the device enters debug mode. 0 Timers continue to run in debug mode. 1 Timers are stopped in debug mode.

50.2.1.2 PIT Upper Lifetime Timer Register (LTMR64H)

The PIT Upper Lifetime Timer Register (LTMR64H) is intended for applications that chain timer 0 and timer 1 to build a 64-bit life timer.

Offset: 0x00E0 Access: User read-only

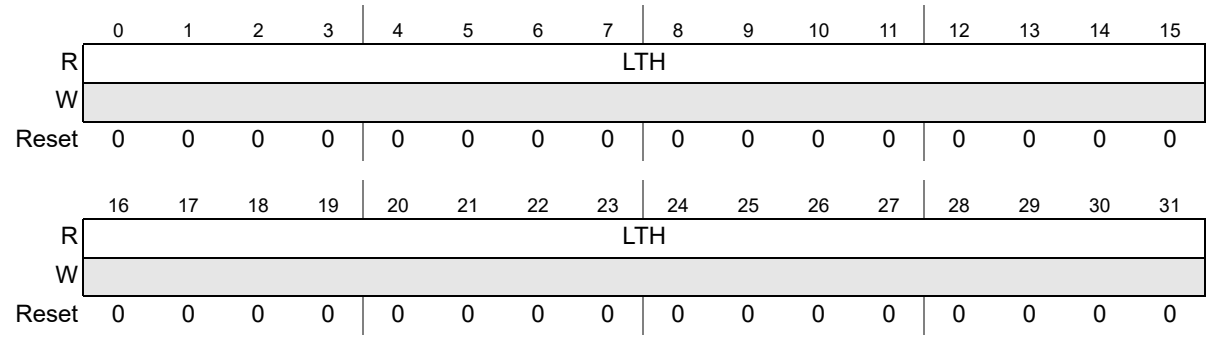


Figure 643. PIT Upper Lifetime Timer Register (LTMR64H)

Table 785. LTMR64H field descriptions

Field	Description
0:31 LTH	Life Timer value Shows the timer value of timer 1. If this register is read at a time t1, LTMR64L shows the value of timer 0 at time t1.

50.2.1.3 PIT Lower Lifetime Timer Register (LTMR64L)

The PIT Lower Lifetime Timer Register (LTMR64L) is intended for applications that chain timer 0 and timer 1 to build a 64-bit life timer.

To use LTMR64H and LTMR64L, timer 0 and timer 1 need to be chained. To obtain the correct value, first read LTMR64H and then LTMR64L. LTMR64H will have the value of CVAL1 at the time of the first access, LTMR64L will have the value of CVAL0 at the time of the first access, therefore the application does not need to worry about carry-over effects of the running counter.

Offset: 0x00E4 Access: User read-only

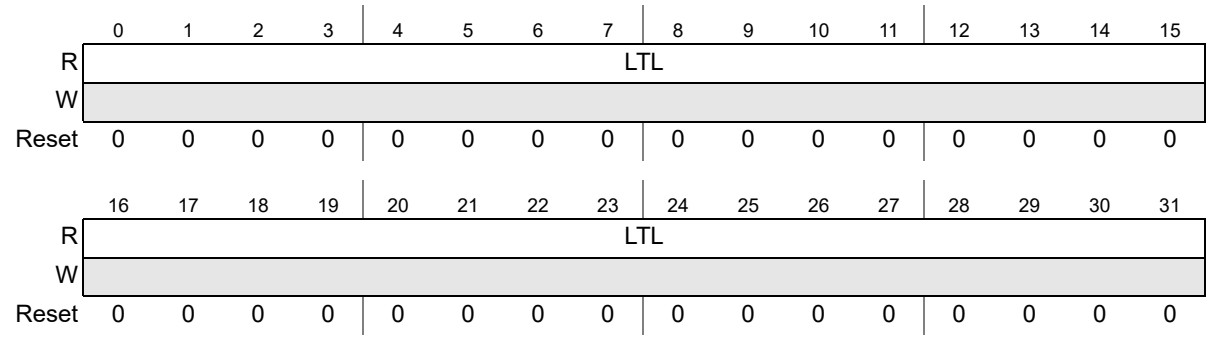


Figure 644. PIT Lower Lifetime Timer Register (LTMR64L)

Table 786. LTMR64L field descriptions

Field	Description
0:31 LTL	Life Timer value Shows the value of timer 0 at the time LTMR64H was last read. It will only update if LTMR64H is read.

50.2.1.4 PIT RTI Timer Load Value Register (RTI_LDVAL)

This register selects the timeout period for the timer interrupts.

In case of the RTI, it takes several cycles until this value is synchronized into the RTI clock domain. For all other timers the value change is visible immediately. The synchronization mechanism allows 0 wait states for RTI in this case.

Offset: 0x00F0 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	TSV															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	TSV															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 645. PIT RTI Timer Load Value Register (RTI_LDVAL)

Table 787. RTI_LDVAL field descriptions

Field	Description
0:31 TSV	Timer Start Value Bits These bits set the timer start value. The timer will count down until it reaches 0, then it will generate an interrupt and load this register value again. Writing a new value to this register will not restart the timer, instead the value will be loaded once the timer expires. To abort the current cycle and start a timer period with the new value, the timer must be disabled and enabled again. **Note:** The RTI timer must not be set to a value lower than 32 cycles, otherwise interrupts may be lost, as it takes several cycles to clear the RTI interrupt. For the other timers, this limit does not apply, however there will be practical limits, since the processor will require several cycles to service an interrupt.

50.2.1.5 PIT RTI Current Timer Value Register (RTI_CVAL)

This register indicates the current timer position.

For the RTI timers, it shows a value which is several cycles old, since it originates from a potentially different clock domain.

Offset: 0x00F4 Access: User read-only

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	TVL															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	TVL															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 646. PIT RTI Current Timer Value Register (RTI_CVAL)

Table 788. RTI_CVAL field descriptions

Field	Description
0:31 TVL	Current Timer Value If the timer is enabled, these bits represent the current timer value. If the timer is disabled, do not use this field as its value is unreliable. **Note:** The timer uses a down counter. The timer values are frozen in debug mode if the MCR[FRZ] bit is set.

50.2.1.6 PIT RTI Timer Control Register (RTI_TCTRL)

This register contains the control bits for each timer.

Offset: 0x00F8 Access: User read/write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	TIE	TEN
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 647. PIT RTI Timer Control Register (RTI_TCTRL)

Table 789. RTI_TCTRL field descriptions

Field	Description
30 TIE	Timer Interrupt Enable Bit 0 Interrupt requests from Timer n are disabled. 1 Interrupt will be requested whenever TIF is set. When an interrupt is pending (TIF set), enabling the interrupt will immediately cause an interrupt event. To avoid this, the associated TIF flag must be cleared first.
31 TEN	Timer Enable Bit This bit enables or disables the timer. 0 Timer n is disabled. 1 Timer n is active.

50.2.1.7 PIT RTI Timer Flag Register (RTI_TFLG)

This register holds the PIT interrupt flags.

Offset: 0x00FC Access: User read/write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	TIF
W																w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 648. PIT RTI Timer Flag Register (RTI_TFLG)

Table 790. RTI_TFLG field descriptions

Field	Description
31 TIF	Timer Interrupt Flag TIF is set to 1 at the end of the timer period. This flag can be cleared only by writing it with 1. Writing 0 has no effect. If enabled (TIE = 1), TIF causes an interrupt request. 0 Timeout has not yet occurred. 1 Timeout has occurred.

50.2.1.8 PIT Timer Load Value Register *n* (LDVAL*n*)

Offset: 0x0100 + n * 0x10 (n = 0 to 7) Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	TSV															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	TSV															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 649. PIT Timer Load Value Register *n* (LDVAL*n*)

Table 791. LDVAL*n* field descriptions

Field	Description
0:31 TSV	Timer Start Value Bits These bits set the timer start value. The timer will count down until it reaches 0, then it will generate an interrupt and load this register value again. Writing a new value to this register will not restart the timer, instead the value will be loaded once the timer expires. To abort the current cycle and start a timer period with the new value, the timer must be disabled and enabled again.

50.2.1.9 PIT Current Timer Value Register *n* (CVAL*n*)

Offset: 0x0104 + *n**0x10 (n = 0 to 7) Access: User read-only

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	TVL															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	TVL															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 650. PIT Current Timer Value Register *n* (CVAL*n*)

Table 792. CVAL*n* field descriptions

Field	Description
0:31 TVL	Current Timer Value If the timer is enabled, these bits represent the current timer value. If the timer is disabled, do not use this field as its value is unreliable. **Note:** The timer uses a down counter. The timer values are frozen in debug mode if the MCR[FRZ] bit is set.

50.2.1.10 PIT Timer Control Register *n* (TCTRL*n*)

Offset: 0x0108 + *n**0x10 (*n* = 0 to 7) Access: User read/write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	CHN	TIE	TEN
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 651. Timer Control Register (TCTRL*n*)

Table 793. TCTRL*n* field descriptions

Field	Description
29 CHN	Chain Mode Bit When activated, timer n-1 needs to expire before timer n can decrement by 1. Timer 0 cannot be chained. 0 Timer is not chained. 1 Timer is chained to previous timer (example, for channel 2 if this bit is set timer2 is chained 1).
30 TIE	Timer Interrupt Enable Bit 0 Interrupt requests from Timer n are disabled. 1 Interrupt will be requested whenever TIF is set. When an interrupt is pending (TIF set), enabling the interrupt will immediately cause an interrupt event. To avoid this, the associated TIF flag must be cleared first.
31 TEN	Timer Enable Bit This bit enables or disables the timer. 0 Timer n is disabled. 1 Timer n is active.

50.2.1.11 PIT Timer Flag Register *n* (TFLG*n*)

Offset: 0x010C + *n**0x10 (*n* = 0 to 7) Access: User read/write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	TIF
W																w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 652. Timer Flag Register *n* (TFLG*n*)

Table 794. TFLG*n* field descriptions

Field	Description
31 TIF	Timer Interrupt Flag TIF is set to '1' at the end of the timer period. This flag can be cleared only by writing it with 1. Writing 0 has no effect. If enabled (TIE = 1), TIF causes an interrupt request. 0 Timeout has not yet occurred. 1 Timeout has occurred.

50.3 Functional description

This section provides the functional description of the module.

50.3.1 General

This section gives detailed information on the internal operation of the module. Each timer can be used to generate trigger pulses as well as to generate interrupts. Each interrupt is available on a separate interrupt line.

50.3.1.1 Timers/RTI

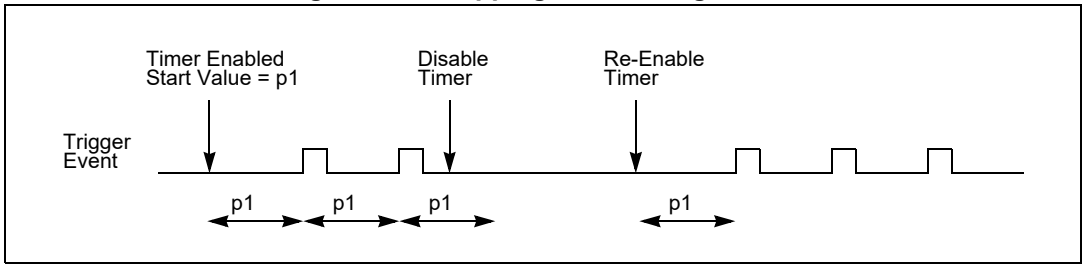
The timers generate triggers at periodic intervals, when enabled. They load their start values, as specified in their LDVAL registers, then count down until they reach 0. Then they load their respective start value again. Each time a timer reaches 0, it will generate a trigger pulse and set the interrupt flag.

All interrupts can be enabled or masked (by setting the TIE bits in the TCTRL registers). A new interrupt can be generated only after the previous one is cleared. Since in the case of the RTI, clearing the interrupt crosses clock domains, a minimum load value of 32 should be maintained.

If desired, the current counter value of the timer can be read via the CVAL registers. The value of the RTI counter can be delayed considerably, as it is synchronized to the bus clock from the RTI clock domain.

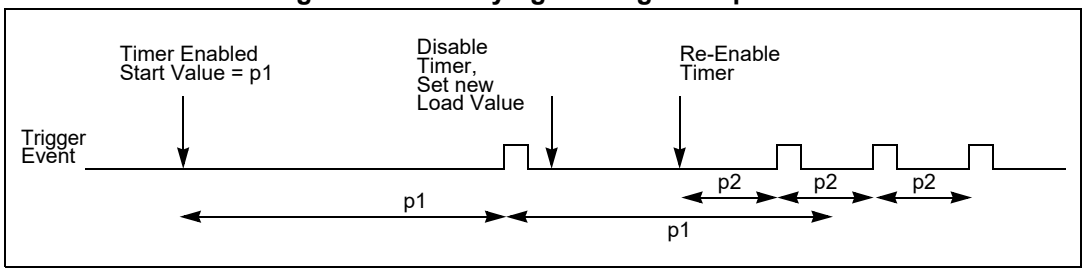
The counter period can be restarted, by first disabling, then enabling the timer with the TEN bit (refer to the following figure).

Figure 653. Stopping and starting a timer



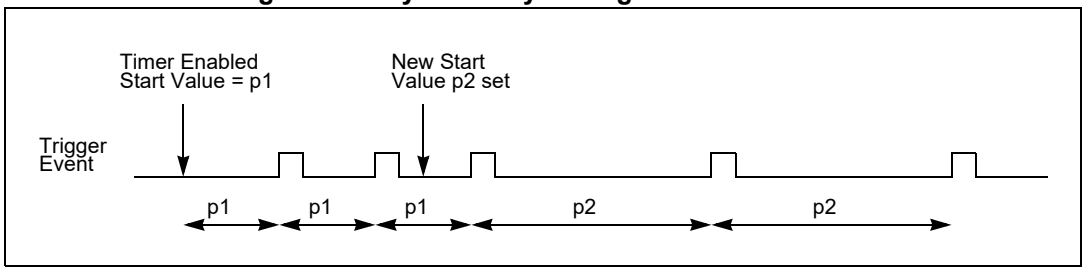
The counter period of a running timer can be modified, by first disabling the timer, setting a new load value and then enabling the timer again (refer to the following figure).

Figure 654. Modifying running timer period



It is also possible to change the counter period without restarting the timer by writing the LDVAL register with the new load value. This value will then be loaded after the next trigger event (refer to the following figure).

Figure 655. Dynamically setting a new load value



50.3.1.2 Debug mode

In debug mode, the timers will be frozen based on field MCR[FRZ]. This is intended to aid software development, allowing the developer to halt the processor, investigate the current state of the system (for example, the timer values) and then continue the operation.

50.3.2 Interrupts

All the timers support interrupt generation. The RTI is typically used for system wakeup, but can be used for interrupt generation as well. Refer to the MCU specification for related vector addresses and priorities.

Timer interrupts can be enabled by setting the TIE bits. The timer interrupt flags (TIF) are set to ‘1’ when a timeout occurs on the associated timer, and are cleared to ‘0’ by writing a ‘1’ to that TIF bit.

The PIT RTI generates a real time interrupt when the selected interrupt time period elapses. The RTI interrupt is disabled locally by setting the TIE bit to zero. The real time interrupt flag (TIF) is set to ‘1’ when a timeout occurs, and is cleared by writing a ‘1’ to the TIF bit. (The flag will be set regardless whether the interrupt is enabled.)

The RTI can be used for periodic wakeup from a low power mode. It can also be used to generate a general purpose interrupt.

50.3.3 Chained timers

When a timer has chain mode enabled, it will only count after the previous timer has expired. So if timer *n*-1 has counted down to 0, counter *n* will decrement the value by one. This allows to chain some of the timers together to form a longer timer. The first timer (timer 0) cannot be chained to any other timer.

50.4 Initialization and application information

In the example configuration:

- The PIT clock has a frequency of 50 MHz.
- The RTI clock has a frequency of 10 MHz.
- The RTI creates a wakeup interrupt every 500 ms.
- Timer 1 creates an interrupt every 5.12 ms.
- Timer 3 creates a trigger event every 30 ms.

First the PIT module must be activated by writing a 0 to field MCR[MDIS].

The 50 MHz clock frequency equates to a clock period of 20 ns and the 10 MHz frequency equates to a clock period of 100 ns. Therefore, the RTI timer needs to trigger every 500 ms/100 ns = 5000000 cycles.

Timer 1 needs to trigger every 5.12 ms/20 ns = 256000 cycles and timer 3 every 30 ms/20 ns = 1500000 cycles.

The value for the LDVAL register trigger is calculated as:

LDVAL trigger = (period / clock period) - 1

This means RTI_LDVAL will be written with 004C4B3F hex, LDVAL1 should be written with 0x0003E7FF, and LDVAL3 should be written with 0x0016E35F.

To generate the wakeup interrupt, the interrupt line must be enabled by writing a ‘1’ to bit RTI_TCTRL[TIE]. To start the RTI, bit RTI_TCTRL[TEN] must also be set.

The interrupt for Timer 1 is enabled by setting bit TCTRL1[TIE]. The timer is started by writing ‘1’ to bit TCTRL1[TEN].

Timer 3 shall be used only for triggering. Therefore Timer 3 is started by writing a ‘1’ to bit TCTRL3[TEN]; bit TIE stays at 0.

The following example code matches the described setup:

Example 9

```
// turn on PIT
PIT_MCR = 0x00;


// RTI
PIT_RTI_LDVAL = 0x004C4B3F; // setup RTI for 5000000 cycles
PIT_RTI_TCTRL = PIT_TIE;    // let RTI generate interrupts
PIT_RTI_TCTRL |= PIT_TEN;   // start RTI

// Timer 1
PIT_LDVAL1 = 0x0003E7FF; // setup timer 1 for 256000 cycles
PIT_TCTRL1 = TIE;        // enable Timer 1 interrupts
PIT_TCTRL1 |= TEN;         // start Timer 1

// Timer 3
PIT_LDVAL3 = 0x0016E35F; // setup timer 3for 1500000 cycles
PIT_TCTRL3 |= TEN;         // start Timer 3
```

50.5 Example Configuration for the Lifetime Timer

To configure the life timer, channels 0 and 1 need to be chained together.

First the PIT module needs to be activated by writing a ‘0’ to field MCR[MDIS], then the LDVAL registers need to be set to the maximum value.

The following example code matches the described setup:

Example 10

```
// turn on PIT
PIT_CTRL = 0x00;

// Timer 1
PIT_LDVAL1 = 0xFFFFFFFF; // setup timer 1 for maximum counting period
PIT_TCTRL1 = 0x0;        // disable timer 2 interrupts
PIT_TCTRL1 |= CHN;       // chain timer 1 to timer 0
PIT_TCTRL1 |= TEN;         // start timer 1

// Timer 0
PIT_LDVAL0 = 0xFFFFFFFF; // setup timer 0for maximum counting period
PIT_TCTRL0 = TEN;          // start timer 0
```

To access the lifetime read first LTMR64H then LTMR64L.

Equation 29

current_uptime = PIT_LTMR64H << 32;

current_uptime = current_uptime + PIT_LTMR64L

The timer is a down counter.

51 Enhanced Modular IO Subsystem (eMIOS)

51.1 Introduction

51.1.1 Overview of the eMIOS module

The eMIOS provides functionality to generate or measure time events on up to 32 unified channels (UCs). Each UC provides a set of functional modes, 16-bit timer resolution, and provides a user interface that is consistent with previous eMIOS implementations.

51.1.2 Features of the eMIOS module

- Two eMIOS blocks with 32 channels
- eMIOS blocks can be synchronized from outside
- One global prescaler
- 16-bit data registers
- 16-bit wide counter buses
  - Counter buses B, C, D, and E can be driven by Unified Channel 0, 8, 16, and 24, respectively
  - Counter bus A is driven by the Unified Channel 23
  - Several channels have their own time base, alternative to the counter buses
  - Shared timebases through the counter buses
  - Synchronization among timebases
- Control and Status bits grouped in a single register
- Shadow FLAG register
- Flag outputs of channels 8-11 of eMIOSs are used to disable the outputs of other channels. They form the ODIS bits
- State of the UC can be frozen for debug purposes
- Motor control capability

51.1.3 Modes of operation

The Unified Channels can be configured to operate in the following modes:

- General purpose input/output
- Single Action Input Capture
- Single Action Output Compare
- Input Pulse Width Measurement
- Input Period Measurement
- Double Action Output Compare
- Modulus Counter
- Modulus Counter Buffered
- Output Pulse Width and Frequency Modulation Buffered
- Output Pulse Width Modulation Buffered
- Output Pulse Width Modulation with Trigger
- Center Aligned Output Pulse Width Modulation Buffered

These modes are described in *Section 51.4.1.1: UC modes of operation*.

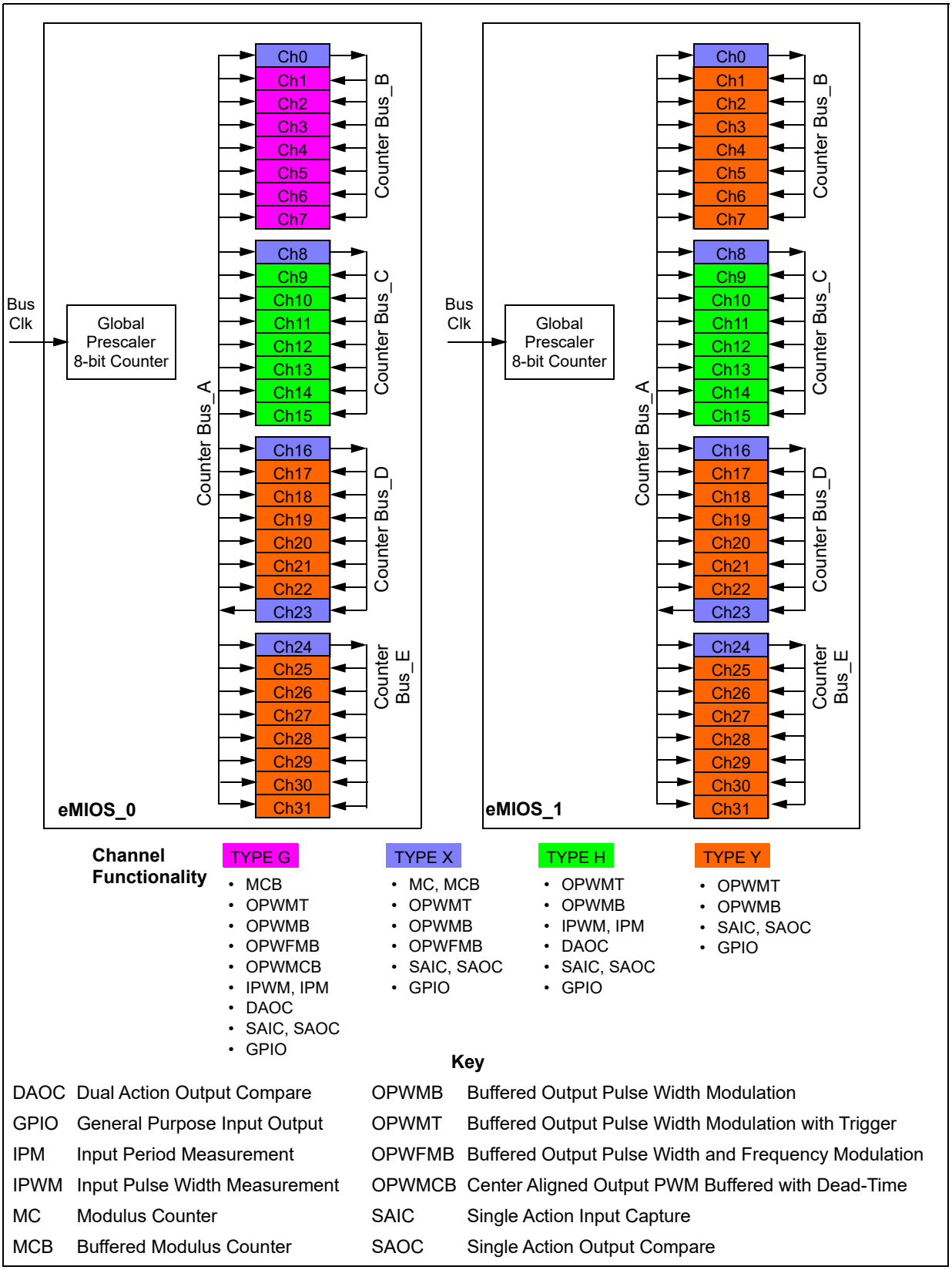
Each channel can have a specific set of modes implemented, according to device requirements.

If an unimplemented mode (reserved) is selected, the results are unpredictable such as writing a reserved value to MODE[25:31] in *Section 51.3.2.8: eMIOS UC Control Register (EMIOSCn)*.

51.1.4 Channel implementation

Figure 656 shows the channel configuration of the eMIOS blocks.

Figure 656. Channel configuration



51.1.4.1 Channel mode selection

Channel modes are selected using the mode selection bits MODE[25:31] in the *Section 51.3.2.8: eMIOS UC Control Register (EMIOSCn)*. *Table 811* provides the specific mode selection settings for the eMIOS implementation on this device.

51.2 External signal description

For information on eMIOS external signals on this device, refer to the signal description chapter of the reference manual.

51.3 Memory map and register description

51.3.1 Memory maps

The overall address map organization is shown in *Table 795*.

51.3.1.1 Unified Channel memory map

Table 795. eMIOS memory map

Address offset	Register	Location
0x0000	eMIOS Module Configuration (EMIOSMCR)	*Section 51.3.2.1*
0x0004	eMIOS Global FLAG (EMIOSGFLAG)	*Section 51.3.2.2*
0x0008	eMIOS Output Update Disable (EMIOSOUDIS)	*Section 51.3.2.3*
0x000C	eMIOS Disable Channel (EMIOSUCDIS)	*Section 51.3.2.4*
0x0010–0x001F	Reserved	
0x0020–0x011F	Channel [0] to Channel [7]	—
0x0120–0x021F	Channel [8] to Channel [15]	—
0x0220–0x031F	Channel [16] to Channel [23]	—
0x0320–0x041F	Channel [24] to Channel [31]	—
0x0420–0x0FFF	Reserved	

Addresses of Unified Channel registers are specified as offsets from the channel's base address; otherwise the eMIOS base address is used as reference.

Table 796 describes the Unified Channel memory map.

Table 796. Unified Channel memory map

UC base address	Register	Location
0x0000	eMIOS UC A (EMIOSAn)	*Section 51.3.2.5*
0x0004	eMIOS UC B (EMIOSBn)	*Section 51.3.2.6*
0x0008	eMIOS UC Counter (EMIOSCNTn)	*Section 51.3.2.7*
0x000C	eMIOS UC Control (EMIOSCn)	*Section 51.3.2.8*
0x0010	eMIOS UC Status (EMIOSSn)	*Section 51.3.2.9*
0x0014	eMIOS UC Alternate A (EMIOSALTAn)	*Section 51.3.2.10*
0x0018–0x001F	Reserved	

51.3.2 Register description

All control registers are 32 bits wide. Data registers and counter registers are 16 bits wide.

51.3.2.1 eMIOS Module Configuration Register (EMIOSMCR)

The EMIOSMCR contains global control bits for the eMIOS block.

Offset: 0x0000 Access: Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	MDIS	FRZ	GTBE	0	GPREN	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	GPRE								0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 657. eMIOS Module Configuration Register (EMIOSMCR)

Table 797. EMIOSMCR field descriptions

Field	Description
1 MDIS	Module Disable Puts the eMIOS in low power mode. The MDIS bit is used to stop the clock of the block, except the access to registers EMIOSMCR, EMIOSOUDIS and EMIOSUCDIS. 0 Clock is running 1 Enter low power mode
2 FRZ	Freeze Enables the eMIOS to freeze the registers of the Unified Channels when Debug Mode is requested at MCU level. Each Unified Channel should have FREN bit set in order to enter freeze state. While in Freeze state, the eMIOS continues to operate to allow the MCU access to the Unified Channels registers. The Unified Channel will remain frozen until the FRZ bit is written to '0' or the MCU exits Debug mode or the Unified Channel FREN bit is cleared. 0 Exit freeze state 1 Stops Unified Channels operation when in Debug mode and the FREN bit is set in the EMIOSC*n* register
3 GTBE	Global Time Base Enable The GTBE bit is used to export a Global Time Base Enable from the module and provide a method to start time bases of several blocks simultaneously. 0 Global Time Base Enable Out signal negated 1 Global Time Base Enable Out signal asserted **Note:** The Global Time Base Enable Out signals of all eMIOS blocks are combined (ORed) together to generate one Global Time Base Enable Input signal that controls the internal counters of all eMIOS blocks. When asserted, internal counter are enable. When negated, internal counters are disabled.
5 GPREN	Global Prescaler Enable The GPREN bit enables the prescaler counter. 0 Prescaler disabled (no clock) and prescaler counter is cleared 1 Prescaler enabled
16:23 GPRE	Global Prescaler The GPRE bits select the clock divider value for the global prescaler, as shown in *Table 798*.

Table 798. Global prescaler clock divider

GPRE	Divide ratio
00000000	1
00000001	2
00000010	3
00000011	4
…	…
11111110	255
11111111	256

51.3.2.2 eMIOS Global FLAG (EMIOSGFLAG) Register

The EMIOSGFLAG is a read-only register that groups the flag bits (F[31:0]) from all unified channels.These bits are mirrors of the FLAG bits in each EMIOSS*n* register. This organization improves interrupt handling on simpler devices. Each bit relates to one channel.

Offset: 0x0004 Access: Read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	F31	F30	F29	F28	F27	F26	F25	F24	F23	F22	F21	F20	F19	F18	F17	F16
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	F15	F14	F13	F12	F11	F10	F9	F8	F7	F6	F5	F4	F3	F2	F1	F0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 658. eMIOS Global FLAG (EMIOSGFLAG) Register

Table 799. EMIOSGFLAG field descriptions

Field	Description
0:31 F[31:0]	Channel n Flag bit (F[n])

51.3.2.3 eMIOS Output Update Disable (EMIOSOUDIS) Register

Offset: 0x0008 Access: Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	OU31	OU30	OU29	OU28	OU27	OU26	OU25	OU24	OU23	OU22	OU21	OU20	OU19	OU18	OU17	OU16
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	OU15	OU14	OU13	OU12	OU11	OU10	OU9	OU8	OU7	OU6	OU5	OU4	OU3	OU2	OU1	OU0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 659. eMIOS Output Update Disable (EMIOSOUDIS) Register

Table 800. EMIOSOUDIS field descriptions

Field	Description
0:31 OU[31:0]	Channel n Output Update Disable bit (OU[n]) When running MC, MCB or an output mode, values are written to registers A2 and B2. OU*n* bits are used to disable transfers from registers A2 to A1 and B2 to B1. Each bit controls one channel. 0 Transfer enabled. Depending on the operation mode, transfer may occur immediately or in the next period. Unless stated otherwise, transfer occurs immediately. 1 Transfers disabled

51.3.2.4 eMIOS Disable Channel (EMIOSUCDIS) Register

Offset: 0x000C Access: Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	CHDIS31	CHDIS30	CHDIS29	CHDIS28	CHDIS27	CHDIS26	CHDIS25	CHDIS24	CHDIS23	CHDIS22	CHDIS21	CHDIS20	CHDIS19	CHDIS18	CHDIS17	CHDIS16
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	CHDIS15	CHDIS14	CHDIS13	CHDIS12	CHDIS11	CHDIS10	CHDIS9	CHDIS8	CHDIS7	CHDIS6	CHDIS5	CHDIS4	CHDIS3	CHDIS2	CHDIS1	CHDIS0
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 660. eMIOS Enable Channel (EMIOSUCDIS) Register

Table 801. EMIOSUCDIS field descriptions

Field	Description
0:31 CHDIS[31:0]	Enable Channel n bit (CHDIS[n]) The CHDIS*n* bit is used to disable each of the channels by stopping its respective clock. 0 Channel *n* enabled 1 Channel *n* disabled

51.3.2.5 eMIOS UC A Register (EMIOSA*n*)

Depending on the mode of operation, internal registers A1 or A2, used for matches and captures, can be assigned to the EMIOSA*n* register. Both A1 and A2 are cleared by reset. *Table 804* summarizes the EMIOSA*n* writing and reading accesses for all operation modes. For more information, refer to *Section 51.4.1.1: UC modes of operation*.

Offset: 0x0020 + n * 0x20 (n = 0 to 31) Access: Read/Write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	A															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 661. eMIOS UC A Register (EMIOSA*n*)

Table 802. eMIOS UC A Register (EMIOSA*n*) field descriptions

Field	Description
16:31 A	A1 or A2 depending on the mode of operation

51.3.2.6 eMIOS UC B Register (EMIOSB*n*)

Depending on the mode of operation, internal registers B1 or B2 can be assigned to the EMIOSB*n* register. Both B1 and B2 are cleared by reset. *Table 804* summarizes the EMIOSB*n* writing and reading accesses for all operation modes. For more information, refer to *Section 51.4.1.1: UC modes of operation*.

Depending on the channel configuration, it may have EMIOSB register or not. This means that, if at least one mode that requires the register is implemented, then the register is present; otherwise it is absent.

Offset: 0x0024 + n * 0x20 (n = 0 to 31) Access: Read/Write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	B															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 662. eMIOS UC B Register (EMIOSB*n*)

Table 803. eMIOS UC B Register (EMIOSB*n*) field descriptions

Field	Description
16:31 B	B1 or B2 depending on the mode of operation

Table 804. EMIOSA*n*, EMIOSB*n* and EMIOSALTA*n* values assignment

Operation mode	Register access					
	write	read	write	read	alt write	alt read
GPIO	A1, A2	A1	B1,B2	B1	A2	A2
SAIC(1)	—	A2	B2	B2	—	—
SAOC(1)	A2	A1	B2	B2	—	—
IPWM	—	A2	—	B1	—	—
IPM	—	A2	—	B1	—	—
DAOC	A2	A1	B2	B1	—	—
MC(1)	A2	A1	B2	B2	—	—
OPWMT	A1	A1	B2	B1	A2	A2
MCB(1)	A2	A1	B2	B2	—	—
OPWFMB	A2	A1	B2	B1	—	—
OPWMCB	A2	A1	B2	B1	—	—
OPWMB	A2	A1	B2	B1	—	—

1. In these modes, the register EMIOSB*n* is not used, but B2 can be accessed.

51.3.2.7 eMIOS UC Counter Register (EMIOSCNT*n*)

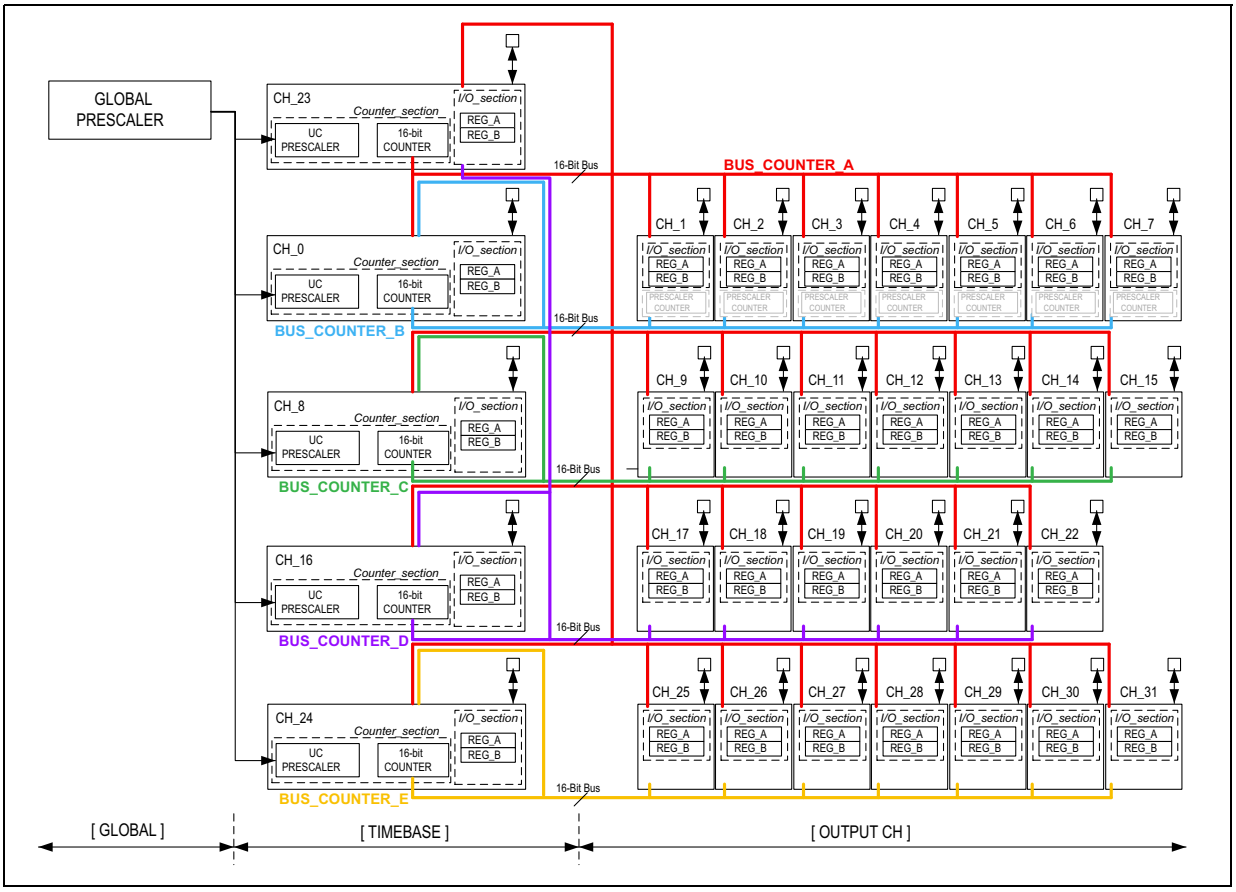
The EMIOSCNT*n* register contains the value of the internal counter. When GPIO mode is selected or the channel is frozen, the EMIOSCNT*n* register is read/write. For all other modes, the EMIOSCNT*n* is a read-only register. When entering some operation modes, this register is automatically cleared (refer to *Section 51.4.1.1: UC modes of operation* for details).

Depending on the channel configuration it may have an internal counter or not. It means that if at least one mode that requires the counter is implemented, then the counter is present; otherwise it is absent.

Channels of type X and G have the internal counter enabled, so their timebase can be selected by channel's BSL[21:22] = 11:eMIOS_A - channels 0 to 8, 16, 23 and 24, eMIOS_B = channels 0, 8, 16, 23 and 24. Other channels from the above list do not have internal counters.

Figure 663 shows the bus counter choices for the unified channels.

Figure 663. eMIOS bus counters



Note: EMIOS_0: counter_section are implemented in CH_0/8/16/23/24 and CH_1 to 7

Note: EMIOS_1: counter_section are implemented in CH_0/8/16/23/24

Offset: 0x0028 + n * 0x20 (n = 0 to 31) Access: Read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W(1)																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	C															
W(1)																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. In GPIO mode or Freeze action, this register is writable.

Figure 664. eMIOS UC Counter Register (EMIOSCNT*n*)

Table 805. eMIOS UC Counter Register (EMIOSCNT*n*) field descriptions

Field	Description
16:31 C	Internal counter value

51.3.2.8 eMIOS UC Control Register (EMIOSC*n*)

The Control register gathers bits reflecting the status of the UC input/output signals and the overflow condition of the internal counter, as well as several read/write control bits.

Offset: 0x002C + n * 0x20 (n = 0 to 31) Access: Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	FREN	ODIS	ODISSL		UCPRE		UCPREN	DMA	0	IF				FCK	FEN	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	BSL		EDSEL	EDPOL	MODE						
W			FORCMA	FORCMB												
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 665. eMIOS UC Control Register (EMIOSC*n*)

Table 806. EMIOSC*n* field descriptions

Field	Description
0 FREN	Freeze Enable bit The FREN bit, if set and validated by FRZ bit in EMIOSMCR register allows the channel to enter freeze state, freezing all registers values when in debug mode and allowing the MCU to perform debug functions. 0 Normal operation 1 Freeze UC registers values
1 ODIS	Output Disable bit The ODIS bit allows disabling the output pin when running any of the output modes with the exception of GPIO mode. 1 If the selected Output Disable Input signal is asserted, the output pin goes to EDPOL for OPWFMB and OPWMB modes and to the complement of EDPOL for other output modes, but the Unified Channel continues to operate normally, i.e., it continues to produce FLAG and matches. When the selected Output Disable Input signal is negated, the output pin operates normally. 0 The output pin operates normally.
2:3 ODISSL	Output Disable Select bits The ODISSL[1:0] bits select one of the four output disable input signals, as shown in *Table 807*.

Table 806. EMIOSC*n* field descriptions (continued)

Field	Description
4:5 UCPRE	Prescaler bits The UCPRE bits select the clock divider value for the internal prescaler of Unified Channel, as shown in *Table 808*.
6 UCPREN	Prescaler Enable bit The UCPREN bit enables the prescaler counter. 0 Prescaler disabled (no clock) 1 Prescaler enabled
7 DMA	Direct Memory Access bit The DMA bit selects if the FLAG signal will be used as an interrupt request, as a DMA request or as a CTU trigger. The choice between a DMA request or a CTU trigger is determined by the value of bit TM in the register CTU_EVTCFGRx (refer to the CTU chapter of the reference manual). 0 Flag/overrun assigned to interrupt request 1 Flag/overrun assigned to DMA request or CTU trigger
9:12 IF	Input Filter The IF field controls the programmable input filter, selecting the minimum input pulse width that can pass through the filter, as shown in *Table 809*. For output modes, these bits have no meaning.
13 FCK	Filter Clock select bit The FCK bit selects the clock source for the programmable input filter. 0 Prescaled clock 1 Main clock
14 FEN	FLAG Enable bit The FEN bit allows the Unified Channel FLAG bit to generate an interrupt signal or a DMA request signal or a CTU trigger signal (The type of signal to be generated is defined by the DMA bit). 0 Disable (FLAG does not generate an interrupt request or DMA request or a CTU trigger) 1 Enable (FLAG will generate an interrupt request or DMA request or a CTU trigger)
18 FORCMA	Force Match A bit For output modes, the FORCMA bit is equivalent to a successful comparison on comparator A (except that the FLAG bit is not set). This bit is cleared by reset and is always read as zero. This bit is valid for every output operation mode which uses comparator A, otherwise it has no effect. 0 Has no effect 1 Force a match at comparator A **Note:** For input modes, the FORCMA bit is not used and writing to it has no effect.
19 FORCMB	Force Match B bit For output modes, the FORCMB bit is equivalent to a successful comparison on comparator B (except that the FLAG bit is not set). This bit is cleared by reset and is always read as zero. This bit is valid for every output operation mode which uses comparator B, otherwise it has no effect. 0 Has no effect 1 Force a match at comparator B **Note:** For input modes, the FORCMB bit is not used and writing to it has no effect.

Table 806. EMIOSC*n* field descriptions (continued)

Field	Description
21:22 BSL	Bus Select The BSL field is used to select either one of the counter buses or the internal counter (if present) to be used by the Unified Channel. Refer to *Table 810* for details.
23 EDSEL	Edge Selection bit For input modes, the EDSEL bit selects whether the internal counter is triggered by both edges of a pulse or just by a single edge as defined by the EDPOL bit. When not shown in the mode of operation description, this bit has no effect. 0 [All input modes]: single edge triggering defined by the EDPOL bit; [GPIO input mode]: a FLAG is generated as defined by the EDPOL bit; [SAOC mode]: the output flip-flop is toggled at each match. 1 [All input modes]: both edges triggering; [GPIO input mode]: no FLAG is generated; [SAOC mode]: the output flip-flop is toggled at each match.
24 EDPOL	Edge Polarity bit For input modes, the EDPOL bit asserts which edge triggers either the internal counter or an input capture or a FLAG. When not shown in the mode of operation description, this bit has no effect. 0 [All input modes]: trigger on a falling edge; [All output modes]: a match on comparator A clears the output flip-flop, while a match on comparator B sets it 1 [All input modes]: trigger on a rising edge; [All output modes]: a match on comparator A sets the output flip-flop, while a match on comparator B clears it
25:31 MODE	Mode selection The MODE field selects the mode of operation of the Unified Channel, as shown in *Table 811*. **Note:** If a reserved value is written to mode the results are unpredictable.

Table 807. UC ODISSL selection

ODISSL	Input signal
00	Output Disable Input 0
01	Output Disable Input 1
10	Output Disable Input 2
11	Output Disable Input 3

Table 808. UC internal prescaler clock divider

UCPRE	Divide ratio
00	1
01	2
10	3
11	4

Table 809. UC input filter bits

IF[1]	Minimum input pulse width [FLT_CLK periods]
0000	Bypassed[2]
0001	02
0010	04
0100	08
1000	16
all others	Reserved

1. Filter latency is 3 main clock periods.
2. The input signal is synchronized before arriving to the digital filter.

Table 810. UC BSL bits

BSL	Selected bus
00	All channels: counter bus[A]
01	Channels 0 to 7: counter bus[B] Channels 8 to 15: counter bus[C] Channels 16 to 23: counter bus[D] Channels 24 to 31: counter bus[E]
10	Reserved
11	UC internal counter (use only if counter present)

Table 811. Channel mode selection

MODE[1]	Mode of operation
0000000	General purpose Input/Output mode (input)
0000001	General purpose Input/Output mode (output)
0000010	Single Action Input Capture
0000011	Single Action Output Compare
0000100	Input Pulse Width Measurement
0000101	Input Period Measurement
0000110	Double Action Output Compare (with FLAG set on B match)
0000111	Double Action Output Compare (with FLAG set on both matches)
0001000 – 0001111	Reserved
001000b	Modulus Counter (Up counter with clear on match start)
001001b	Modulus Counter (Up counter with clear on match end)
00101bb	Modulus Counter (Up/Down counter)
0011000 – 0100101	Reserved
0100110	Output Pulse Width Modulation with Trigger
0100111 – 1001111	Reserved

Table 811. Channel mode selection (continued)

MODE[1]	Mode of operation
101000b	Modulus Counter Buffered (Up counter)
101001b	Reserved
10101bb	Modulus Counter Buffered (Up/Down counter)
10110b0	Output Pulse Width and Frequency Modulation Buffered
10110b1	Reserved
10111b0	Center Aligned Output Pulse Width Modulation Buffered (with trail edge dead-time)
10111b1	Center Aligned Output Pulse Width Modulation Buffered (with lead edge dead-time)
11000b0	Output Pulse Width Modulation Buffered
1100001 – 1111111	Reserved

1. b = adjust parameters for the mode of operation. Refer to *UC modes of operation* for details.

51.3.2.9 eMIOS UC Status Register (EMIOSS*n*)

Offset: 0x0030 + n * 0x20 (n = 0 to 31) Access: Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	OVR	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W	w1c															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	OVFL	0	0	0	0	0	0	0	0	0	0	0	0	UCIN	UCOUT	FLAG
W	w1c															w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 666. eMIOS UC Status Register (EMIOSS*n*)

Table 812. EMIOSS*n* field descriptions

Field	Description
0 OVR	Overrun bit The OVR bit indicates that FLAG generation occurred when the FLAG bit was already set. 0 Overrun has not occurred 1 Overrun has occurred
16 OVFL	Overflow bit The OVFL bit indicates that an overflow has occurred in the internal counter. OVFL must be cleared by software writing a 1 to the OVFLC bit. 0 No overflow 1 An overflow had occurred

Table 812. EMIOSS*n* field descriptions (continued)

Field	Description
29 UCIN	Unified Channel Input pin bit The UCIN bit reflects the input pin state after being filtered and synchronized.
30 UCOUT	Unified Channel Output pin bit The UCOUT bit reflects the output pin state.
31 FLAG	FLAG bit The FLAG bit is set when an input capture or a match event in the comparators occurred. 0 FLAG cleared 1 FLAG set event has occurred **Note:** When DMA bit is set, the FLAG bit can be cleared by the DMA controller or the CTU.

51.3.2.10 eMIOS UC Alternate A Register (EMIOSALTA*n*)

The EMIOSALTA*n* register provides an alternate register to access A2 channel registers in restricted modes (GPIO, OPWMT) only. If EMIOSA*n* register is used along with EMIOSALTA*n*, both A1 and A2 registers can be accessed in these modes. *Figure 804* summarizes the EMIOSALTA*n* writing and reading accesses for all operation modes. Refer to *Section 51.4.1.1.1: General purpose Input/Output (GPIO) mode*, *Section 51.4.1.1.12: Output Pulse Width Modulation with Trigger (OPWMT) mode* for a more detailed description of the use of EMIOSALTA*n* register.

Offset: 0x0034 + n * 0x20 (n = 0 to 31) Access: Read/Write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	ALTA															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 667. eMIOS UC Alternate A register (EMIOSALTA*n*)

Table 813. eMIOS UC Alternate A register (EMIOSALTA*n*) field descriptions

Field	Description
16:31 ALTA	Alternate address to access the A2 channel registers in restricted mode

51.4 Functional description

The four types of channels of the eMIOS (types X, Y, G and H) can operate in the modes as listed in *Figure 656*. The eMIOS provides independently operating unified channels (UC)

that can be configured and accessed by a host MCU. Up to three time bases[21] can be shared by the channels through five counter buses[22] and each unified channel can generate its own time base[23]. The eMIOS block is reset at positive edge of the clock (synchronous reset). All registers are cleared on reset.

51.4.1 Unified Channel (UC)

Each Unified Channel consists of:

- Counter bus selector, which selects the time base to be used by the channel for all timing functions
- A programmable clock prescaler
- Two double buffered data registers A and B that allow up to two input capture and/or output compare events to occur before software intervention is needed
- Two comparators (equal only) A and B, which compare the selected counter bus with the value in the data registers
- Internal counter, which can be used as a local time base or to count input events
- Programmable input filter, which ensures that only valid pin transitions are received by channel
- Programmable input edge detector, which detects the rising, falling or either edges
- An output flip-flop, which holds the logic level to be applied to the output pin
- eMIOS Status and Control register

51.4.1.1 UC modes of operation

The mode of operation of the Unified Channel is determined by the mode select bits MODE[25:31] in the *Section 51.3.2.8: eMIOS UC Control Register (EMIOSCn)* (refer to *Figure 665* for details).

As the internal counter EMIOSCNT*n* continues to run in all modes (except for GPIO mode), it is possible to use this local counter as a timebase for other UCs if it is not used for the current local mode.

In order to provide smooth waveform generation, a few specific modes allow A and B registers to be changed on the fly. These modes are suffixed with a “B” (MCB, OPWFMB, OPWMB and OPWMCB), meaning that in these modes A and B registers are double buffered.

21. Time bases can be supplied by:
i) channel 23 to all unified channels
ii) channel 0 to channels 0 to 7, by channel 8 to channels 8 to 15, by channel 16 to channels 16 to 23, by channel 24 to channels 24 to 31
iii) channel's internal counter when available.

22. Internal eMIOS architecture has one global counter bus A and four local counter buses B, C, D, and E, that distribute the time bases described in footnote *21.* (i) and (ii).

23. Channels of type X and G have the internal counter enabled, so their timebase can be selected by channel's BSL[21:22]=11: eMIOS_A - channels 0 to 8, 16, 23 and 24 eMIOS_B = channels 0, 8, 16, 23 and 24.

51.4.1.1.1 General purpose Input/Output (GPIO) mode

In GPIO mode, all input capture and output compare functions of the UC are disabled, the internal counter (EMIOSCNT*n* register) is cleared and disabled. All control bits remain accessible. In order to prepare the UC for a new operation mode, writing to registers EMIOSA*n* or EMIOSB*n* stores the same value in registers A1/A2 or B1/B2, respectively. Writing to register EMIOSALTA*n* stores a value only in register A2.

MODE[31] bit selects between input (MODE[31] = 0) and output (MODE[31] = 1) modes.

It is required that when changing MODE[25:31], the application software goes to GPIO mode first in order to reset the UC's internal functions properly. Failure to do this could lead to invalid and unexpected output compare or input capture results or the FLAGs being set incorrectly.

In GPIO input mode (MODE[25:31] = 0000000), the FLAG generation is determined according to EDPOL and EDSEL bits and the input pin status can be determined by reading the UCIN bit.

In GPIO output mode (MODE[25:31] = 0000001), the Unified Channel is used as a single output port pin and the value of the EDPOL bit is permanently transferred to the output flip-flop.

51.4.1.1.2 Single Action Input Capture (SAIC) mode

In SAIC mode (MODE[25:31] = 0000010), when a triggering event occurs on the input pin, the value on the selected time base is captured into register A2. The FLAG bit is set along with the capture event to indicate that an input capture has occurred. Register EMIOSA*n* returns the value of register A2. As soon as the SAIC mode is entered coming out from GPIO mode the channel is ready to capture events. The events are captured as soon as they occur thus reading register A always returns the value of the latest captured event. Subsequent captures are enabled with no need of further reads from EMIOSA*n* register. The FLAG is set at any time a new event is captured.

The input capture is triggered by a rising, falling or either edges in the input pin, as configured by EDPOL and EDSEL bits in EMIOSC*n* register.

Figure 668 and *Figure 669* show how the Unified Channel can be used for input capture.

Figure 668. Single action input capture with rising edge triggering example

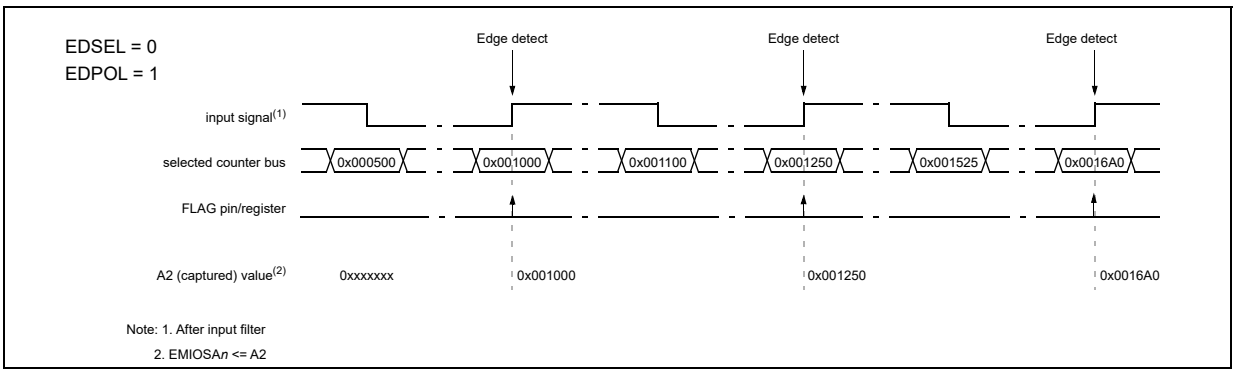
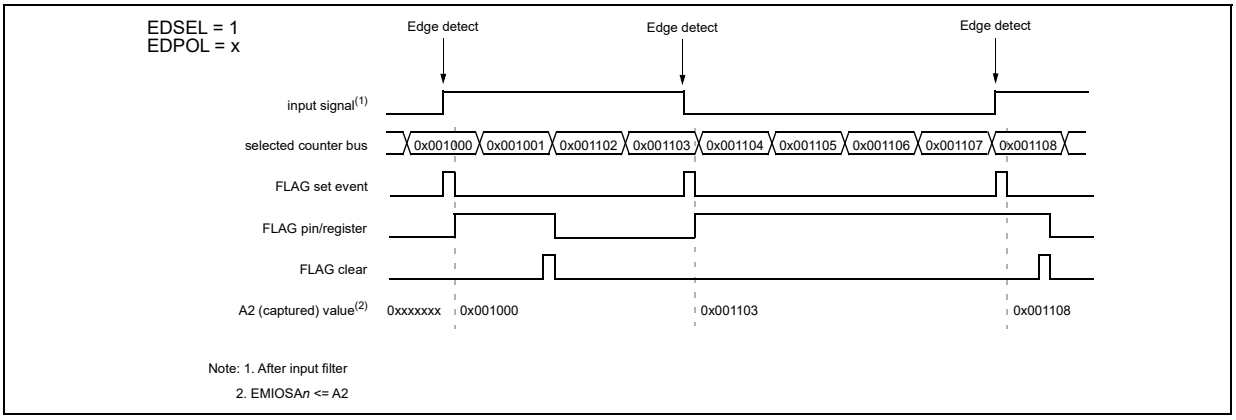


Figure 669. Single action input capture with both edges triggering example



51.4.1.1.3 Single Action Output Compare (SAOC) mode

In SAOC mode (MODE[25:31] = 0000011) a match value is loaded in register A2 and then immediately transferred to register A1 to be compared with the selected time base. When a match occurs, the EDSEL bit selects whether the output flip-flop is toggled or the value in EDPOL is transferred to it. Along with the match the FLAG bit is set to indicate that the output compare match has occurred. Writing to register EMIOSA*n* stores the value in register A2 and reading to register EMIOSA*n* returns the value of register A1.

An output compare match can be simulated in software by setting the FORCMA bit in EMIOSC*n* register. In this case, the FLAG bit is not set.

When SAOC mode is entered coming out from GPIO mode the output flip-flop is set to the complement of the EDPOL bit in the EMIOSC*n* register.

Counter bus can be either internal or external and is selected through bits BSL[21:22].

Figure 670 and *Figure 671* show how the Unified Channel can be used to perform a single output compare with EDPOL value being transferred to the output flip-flop and toggling the output flip-flop at each match, respectively. Note that once in SAOC mode the matches are enabled thus the desired match value on register A1 must be written before the mode is entered. A1 register can be updated at any time thus modifying the match value which will reflect in the output signal generated by the channel. Subsequent matches are enabled with no need of further writes to EMIOSA*n* register. The FLAG is set at the same time a match occurs (refer to *Figure 672*).

Note: The channel internal counter in SAOC mode is free-running. It starts counting as soon as the SAOC mode is entered.

Figure 670. SAOC example with EDPOL value being transferred to the output flip-flop

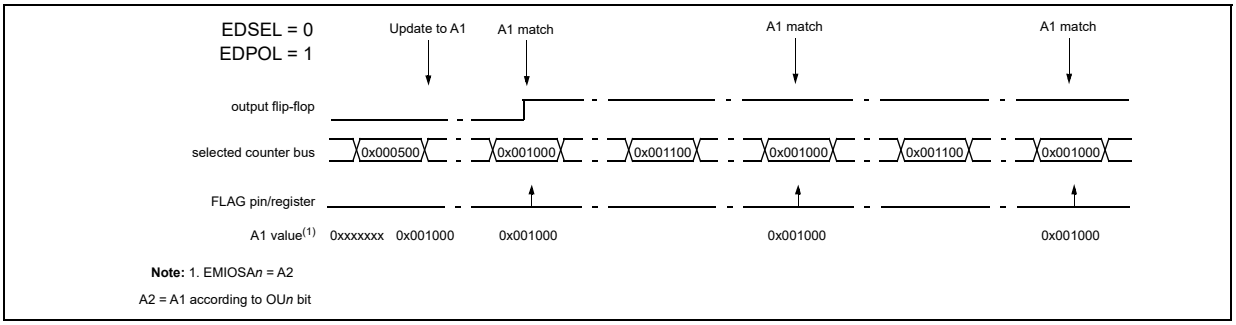


Figure 671. SAOC example toggling the output flip-flop

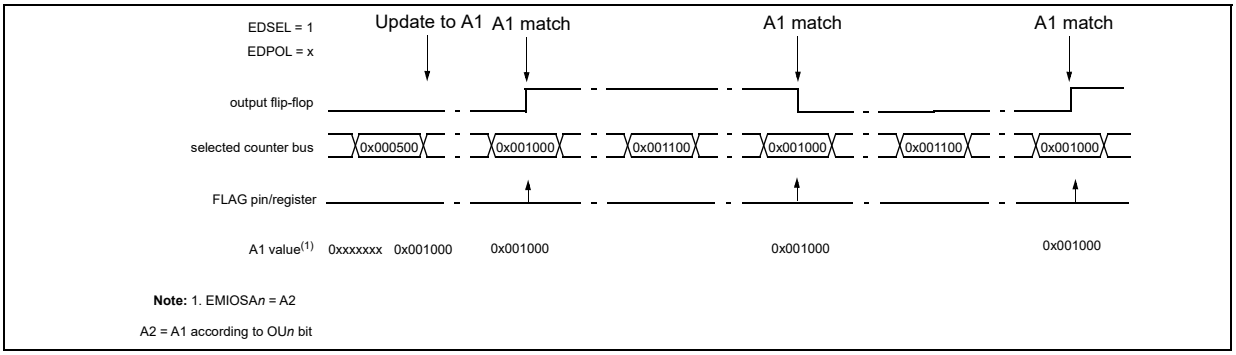
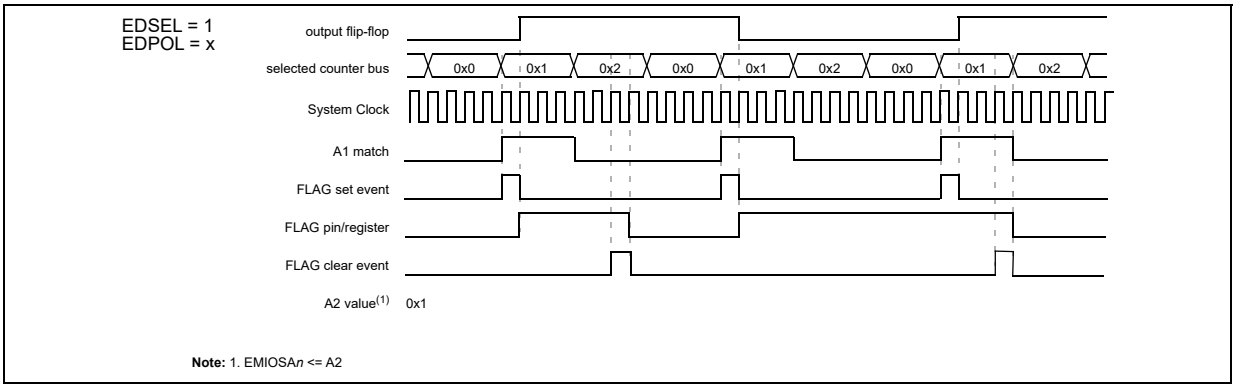


Figure 672. SAOC example with flag behavior



51.4.1.1.4 Input Pulse Width Measurement (IPWM) Mode

The IPWM mode (MODE[25:31] = 0000100) allows the measurement of the width of a positive or negative pulse by capturing the leading edge on register B1 and the trailing edge on register A2. Successive captures are done on consecutive edges of opposite polarity. The leading edge sensitivity (that is, pulse polarity) is selected by EDPOL bit in the EMIOSC*n* register. Registers EMIOSA*n* and EMIOSB*n* return the values in register A2 and B1, respectively.

The capture function of register A2 remains disabled until the first leading edge triggers the first input capture on register B2. When this leading edge is detected, the count value of the selected time base is latched into register B2; the FLAG bit is not set. When the trailing edge is detected, the count value of the selected time base is latched into register A2 and, at the same time, the FLAG bit is set and the content of register B2 is transferred to register B1 and to register A1.

If subsequent input capture events occur while the corresponding FLAG bit is set, registers A2, B1 and A1 will be updated with the latest captured values and the FLAG will remain set. Registers EMIOSA*n* and EMIOSB*n* return the value in registers A2 and B1, respectively.

In order to guarantee coherent access, reading EMIOSA*n* forces B1 be updated with the content of register A1. At the same time transfers between B2 and B1 are disabled until the next read of EMIOSB*n* register. Reading EMIOSB*n* register forces B1 be updated with A1 register content and re-enables transfers from B2 to B1, to take effect at the next trailing edge capture. Transfers from B2 to A1 are not blocked at any time.

The input pulse width is calculated by subtracting the value in B1 from A2.

Figure 673 shows how the Unified Channel can be used for input pulse width measurement.

Figure 673. Input pulse width measurement example

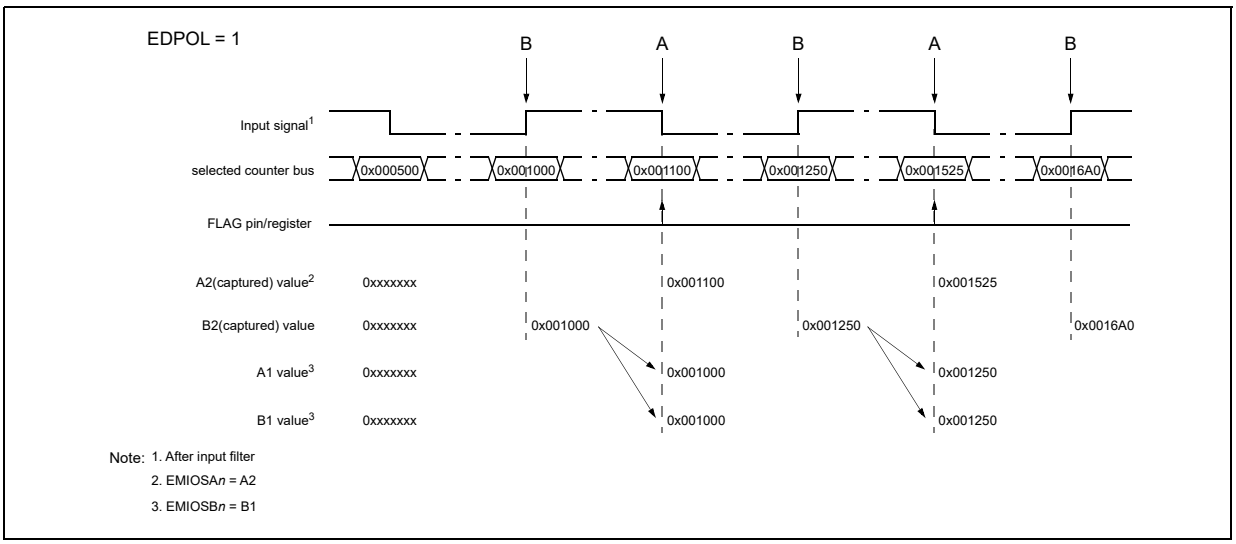
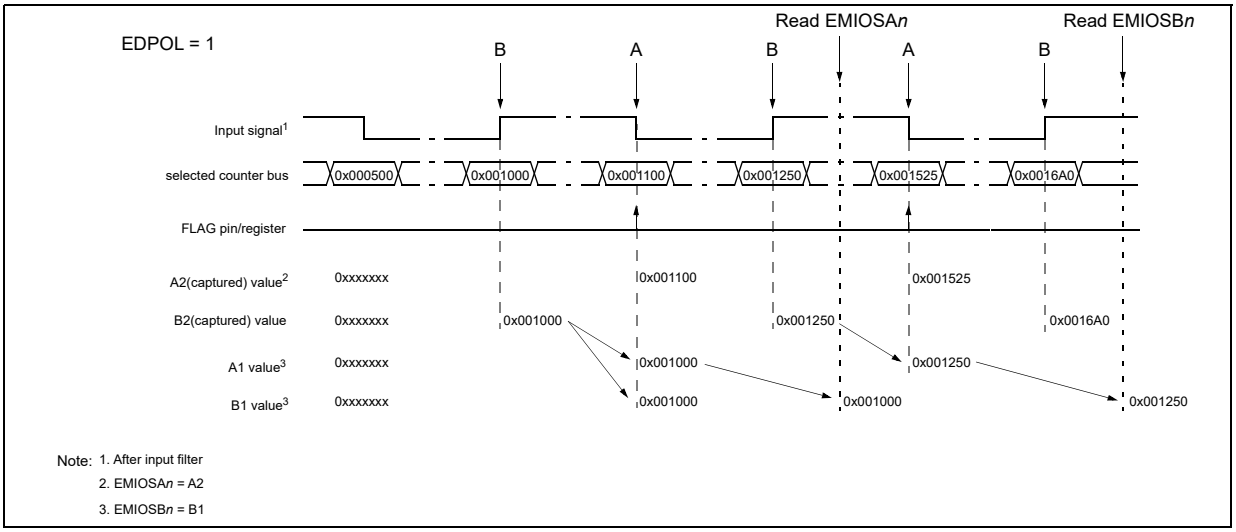


Figure 674 shows the A1 and B1 updates when EMIOSA*n* and EMIOSB*n* register reads occur. Note that A1 register has always coherent data related to A2 register. Note also that when EMIOSA*n* read is performed B1 register is loaded with A1 register content. This guarantees that the data in register B1 has always the coherent data related to the last EMIOSA*n* read. The B1 register updates remain locked until EMIOSB*n* read occurs. If EMIOSA*n* read is performed B1 is updated with A1 register content, even if B1 update is locked by a previous EMIOSA*n* read operation.

Figure 674. B1 and A1 updates at EMIOSA*n* and EMIOSB*n* reads



Reading EMIOSA*n* followed by EMIOSB*n* always provides coherent data. If not coherent data is required for any reason, the sequence of reads should be inverted, therefore EMIOSB*n* should be read prior to EMIOSA*n* register. Note that even in this case B1 register updates will be blocked after EMIOSA*n* read, thus a second EMIOSB*n* is required in order to release B1 register updates.

51.4.1.1.5 Input Period Measurement (IPM) mode

The IPM mode (MODE[25:31] = 0000101) allows the measurement of the period of an input signal by capturing two consecutive rising edges or two consecutive falling edges. Successive input captures are done on consecutive edges of the same polarity. The edge polarity is defined by the EDPOL bit in the EMIOSC*n* register.

Note: The input signal must have at least four system clock cycles period in order to be properly captured by the synchronization logic at the channel input even if the input filter is in by-pass mode.

When the first edge of selected polarity is detected, the selected time base is latched into the registers A2 and B2, and the data previously held in register B2 is transferred to register B1. On this first capture the FLAG line is not set, and the values in registers B1 is meaningless. On the second and subsequent captures, the FLAG line is set and data in register B2 is transferred to register B1.

When the second edge of the same polarity is detected, the counter bus value is latched into registers A2 and B2, the data previously held in register B2 is transferred to data register B1 and to register A1. The FLAG bit is set to indicate the start and end points of a complete period have been captured. This sequence of events is repeated for each subsequent capture. Registers EMIOSA*n* and EMIOSB*n* return the values in register A2 and B1, respectively.

In order to allow coherent data, reading EMIOSA*n* forces A1 content be transferred to B1 register and disables transfers between B2 and B1. These transfers are disabled until the next read of the EMIOSB*n* register. Reading EMIOSB*n* register forces A1 content to be transferred to B1 and re-enables transfers from B2 to B1, to take effect at the next edge capture.

The input pulse period is calculated by subtracting the value in B1 from A2.

Figure 675 shows how the Unified Channel can be used for input period measurement.

Figure 675. Input period measurement example

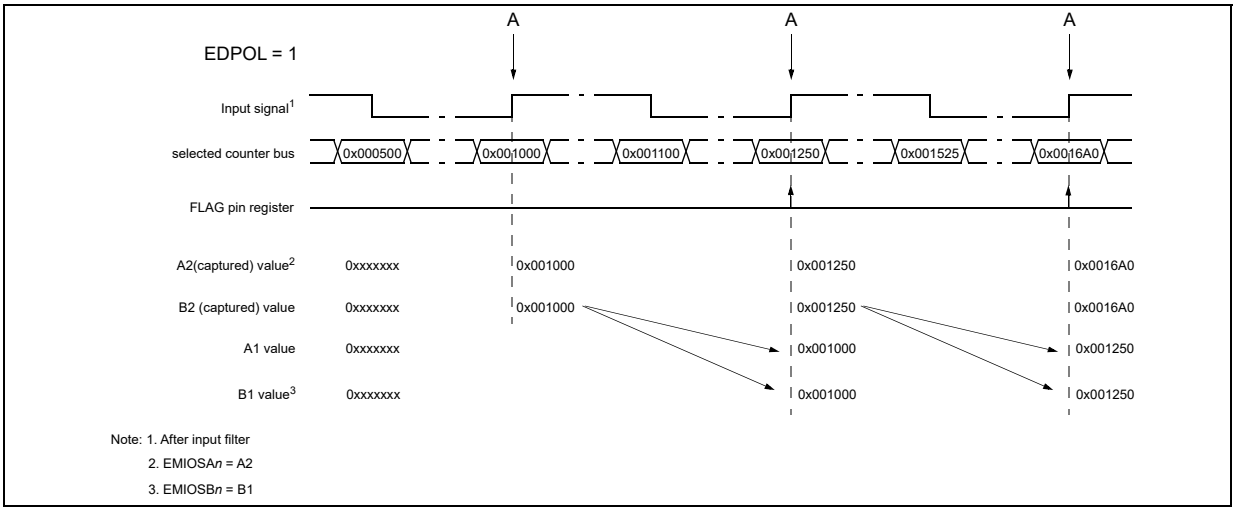
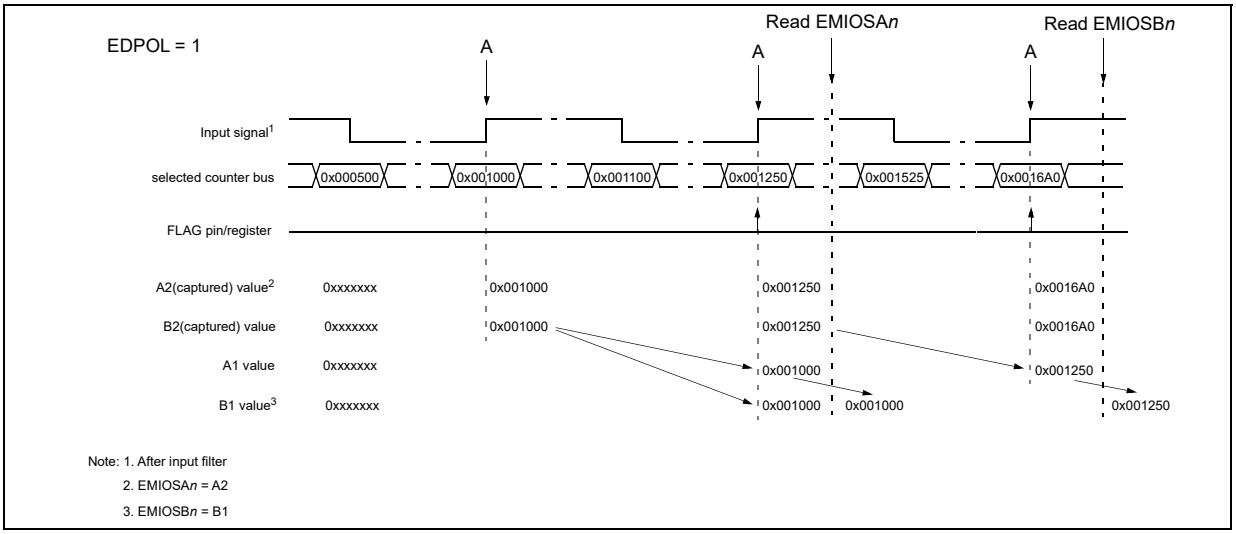


Figure 676 describes the A1 and B1 register updates when EMIOSA*n* and EMIOSB*n* read operations are performed. When EMIOSA*n* read occurs the content of A1 is transferred to B1 thus providing coherent data in A2 and B1 registers. Transfers from B2 to B1 are then blocked until EMIOSB*n* is read. After EMIOSB*n* is read, register A1 content is transferred to

register B1 and the transfers from B2 to B1 are re-enabled to occur at the transfer edges, which is the leading edge in the *Figure 676* example.

Figure 676. A1 and B1 updates at EMIOSA*n* and EMIOSB*n* reads



51.4.1.1.6 Double Action Output Compare (DAOC) mode

In the DAOC mode the leading and trailing edges of the variable pulse width output are generated by matches occurring on comparators A and B. There is no restriction concerning the order in which A and B matches occur.

When the DAOC mode is entered, coming out from GPIO mode both comparators are disabled and the output flip-flop is set to the complement of the EDPOL bit in the EMIOSC*n* register.

Data written to A2 and B2 are transferred to A1 and B1, respectively, on the next system clock cycle if bit OU*n* of the EMIOSOUDIS register is cleared (refer to *Figure 679*). The transfer is blocked if bit OU*n* is set. Comparator A is enabled only after the transfer to A1 register occurs and is disabled on the next A match. Comparator B is enabled only after the transfer to B1 register occurs and is disabled on the next B match. Comparators A and B are enabled and disabled independently.

The output flip-flop is set to the value of EDPOL when a match occurs on comparator A and to the complement of EDPOL when a match occurs on comparator B.

MODE[31] controls if the FLAG is set on both matches (MODE[25:31] = 0000111) or just on the B match (MODE[25:31] = 0000110). FLAG bit assertion depends on comparator enabling.

If subsequent enabled output compares occur on registers A1 and B1, pulses will continue to be generated, regardless of the state of the FLAG bit.

At any time, the FORCMA and FORCMB bits allow the software to force the output flip-flop to the level corresponding to a comparison event in comparator A or B, respectively. Note that the FLAG bit is not affected by these forced operations.

Note: If both registers (A1 and B1) are loaded with the same value, the B match prevails concerning the output pin state (output flip-flop is set to the complement of EDPOL), the FLAG bit is set and both comparators are disabled.

Figure 677 and *Figure 678* show how the Unified Channel can be used to generate a single output pulse with FLAG bit being set on the second match or on both matches, respectively.

Figure 677. Double action output compare with FLAG set on the second match

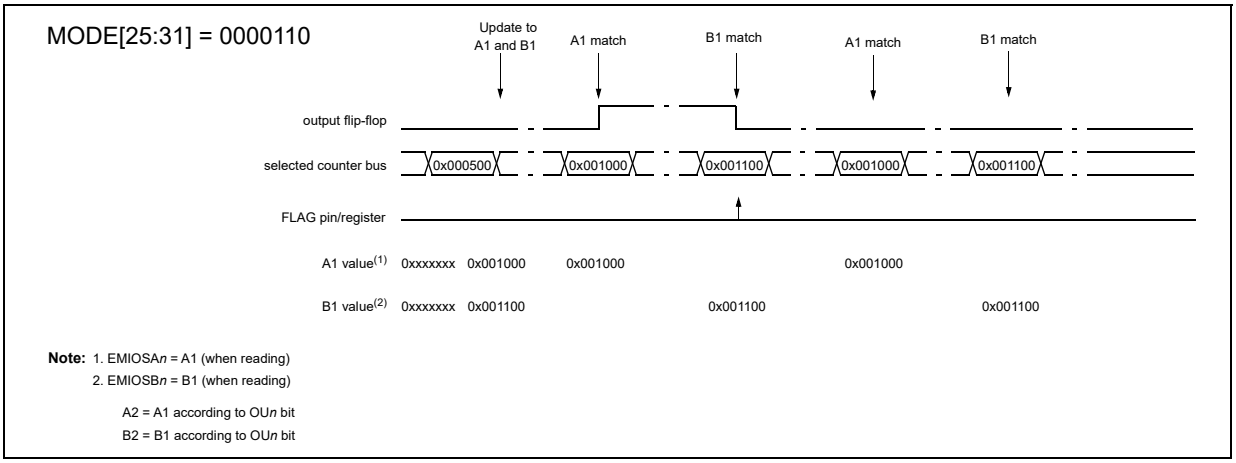


Figure 678. Double action output compare with FLAG set on both matches

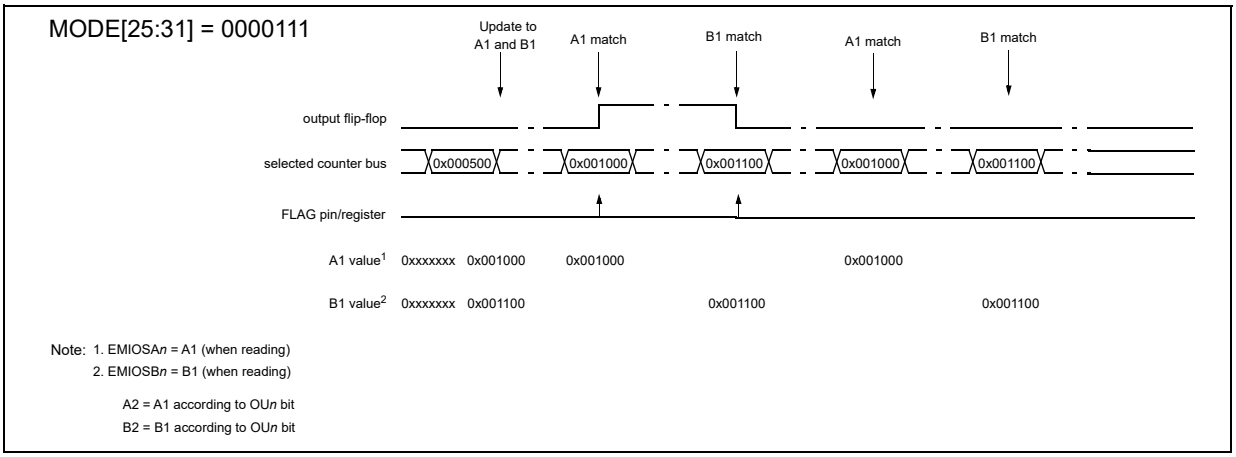
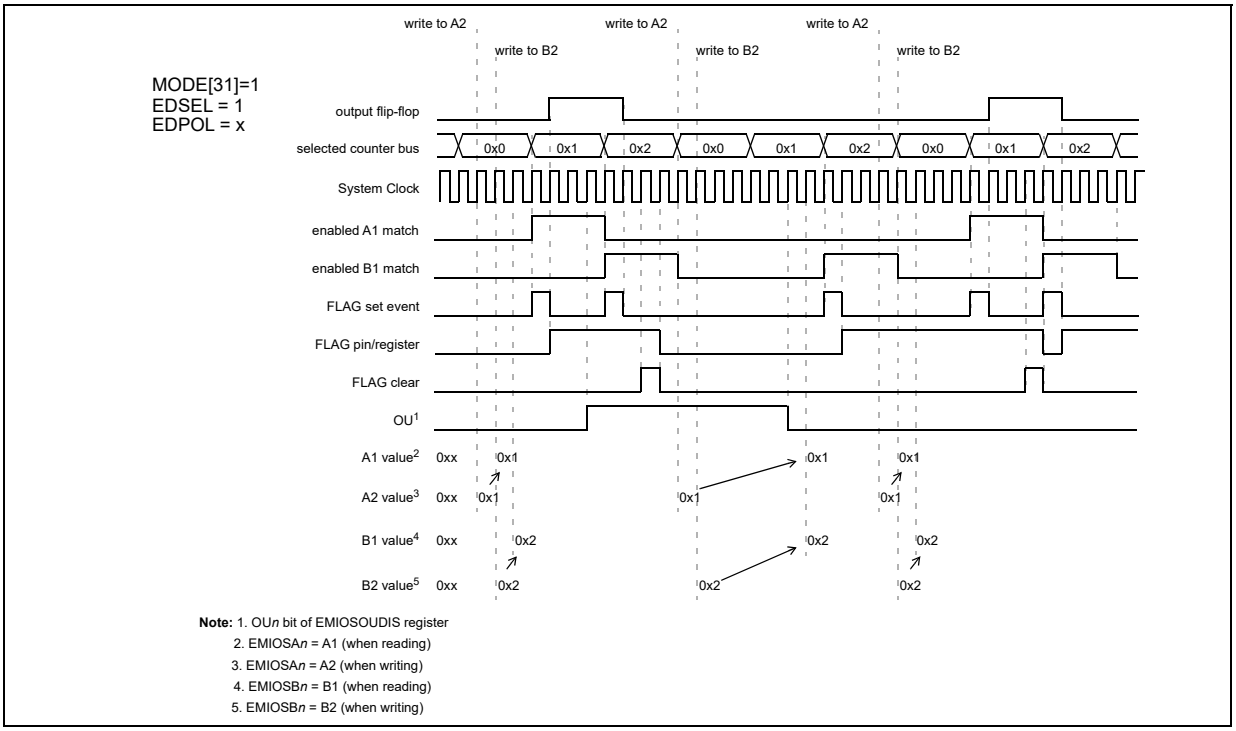


Figure 679. DAOC with transfer disabling example



51.4.1.1.7 Modulus Counter (MC) mode

The MC mode can be used to provide a time base for a counter bus or as a general purpose timer.

Bit MODE[31] selects internal or external clock source when cleared or set, respectively. When external clock is selected, the input signal pin is used as the source and the triggering polarity edge is selected by the EDPOL and EDSEL in the EMIOSC*n* register.

The internal counter counts up from the current value until it matches the value in register A1. Register B1 is cleared and is not accessible to the MCU. Bit MODE[29] selects up mode or up/down mode, when cleared or set, respectively.

When in up count mode, a match between the internal counter and register A1 sets the FLAG and clears the internal counter. The timing of those events varies according to the MC mode setup as follows:

- Internal counter clearing on match start (MODE[25:31] = 001000b)
  - External clock is selected if MODE[31] is set. In this case the internal counter clears as soon as the match signal occurs. The channel FLAG is set at the same time the match occurs. Note that by having the internal counter cleared as soon as the match occurs and incremented at the next input event a shorter zero count is generated. Refer to *Figure 702* and *Figure 703*.
  - Internal clock source is selected if MODE[31] is cleared. In this case the counter clears as soon as the match signal occurs. The channel FLAG is set at the same time the match occurs. At the next prescaler tick after the match the internal

counter remains at zero and only resumes counting on the following tick. Refer to *Figure 702* and *Figure 704*.

- Internal counter clearing on match end (MODE[25:31] = 001001b)
  - External clock is selected if MODE[31] is set. In this case the internal counter clears when the match signal is asserted and the input event occurs. The channel FLAG is set at the same time the counter is cleared. Refer to *Figure 702* and *Figure 705*.
  - Internal clock source is selected if MODE[31] is cleared. In this case the internal counter clears when the match signal is asserted and the prescaler tick occurs. The channel FLAG is set at the same time the counter is cleared. Refer to *Figure 702* and *Figure 705*.

Note: If the internal clock source is selected and the prescaler of the internal counter is set to '1', the MC mode behaves the same way even in Clear on Match Start or Clear on Match End submodes.

When in up/down count mode (MODE[25:31] = 00101bb), a match between the internal counter and register A1 sets the FLAG and changes the counter direction from increment to decrement. A match between register B1 and the internal counter changes the counter direction from decrement to increment and sets the FLAG only if MODE[30] bit is set.

Only values different than 0x0 must be written at A register. Loading 0x0 leads to unpredictable results.

Updates on A register or counter in MC mode may cause loss of match in the current cycle if the transfer occurs near the match. In this case, the counter may rollover and resume operation in the next cycle.

Register B2 has no effect in MC mode. Nevertheless, register B2 can be accessed for reads and writes by addressing EMIOSB.

Figure 680 and *Figure 681* show how the Unified Channel can be used as modulus counter in up mode and up/down mode, respectively.

Figure 680. Modulus Counter Up mode example

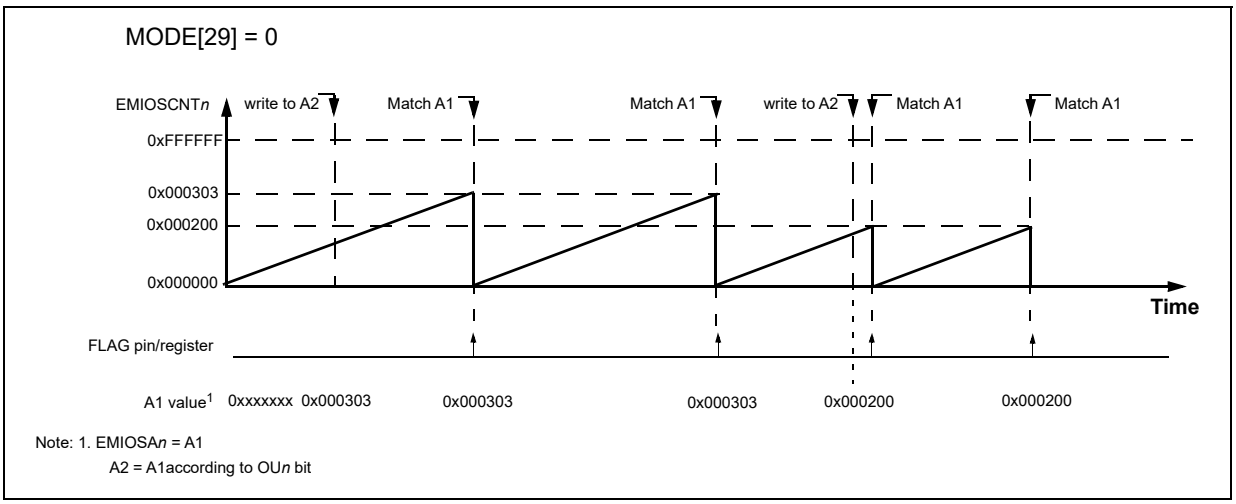
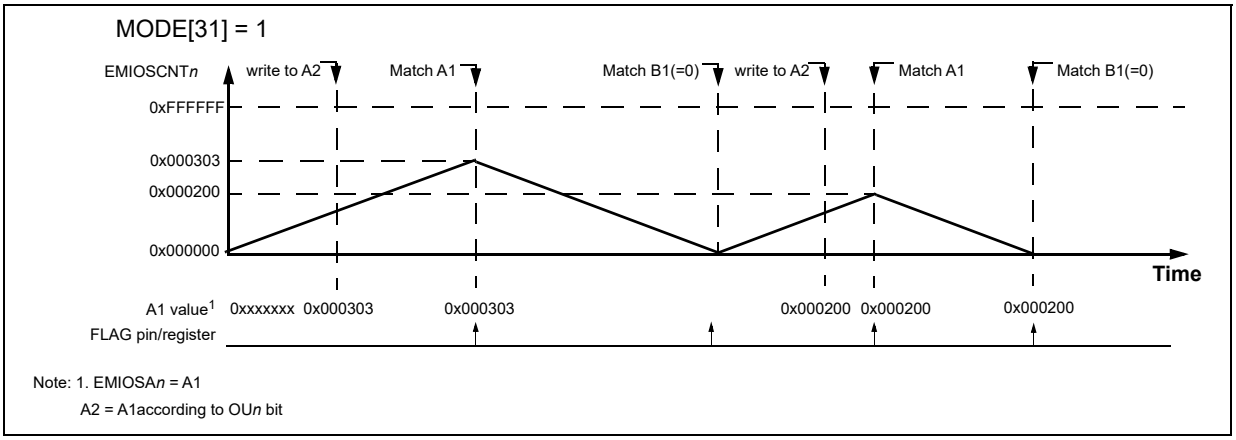


Figure 681. Modulus Counter Up/Down mode example



51.4.1.1.8 Modulus Counter Buffered (MCB) mode

The MCB mode provides a time base which can be shared with other channels through the internal counter buses. Register A1 is double buffered thus allowing smooth transitions between cycles when changing A2 register value on the fly. A1 register is updated at the cycle boundary, which is defined as when the internal counter transitions to 0x1.

The internal counter values operates within a range from 0x1 up to register A1 value. If when entering MCB mode coming out from GPIO mode the internal counter value is not within that range then the A match will not occur causing the channel internal counter to wrap at the maximum counter value which is 0xFFFF for a 16-bit counter. After the counter wrap occurs it returns to 0x1 and resume normal MCB mode operation. Thus in order to avoid the counter wrap condition make sure its value is within the 0x1 to A1 register value range when the MCB mode is entered.

Bit MODE[31] selects internal clock source if cleared or external if set. When external clock is selected the input channel pin is used as the channel clock source. The active edge of this clock is defined by EDPOL and EDSEL bits in the EMIOSC*n* channel register.

When entering in MCB mode, if up counter is selected by MODE[29] = 0 (MODE[25:31] = 101000b), the internal counter starts counting from its current value to up direction until A1 match occurs. The internal counter is set to 0x1 when its value matches A1 value and a clock tick occurs (either prescaled clock or input pin event).

If up/down counter is selected by setting MODE[29] = 1, the counter changes direction at A1 match and counts down until it reaches the value 0x1. After it has reached 0x1 it is set to count in up direction again. B1 register is used to generate a match in order to set the internal counter in up-count direction if up/down mode is selected. Register B1 cannot be changed while this mode is selected.

Note that differently from the MC mode, the MCB mode counts between 0x1 and A1 register value. Only values greater than 0x1 must be written at A1 register. Loading values other than those leads to unpredictable results. The counter cycle period is equal to A1 value in up counter mode. If in up/down counter mode the period is defined by the expression: (2*A1)-2.

Figure 682 describes the counter cycle for several A1 values. Register A1 is loaded with A2 register value at the cycle boundary. Thus any value written to A2 register within cycle **n** will be updated to A1 at the next cycle boundary and therefore will be used on cycle **n+1**. The cycle boundary between cycle **n** and cycle **n+1** is defined as when the internal counter

transitions from A1 value in cycle **n** to 0x1 in cycle **n+1**. Note that the FLAG is generated at the cycle boundary and has a synchronous operation, meaning that it is asserted one system clock cycle after the FLAG set event.

Figure 682. Modulus Counter Buffered (MCB) Up Count mode

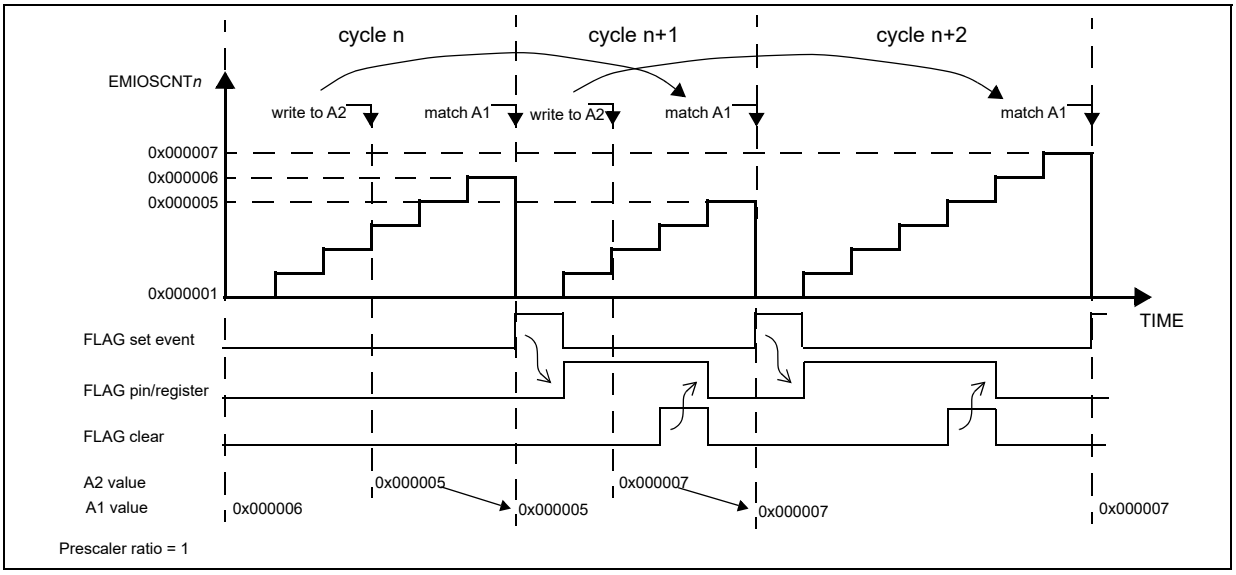


Figure 683 describes the MCB in up/down counter mode (MODE[25:31] = 10101bb). A1 register is updated at the cycle boundary. If A2 is written in cycle **n**, this new value will be used in cycle **n+1** for A1 match. Flags are generated only at A1 match start if MODE[30] is 0. If MODE[30] is set to 1 flags are also generated at the cycle boundary.

Figure 683. Modulus Counter Buffered (MCB) Up/Down mode

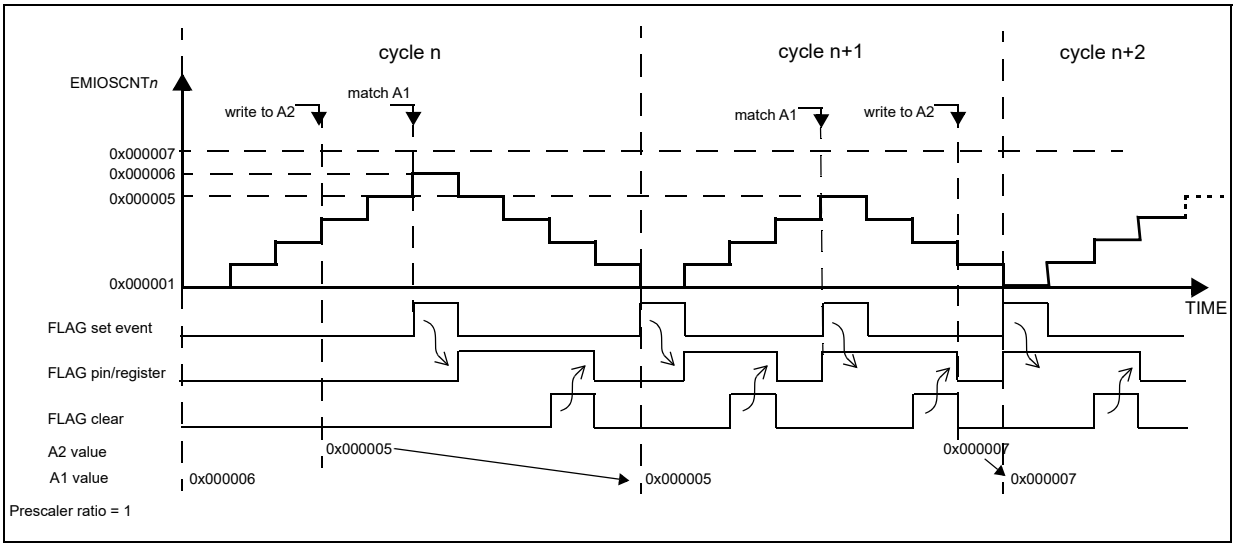


Figure 684 describes in more detail the A1 register update process in up counter mode. The A1 load signal is generated at the last system clock period of a counter cycle. Thus, A1 is updated with A2 value at the same time that the counter (EMIOSCNT*n*) is loaded with 0x1. The load signal pulse has the duration of one system clock period. If A2 is written within cycle **n** its value is available at A1 at the first clock of cycle **n+1** and the new value is used

for match at cycle **n+1**. The update disable bits OU*n* of EMIOSOUDIS register can be used to control the update of this register, thus allowing to delay the A1 register update for synchronization purposes.

Figure 684. MCB Mode A1 Register Update in Up Counter mode

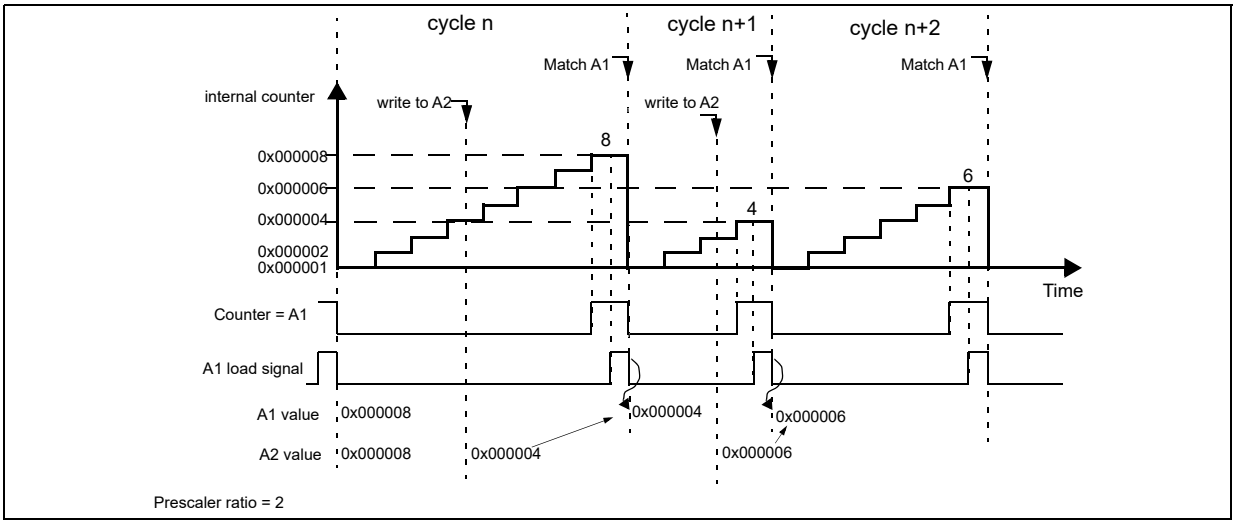
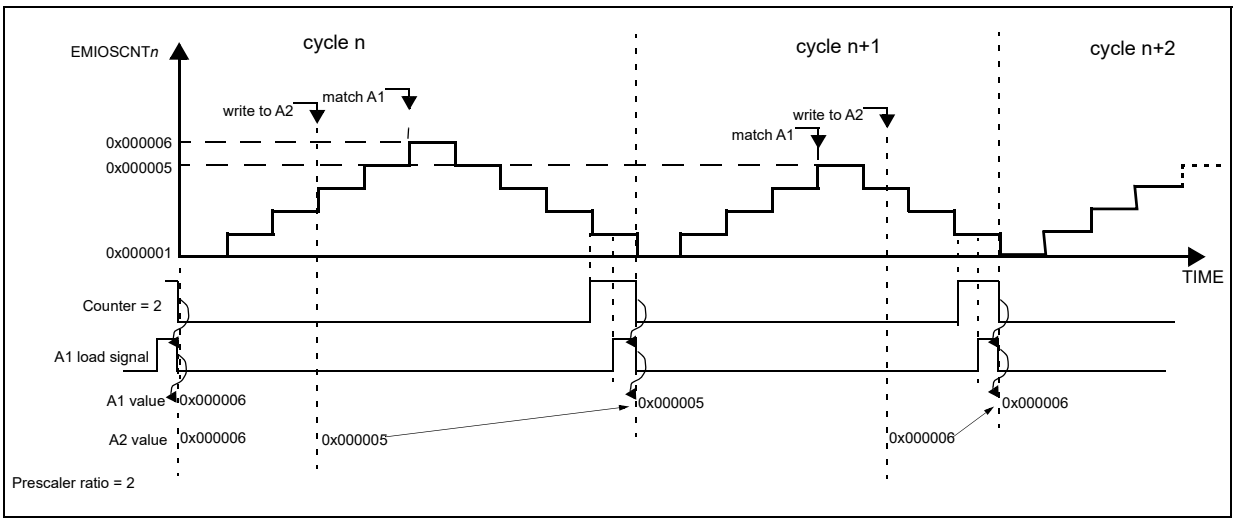


Figure 685 describes the A1 register update in up/down counter mode. Note that A2 can be written at any time within cycle **n** in order to be used in cycle **n+1**. Thus A1 receives this new value at the next cycle boundary. Note that the update disable bits OU*n* of EMIOSOUDIS register can be used to disable the update of A1 register.

Figure 685. MCB Mode A1 Register Update in Up/Down Counter mode



51.4.1.1.9 Output Pulse Width and Frequency Modulation Buffered (OPWFMB) mode

This mode (MODE[25:31] = 10110b0) provides waveforms with variable duty cycle and frequency. The internal channel counter is automatically selected as the time base when this mode is selected. A1 register indicates the duty cycle and B1 register the frequency. Both A1 and B1 registers are double buffered to allow smooth signal generation when changing the registers values on the fly. 0% and 100% duty cycles are supported.

At OPWFMB mode entry the output flip-flop is set to the value of the EDPOL bit in the EMIOSC*n* register.

If when entering OPWFMB mode coming out from GPIO mode the internal counter value is not within that range then the B match will not occur causing the channel internal counter to wrap at the maximum counter value which is 0xFFFF for a 16-bit counter. After the counter wrap occurs it returns to 0x1 and resume normal OPWFMB mode operation. Thus in order to avoid the counter wrap condition make sure its value is within the 0x1 to B1 register value range when the OPWFMB mode is entered.

When a match on comparator A occurs the output register is set to the value of EDPOL. When a match on comparator B occurs the output register is set to the complement of EDPOL. B1 match also causes the internal counter to transition to 0x1, thus restarting the counter cycle.

Only values greater than 0x1 are allowed to be written to B1 register. Loading values other than those leads to unpredictable results. If you want to configure the module for OPWFMB mode, ensure that the B1 register is modified before the mode is set.

Figure 686 describes the operation of the OPWFMB mode regarding output pin transitions and A1/B1 registers match events. Note that the output pin transition occurs when the A1 or B1 match signal is deasserted which is indicated by the A1 match negedge detection signal. If register A1 is set to 0x4 the output pin transitions 4 counter periods after the cycle had started, plus one system clock cycle. Note that in the example shown in *Figure 686* the internal counter prescaler has a ratio of two.

Figure 686. OPWFMB A1 and B1 match to Output Register Delay

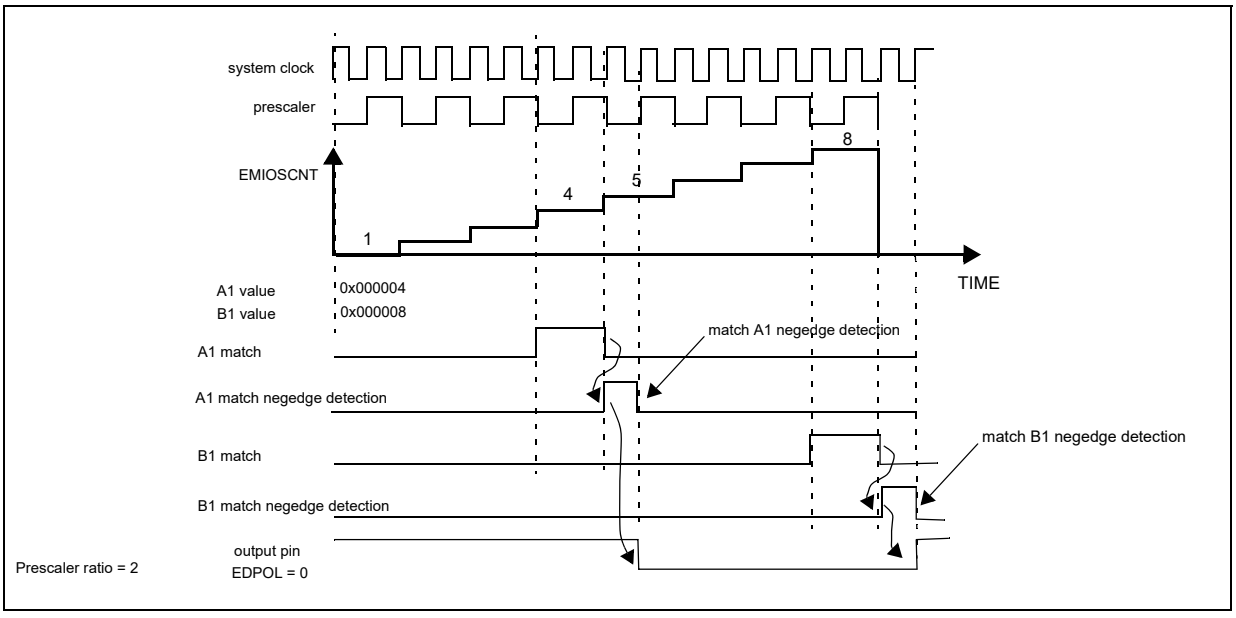


Figure 687 describes the generated output signal if A1 is set to 0x0. Since the counter does not reach zero in this mode, the channel internal logic infers a match as if A1 = 0x1 with the difference that in this case, the posedge of the match signal is used to trigger the output pin transition instead of the negedge used when A1 = 0x1. Note that A1 posedge match signal from cycle **n+1** occurs at the same time as B1 negedge match signal from cycle **n**. This allows to use the A1 posedge match to mask the B1 negedge match when they occur at the same time. The result is that no transition occurs on the output flip-flop and a 0% duty cycle is generated.

Figure 687. OPWFMB Mode with A1 = 0 (0% duty cycle)

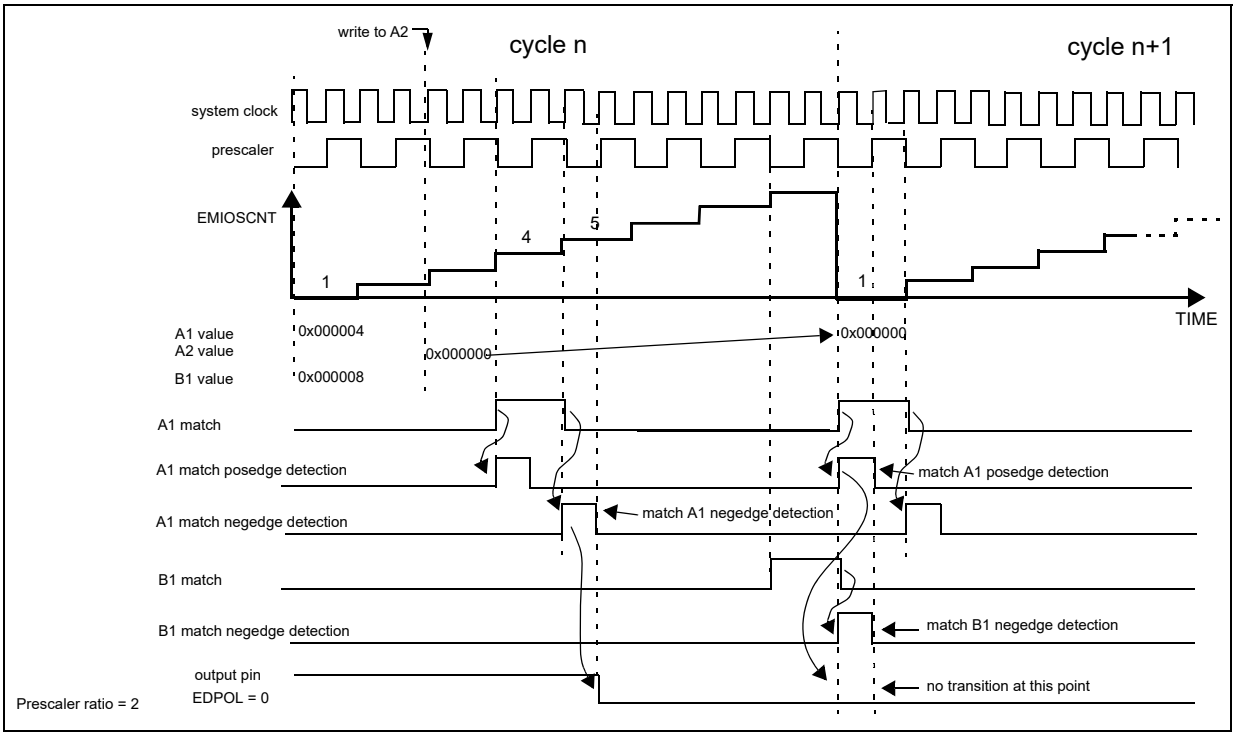
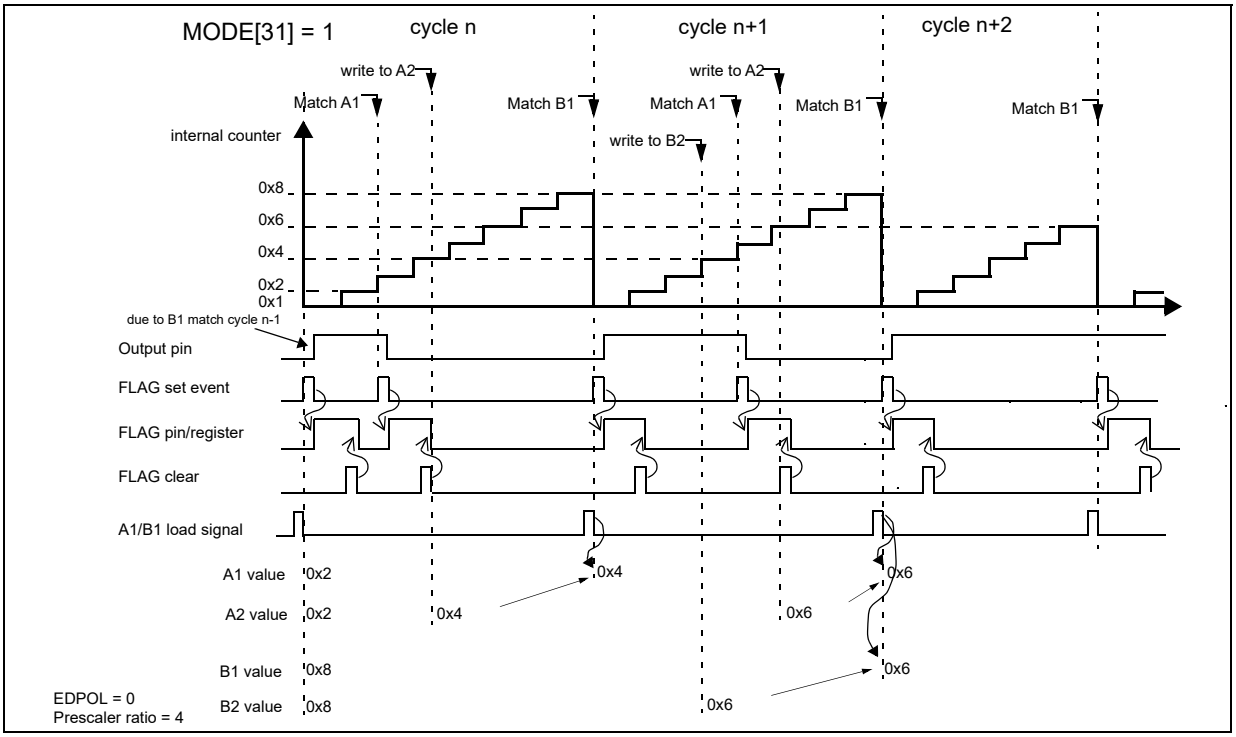


Figure 688 describes the timing for the A1 and B1 registers load. The A1 and B1 load use the same signal which is generated at the last system clock period of a counter cycle. Thus, A1 and B1 are updated respectively with A2 and B2 values at the same time that the counter (EMIOSCNT*n*) is loaded with 0x1. This event is defined as the cycle boundary. The load signal pulse has the duration of one system clock period. If A2 and B2 are written within cycle **n** their values are available at A1 and B1, respectively, at the first clock of cycle **n+1** and the new values are used for matches at cycle **n+1**. The update disable bits OU*n* of EMIOSOUDIS register can be used to control the update of these registers, thus allowing to delay the A1 and B1 registers update for synchronization purposes.

In *Figure 688* it is assumed that both the channel and global prescalers are set to 0x1 (each divide ratio is two), meaning that the channel internal counter transitions at every four system clock cycles. FLAGs can be generated only on B1 matches when MODE[30] is cleared, or on both A1 and B1 matches when MODE[30] is set. Since B1 flag occurs at the cycle boundary, this flag can be used to indicate that A2 or B2 data written on cycle **n** were loaded to A1 or B1, respectively, thus generating matches in cycle **n+1**. Note that the FLAG has a synchronous operation, meaning that it is asserted one system clock cycle after the FLAG set event.

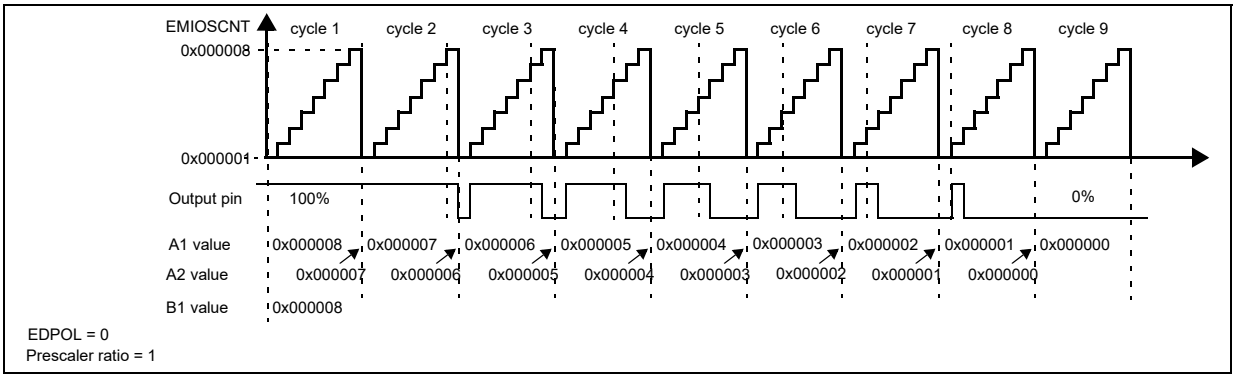
Figure 688. OPWFMB A1 and B1 registers update and flags



The FORCMA and FORCMB bits allow the software to force the output flip-flop to the level corresponding to a match on comparators A or B respectively. Similarly to a B1 match FORCMB sets the internal counter to 0x1. The FLAG bit is not set by the FORCMA or FORCMB bits being asserted.

Figure 689 describes the generation of 100% and 0% duty cycle signals. It is assumed EDPOL = 0 and the resultant prescaler value is 1. Initially A1 = 0x8 and B1 = 0x8. In this case, B1 match has precedence over A1 match, thus the output flip-flop is set to the complement of EDPOL bit. This cycle corresponds to a 100% duty cycle signal. The same output signal can be generated for any A1 value greater than or equal to B1.

Figure 689. OPWFMB mode from 100% to 0% duty cycle



A 0% duty cycle signal is generated if A1 = 0x0 as shown in *Figure 689* cycle 9. In this case B1 = 0x8 match from cycle 8 occurs at the same time as the A1 = 0x0 match from cycle 9. Refer to *Figure 687* for a description of the A1 and B1 match generation. In this case A1 match has precedence over B1 match and the output signal transitions to EDPOL.

51.4.1.1.10 Center Aligned Output PWM Buffered with Dead-Time (OPWMCB) mode

This operation mode generates a center aligned PWM with dead time insertion to the leading (MODE[25:31] = 10111b1) or trailing edge (MODE[25:31] = 10111b0). A1 and B1 registers are double buffered to allow smooth output signal generation when changing A2 or B2 registers values on the fly.

Bits BSL[21:22] select the time base. The time base selected for a channel configured to OPWMCB mode should be a channel configured to MCB Up/Down mode, as shown in *Figure 683*. It is recommended to start the MCB channel time base after the OPWMCB mode is entered in order to avoid missing A matches at the very first duty cycle.

Register A1 contains the ideal duty cycle for the PWM signal and is compared with the selected time base.

Register B1 contains the dead time value and is compared against the internal counter. For a leading edge dead time insertion, the output PWM duty cycle is equal to the difference between register A1 and register B1, and for a trailing edge dead time insertion, the output PWM duty cycle is equal to the sum of register A1 and register B1. Bit Mode[31] selects between trailing and leading dead time insertion, respectively.

Note: The internal counter runs in the internal prescaler ratio, while the selected time base may be running in a different prescaler ratio.

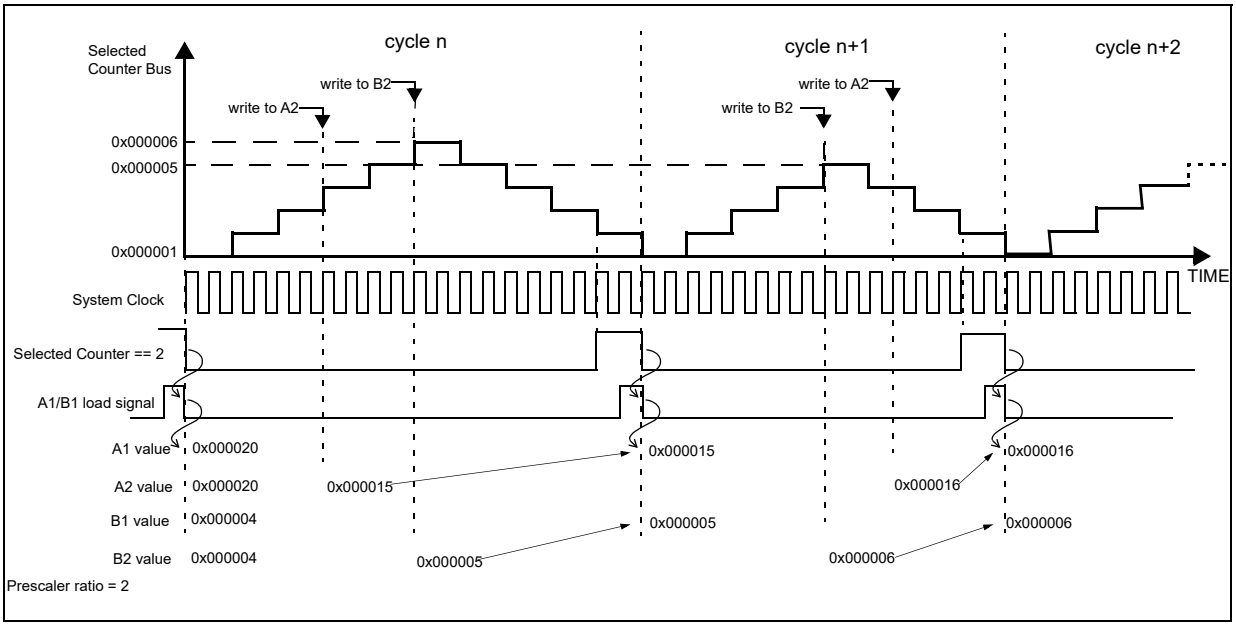
When OPWMCB mode is entered, coming out from GPIO mode, the output flip-flop is set to the complement of the EDPOL bit in the EMIOSC*n* register.

The following basic steps summarize the proper OPWMCB startup, assuming the channels are initially in GPIO mode:

1. *[global]* Disable Global Prescaler;
2. *[MCB channel]* Disable Channel Prescaler;
3. *[MCB channel]* Write 0x1 at internal counter;
4. *[MCB channel]* Set A register;
5. *[MCB channel]* Set channel to MCB Up mode;
6. *[MCB channel]* Set prescaler ratio;
7. *[MCB channel]* Enable Channel Prescaler;
8. *[OPWMCB channel]* Disable Channel Prescaler;
9. *[OPWMCB channel]* Set A register;
10. *[OPWMCB channel]* Set B register;
11. *[OPWMCB channel]* Select time base input through BSL[21:22] bits;
12. *[OPWMCB channel]* Enter OPWMCB mode;
13. *[OPWMCB channel]* Set prescaler ratio;
14. *[OPWMCB channel]* Enable Channel Prescaler;
15. *[global]* Enable Global Prescaler.

Figure 690 describes the load of A1 and B1 registers which occurs when the selected counter bus transitions from 0x2 to 0x1. This event defines the cycle boundary. Note that values written to A2 or B2 within cycle **n** are loaded into A1 or B1 registers, respectively, and used to generate matches in cycle **n+1**.

Figure 690. OPWMCB A1 and B1 registers load

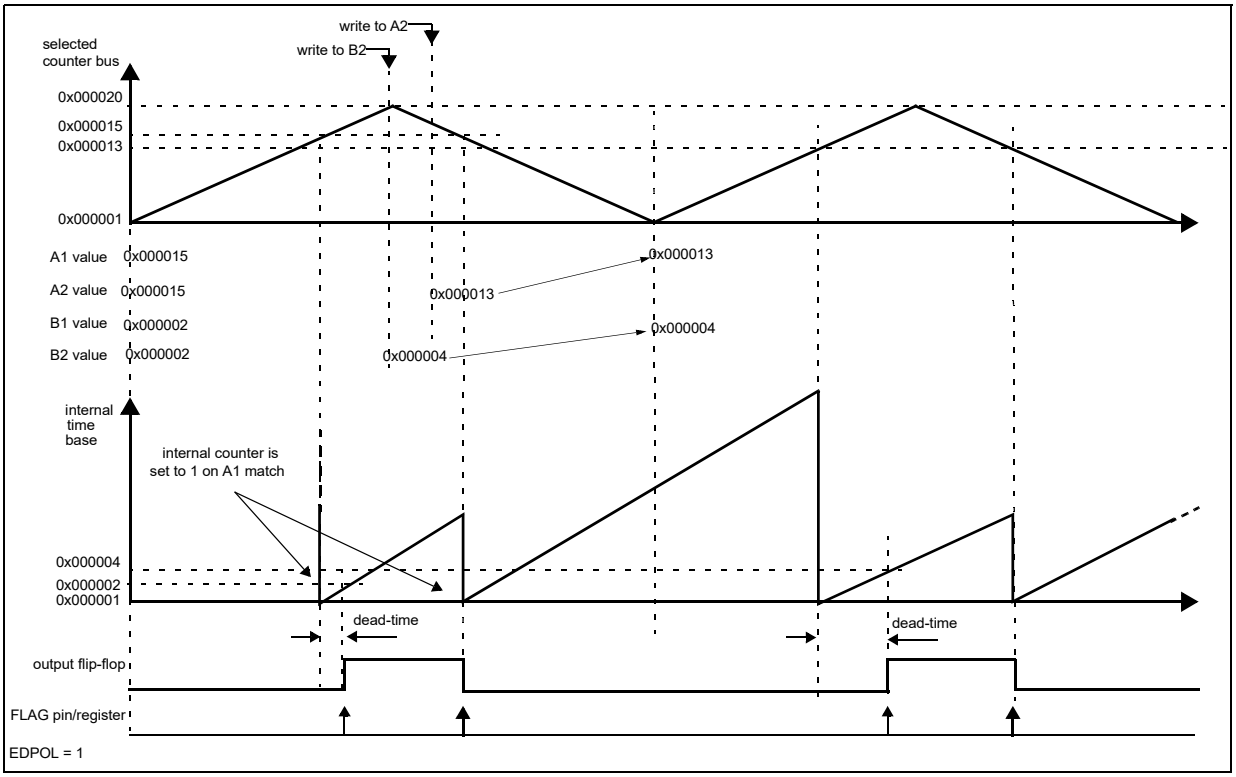


Bit OU*n* of the EMIOSOUDIS register can be used to disable the A1 and B1 updates, thus allowing to synchronize the load on these registers with the load of A1 or B1 registers in others channels. Note that, using the update disable bit A1 and B1 registers can be updated at the same counter cycle thus allowing to change both registers at the same time.

In this mode, A1 matches always set the internal counter to 0x1. When operating with leading edge dead time insertion the first A1 match sets the internal counter to 0x1. When a match occurs between register B1 and the internal time base, the output flip-flop is set to the value of the EDPOL bit. In the following match between register A1 and the selected time base, the output flip-flop is set to the complement of the EDPOL bit. This sequence repeats continuously. The internal counter should not reach 0x0 as a consequence of a rollover. In order to avoid it, the user should not write to the EMIOSB register a value greater than twice the difference between external count up limit and EMIOSA value.

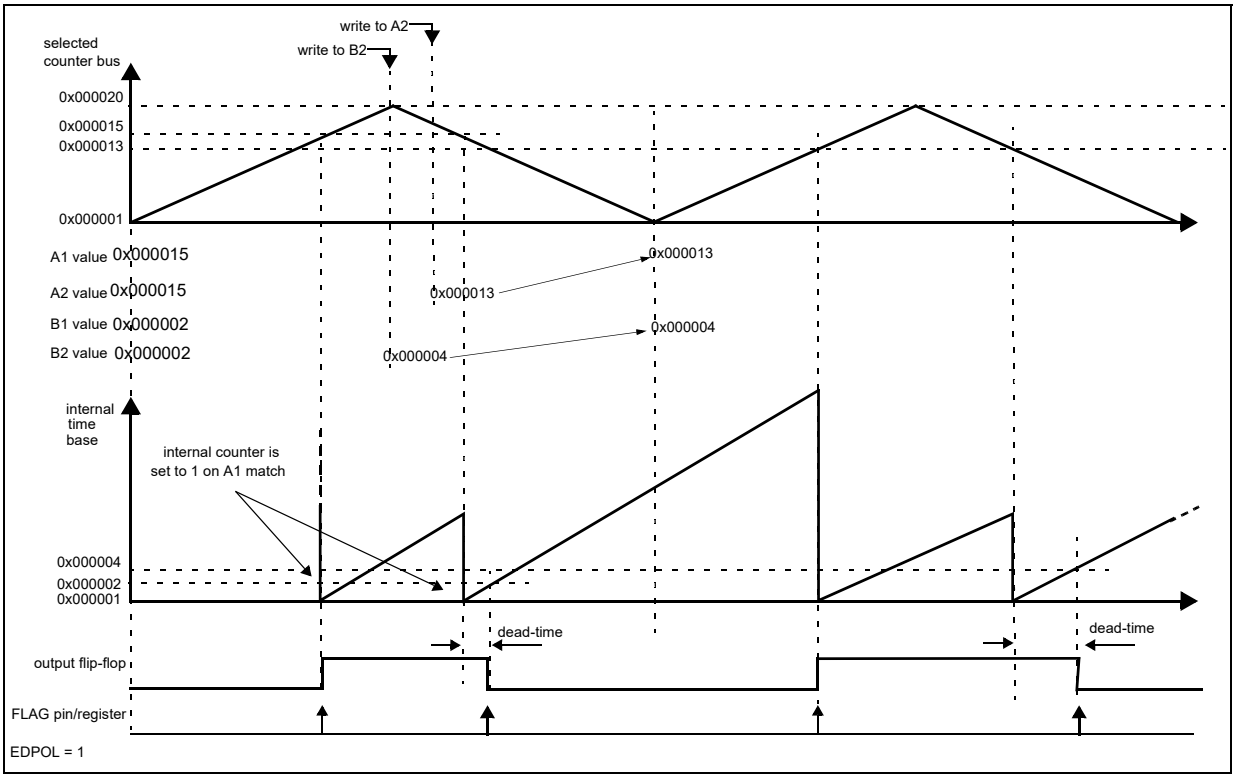
Figure 691 shows two cycles of a Center Aligned PWM signal. Note that, both A1 and B1 register values are changing within the same cycle which allows to vary at the same time the duty cycle and dead time values.

Figure 691. OPWMCB with lead dead time insertion



When operating with trailing edge dead time insertion, the first match between A1 and the selected time base sets the output flip-flop to the value of the EDPOL bit and sets the internal counter to 0x1. In the second match between register A1 and the selected time base, the internal counter is set to 0x1 and B1 matches are enabled. When the match between register B1 and the selected time base occurs the output flip-flop is set to the complement of the EDPOL bit. This sequence repeats continuously.

Figure 692. OPWMCB with trail dead time insertion



FLAG can be generated in the trailing edge of the output PWM signal when MODE[30] is cleared, or in both edges, when MODE[30] is set. If subsequent matches occur on comparators A and B, the PWM pulses continue to be generated, regardless of the state of the FLAG bit.

Note: In OPWMCB mode, FORCMA and FORCMB do not have the same behavior as a regular match. Instead, they force the output flip-flop to constant value which depends upon the selected dead time insertion mode, lead or trail, and the value of the EDPOL bit.

FORCMA has different behaviors depending upon the selected dead time insertion mode, lead or trail. In lead dead time insertion FORCMA force a transition in the output flip-flop to the opposite of EDPOL. In trail dead time insertion the output flip-flop is forced to the value of EDPOL bit.

If bit FORCMB is set, the output flip-flop value depends upon the selected dead time insertion mode. In lead dead time insertion FORCMB forces the output flip-flop to transition to EDPOL bit value. In trail dead time insertion the output flip-flop is forced to the opposite of EDPOL bit value.

Note: FORCMA bit set does not set the internal time-base to 0x1 as a regular A1 match.

The FLAG bit is not set either in case of a FORCMA or FORCMB or even if both forces are issued at the same time.

Note: FORCMA and FORCMB have the same behavior even in Freeze or normal mode regarding the output pin transition.

When FORCMA is issued along with FORCMB the output flip-flop is set to the opposite of EDPOL bit value. This is equivalent of saying that.FORCMA has precedence over FORCMB

when lead dead time insertion is selected and FORCMB has precedence over FORCMA when trail dead time insertion is selected.

Duty cycle from 0% to 100% can be generated by setting appropriate values to A1 and B1 registers relatively to the period of the external time base. Setting A1 = 1 generates a 100% duty cycle waveform. Assuming EDPOL is set to '1' and OPWMCB mode with trail dead time insertion, 100% duty cycle signals can be generated if B1 occurs at or after the cycle boundary (external counter = 1). If A1 is greater than the maximum value of the selected counter bus period, then a 0% duty cycle is produced, only if the pin starts the current cycle in the opposite of EDPOL value. In case of 100% duty cycle, the transition from EDPOL to the opposite of EDPOL may be obtained by forcing pin, using FORCMA or FORCMB, or both.

Note: If A1 is set to 0x1 at OPWMCB entry the 100% duty cycle may not be obtained in the very first PWM cycle due to the pin condition at mode entry.

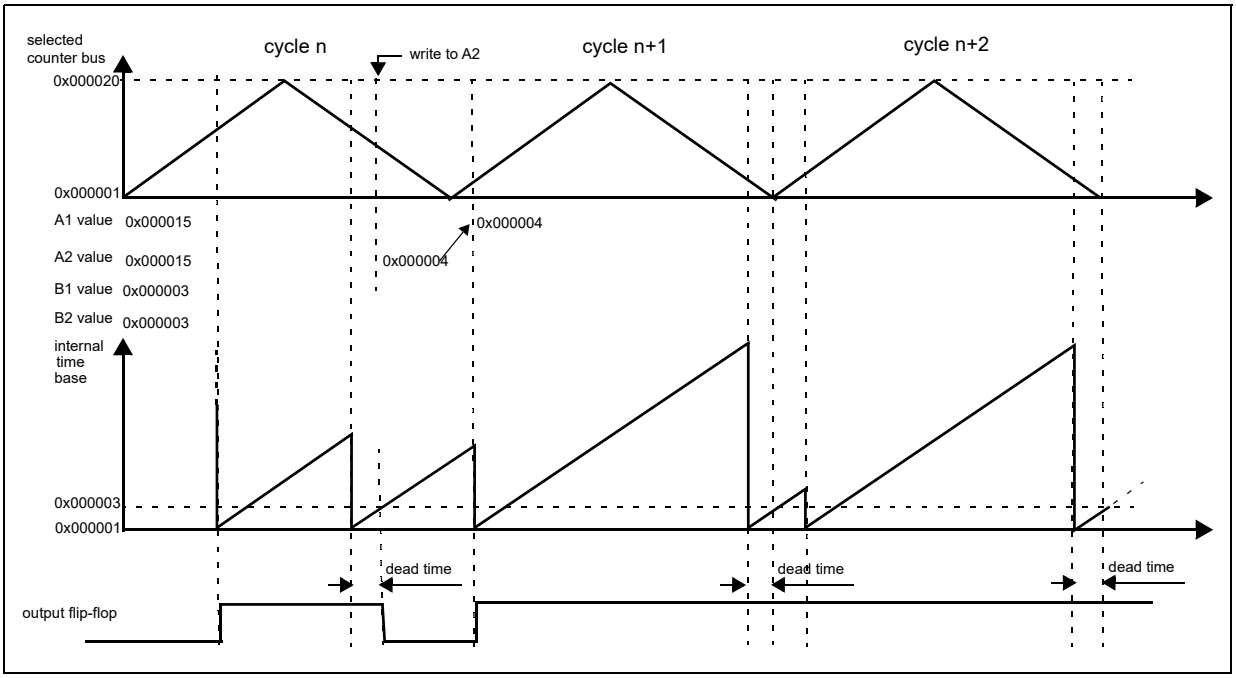
Only values different than 0x0 are allowed to be written to A1 register. If 0x0 is loaded to A1 the results are unpredictable.

Note: A special case occurs when A1 is set to (external counter bus period)/2, which is the maximum value of the external counter. In this case the output flip-flop is constantly set to the EDPOL bit value.

The internal channel logic prevents matches from one cycle to propagate to the next cycle. In trail dead time insertion B1 match from cycle **n** could eventually cross the cycle boundary and occur in cycle **n+1**. In this case B1 match is masked out and does not cause the output flip-flop to transition. Therefore matches in cycle **n+1** are not affected by the late B1 matches from cycle **n**.

Figure 693 shows a 100% duty cycle output signal generated by setting A1 = 4 and B1 = 3. In this case the trailing edge is positioned at the boundary of cycle **n+1**, which is actually considered to belong to cycle **n+2** and therefore does not cause the output flip-flip to transition.

Figure 693. OPWMCB with 100% Duty Cycle (A1 = 4 and B1 = 3)



It is important to notice that, such as in OPWMB and OPWFMB modes, the match signal used to set or clear the channel output flip-flop is generated on the deassertion of the channel combinational comparator output signal which compares the selected time base with A1 or B1 register values. Refer to *Figure 686* which describes the delay from matches to output flip-flop transition in OPWFMB mode. The operation of OPWMCB mode is similar to OPWFMB regarding matches and output pin transition.

51.4.1.1.11 Output Pulse Width Modulation Buffered (OPWMB) Mode

OPWMB mode (MODE[25:31] = 11000b0) is used to generate pulses with programmable leading and trailing edge placement. An external counter driven in MCB Up mode must be selected from one of the counter buses. A1 register value defines the first edge and B1 the second edge. The output signal polarity is defined by the EDPOL bit. If EDPOL is zero, a negative edge occurs when A1 matches the selected counter bus and a positive edge occurs when B1 matches the selected counter bus.

The A1 and B1 registers are double buffered and updated from A2 and B2, respectively, at the cycle boundary. The load operation is similar to the OPWFMB mode. Refer to *Figure 688* for more information about A1 and B1 registers update.

FLAG can be generated at B1 matches, when MODE[30] is cleared, or in both A1 and B1 matches, when MODE[30] is set. If subsequent matches occur on comparators A and B, the PWM pulses continue to be generated, regardless of the state of the FLAG bit.

FORCMA and FORCMB bits allow the software to force the output flip-flop to the level corresponding to a match on A1 or B1 respectively. FLAG bit is not set by the FORCMA and FORCMB operations.

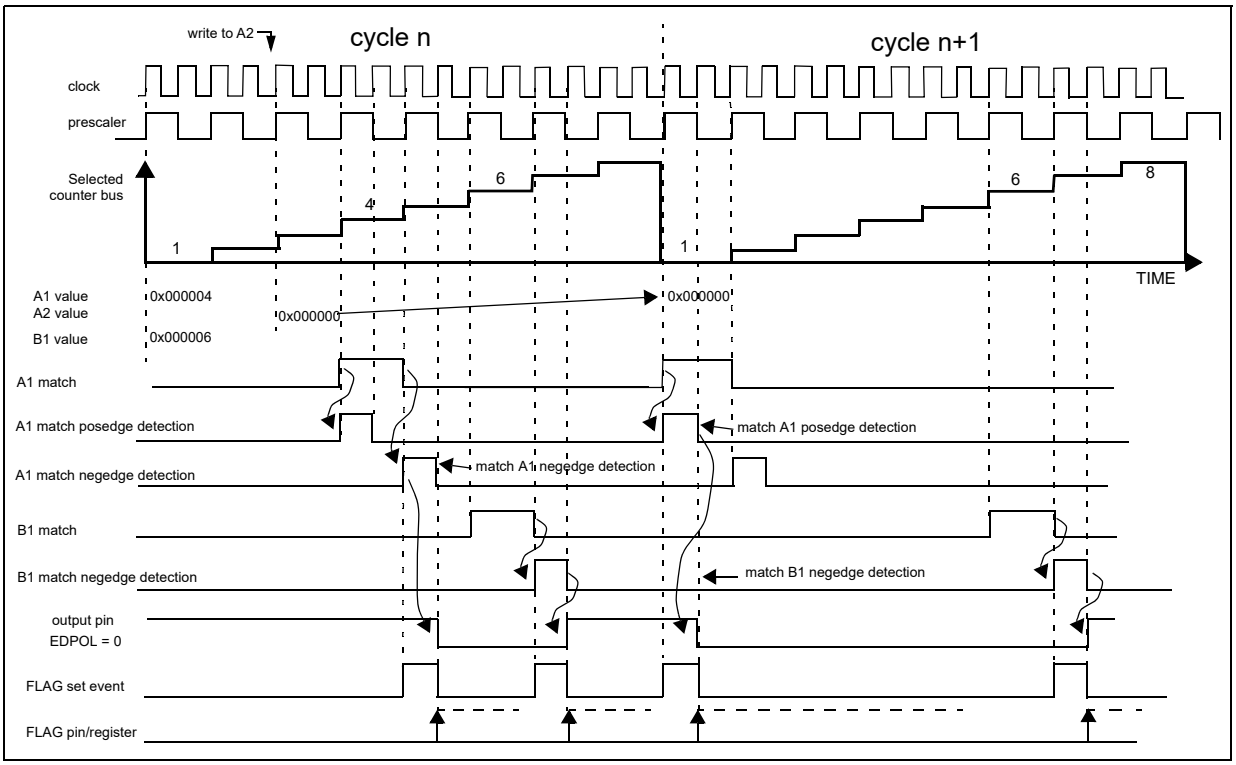
At OPWMB mode entry the output flip-flop is set to the value of the EDPOL bit in the EMIOSC*n* register.

Some rules applicable to the OPWMB mode are:

- B1 matches have precedence over A1 matches if they occur at the same time within the same counter cycle
- A1 = 0 match from cycle **n** has precedence over B1 match from cycle **n-1**
- A1 matches are masked out if they occur after B1 match within the same cycle
- Any value written to A2 or B2 on cycle **n** is loaded to A1 and B1 registers at the following cycle boundary (assuming OU*n* bit of EMIOSOUDIS register is not asserted). Thus the new values will be used for A1 and B1 matches in cycle **n+1**

Figure 694 describes the operation of the OPWMB mode regarding A1 and B1 matches and the transition of the channel output pin. In this example EDPOL is set to '0'.

Figure 694. OPWMB mode matches and flags



Note that the output pin transitions are based on the negedges of the A1 and B1 match signals. *Figure 694* shows in cycle **n+1** the value of A1 register being set to '0'. In this case the match posedge is used instead of the negedge to transition the output flip-flop.

Figure 695 describes the channel operation for 0% duty cycle. Note that the A1 match posedge signal occurs at the same time as the B1 = 0x8 negedge signal. In this case A1 match has precedence over B1 match, causing the output pin to remain at EDPOL bit value, thus generating a 0% duty cycle signal.

Figure 695. OPWMB mode with 0% duty cycle

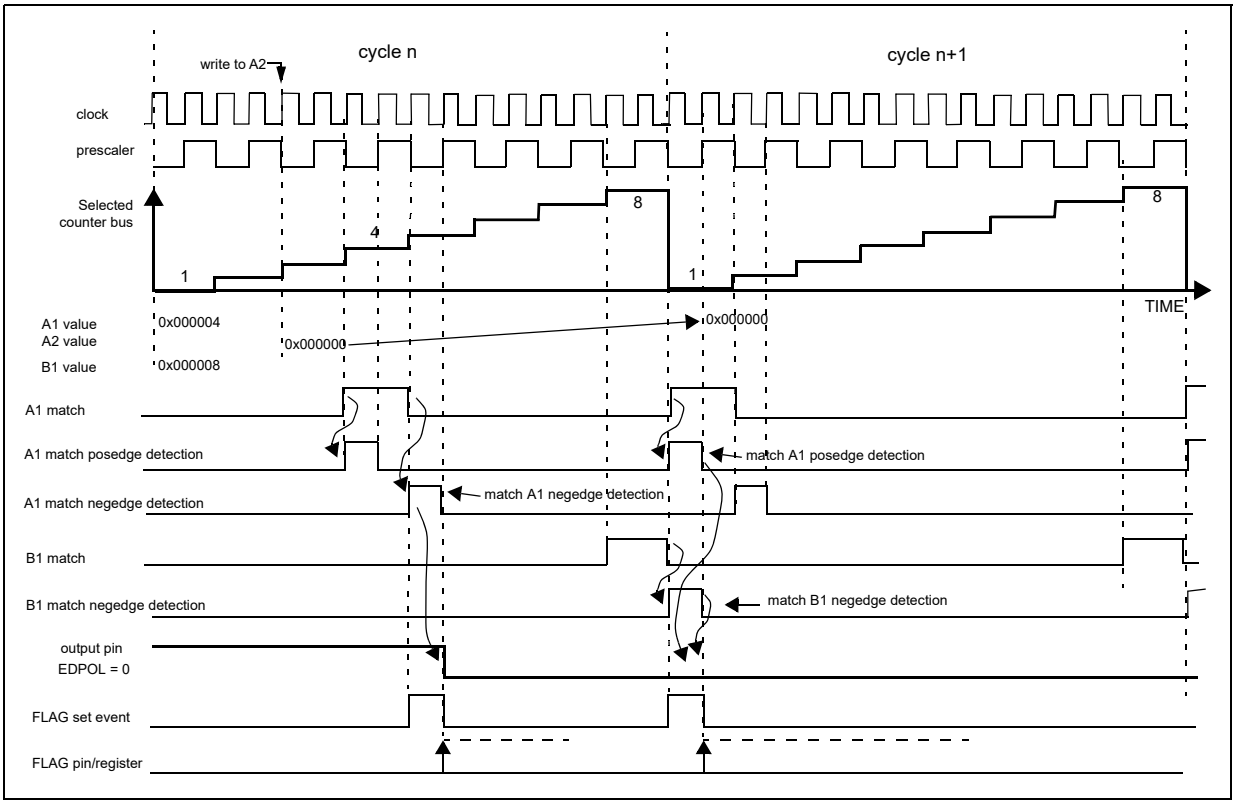
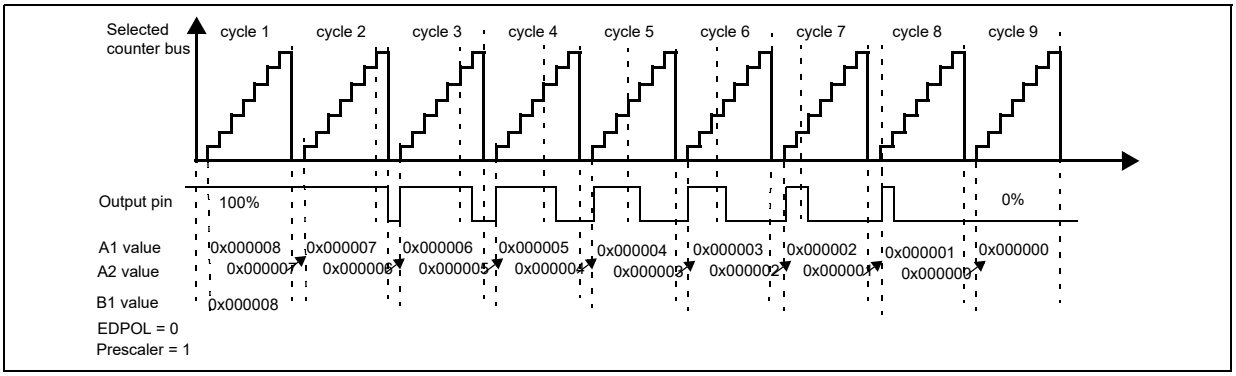


Figure 696 shows a waveform changing from 100% to 0% duty cycle. EDPOL in this case is zero. In this example B1 is programmed to the same value as the period of the external selected time base.

Figure 696. OPWMB mode from 100% to 0% duty cycle



In *Figure 696* if B1 is set to a value lower than 0x8 it is not possible to achieve 0% duty cycle by only changing A1 register value. Since B1 matches have precedence over A1 matches the output pin transitions to the opposite of EDPOL bit at B1 match. Note also that if B1 is set to 0x9, for instance, B1 match does not occur, thus a 0% duty cycle signal is generated.

51.4.1.1.12 Output Pulse Width Modulation with Trigger (OPWMT) mode

OPWMT mode (MODE[25:31] = 0100110) is intended to support the generation of pulse width modulation signals where the period is not modified while the signal is being output,

but where the duty cycle will be varied and must not create glitches. The mode is intended to be used in conjunction with other channels executing in the same mode and sharing a common timebase. It will support each channel with a fixed PWM leading edge position with respect to the other channels and the ability to generate a trigger signal at any point in the period that can be output from the module to initiate activity in other parts of the device such as starting ADC conversions.

An external counter driven in either MC Up or MCB Up mode must be selected from one of the counter buses.

Register A1 defines the leading edge of the PWM output pulse and as such the beginning of the PWM's period. This makes it possible to insure that the leading edge of multiple channels in OPWMT mode can occur at a specific time with respect to the other channels when using a shared timebase. This can allow the introduction of a fixed offset for each channel which can be particularly useful in the generation of lighting PWM control signals where it is desirable that edges are not coincident with each other to help eliminate noise generation. The value of register A1 represents the shift of the PWM channel with respect to the selected timebase. A1 can be configured with any value within the range of the selected time base. Note that registers loaded with 0x0 will not produce matches if the timebase is driven by a channel in MCB mode.

A1 is not buffered as the shift of a PWM channel must not be modified while the PWM signal is being generated. In case A1 is modified it is immediately updated and one PWM pulse could be lost.

EMIOSB*n* address gives access to B2 register for write and B1 register for read. Register B1 defines the trailing edge of the PWM output pulse and as such the duty cycle of the PWM signal. To synchronize B1 update with the PWM signal and so ensure a correct output pulse generation the transfer from B2 to B1 is done at every match of register A1.

EMIOSOUDIS register affects transfers between B2 and B1 only.

In order to account for the shift in the leading edge of the waveform defined by register A1 it will be necessary that the trailing edge, held in register B1, can roll over into the next period. This means that a match against the B1 register should not have to be qualified by a match in the A1 register. The impact of this would mean that incorrectly setting register B1 to a value less that register A1 will result in the output being held over a cycle boundary until the B1 value is encountered.

This mode provides a buffered update of the trailing edge by updating register B1 with register B2 contents only at a match of register A1.

The value loaded in register A1 is compared with the value on the selected time base. When a match on comparator A1 occurs, the output flip-flop is set to the value of the EDPOL bit. When a match occurs on comparator B, the output flip-flop is set to the complement of the EDPOL bit.

Note that the output pin and flag transitions are based on the posedges of the A1, B1 and A2 match signals. Refer to *Figure 694* at *Section 51.4.1.1.11: Output Pulse Width Modulation Buffered (OPWMB) Mode* for details on match posedge.

Register A2 defines the generation of a trigger event within the PWM period and A2 should be configured with any value within the range of the selected time base, otherwise no trigger will be generated. A match on the comparator will generate the FLAG signal but it has no effect on the PWM output signal generation. The typical setup to obtain a trigger with FLAG is to enable DMA and to drive the channel's ipd_done input high.

A2 is not buffered and therefore its update is immediate. If the channel is running when a change is made this could cause either the loss of one trigger event or the generation of two trigger events within the same period. Register A2 can be accessed by reading or writing the eMIOS UC Alternate A Register (EMIOSALTA) at UC*n* base address +0x14.

FLAG signal is set only at match on the comparator with A2. A match on the comparator with A1 or B1 or B2 has no effect on FLAG.

At any time, the FORCMA and FORCMB bits allow the software to force the output flip-flop to the level corresponding to a match on A or B respectively. Any FORCMA and/or FORCMB has priority over any simultaneous match regarding to output pin transitions. Note that the load of B2 content on B1 register at an A match is not inhibited due to a simultaneous FORCMA/FORCMB assertion. If both FORCMA and FORCMB are asserted simultaneously the output pin goes to the opposite of EDPOL value such as if A1 and B1 registers had the same value. FORCMA assertion causes the transfer from register B2 to B1 such as a regular A match, regardless of FORCMB assertion.

If subsequent matches occur on comparators A1 and B, the PWM pulses continue to be generated, regardless of the state of the FLAG bit.

At OPWMT mode entry the output flip-flop is set to the complement of the EDPOL bit in the EMIOSC*n* register.

In order to achieve 0% duty cycle both registers A1 and B must be set to the same value. When a simultaneous match on comparators A and B occur, the output flip-flop is set at every period to the complement value of EDPOL.

In order to achieve 100% duty cycle the register B1 must be set to a value greater than maximum value of the selected time base. As a consequence, if 100% duty cycle must be implemented, the maximum counter value for the time base is 0xFFFE for a 16-bit counter. When a match on comparator A1 occurs the output flip-flop is set at every period to the value of EDPOL bit. The transfer from register B2 to B1 is still triggered by the match at comparator A.

Figure 697 shows the Unified Channel running in OPWMT mode with Trigger Event Generation and duty cycle update on next period update.

Figure 697. OPWMT example

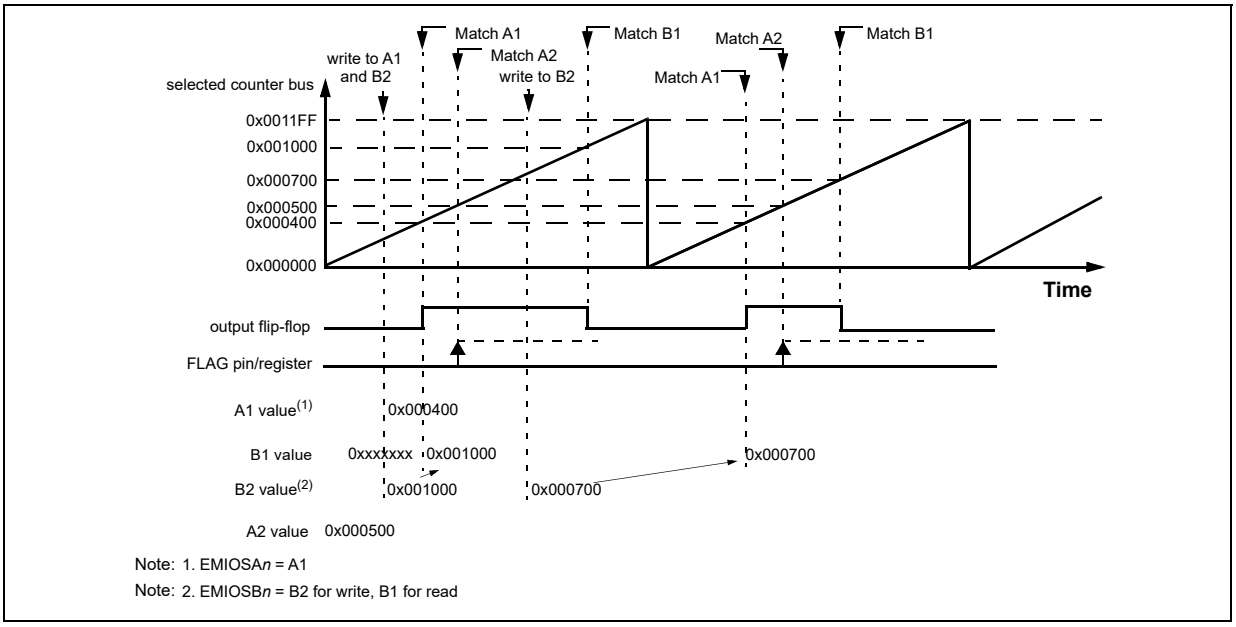


Figure 698 shows the Unified Channel running in OPWMT mode with Trigger Event Generation and 0% duty.

Figure 698. OPWMT with 0% Duty Cycle

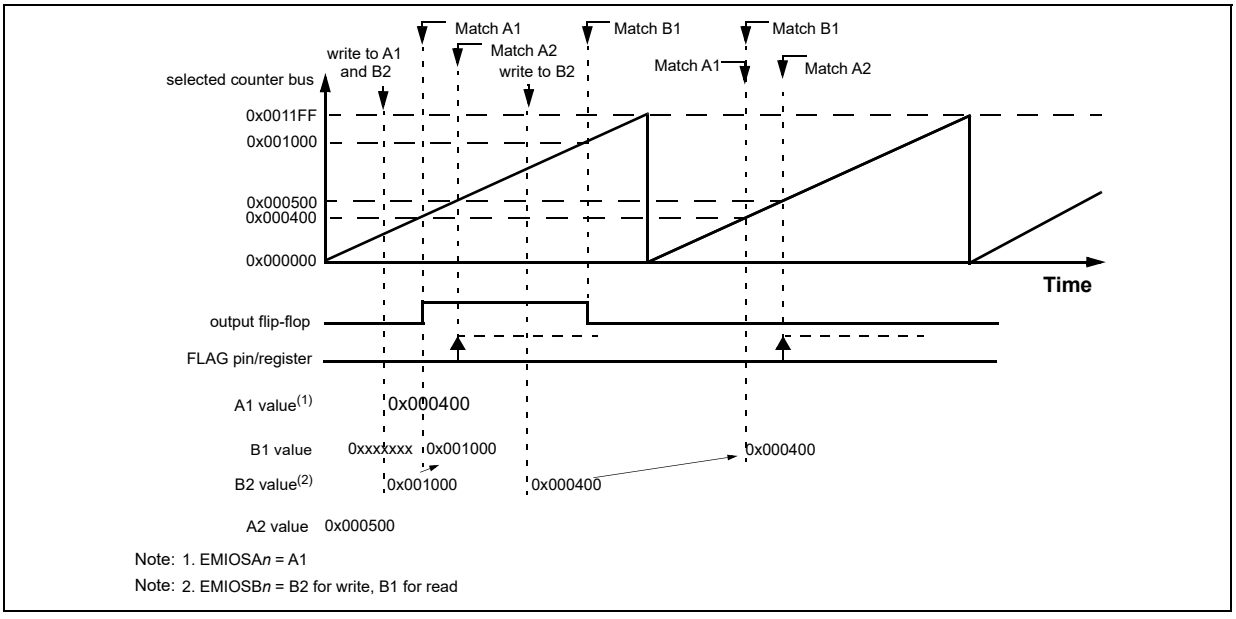
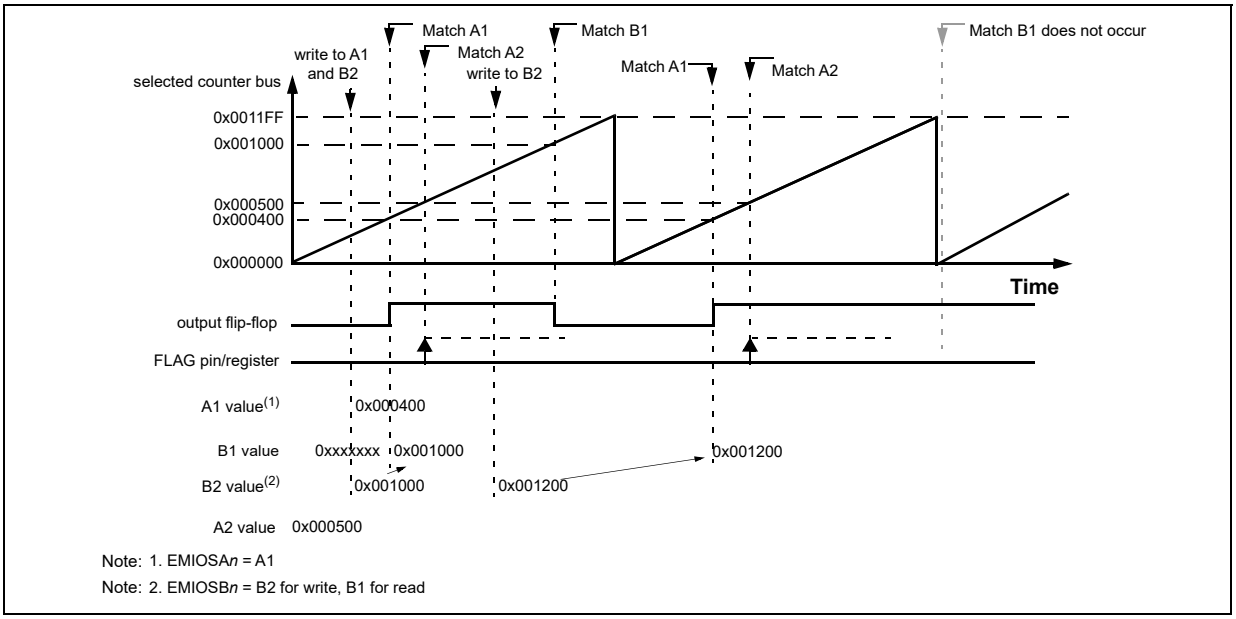


Figure 699 shows the Unified Channel running in OPWMT mode with Trigger Event Generation and 100% duty cycle.

Figure 699. OPWMT with 100% duty cycle

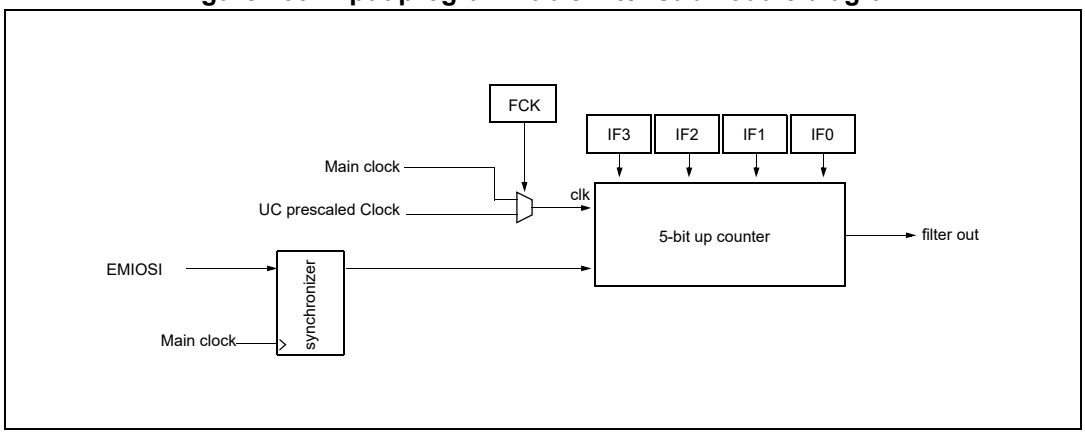


51.4.1.2 Input Programmable Filter (IPF)

The IPF ensures that only valid input pin transitions are received by the Unified Channel edge detector. A block diagram of the IPF is shown in *Figure 700*.

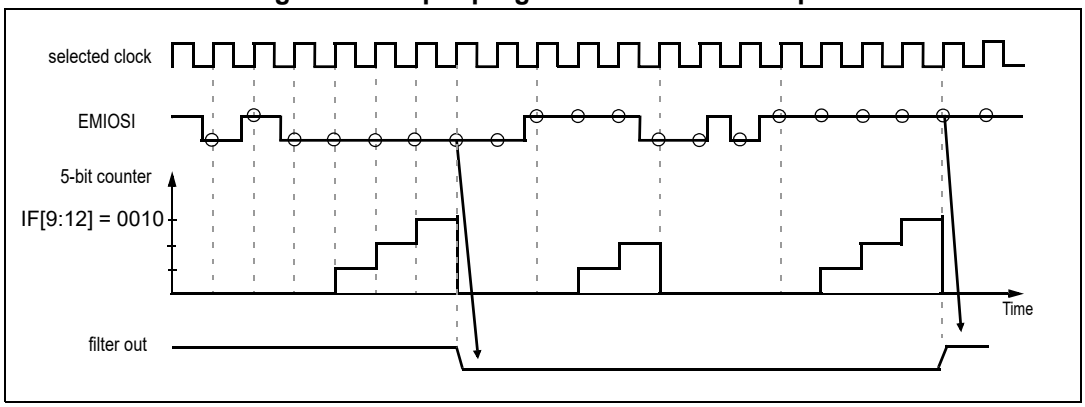
The IPF is a 5-bit programmable up counter that is incremented by the selected clock source, according to bits IF[9:12] in EMIOSC*n* register.

Figure 700. input programmable filter submodule diagram



The input signal is synchronized by the main clock. When a state change occurs in this signal, the 5-bit counter starts counting up. As long as the new state is stable on the pin, the counter remains incrementing. If a counter overflows occurs, the new pin value is validated. In this case, it is transmitted as a pulse edge to the edge detector. If the opposite edge appears on the pin before validation (overflow), the counter is reset. At the next pin transition, the counter starts counting again. Any pulse that is shorter than a full range of the masked counter is regarded as a glitch and it is not passed on to the edge detector. A timing diagram of the input filter is shown in *Figure 701*.

Figure 701. Input programmable filter example



The filter is not disabled during either freeze state or negated GTBE input.

51.4.1.3 Clock Prescaler (CP)

The CP divides the GCP output signal to generate a clock enable for the internal counter of the Unified Channels. The GCP output signal is prescaled by the value defined in *Figure 808* according to the UCPRE[4:5] bits in EMIOSC*n* register. The prescaler is enabled by setting the UCPREN bit in the EMIOSC*n* and can be stopped at any time by clearing this bit, thereby stopping the internal counter in the Unified Channel.

In order to ensure safe working and avoid glitches the following steps must be performed whenever any update in the prescaling rate is desired:

1. Write 0 at both GPREN bit in EMIOSMCR register and UCPREN bit in EMIOSC*n* register, thus disabling prescalers;
2. Write the desired value for prescaling rate at UCPRE[4:5] bits in EMIOSC*n* register;
3. Enable channel prescaler by writing 1 at UCPREN bit in EMIOSC*n* register;
4. Enable global prescaler by writing 1 at GPREN bit in EMIOSMCR register.

The prescaler is not disabled during either freeze state or negated GTBE input.

51.4.1.4 Effect of Freeze on the Unified Channel

When in debug mode, bit FRZ in the EMIOSMCR and bit FREN in the EMIOSC*n* register are both set, the internal counter and Unified Channel capture and compare functions are halted. The UC is frozen in its current state.

During freeze, all registers are accessible. When the Unified Channel is operating in an output mode, the force match functions remain available, allowing the software to force the output to the desired level.

Note that for input modes, any input events that may occur while the channel is frozen are ignored.

When exiting debug mode or freeze enable bit is cleared (FRZ in the EMIOSMCR or FREN in the EMIOSC*n* register) the channel actions resume, but may be inconsistent until channel enters GPIO mode again.

51.4.2 IP Bus Interface Unit (BIU)

The BIU provides the interface between the Internal Interface Bus (IIB) and the Peripheral Bus, allowing communication among all submodules and this IP interface.

The BIU allows 8, 16 and 32-bit access. They are performed over a 32-bit data bus in a single cycle clock.

51.4.2.1 Effect of Freeze on the BIU

When the FRZ bit in the EMIOSMCR is set and the module is in debug mode, the operation of BIU is not affected.

51.4.3 Global Clock Prescaler Submodule (GCP)

The GCP divides the system clock to generate a clock for the CPs of the channels. The main clock signal is prescaled by the value defined in *Figure 798* according to bits GPRE[16:23] in the EMIOSMCR. The global prescaler is enabled by setting the GPREN bit in the EMIOSMCR and can be stopped at any time by clearing this bit, thereby stopping the internal counters in all the channels.

In order to ensure safe working and avoid glitches the following steps must be performed whenever any update in the prescaling rate is desired:

1. Write ‘0’ at GPREN bit in EMIOSMCR, thus disabling global prescaler;
2. Write the desired value for prescaling rate at GPRE[16:23] bits in EMIOSMCR;
3. Enable global prescaler by writing ‘1’ at GPREN bit in EMIOSMCR.

The prescaler is not disabled during either freeze state or negated GTBE input.

51.4.3.1 Effect of Freeze on the GCP

When the FRZ bit in the EMIOSMCR is set and the module is in debug mode, the operation of GCP submodule is not affected, that is, there is no freeze function in this submodule.

51.5 Initialization/Application information

On resetting the eMIOS the Unified Channels enter GPIO input mode.

51.5.1 Considerations

Before changing an operating mode, the UC must be programmed to GPIO mode and EMIOSA*n* and EMIOSB*n* registers must be updated with the correct values for the next operating mode. Then the EMIOSC*n* register can be written with the new operating mode. If a UC is changed from one mode to another without performing this procedure, the first operation cycle of the selected time base can be random, that is, matches can occur in random time if the contents of EMIOSA*n* or EMIOSB*n* were not updated with the correct value before the time base matches the previous contents of EMIOSA*n* or EMIOSB*n*.

When interrupts are enabled, the software must clear the FLAG bits before exiting the interrupt service routine.

51.5.2 Application information

Correlated output signals can be generated by all output operation modes. Bits OU*n* of the EMIOSOUDIS register can be used to control the update of these output signals.

In order to guarantee that the internal counters of correlated channels are incremented in the same clock cycle, the internal prescalers must be set up before enabling the global

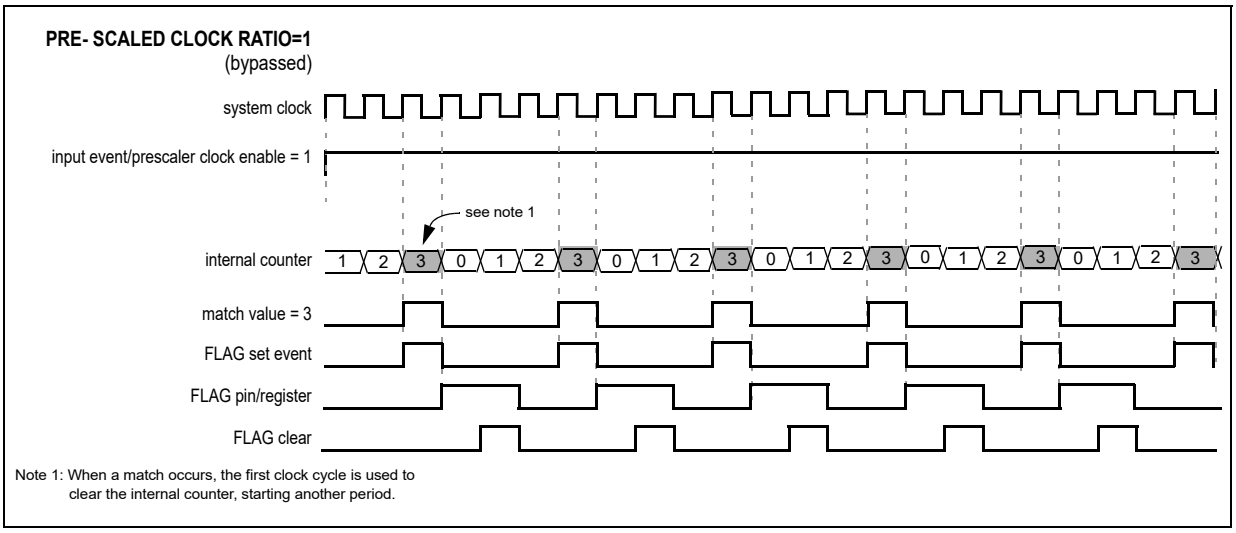
prescaler. If the internal prescalers are set after enabling the global prescaler, the internal counters may increment in the same ratio, but at a different clock cycle.

51.5.2.1 Time base generation

For MC with internal clock source operation modes, the internal counter rate can be modified by configuring the clock prescaler ratio. *Figure 702* shows an example of a time base with prescaler ratio equal to one.

Note: MCB and OPWFMB modes have a different behavior.

Figure 702. Time base period when running in the fastest prescaler ratio

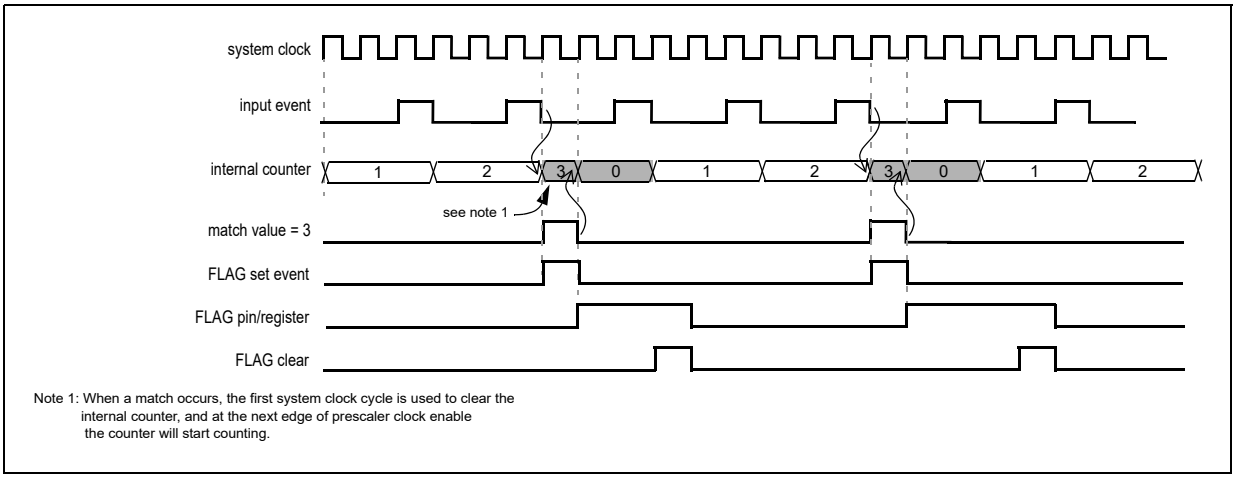


If the prescaler ratio is greater than one or external clock is selected, the counter may behave in three different ways depending on the channel mode:

- If MC mode and Clear on Match Start and External Clock source are selected the internal counter behaves as described in *Figure 703*.
- If MC mode and Clear on Match Start and Internal Clock source are selected the internal counter behaves as described in *Figure 704*.
- If MC mode and Clear on Match End are selected the internal counter behaves as described in *Figure 705*.

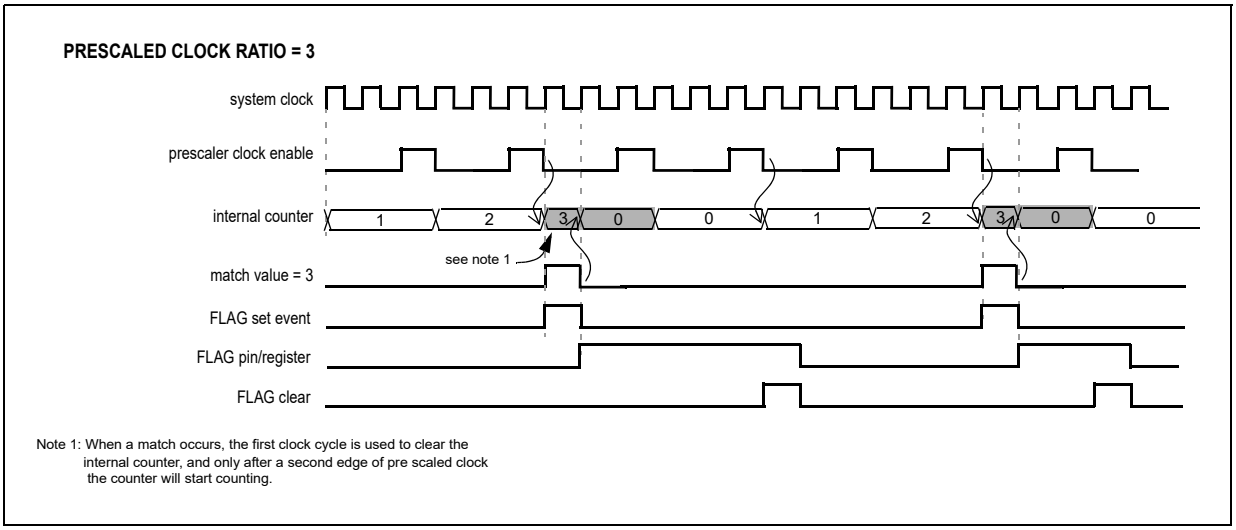
Note: MCB and OPWFMB modes have a different behavior.

Figure 703. Time base generation with external clock and clear on match start



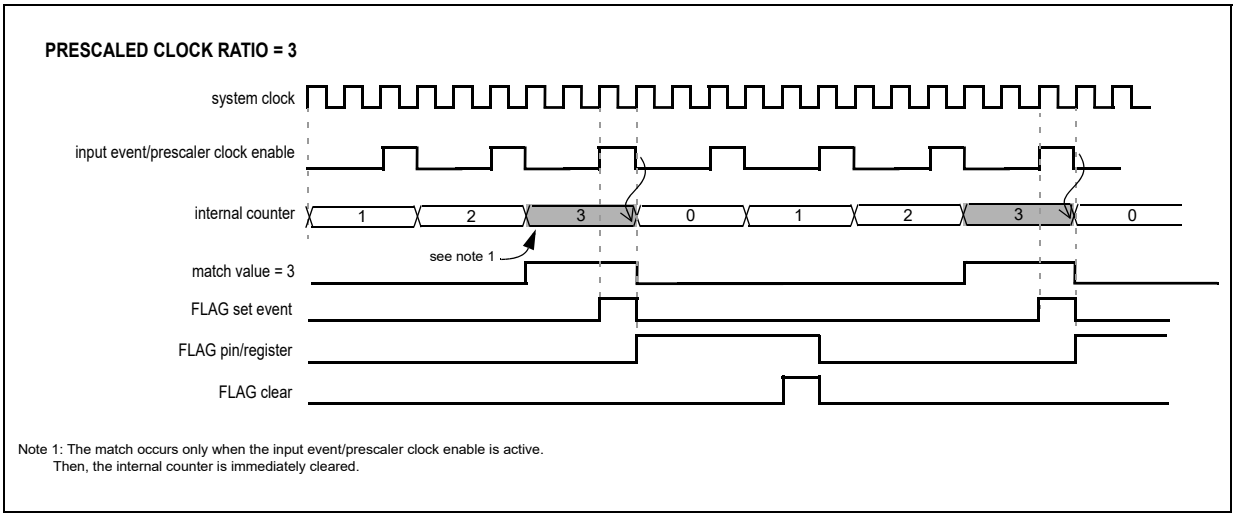
Note 1: When a match occurs, the first system clock cycle is used to clear the internal counter, and at the next edge of prescaler clock enable the counter will start counting.

Figure 704. Time base generation with internal clock and clear on match start



Note 1: When a match occurs, the first clock cycle is used to clear the internal counter, and only after a second edge of pre scaled clock the counter will start counting.

Figure 705. Time base generation with clear on match end



Note 1: The match occurs only when the input event/prescaler clock enable is active. Then, the internal counter is immediately cleared.

51.5.2.2 Coherent accesses

It is highly recommended that the software waits for a new FLAG set event before starting to read the EMIOSA*n* and EMIOSB*n* registers to get a new measurement. The FLAG indicates that new data has been captured and it is the only way to assure data coherency.

The FLAG set event can be detected by polling the FLAG bit or by enabling the interrupt request or CTU trigger generation.

Reading the EMIOSA*n* register again in the same period of the last read of EMIOSB*n* register may lead to incoherent results. This will occur if the last read of EMIOSB*n* register occurred after a disabled B2 to B1 transfer.

51.5.2.3 Channel/Modes initialization

The following basic steps summarize basic output mode startup, assuming the channels are initially in GPIO mode:

1. *[global]* Disable Global Prescaler.
2. *[timebase channel]* Disable Channel Prescaler.
3. *[timebase channel]* Write initial value at internal counter.
4. *[timebase channel]* Set A/B register.
5. *[timebase channel]* Set channel to MC(B) Up mode.
6. *[timebase channel]* Set prescaler ratio.
7. *[timebase channel]* Enable Channel Prescaler.
8. *[output channel]* Disable Channel Prescaler.
9. *[output channel]* Set A/B register.
10. *[output channel]* Select timebase input through bits BSL[21:22].
11. *[output channel]* Enter output mode.
12. *[output channel]* Set prescaler ratio (same ratio as timebase channel).
13. *[output channel]* Enable Channel Prescaler.
14. *[global]* Enable Global Prescaler.
15. *[global]* Enable Global Time Base.

The timebase channel and the output channel may be the same for some applications such as in OPWFM(B) mode or whenever the output channel is intended to run the timebase itself.

The flags can be configured at any time.

52 CAN Subsystem

52.1 Introduction

SPC58xEx/SPC58xGx has two CAN subsystems implemented:

- CAN Subsystem 0
- CAN Subsystem 1

Note: Refer to Section 7.8.2: CAN subsystem configuration in Chapter 7: Device configuration to see the subsystems implementation.

The Controller Area Network (CAN) subsystem consists of the modular CAN (M_CAN) modules and, for at least one subsystem, Time triggered CAN (M_TTCAN) modules along with an integrated intelligent CAN RAM controller. The CAN RAM controller consists of additional logic for arbitration between the requests for the RAM access by the various CANs and CPU, ECC encoder/decoder for the Message RAM data and active transmit message buffer protection and M_TTCAN trigger memory from CPU write access. The subsystem follows the little endian format.

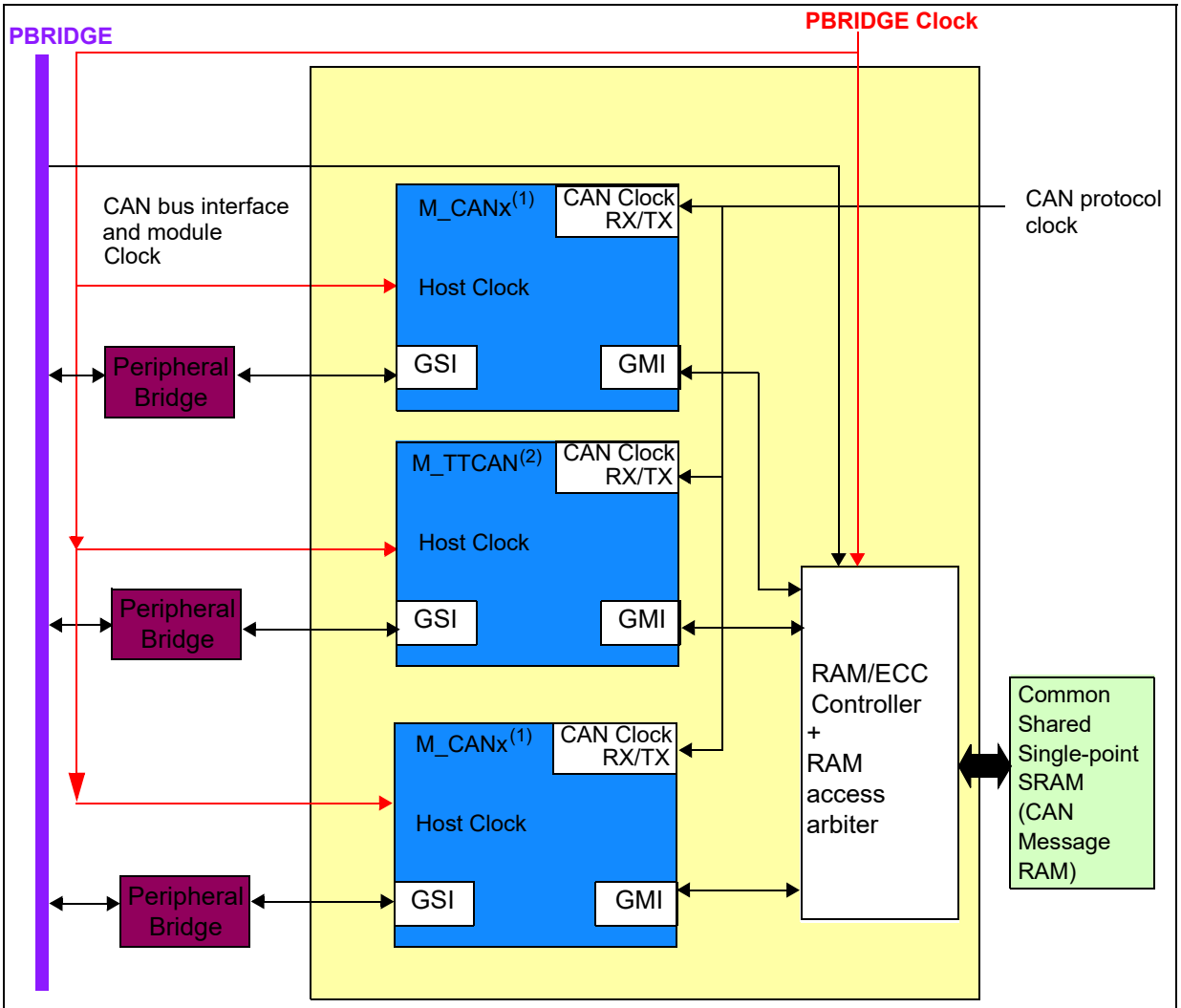
Note: Refer to Section 7.8.2: CAN subsystem configuration in Chapter 7: Device configuration to see the number of CAN nodes used in the device.

Table 814. IP user manual versions

IP Name	Version
M_CAN IP	Revision 3.2.1
M_TTCAN IP	Revision 3.2.1

The general CAN subsystem implementation is shown in Section 52.2. It interfaces with the Host processor bus using the peripheral bus. The RAM is not implemented inside the subsystem.

Figure 706. CAN subsystem generic block diagram



1: Number of M_CAN nodes varies per device. x = 1 to n. Please refer to Device Configuration chapter to see the number of M_CAN nodes used in the device.
2: Please refer to Device Configuration chapter to see the number of M_TTCAN nodes used in the device.

52.2 Features

The CAN subsystem consists of the following major blocks:

- Modular CAN cores: The registers of the CAN module can be accessed using the Generic Slave Interface (GSI)
- Time triggered CAN core. The registers of the CAN module can be accessed using the Generic Slave Interface (GSI)
- CAN-RAM arbiter
- SRAM interface and memory organization
- ECC Controller

52.3 Modular CAN (M_CAN) cores

M_CAN functionality conforms to CAN specification V2.0B active for each M_CAN node.The M_CAN performs communication according to the CAN protocol specification 2.0 part A,B and to CAN FD 1.0.Flexible assignment of Message Objects to nodes.
The bit rate can be programmed to values up to 1 Mbit/s for standard CAN frames. High bit rates are possible in CAN FD mode. Additional transceiver hardware is required for connection to the physical layer.

Note: Implemented CAN nodes are fully compliant to ISO CAN FD specification ISO 11898-1: 2015.

The message storage is intended to be a single-ported Message RAM outside of the module. It is connected to the M_CAN via the Generic Master Interface. Depending on the chosen device, multiple M_CAN controllers can share the same Message RAM.

All functions concerning the handling of messages are implemented by the Rx Handler and the Tx Handler. The Rx Handler manages message acceptance filtering, the transfer of received messages from the CAN core to the Message RAM as well as providing receive message status information. The Tx Handler is responsible for the transfer of transmit messages from the Message RAM to the CAN core as well as providing transmit status information.

Acceptance filtering is implemented by a combination of up to 128 filter elements where each one can be configured as a range, as a bit mask, or as a dedicated ID filter.

The M_CAN’s clock domain concept allows the separation between the high precision CAN clock and the Host clock, which may be generated by an FMPLL.

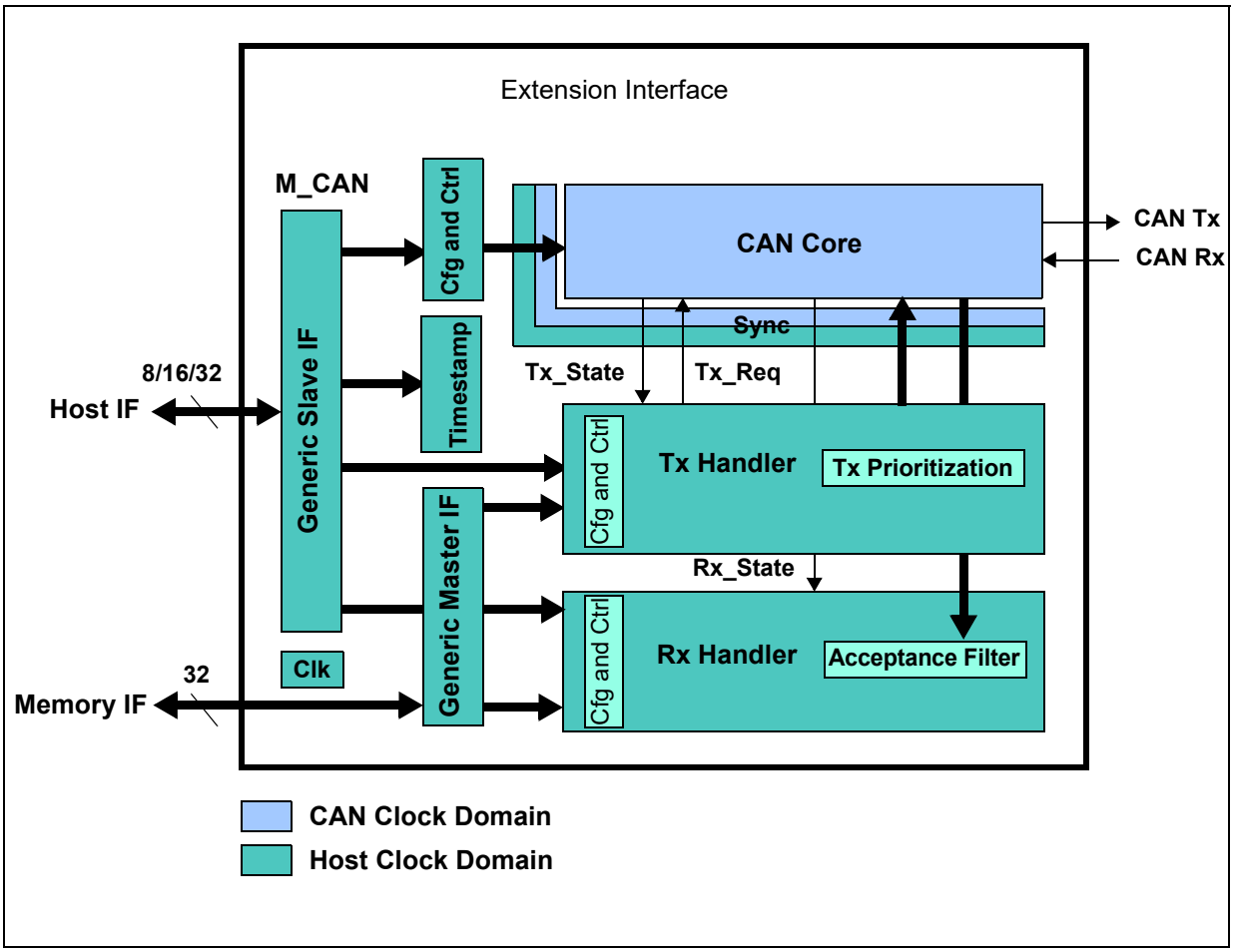
52.3.1 Features

The following are the features of Modular CAN cores.

- Conforms with CAN protocol version 2.0 part A, B and ISO 11898-1: 2015
- CAN Flexible data-rate (ISO CAN FD) protocol with up to 64 data bytes on M_TTCAN and 64 data bytes on M_CAN is supported
- Bit rates up to 1 Mbit/s in standard CAN mode
- Bit rates up to 8 Mbit/s in ISO CAN FD mode
- CAN error logging
- AUTOSAR optimized
- SAE J1939 optimized
- Improved acceptance filtering
- Two configurable Receive FIFOs
- Separate signaling on reception of High Priority Messages
- Up to 64 dedicated Receive Buffers
- Up to 32 dedicated Transmit Buffers
- Configurable Transmit FIFO
- Configurable Transmit Queue
- Configurable Transmit Event FIFO
- Direct Message RAM access for Host CPU
- Multiple M_CANs share the same Message RAM
- Programmable loop-back test mode
- Maskable module interrupts
- 8-/16-/32-bits Generic Slave Interface for connection customer-specific Host CPUs
- Two clock domains (CAN clock and Host clock)
- Power-down support
- Debug over M_CAN
- Clock calibration over M_CAN is supported

52.3.2 Block diagram

Figure 707. M_CAN core block diagram



- CAN core: CAN Protocol Controller and Rx/Tx Shift Register. Handles all ISO 11898-1 protocol functions. Supports 11-bit and 29-bit identifiers.
- Sync: Synchronizes signals from the Host clock domain to the CAN clock domain and vice versa.
- Clk: Synchronizes reset signal to the Host clock domain and to the CAN clock domain.
- Cfg and Ctrl: CAN core related configuration and control bits.
- Interrupt and Timestamp: Interrupt control and 16-bit CAN bit time counter for receive and transmit timestamp generation.
- Tx Handler: Controls the message transfer from the external Message RAM to the CAN core. A maximum of 32 Tx Buffers can be configured for transmission. Tx buffers can be used as dedicated Tx Buffers, as Tx FIFO, part of a Tx Queue, or as a combination of them. A Tx Event FIFO stores Tx timestamps together with the corresponding Message ID. Transmit cancellation is also supported.
- Rx Handler: Controls the transfer of received messages from the CAN core to the external Message RAM. The Rx Handler supports two Receive FIFOs, each of configurable size, and up to 64 dedicated Rx Buffers for storage of all messages that have passed acceptance filtering. A dedicated Rx Buffer, in contrast to a Receive FIFO, is used to store only messages with a specific identifier. An Rx timestamp is stored

together with each message. Up to 128 filters can be defined for 11-bit IDs and up to 64 filters for 29-bit IDs.

- Generic Slave Interface: Connects the M_CAN to a customer specific Host CPU. The Generic Slave Interface is capable of connecting to an 8-/16-/32-bit bus to support a wide range of interconnection structures.
- Generic Master Interface: Connects the M_CAN access to an external 32-bit Message RAM. The maximum Message RAM size is 16 KB $\times$ 32 bits.
- Extension Interface: All flags from the Interrupt Register IR as well as selected internal status and control signals are routed to this interface. The interface is intended for connection of the M_CAN to a module-external interrupt unit or to other module-external components. The connection of these signals is optional.

52.3.3 Dual clock sources

To improve the EMC behavior, a spread spectrum clock can be used for the Host clock domain. Due to the high precision clocking requirements of the CAN core, a separate clock without any modulation has to be provided as CAN clock.

Within the M_CAN module there is a synchronization mechanism implemented to ensure save data transfer between the two clock domains.

Note: In order to achieve a stable function of the M_CAN, the Host clock must always be faster than or equal to the CAN clock. Also the modulation depth of the spread spectrum clock has to be regarded.

Caution: Stop this module when the device is in STOP mode. If CAN message reception is mandatory, check that the PLL is unlocked before transmitting any message. Use the external oscillator (XOSC) instead of the internal RC oscillator (IRCOSC) during the STOP mode.

52.3.4 Dual interrupt lines

The module provides two interrupt lines. Interrupts can be routed either to EINT0 or to EINT1. By default all interrupts are routed to interrupt line EINT0. By programming EINT0 and EINT1 bits of the Interrupt Line Enable (ILE) register, the interrupt lines can be enabled or disabled separately.

52.3.5 Memory map and register descriptions

52.3.5.1 Hardware reset description

After hardware reset, the registers of the M_CAN hold the reset values listed in *Table 815*. Additionally the *Bus_Off* state is reset and the M_CAN Tx output is set to *recessive* (HIGH). The value 0x0001 (bit 0 [INIT] of the CCCR register = ‘1’) in the CC Control Register enables software initialization. The M_CAN does not influence the CAN bus until the CPU resets bit 0 [INIT] of the CCCR register = ‘0’.

52.3.5.2 Register map

The M_CAN module allocates an address space of 256 bytes. All registers are organized as 32-bit registers. The M_CAN is accessible by the Host CPU via the Generic Slave Interface using a data width of 8-bit (byte access), 16-bit (half-word access), or 32-bit (word access).

Note: Write access by the Host CPU to registers/bits marked with "P=Protected Write" is possible only when the bit 1 [CCE] and bit 0 [INIT] of the CCCR register is set to '1'.

There is a delay from writing to a command register until the update of the related status register bits due to clock domain crossing.

Caution: Any write access to reserved or not implemented registers in the 16 KB slot assigned by the peripherals bridge to the M_CAN IP will not generate any bus access error.

Table 815. M_CAN memory map

Address Offset	Register Name	Access(1)	Reset value	Location
0x0000	Core Release Register (CREL)	R	0x3215_0320	*Section 52.3.5.2.1*
0x0004	Endian Register (ENDN)	R	0x8765_4321	*Section 52.3.5.2.2*
0x0008–0x000B	Reserved			
0x000C	Data Bit Timing and Prescaler Register (DBTP)	RP	0x0000_0A33	*Section 52.3.5.2.3*
0x0010	Test Register (TEST)	RP	0x0000_0000	*Section 52.3.5.2.4*
0x0014	RAM Watchdog Register (RWD)	RP	0x0000_0000	*Section 52.3.5.2.5*
0x0018	CC Control Register (CCCR)	RWPp	0x0000_0001	*Section 52.3.5.2.6*
0x001C	Nominal Bit Timing and Prescaler Register (NBTP)	RP	0x0600_0A03	*Section 52.3.5.2.7*
0x0020	Timestamp Counter Configuration Register (TSCC)	RP	0x0000_0000	*Section 52.3.5.2.8*
0x0024	Timestamp Counter Value Register (TSCV)	RC	0x0000_0000	*Section 52.3.5.2.9*
0x0028	Timeout Counter Configuration Register (TOCC)	RP	0xFFFF_0000	*Section 52.3.5.2.10*
0x002C	Timeout Counter Value Register (TOCV)	RC	0x0000_FFFF	*Section 52.3.5.2.11*
0x0030–0x003F	Reserved			
0x0040	Error Counter Register (ECR)	RX	0x0000_0000	*Section 52.3.5.2.12*
0x0044	Protocol Status Register (PSR)	RXS	0x0000_0707	*Section 52.3.5.2.13*
0x0048	Transmitter Delay Compensation Register (TDCR)	RP	0x0000_0000	*Section 52.3.5.2.14*
0x004C–0x004F	Reserved			
0x0050	Interrupt Register (IR)	R/W	0x0000_0000	*Section 52.3.5.2.15*
0x0054	Interrupt Enable Register (IE)	R/W	0x0000_0000	*Section 52.3.5.2.16*
0x0058	Interrupt Line Select Register (ILS)	R/W	0x0000_0000	*Section 52.3.5.2.17*
0x005C	Interrupt Line Enable Register (ILE)	R/W	0x0000_0000	*Section 52.3.5.2.18*
0x0060–0x007F	Reserved			
0x0080	Global Filter Configuration Register (GFC)	RP	0x0000_0000	*Section 52.3.5.2.19*
0x0084	Standard ID Filter Configuration Register (SIDFC)	RP	0x0000_0000	*Section 52.3.5.2.20*

Table 815. M_CAN memory map (continued)

Address Offset	Register Name	Access(1)	Reset value	Location
0x0088	Extended ID Filter Configuration Register (XIDFC)	RP	0x0000_0000	Section 52.3.5.2.21
0x008C–0x008F	Reserved			
0x0090	Extended ID AND Mask Register (XIDAM)	RP	0x1FFF_FFFF	Section 52.3.5.2.22
0x0094	High Priority Message Status Register (HPMS)	R	0x0000_0000	Section 52.3.5.2.23
0x0098	New Data 1 Register (NDAT1)	R/W	0x0000_0000	Section 52.3.5.2.24
0x009C	New Data 2 Register (NDAT2)	R/W	0x0000_0000	Section 52.3.5.2.25
0x00A0	Rx FIFO 0 Configuration Register (RXF0C)	RP	0x0000_0000	Section 52.3.5.2.26
0x00A4	Rx FIFO 0 Status Register (RXF0S)	R	0x0000_0000	Section 52.3.5.2.27
0x00A8	Rx FIFO 0 Acknowledge Register (RXF0A)	R/W	0x0000_0000	Section 52.3.5.2.28
0x00AC	Rx Buffer Configuration Register (RXBC)	RP	0x0000_0000	Section 52.3.5.2.29
0x00B0	Rx FIFO 1 Configuration Register (RXF1C)	RP	0x0000_0000	Section 52.3.5.2.30
0x00B4	Rx FIFO 1 Status Register (RXF1S)	R	0x0000_0000	Section 52.3.5.2.31
0x00B8	Rx FIFO 1 Acknowledge Register (RXF1A)	R/W	0x0000_0000	Section 52.3.5.2.32
0x00BC	Rx Buffer / FIFO Element Size Configuration Register (RXESC)	RP	0x0000_0000	Section 52.3.5.2.33
0x00C0	Tx Buffer Configuration Register (TXBC)	RP	0x0000_0000	Section 52.3.5.2.34
0x00C4	Tx FIFO/Queue Status Register (TXFQS)	R	0x0000_0000	Section 52.3.5.2.35
0x00C8	Tx Buffer Element Size Configuration Register (TXESC)	RP	0x0000_0000	Section 52.3.5.2.36
0x00CC	Tx Buffer Request Pending Register (TXBRP)	R	0x0000_0000	Section 52.3.5.2.37
0x00D0	Tx Buffer Add Request Register (TXBAR)	R/W	0x0000_0000	Section 52.3.5.2.38
0x00D4	Tx Buffer Cancellation Register (TXBCR)	R/W	0x0000_0000	Section 52.3.5.2.39
0x00D8	Tx Buffer Transmission Occurred Register (TXBTO)	R	0x0000_0000	Section 52.3.5.2.40
0x00DC	Tx Buffer Cancellation Finished Register (TXBCF)	R	0x0000_0000	Section 52.3.5.2.41
0x00E0	Tx Buffer Transmission Interrupt Enable (TXBTIE)	R/W	0x0000_0000	Section 52.3.5.2.42
0x00E4	Tx Buffer Cancellation Finished Interrupt Enable Register (TXBCIE)	R/W	0x0000_0000	Section 52.3.5.2.43
0x00E8–0x00EF	Reserved			
0x00F0	Tx Event FIFO Configuration Register (TXEFC)	RP	0x0000_0000	Section 52.3.5.2.44
0x00F4	Tx Event FIFO Status Register (TXEFS)	R	0x0000_0000	Section 52.3.5.2.45

Table 815. M_CAN memory map (continued)

Address Offset	Register Name	Access(1)	Reset value	Location
0x00F8	Tx Event FIFO Acknowledge Register (TXEFA)	R/W	0x0000_0000	*Section 52.3.5.2.46*
0x00FC–0x00FF	Reserved			

1. R = Read, S = Set on read, W = Write, P = Protected Write, C = Clear/preset on write, p = Protected set, X= Reset on read

52.3.5.2.1 Core Release Register (CREL)

Offset: 0x0000 Access: Read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	REL				STEP				SUBSTEP				YEAR			
W																
Reset(1)	-r	-r	-r	-r	-r	-r	-r	-r	-r	-r	-r	-r	-d	-d	-d	-d

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	MON								DAY							
W																
Reset(1)	-d	d-	-d	-d	-d	-d	-d	-d	-d	-d	-d	-d	-d	-d	-d	-d

1. -r = release, -d = time stamp. The coding of revisions depends on the module version used in the device. For the device reset value refer to *Table 815: M_CAN memory map*.

Figure 708. Core Release Register (CREL)

Table 816. CREL field descriptions

Field	Description
0:3 REL	Core Release One digit, BCD.
4:7 STEP	Step of Core Release One digit, BCD.
8:11 SUBSTEP	Substep of Core Release One digit, BCD.
12:15 YEAR	Time Stamp Year One digit, BCD.
16:23 MON	Time Stamp Month Two digits, BCD.
24:31 DAY	Time Stamp Day Two digits, BCD.

52.3.5.2.2 Endian Register (ENDN)

Offset: 0x0004 Access: Read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	ETV															
W																
Reset	1	0	0	0	0	1	1	1	0	1	1	0	0	1	0	1

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	ETV															
W																
Reset	0	1	0	0	0	0	1	1	0	0	1	0	0	0	0	1

Figure 709. Endian Register (ENDN)

Table 817. ENDN field descriptions

Field	Description
0:31 ETV	Endianness Test Value The endianness test value is 0x87654321.

52.3.5.2.3 Data Bit Timing and Prescaler Register (DBTP)

This register is only writable if bits **CCCR.CCE** and **CCCR.INIT** are set. The CAN bit time may be programed in the range of 4 to 49 time quanta. The CAN time quantum may be programmed in the range of 1 to 32 M_CAN clock periods. tq = (FBRP + 1) M_CAN clock period.

FTSEG1 is the sum of Prop_Seg and Phase_Seg1. **FTSEG2** is Phase_Seg2.

Therefore the length of the bit time is (programmed values) **[FTSEG1 + FTSEG2 + 3]** tq or (functional values) [Sync_Seg + Prop_Seg + Phase_Seg1 + Phase_Seg2] tq.

The Information Processing Time (IPT) is zero, meaning that the data for the next bit is available at the first clock edge after the sample point.

Offset: 0x000C Access: Read Protected

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	TDC(1)	0	0	DBRP(1)				
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	DTSEG1(1)					DTSEG2(1)				DSJW(1)			
W																
Reset	0	0	0	0	1	0	1	0	0	0	1	1	0	0	1	1

1. These are protected write (P) bits which means that write access by the bits is possible only when the bit 1 [CCE] and bit 0 [INIT] of CCCR register are set to ‘1’.

Figure 710. Data Bit Timing and Prescaler Register (DBTP)

Table 818. DBTP field descriptions

Field	Description
8 TDC	Transceiver Delay Compensation 0 Transceiver Delay Compensation disabled 1 Transceiver Delay Compensation enabled
11:15 DBRP	Data Baud Rate Prescaler (0x00–0x1F) The value by which the oscillator frequency is divided for generating the bit time quanta. The bit time is built up from a multiple of this quanta. Valid values for the Baud Rate Prescaler are 0 to 31. The actual interpretation by the hardware of this value is such that one more than the value programmed here is used.
19:23 DTSEG1	Data time segment before sample point (0x0–0x1F) Valid values are 0 to 31. The actual interpretation by the hardware of this value is such that one more than the programmed value is used.
24:27 DTSEG2	Data time segment after sample point (0x0–0xF) Valid values are 0 to 15. The actual interpretation by the hardware of this value is such that one more than the programmed value is used.
28:31 DSJW	Data (Re) Synchronization Jump Width (0x0–0xF) Valid values are 0 to 15. The actual interpretation by the hardware of this value is such that one more than the value programmed here is used.

Note: With a CAN clock of 8 MHz, the reset value of 0x00000A33 configures the M_CAN for a fast bit rate of 500 kbit/s.

The bit rate configured for the CAN FD data phase via DBTP must be higher than or equal to the bit rate configured for the arbitration phase via NBTP.

To achieve high bit rates in CAN FD mode for the data phase up to 8 Mbit/s the number of time quanta bits per CAN bit can be chosen down to 4 time quanta per bit time.

52.3.5.2.4 Test Register (TEST)

Write access to the Test Register has to be enabled by setting CCCR[TEST] to ‘1’. All Test Register functions are set to their reset values when CCCR[TEST] is reset. Loop Back mode and software control of M_CAN Tx pin are hardware test modes. Programming of **TX** ≠ ‘00’ may disturb the message transfer on the CAN bus.

Offset: 0x0010 Access: Read-Write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	RX	TX(1)		LBCK(1)	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	—	0	0	0	0	0	0	0

1. These are protected write (P) bits which means that write access by the bits is possible only when the bit 1 [CCE] and bit 0 [INIT] of CCCR register are set to ‘1’.

Figure 711. Test Register (TEST)

Table 819. TEST field descriptions

Field	Description
24 RX	Receive Pin Monitors the actual value of CAN Rx pin. 0 The CAN bus is dominant (CAN Rx input = ‘0’) 1 The CAN bus is recessive (CAN Rx input = ‘1’)
25:26 TX	Control of Transmit Pin 00 Reset value, CAN Tx output is controlled by the CAN core, updated at the end of the CAN bit time. 01 Sample Point can be monitored at pin CAN Tx output 10 Dominant (‘0’) level at pin CAN Tx output 11 Recessive (‘1’) at pin CAN Tx output
27 LBCK	Loop Back mode 0 Reset value, Loop Back mode is disabled 1 Loop Back mode is enabled (refer to *Section 52.3.7.9: Test modes*)

52.3.5.2.5 RAM Watchdog Register (RWD)

The RAM Watchdog monitors the READY output of the Message RAM. A Message RAM access via the M_CAN’s Generic Master Interface starts the Message RAM Watchdog Counter with the value configured by the WDC bits. The counter is reloaded with WDC bits when the Message RAM signals successful completion by activating its READY output. In case there is no response from the Message RAM until the counter has counted down to zero, the counter stops and interrupt flag WDI bit of the IR is set. The RAM Watchdog Counter is clocked by the Host clock.

Offset: 0x0014 Access: Read Protected

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	WDV								WDC(1)							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. These are protected write (P) bits which means that write access by the bits is possible only when the bit 1 [CCE] and bit 0 [INIT] of CCCR register are set to ‘1’.

Figure 712. RAM Watchdog Register (RWD)

Table 820. RWD field descriptions

Field	Description
16:23 WDV	Watchdog Value Actual Message RAM Watchdog Counter Value.
24:31 WDC	Watchdog Configuration Start value of the Message RAM Watchdog Counter. With the reset value of ‘00’ the counter is disabled.

52.3.5.2.6 CC Control Register (CCCR)

The following section shows the CAN Control register.For details about setting and resetting of single bits refer to *Section 52.3.7.1.1: Software initialization*.

Offset: 0x0018 Access: Read Protected

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	NISO(1)	TXP(1)	EFBI(1)	PXHD(1)	0	0	BRSE(1)	FDOE(1)	TEST(2)	DAR(1)	MON(2)	CSR	CSA	ASM(2)	CCE(1)	INIT
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

1. These are protected write (P) bits which means that write access by the bits is possible only when the bit 1 [CCE] and bit 0 [INIT] of CCCR register are set to ‘1’.
2. These are protected set (p) bits.

Figure 713. CC Control register (CCCR)

Table 821. CCCR field descriptions

Field	Description
16 NISO	Non ISO Operation If this bit is set, the M_CAN uses the CAN FD frame format as specified by the Bosch CAN FD Specification V1.0. 0 CAN FD frame format according to ISO11898-1 1 CAN FD frame format according to Bosch CAN FD Specification V1.0
17 TXP	Transmit Pause If this bit is set, the M_CAN pauses for two CAN bit times before starting the next transmission after itself has successfully transmitted a frame. 0 Transmit pause disabled 1 Transmit pause enabled
18 EFBI	Edge Filtering during Bus Integration 0 Edge filtering disabled 1 Two consecutive dominant tq required to detect an edge for hard synchronization
19 PXHD	Protocol Exception Handling Disable 0 Protocol exception handling enabled 1 Protocol exception handling disabled **Note:** When protocol exception handling is disabled, the M_CAN will transmit an error frame when it detects a protocol exception condition.
22 BRSE	Bit Rate Switch Enable 0 Bit rate switching for transmissions disabled 1 Bit rate switching for transmissions enabled **Note:** When CAN FD operation is disabled FDOE = ‘0’, BRSE is not evaluated.
23 FDOE	FD Operation Enable 0 FD operation disabled 1 FD operation enabled
24 TEST	Test Mode Enable 0 Normal operation, register TEST holds reset values 1 Test Mode, write access to register TEST enabled
25 DAR	Disable Automatic Retransmission 0 Automatic retransmission of messages not transmitted successfully enabled 1 Automatic retransmission disabled
26 MON	Bus Monitoring Mode Bit MON can only be set by the Host when both CCE and INIT are set to ‘1’. The bit can be reset by the Host at any time. 0 Bus Monitoring Mode is disabled 1 Bus Monitoring Mode is enabled
27 CSR	Clock Stop Request 0 No clock stop is requested 1 Clock stop requested. When clock stop is requested, first INIT and then CSA will be set after all pending transfer requests have been completed and the CAN bus reached idle.
28 CSA	Clock Stop Acknowledge 0 No clock stop acknowledged 1 M_CAN may be set in power down by stopping Host clock and core clock

Table 821. CCCR field descriptions (continued)

Field	Description
29 ASM	Restricted Operation Mode Bit ASM is only set by the Host when both CCE and INIT are set to '1'. The bit is reset by the Host at any time. 0 Normal CAN operation 1 Restricted Operation Mode active
30 CCE	Configuration Change Enable 0 The CPU has no write access to the protected configuration registers 1 The CPU has write access to the protected configuration registers (while CCCR.INIT = '1')
31 INIT	Initialization 0 Normal Operation 1 Initialization is started **Note:** Due to the synchronization mechanism between the two clock domains, there may be a delay until the value written to INIT can be read back. Therefore the programmer has to ensure that the previous value written to INIT has been accepted by reading INIT before setting INIT to a new value.

52.3.5.2.7 Nominal Bit Timing and Prescaler Register (NBTP)

This register is only writable if bits CCCR[CCE] and CCCR[INIT] are set. The CAN bit time may be programmed in the range of [4…385] time quanta. The CAN time quantum may be programmed in the range of [1…512] M_CAN clock periods.

Equation 30 tq = (NBRP + 1) M_CAN clock period

NTSEG1 is the sum of Prop_Seg and Phase_Seg1. NTSEG2 is Phase_Seg2.

Therefore the length of the bit time is (prog rammed values) [NTSEG1 + NTSEG2 + 3] tq or (functional values) [Sync_Seg + Prop_Seg + Phase_Seg1 + Phase_Seg2] tq.

The Information Processing Time (IPT) is zero, meaning the data for the next bit is available at the first clock edge after the sample point.

Offset: 0x001C Access: Read Protected

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	NSJW(1)								NBRP(1)							
Reset	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	NTSEG1(1)								0	NTSEG2(1)						
Reset	0	0	0	0	1	0	1	0	0	0	0	0	0	0	1	1

1. These are protected write (P) bits which means that write access by the bits is possible only when the bit 1 [CCE] and bit 0 [INIT] of CCCR register are set to '1'.

Figure 714. Nominal Bit Timing and Prescaler Register (NBTP)

Table 822. NBTP field descriptions

Field	Description
0:6 NSJW	(Re) Synchronization Jump Width (0x00–0x7F) Valid values are 0 to 127. The actual interpretation by the hardware of this value is such that one more than the value programmed here is used.
7:15 NBRP	Baud Rate Prescaler (0x000–0x1FF) The value by which the oscillator frequency is divided for generating the bit time quanta. The bit time is built up from a multiple of this quanta. Valid values for the Baud Rate Prescaler are 0 to 511. The actual interpretation by the hardware of this value is such that one more than the value programmed here is used.
16:23 NTSEG1	Time segment before sample point (0x01–0xFF) Valid values are 1 to 255. The actual interpretation by the hardware of this value is such that one more than the programmed value is used.
25:31 NTSEG2	The time segment after the sample point (0x00–0x7F) Valid values are 0 to 127. The actual interpretation by the hardware of this value is such that one more than the programmed value is used.

Note: With a CAN clock of 8 MHz, the reset value of 0x06000A03 configures the M_CAN for a bit rate of 500 kbit/s.

52.3.5.2.8 Timestamp Counter Configuration Register (TSCC)

Offset: 0x0020 Access: RP

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	TCP(1)			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	TSS(1)	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. These are protected write (P) bits which means that write access by the bits is possible only when the bit 1 [CCE] and bit 0 [INIT] of CCCR register are set to ‘1’.

Figure 715. Timestamp Counter Configuration Register (TSCC)

Table 823. TSCC field descriptions

Field	Description
12:15 TCP	Timestamp Counter Prescaler (0x0–0xF) Configures the timestamp and timeout counters time unit in multiples of CAN bit times [1 to16]. The actual interpretation by the hardware of this value is such that one more than the value programmed here is used.
30:31 TSS	Timestamp Select 00 Timestamp counter value always 0x0000 01 Timestamp counter value incremented according to Timestamp Counter Prescaler (TCP) 10 Reserved for SPC58xEx only (reserved for SPC58xGx) 11 Same as '00' **Note:** In case of CAN FD mode the absolute duration of a bit time does change in one CAN message due to the different baud rates for arbitration phase and data phase. Therefore an external counter (available through GTM supported only by SPC58xEx) is request for timestamp generation. Take care on SPC58xGx where the internal timestamp counter does still reflect increasing number on bit times but the fix relationship to absolute time is not given any more. It is possible to establish in which order the frames are received.

52.3.5.2.9 Timestamp Counter Value Register (TSCV)

Offset: 0x0024 Access: R/W

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	TSC															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 716. Timestamp Counter Value Register (TSCV)

Table 824. TSCV field descriptions

Field	Description
16:31 TSC	Timestamp Counter The internal Timestamp Counter value is captured on start of frame (both Rx and Tx). When TSCC.TSS = '01', the Timestamp Counter is incremented in multiples of CAN bit times [1 to 16] depending on the configuration of TSCC.TCP. A wrap around sets interrupt flag IR.TSW. Write access resets the counter to zero.

Note: A "wrap around" is a change of the Timestamp Counter value from non-zero to zero not caused by write access to TSCV.

52.3.5.2.10 Timeout Counter Configuration Register (TOCC)

For a description of the Timeout Counter, refer to *Section 52.3.9*.

Offset: 0x0028 Access: RP

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	TOP(1)															
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	TOS(1)		ETOC(1)
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. These are protected write (P) bits which means that write access by the bits is possible only when the bit 1 [CCE] and bit 0 [INIT] of CCCR register are set to '1'.

Figure 717. Timeout Counter Configuration Register (TOCC)

Table 825. TOCC field descriptions

Field	Description
0:15 TOP	Timeout Period Start value of the Timeout Counter (down-counter). Configures the Timeout Period.
29:30 TOS	Timeout Select When operating in Continuous mode, a write to **TOCV** presets the counter to the value configured by **TOCC[TOP]** and continues down-counting. When the Timeout Counter is controlled by one of the FIFOs, an empty FIFO presets the counter to the value configured by **TOCC[TOP]**. Down-counting is started when the first FIFO element is stored. 00 Continuous operation 01 Timeout controlled by Tx Event FIFO 10 Timeout controlled by Rx FIFO 0 11 Timeout controlled by Rx FIFO 1
31 ETOC	Enable Timeout Counter 0 Timeout Counter disabled 1 Timeout Counter enabled **Note:** For use of timeout function with CAN FD refer to *Section 52.3.9*.

52.3.5.2.11 Timeout Counter Value Register (TOCV)

Offset: 0x002C Access: R/W

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	TOC															
W																
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Figure 718. Timeout Counter Value Register (TOCV)

Table 826. TOCV field descriptions

Field	Description
16:31 TOC	The Timeout Counter is decremented in multiples of CAN bit times (1 to 16) depending on the configuration of TSCC.TCP. When decremented to zero, interrupt flag IR.TOO is set and the Timeout Counter is stopped. Start and reset/restart conditions are configured via TOCC.TOS.

52.3.5.2.12 Error Counter Register (ECR)

Offset: 0x0040 Access: Read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	CEL(1)							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	RP	REC							TEC							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. Access type is RX: reset on read.

Figure 719. Error Counter Register (ECR)

Table 827. ECR field descriptions

Field	Description
8:15 CEL	CAN Error Logging The counter is incremented each time when a CAN protocol error causes the Transmit Error Counter or the Receive Error Counter to be incremented. It is reset by read access to CEL. The counter stops at 0xFF; the next increment of TEC or REC sets interrupt flag ELO bit of the IR register.
16 RP	Receive Error Passive 0 The Receive Error Counter is below the error passive level of 128 1 The Receive Error Counter has reached the error passive level of 128
17:23 REC	Receive Error Counter Actual state of the Receive Error Counter, values between 0 and 127.
24:31 TEC	Transmit Error Counter Actual state of the Transmit Error Counter, values between 0 and 255.

Note: When CCCR.ASM is set, the CAN protocol controller does not increment TEC and REC when a CAN protocol error is detected, but CEL is still incremented.

52.3.5.2.13 Protocol Status Register (PSR)

Offset: 0x0044 Access: RXS

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	TDCV						
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	PXE(1)	RFDF(1)	RBRS(1)	RESI(1)	DLEC(2)			BO	EW	EP	ACT		LEC(2)		
W																
Reset	0	0	0	0	0	1	1	1	0	0	0	0	0	1	1	1

1. Access type is RX: reset on read.
2. Access type is RS: set on read.

Figure 720. Protocol Status Register (PSR)

Table 828. PSR field descriptions

Field	Description
9:15 TDCV	Transmitter Delay Compensation Value (0x00-0x7F) Position of the secondary sample point, defined by the sum of the measured delay from CAN Tx to CAN Rx and TDCR.TDCO. The SSP position is, in the data phase, the number of mtq between the start of the transmitted bit and the secondary sample point. Valid values are 0 to 127 mtq.
17 PXE	Protocol Exception Event 0 No protocol exception event occurred since last read access 1 Protocol exception event occurred
18 RFDF	Received CAN FD Message This bit is set independent of acceptance filtering. 0 Since this bit was reset by the CPU, no CAN FD message has been received 1 Message in CAN FD format with FDF flag set has been received
19 RBRS	BRS flag of last received CAN FD Message This bit is set together with RFDF, independent of acceptance filtering. 0 Last received CAN FD message did not have its BRS flag set. 1 Last received CAN FD message had its BRS flag set.
20 RESI	ESI CAN FD Message with ESI flag This bit is set together with RFDF, independent of acceptance filtering. 0 Last received CAN FD message did not have its ESI flag set. 1 Last received CAN FD message had its ESI flag set
21:23 DLEC	Data Phase Last Error Code Type of last error that occurred in the data phase of a CAN FD format frame with its BRS flag set. Coding is the same as for LEC. This field will be cleared to zero when a CAN FD format frame with its BRS flag set has been transferred (reception or transmission) without error.
24 BO	Bus_Off Status 0 The M_CAN is not Bus_Off 1 The M_CAN is in Bus_Off state
25 EW	Warning Status 0 Both error counters are below the Error_Warning limit of 96 1 At least one of error counter has reached the Error_Warning limit of 96
26 EP	Error Passive 0 The M_CAN is in the Error_Active state. It normally takes part in bus communication and sends an active error flag when an error has been detected 1 The M_CAN is in the Error_Passive state

Table 828. PSR field descriptions (continued)

Field	Description
27:28 ACT	Activity Monitors the module's CAN communication state. 00 Synchronizing – node is synchronizing on CAN communication 01 Idle-node is neither receiver nor transmitter 10 Receiver-node is operating as receiver 11 Transmitter-node is operating as transmitter
29:31 LEC	Last Error Code The LEC indicates the type of the last error to occur on the CAN bus. This field will be cleared to '0' when a message has been transferred (reception or transmission) without error. 000 No Error: No error occurred since LEC has been reset by successful reception or transmission. 001 Stuff Error: More than 5 equal bits in a sequence have occurred in a part of a received message where this is not allowed. 010 Form Error: A fixed format part of a received frame has the wrong format. 011 AckError: The message transmitted by the M_CAN was not acknowledged by another node. 100 Bit1Error: During the transmission of a message (with the exception of the arbitration field), the device wanted to send a recessive level (bit of logical value '1'), but the monitored bus value was dominant. 101 Bit0Error: During the transmission of a message (or acknowledge bit, or active error flag, or overload flag), the device wanted to send a dominant level (data or identifier bit logical value '0'), but the monitored bus value was recessive. During Bus_Off recovery this status is set each time a sequence of 11 recessive bits has been monitored. This enables the CPU to monitor the proceeding of the Bus_Off recovery sequence (indicating the bus is not stuck at dominant or continuously disturbed). 110 CRCError: The CRC check sum of a received message was incorrect. The CRC of an incoming message does not match with the CRC calculated from the received data. 111 NoChange: Any read access to the Protocol Status Register re-initializes the LEC to '7'. When the LEC shows the value '7', no CAN bus event was detected since the last CPU read access to the Protocol Status Register.

Note: When a frame in CAN FD format has reached the data phase with BRS flag set, the next CAN event (error or valid frame) will be shown in DLEC instead of LEC. An error in a fixed stuff bit of a CAN FD CRC sequence will be shown as a Form Error, not Stuff Error.

The Bus_Off recovery sequence (refer to CAN Specification Rev. 2.0 or ISO11898-1) cannot be shortened by setting or resetting CCCR[INIT.] If the device goes Bus_Off, it will set CCCR[INIT] of its own accord, stopping all bus activities. Once CCCR[INIT] has been cleared by the CPU, the device will then wait for 129 occurrences of Bus Idle (129 × 11 consecutive recessive bits) before resuming normal operation. At the end of the Bus_Off recovery sequence, the Error Management Counters will be reset. During the waiting time after the resetting of CCCR[INIT], each time a sequence of 11 recessive bits has been monitored, a Bit 0 Error code is written to PSR[LEC], enabling the CPU to readily check up whether the CAN bus is stuck at dominant or continuously disturbed and to monitor the Bus_Off recovery sequence. ECR[REC] is used to count these sequences.

52.3.5.2.14 Transmitter Delay compensation Register (TDCR)

Offset: 0x0048 Access: RP

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	TDCO(1)							0	TDCF(1)						
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. These are protected write (P) bits which means that write access by the bits is possible only when the bit 1 [CCE] and bit 0 [INIT] of CCCR register are set to ‘1’.

Figure 721. Transmitter Delay compensation Register (TDCR)

Table 829. TDCR field descriptions

Field	Description
17:23 TDCO	Transmitter Delay Compensation Offset (0x00-0x7F) Offset value defining the distance between the measured delay from CAN Tx to CAN Rx and the secondary sample point. Valid values are 0 to 127 mtq.
25:31 TDCF	Transmitter Delay Compensation Filter Window Length (0x00-0x7F) Defines the minimum value for the SSP position, dominant edges on CAN Rx that would result in an earlier SSP position are ignored for transmitter delay measurement. The feature is enabled when TDCF is configured to a value greater than TDCO. Valid values are 0 to 127 mtq.

52.3.5.2.15 Interrupt Register (IR)

The flags are set when one of the listed conditions is detected (edge-sensitive). The flags remain set until the Host clears them. A flag is cleared by writing a ‘1’ to the corresponding bit position. Writing a ‘0’ has no effect. A hard reset will clear the register. The configuration of IE controls whether an interrupt is generated. The configuration of ILS controls on which interrupt line an interrupt is signaled.

Offset: 0x0050 Access: R/W

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	ARA	PED	PEA	WDI	BO	EW	EP	ELO	BEU	BEC	DRX	TOO	MRAF	TSW
W			w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	TEFL	TEFF	TEFW	TEFN	TFE	TCF	TC	HPM	RF1L	RF1F	RF1W	RF1N	RF0L	RF0F	RF0W	RF0N
W	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 722. Interrupt Register (IR)

Table 830. IR field descriptions

Field	Description
2 ARA	Access to Reserved Address 0 No access to reserved address occurred 1 Access to reserved address occurred
3 PED	Protocol Error in Data Phase (Data Bit Time is used) 0 No protocol error in data phase 1 Protocol error in data phase detected (PSR.DLEC ≠ 0,7)
4 PEA	Protocol Error in Arbitration Phase (Nominal Bit Time is used) 0= No protocol error in arbitration phase 1= Protocol error in arbitration phase detected (PSR.LEC ≠ 0,7)
5 WDI	Watchdog Interrupt 0 No Message RAM Watchdog event occurred 1 Message RAM Watchdog event due to missing READY
6 BO	Bus_Off Status 0 Bus_Off status unchanged 1 Bus_Off status changed
7 EW	Warning Status 0 Error_Warning status unchanged 1 Error_Warning status changed
8 EP	Error Passive 0 Error_Passive status unchanged 1 Error_Passive status changed
9 ELO	Error Logging Overflow 0 CAN Error Logging Counter did not overflow 1 Overflow of CAN Error Logging Counter occurred

Table 830. IR field descriptions (continued)

Field	Description
10 BEU	Bit Error Uncorrected Message RAM bit error detected, uncorrected. Controlled by Message RAM bit error input signal generated by an optional external parity / ECC logic attached to the Message RAM. An uncorrected Message RAM bit error sets CCCR[INIT] to '1'. This is done to avoid transmission of corrupted data. 0 No bit error detected when reading from Message RAM 1 Bit error detected, uncorrected (for example parity logic)
11 BEC	Bit Error Corrected Message RAM bit error detected and corrected. Controlled by Message RAM bit error input signal generated by an optional external parity / ECC logic attached to the Message RAM. 0 No bit error detected when reading from Message RAM 1 Bit error detected and corrected (for example ECC)
12 DRX	Message stored to Dedicated Rx Buffer The flag is set whenever a received message has been stored into a dedicated Rx Buffer. 0 No Rx Buffer updated 1 At least one received message stored into a Rx Buffer
13 TOO	Timeout Occurred 0 No timeout 1 Timeout reached
14 MRAF	Message RAM Access Failure The flag is set, when the Rx Handler – has not completed acceptance filtering or storage of an accepted message until the arbitration field of the following message has been received. In this case acceptance filtering or message storage is aborted and the Rx Handler starts processing of the following message. – was not able to write a message to the Message RAM. In this case message storage is aborted. In both cases the FIFO put index is not updated resp. the New Data flag for a dedicated Rx Buffer is not set, a partly stored message is overwritten when the next message is stored to this location. The flag is also set when the Tx Handler is not able to read a message from the Message RAM in time. In this case message transmission is aborted. To leave Restricted Operation Mode, the Host CPU has to reset CCCR.ASM. 0 No Message RAM access failure occurred 1 Message RAM access failure occurred
15 TSW	Timestamp Wraparound 0 No timestamp counter wrap-around 1 Timestamp counter wrapped around
16 TEFL	Tx Event FIFO Event Lost 0 No Tx Event FIFO element lost 1 Tx Event FIFO element lost, also set after write attempt to Tx Event FIFO of size zero
17 TEFF	Tx Event FIFO Full

Table 830. IR field descriptions (continued)

Field	Description
18 TEFW	Tx Event FIFO Watermark Reached 0 Tx Event FIFO fill level below watermark 1 Tx Event FIFO fill level reached watermark
19 TEFN	Tx Event FIDO New Entry 0 Tx Event FIFO unchanged 1 Tx Handler wrote Tx Event FIFO element
20 TFE	Tx FIFO Empty 0 Tx FIFO non-empty 1 Tx FIFO empty
21 TCF	Transmission Cancellation Finished 0 No transmission cancellation finished 1 Transmission cancellation finished
22 TC	Transmission Completed 0 No transmission completed 1 Transmission completed
23 HPM	High Priority Message 0 No high priority message received 1 High priority message received
24 RF1L	Rx FIFO 1 Message Lost 0 No Rx FIFO 1 message lost 1 Rx FIFO 1 message lost, also set after write attempt to Rx FIFO 1 of size zero
25 RF1F	Rx FIFO 1 Full 0 Rx FIFO 1 not full 1 Rx FIFO 1 full
26 RF1W	Rx FIFO 1 Watermark Reached 0 Rx FIFO 1 fill level below watermark 1 Rx FIFO 1 fill level reached watermark
27 RF1N	Rx FIFO 1 New Message 0 No new message written to Rx FIFO 1 1 New message written to Rx FIFO 1
28 RF0L	Rx FIFO 0 Message Lost 0 No Rx FIFO 0 message lost 1 Rx FIFO 0 message lost, also set after write attempt to Rx FIFO 0 of size zero
29 RF0F	Rx FIFO 0 Full 0 Rx FIFO 0 not full 1 Rx FIFO 0 full
30 RF0W	Rx FIFO 0 Watermark Reached 0 Rx FIFO 0 fill level below watermark 1 Rx FIFO 0 fill level reached watermark
31 RF0N	Rx FIFO 0 New Message 0 No new message written to Rx FIFO 0 1 New message written to Rx FIFO 0

52.3.5.2.16 Interrupt Enable Register (IE)

The settings in the Interrupt Enable register determine which status changes in the Interrupt register will be signaled on an interrupt line.

Offset: 0x0054 Access: R/W

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	ARAE	PEDE	PEAE	WDIE	BOE	EWE	EPE	ELOE	BEUE	BECE	DRXE	TOOE	MRAFE	TSWE
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	TEFLE	TEFFE	TEFWE	TEFNE	TFEE	TCFE	TCE	HPME	RF1LE	RF1FE	RF1WE	RF1NE	RF0LE	RF0FE	RF0WE	RF0NE
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 723. Interrupt Enable Register (IE)

Table 831. IE field descriptions

Field	Description
2 ARAE	Access to Reserved Address Enable 0 Interrupt disabled 1 Interrupt enabled
3 PEDE	Protocol Error in Data Phase Enable 0 Interrupt disabled 1 Interrupt enabled
4 PEAE	Protocol Error in Arbitration Phase Enable 0 Interrupt disabled 1 Interrupt enabled
5 WDIE	Watchdog Interrupt Enable 0 Interrupt disabled 1 Interrupt enabled
6 BOE	Bus_Off Status Interrupt Enable 0 Interrupt disabled 1 Interrupt enabled
7 EWE	Warning Status Interrupt Enable 0 Interrupt disabled 1 Interrupt enabled
8 EPE	Error Passive Interrupt Enable 0 Interrupt disabled 1 Interrupt enabled
9 ELOE	Error Logging Overflow Interrupt Enable 0 Interrupt disabled 1 Interrupt enabled

Table 831. IE field descriptions (continued)

Field	Description
10 BEUE	Bit Error Uncorrected Interrupt Enable 0 Interrupt disabled 1 Interrupt enabled
11 BECE	Bit Error Corrected Interrupt Enable 0 Interrupt disabled 1 Interrupt enabled
12 DRXE	Message stored to Dedicated Rx Buffer Interrupt Enable 0 Interrupt disabled 1 Interrupt enabled
13 TOOE	Timeout Occurred Interrupt Enable 0 Interrupt disabled 1 Interrupt enabled
14 MRAFE	Message RAM Access Failure Interrupt Enable 0 Interrupt disabled 1 Interrupt enabled
15 TSWE	Timestamp Wraparound Interrupt Enable 0 Interrupt disabled 1 Interrupt enabled
16 TEFLE	Tx Event FIFO Event Lost Interrupt Enable 0 Interrupt disabled 1 Interrupt enabled
17 TEFFE	Tx Event FIFO Full Interrupt Enable 0 Interrupt disabled 1 Interrupt enabled
18 TEFWE	Tx Event FIFO Watermark Reached Interrupt Enable 0 Interrupt disabled 1 Interrupt enabled
19 TEFNE	Tx Event FIFO New Entry Interrupt Enable 0 Interrupt disabled 1 Interrupt enabled
20 TFEE	Tx FIFO Empty Interrupt Enable 0 Interrupt disabled 1 Interrupt enabled
21 TCFE	Transmission Cancellation Finished Interrupt Enable 0 Interrupt disabled 1 Interrupt enabled
22 TCE	Transmission Completed Interrupt Enable 0 Interrupt disabled 1 Interrupt enabled
23 HPME	High Priority Message Interrupt Enable 0 Interrupt disabled 1 Interrupt enabled

Table 831. IE field descriptions (continued)

Field	Description
24 RF1LE	Rx FIFO 1 Message Lost Interrupt Enable 0 Interrupt disabled 1 Interrupt enabled
25 RF1FE	Rx FIFO 1 Full Interrupt Enable 0 Interrupt disabled 1 Interrupt enabled
26 RF1WE	Rx FIFO 1 Watermark Reached Interrupt Enable 0 Interrupt disabled 1 Interrupt enabled
27 RF1NE	Rx FIFO 1 New Message Interrupt Enable 0 Interrupt disabled 1 Interrupt enabled
28 RF0LE	Rx FIFO 0 Message Lost Interrupt Enable 0 Interrupt disabled 1 Interrupt enabled
29 RF0FE	Rx FIFO 0 Full Interrupt Enable 0 Interrupt disabled 1 Interrupt enabled
30 RF0WE	Rx FIFO 0 Watermark Reached Interrupt Enable 0 Interrupt disabled 1 Interrupt enabled
31 RF0NE	Rx FIFO 0 New Message Interrupt Enable 0 Interrupt disabled 1 Interrupt enabled

52.3.5.2.17 Interrupt Line Select Register (ILS)

The Interrupt Line Select register assigns an interrupt generated by a specific interrupt flag from the Interrupt Register to one of the two module interrupt lines.

Offset: 0x0058 Access: R/W

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	ARAL	PEDL	PEAL	WDIL	BOL	EWL	EPL	ELOL	BEUL	BECL	DRXL	TOOL	MRAFL	TSWL
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	TEFLL	TEFFL	TEFWL	TEFNL	TFEL	TCFL	TCL	HPML	RF1LL	RF1FL	RF1WL	RF1NL	RF0LL	RF0FL	RF0WL	RF0NL
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 724. Interrupt Line Select Register (ILS)

Table 832. ILS field descriptions

Field	Description
2 ARAL	Access to Reserved Address Line 0 Interrupt assigned to interrupt line M_CAN INT0 1 Interrupt assigned to interrupt line M_CAN INT1
3 PEDL	Protocol Error in Data Phase Line 0 Interrupt assigned to interrupt line M_CAN INT0 1 Interrupt assigned to interrupt line M_CAN INT1
4 PEAL	Protocol Error in Arbitration Phase Line 0 Interrupt assigned to interrupt line M_CAN INT0 1 Interrupt assigned to interrupt line M_CAN INT1
5 WDIL	Watchdog Interrupt Line 0 Interrupt assigned to interrupt line M_CAN INT0 1 Interrupt assigned to interrupt line M_CAN INT1
6 BOL	Bus_Off Status Interrupt Line 0 Interrupt assigned to interrupt line M_CAN INT0 1 Interrupt assigned to interrupt line M_CAN INT1
7 EWL	Warning Status Interrupt Line 0 Interrupt assigned to interrupt line M_CAN INT0 1 Interrupt assigned to interrupt line M_CAN INT1
8 EPL	Error Passive Interrupt Line 0 Interrupt assigned to interrupt line M_CAN INT0 1 Interrupt assigned to interrupt line M_CAN INT1
9 ELOL	Error Logging Overflow Interrupt Line 0 Interrupt assigned to interrupt line M_CAN INT0 1 Interrupt assigned to interrupt line M_CAN INT1
10 BEUL	Bit Error Uncorrected Interrupt Line 0 Interrupt assigned to interrupt line M_CAN INT0 1 Interrupt assigned to interrupt line M_CAN INT1
11 BECL	Bit Error Corrected Interrupt Line 0 Interrupt assigned to interrupt line M_CAN INT0 1 Interrupt assigned to interrupt line M_CAN INT1
12 DRXL	Message stored to Dedicated Rx Buffer Interrupt Line 0 Interrupt assigned to interrupt line M_CAN INT0 1 Interrupt assigned to interrupt line M_CAN INT1
13 TOOL	Timeout Occurred Interrupt Line 0 Interrupt assigned to interrupt line M_CAN INT0 1 Interrupt assigned to interrupt line M_CAN INT1
14 MRAFL	Message RAM Access Failure Interrupt Line 0 Interrupt assigned to interrupt line M_CAN INT0 1 Interrupt assigned to interrupt line M_CAN INT1
15 TSWL	TSWL: Timestamp Wraparound Interrupt Line 0 Interrupt assigned to interrupt line M_CAN INT0 1 Interrupt assigned to interrupt line M_CAN INT1

Table 832. ILS field descriptions (continued)

Field	Description
16 TEFLL	Tx Event FIFO Event Lost Interrupt Line 0 Interrupt assigned to interrupt line M_CAN INT0 1 Interrupt assigned to interrupt line M_CAN INT1
17 TEFFL	Tx Event FIFO Full Interrupt Line 0 Interrupt assigned to interrupt line M_CAN INT0 1 Interrupt assigned to interrupt line M_CAN INT1
18 TEFWL	Tx Event FIFO Watermark Reached Interrupt Line 0 Interrupt assigned to interrupt line M_CAN INT0 1 Interrupt assigned to interrupt line M_CAN INT1
19 TEFNL	Tx Event FIFO New Entry Interrupt Line 0 Interrupt assigned to interrupt line M_CAN INT0 1 Interrupt assigned to interrupt line M_CAN INT1
20 TFEL	Tx FIFO Empty Interrupt Line 0 Interrupt assigned to interrupt line M_CAN INT0 1 Interrupt assigned to interrupt line M_CAN INT1
21 TCFL	Transmission Cancellation Finished Interrupt Line 0 Interrupt assigned to interrupt line M_CAN INT0 1 Interrupt assigned to interrupt line M_CAN INT1
22 TCL	Transmission Completed Interrupt Line 0 Interrupt assigned to interrupt line M_CAN INT0 1 Interrupt assigned to interrupt line M_CAN INT1
23 HPML	High Priority Message Interrupt Line 0 Interrupt assigned to interrupt line M_CAN INT0 1 Interrupt assigned to interrupt line M_CAN INT1
24 RF1LL	Rx FIFO 1 Message Lost Interrupt Line 0 Interrupt assigned to interrupt line M_CAN INT0 1 Interrupt assigned to interrupt line M_CAN INT1
25 RF1FL	Rx FIFO 1 Full Interrupt Line 0 Interrupt assigned to interrupt line M_CAN INT0 1 Interrupt assigned to interrupt line M_CAN INT1
26 RF1WL	Rx FIFO 1 Watermark Reached Interrupt Line 0 Interrupt assigned to interrupt line M_CAN INT0 1 Interrupt assigned to interrupt line M_CAN INT1
27 RF1NL	Rx FIFO 1 New Message Interrupt Line 0 Interrupt assigned to interrupt line M_CAN INT0 1 Interrupt assigned to interrupt line M_CAN INT1
28 RF0LL	Rx FIFO 0 Message Lost Interrupt Line 0 Interrupt assigned to interrupt line M_CAN INT0 1 Interrupt assigned to interrupt line M_CAN INT1
29 RF0FL	Rx FIFO 0 Full Interrupt Line 0 Interrupt assigned to interrupt line M_CAN INT0 1 Interrupt assigned to interrupt line M_CAN INT1

Table 832. ILS field descriptions (continued)

Field	Description
30 RF0WL	Rx FIFO 0 Watermark Reached Interrupt Line 0 Interrupt assigned to interrupt line M_CAN INT0 1 Interrupt assigned to interrupt line M_CAN INT1
31 RF0NL	Rx FIFO 0 New Message Interrupt Line 0 Interrupt assigned to interrupt line M_CAN INT0 1 Interrupt assigned to interrupt line M_CAN INT1

52.3.5.2.18 Interrupt Line Enable Register (ILE)

Each of the two interrupt lines to the CPU can be enabled / disabled separately by programming bits EINT0 and EINT1

Offset: 0x005C Access: R/W

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	EINT1	EINT0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 725. Interrupt Line Enable Register (ILE)

Table 833. ILE field descriptions

Field	Description
30 EINT1	Enable Interrupt Line 1 0 Interrupt line disabled 1 Interrupt line enabled
31 EINT0	Enable Interrupt Line 0 0 Interrupt line disabled 1 Interrupt line enabled

52.3.5.2.19 Global Filter Configuration Register (GFC)

Global settings for Message ID filtering. The Global Filter Configuration controls the filter path for standard and extended messages as described in *Figure 764* and *Figure 765*.

Offset: 0x0080 Access: RP

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	ANFS(1)		ANFE(1)		RRFS(1)	RRFE(1)
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. These are protected write (P) bits which means that write access by the bits is possible only when the bit 1 [CCE] and bit 0 [INIT] of CCCR register are set to '1'.

Figure 726. Global Filter Configuration Register (GFC)

Table 834. GFC field descriptions

Field	Description
26:27 ANFS	Accept Non-matching Frames Standard Defines how received messages with 11-bit IDs that do not match any element of the filter list are treated. 00 Accept in Rx FIFO 0 01 Accept in Rx FIFO 1 10 Reject 11 Reject
28:29 ANFE	ANFE[1:0]: Accept Non-matching Frames Extended Defines how received messages with 29-bit IDs that do not match any element of the filter list are treated. 00 Accept in Rx FIFO 0 01 Accept in Rx FIFO 1 10 Reject 11 Reject
30 RRFS	Reject Remote Frames Standard 0 Filter remote frames with 11-bit standard IDs 1 Reject all remote frames with 11-bit standard IDs
31 RRFE	Reject Remote Frames Extended 0 Filter remote frames with 29-bit extended IDs 1 Reject all remote frames with 29-bit extended IDs

52.3.5.2.20 Standard ID Filter Configuration Register (SIDFC)

Settings for 11-bit standard Message ID filtering. The Standard ID Filter Configuration controls the filter path for standard messages as described in *Figure 758*.

Offset: 0x0084 Access: RP

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	LSS(1)							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	FLSSA(1)														0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. These are protected write (P) bits which means that write access by the bits is possible only when the bit 1 [CCE] and bit 0 [INIT] of CCCR register are set to ‘1’.

Figure 727. Standard ID Filter Configuration register (SIDFC)

Table 835. SIDFC field descriptions

Field	Description
8:15 LSS	List Size Standard 0x00: No standard Message ID filter 0x01: Number of standard Message ID filter elements ... 0x80: Number of standard Message ID filter elements 0x81: Values greater than 128 are interpreted as 128 ... 0xFF: Values greater than 128 are interpreted as 128
16:29 FLSSA	Filter List Standard Start Address Start address of standard Message ID filter list (32-bit word address, refer to *Figure 754*).

52.3.5.2.21 Extended ID Filter Configuration Register (XIDFC)

Settings for 29-bit extended Message ID filtering. The Extended ID Filter Configuration controls the filter path for standard messages as described in *Figure 759*.

Offset: 0x0088 Access: RP

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	LSE(1)						
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	FLESA(1)														0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. These are protected write (P) bits which means that write access by the bits is possible only when the bit 1 [CCE] and bit 0 [INIT] of CCCR register are set to ‘1’.

Figure 728. Extended ID Filter Configuration register (XIDFC)

Table 836. XIDFC field descriptions

Field	Description
9:15 LSE	List Size Extended 0x00: No extended Message ID filter 0x01: Number of extended Message ID filter elements ... 0x40: Number of extended Message ID filter elements 0x41: Values greater than 64 are interpreted as 64 ... 0x7F: Values greater than 64 are interpreted as 64
16:29 FLESA	Filter List Extended Start Address Start address of extended Message ID filter list (32-bit word address, refer to *Figure 754*).

52.3.5.2.22 Extended ID and Mask Register (XIDAM)

Offset: 0x0090 Access: RP

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	EIDM(1)												
W																
Reset	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	EIDM(1)															
W																
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

1. These are protected write (P) bits which means that write access by the bits is possible only when the bit 1 [CCE] and bit 0 [INIT] of CCCR register are set to ‘1’.

Figure 729. Extended ID and Mask Register (XIDAM)

Table 837. XIDAM field descriptions

Field	Description
3:31 EIDM	Extended ID Mask For acceptance filtering of extended frames the Extended ID AND Mask is ANDed with the Message ID of a received frame. Intended for masking of 29-bit IDs in SAE J1939. With the reset value of all bits set to one the mask is not active.

52.3.5.2.23 High Priority Message Status Register (HPMS)

This register is updated every time a Message ID filter element configured to generate a priority event matches. This can be used to monitor the status of incoming high priority messages and to enable fast access to these messages.

Offset: 0x0094 Access: R

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	FLST	FIDX							MSI		BIDX					
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 730. High Priority Message Status Register (HPMS)

Table 838. HPMS field descriptions

Field	Description
16 FLST	Filter List Indicates the filter list of the matching filter element. 0 Standard Filter List 1 Extended Filter List
17:23 FIDX	Filter Index Index of matching filter element. Range is 0 to SIDFC[LSS] - 1 resp. XIDFC[LSE] - 1.
24:25 MSI	Message Storage Indicator 00 No FIFO selected 01 FIFO overrun 10 Message stored in FIFO 0 11 Message stored in FIFO 1
26:31 BIDX	Buffer Index Index of Rx FIFO element to which the message was stored. Only valid when MSI[1] = ‘1’.

52.3.5.2.24 New Data 1 Register (NDAT1)

Offset: 0x0098 Access: R/W

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	ND31	ND30	ND29	ND28	ND27	ND26	ND25	ND24	ND23	ND22	ND21	ND20	ND19	ND18	ND17	ND16
W	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	ND15	ND14	ND13	ND12	ND11	ND10	ND9	ND8	ND7	ND6	ND5	ND4	ND3	ND2	ND1	ND0
W	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 731. New Data 1 Register (NDAT1)

Table 839. NDAT1 field descriptions

Field	Description
0:31 ND[31:0]	New Data[31:0] The register holds the New Data flags of Rx Buffers 0 to 31. The flags are set when the respective Rx Buffer has been updated from a received frame. The flags remain set until the Host clears them. A flag is cleared by writing a ‘1’ to the corresponding bit position. Writing a ’0’ has no effect. A hard reset will clear the register. 0 Rx Buffer not updated 1 Rx Buffer updated from new message

52.3.5.2.25 New Data 2 Register (NDAT2)

Offset: 0x009C Access: R/W

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	ND63	ND62	ND61	ND60	ND59	ND58	ND57	ND56	ND55	ND54	ND53	ND52	ND51	ND50	ND49	ND48
W	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	ND47	ND46	ND45	ND44	ND43	ND42	ND41	ND40	ND39	ND38	ND37	ND36	ND35	ND34	ND33	ND32
W	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 732. New Data 2 Register (NDAT2)

Table 840. NDAT2 field descriptions

Field	Description
0:31 ND[63:32]	New Data[63:32] The register holds the New Data flags of Rx Buffers 32 to 63. The flags are set when the respective Rx Buffer has been updated from a received frame. The flags remain set until the Host clears them. A flag is cleared by writing a ‘1’ to the corresponding bit position. Writing a ’0’ has no effect. A hard reset will clear the register. 0 Rx Buffer not updated 1 Rx Buffer updated from new message

52.3.5.2.26 Rx FIFO 0 Configuration Register (RXF0C)

Offset: 0x00A0 Access: RP

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	F0OM	F0WM(1)							0	F0S(1)						
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	F0SA(1)														0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. These are protected write (P) bits which means that write access by the bits is possible only when the bit 1 [CCE] and bit 0 [INIT] of CCCR register are set to '1'.

Figure 733. Rx FIFO 0 Configuration Register (RXF0C)

Table 841. RXF0C field descriptions

Field	Description
0 F0OM	FIFO 0 Operation Mode FIFO 0 can be operated in blocking or in overwrite mode. 0 FIFO 0 blocking mode 1 FIFO 0 overwrite mode
1:7 F0WM	Rx FIFO 0 Watermark 0x00: Watermark interrupt disabled 0x01: Level for Rx FIFO 0 watermark interrupt (IR[RF0W]) 0x40: Level for Rx FIFO 0 watermark interrupt (IR[RF0W]) 0x41: Watermark interrupt disabled 0x7F: Watermark interrupt disabled
9:15 F0S	Rx FIFO 0 Size 0x00: No Rx FIFO 0 0x01: Number of Rx FIFO 0 elements ... 0x40: Number of Rx FIFO 0 elements 0x41: Values greater than 64 are interpreted as 64 ... 0x7F: Values greater than 64 are interpreted as 64 The Rx FIFO 0 elements are indexed from 0 to F0S-1.
16:29 F0SA	Rx FIFO 0 Start Address Start address of Rx FIFO 0 in Message RAM (32-bit word address, *Figure 754*).

52.3.5.2.27 Rx FIFO 0 Status Register (RXF0S)

Offset: 0x00A4 Access: R

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	RF0L	F0F	0	0	F0PI					
W	0															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	F0GI						0	F0FL						
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 734. Rx FIFO 0 Status Register (RXF0S)

Table 842. RXF0S field descriptions

Field	Description
6 RF0L	Rx FIFO 0 Message Lost This bit is a copy of interrupt flag IR[RF0L]. When IR[RF0L] is reset, this bit is also reset. 0 No Rx FIFO 0 message lost 1 Rx FIFO 0 message lost, also set after write attempt to Rx FIFO 0 of size zero **Note:** Overwriting the oldest message when RXF0C.F0OM = ‘1’ will not set this flag.
7 F0F	Rx FIFO 0 Full 0 Rx FIFO 0 not full 1 Rx FIFO 0 full
10:15 F0PI	Rx FIFO 0 Put Index Rx FIFO 0 write index pointer, range 0 to 63.
18:23 F0GI	Rx FIFO 0 Get Index Rx FIFO 0 read index pointer, range 0 to 63.
25:31 F0FL	Rx FIFO 0 Fill Level Number of elements stored in Rx FIFO 0, range 0 to 64.

52.3.5.2.28 Rx FIFO 0 Acknowledge Register (RXF0A)

Offset: 0x00A8 Access: R/W

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	F0AI					
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 735. Rx FIFO 0 Acknowledge Register (RXF0A)

Table 843. RXF0A field descriptions

Field	Description
26:31 F0AI	Rx FIFO 0 Acknowledge Index After the Host has read a message or a sequence of messages from Rx FIFO 0 it has to write the buffer index of the last element read from Rx FIFO 0 to F0AI. This will set the Rx FIFO 0 Get Index RXF0S[F0GI] to F0AI + 1 and update the FIFO 0 Fill Level RXF0S[F0FL].

52.3.5.2.29 Rx Buffer Configuration Register (RXBC)

Offset: 0x00AC Access: RP

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	RBSA(1)														0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. These are protected write (P) bits which means that write access by the bits is possible only when the bit 1 [CCE] and bit 0 [INIT] of CCCR register are set to ‘1’.

Figure 736. Rx Buffer Configuration Register (RXBC)

Table 844. RXBC field descriptions

Field	Description
16:29 RBSA	Rx Buffer Start Address Configures the start address of the Rx Buffers section in the Message RAM (32-bit word address). Also used to reference debug messages A,B,C.

52.3.5.2.30 Rx FIFO 1 Configuration Register (RXF1C)

Offset: 0x00B0 Access: RP

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	F1OM(1)	F1WM(1)							0	F1S(1)						
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	F1SA(1)														0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. These are protected write (P) bits which means that write access by the bits is possible only when the bit 1 [CCE] and bit 0 [INIT] of CCCR register are set to '1'.

Figure 737. Rx FIFO 1 Configuration Register (RXF1C)

Table 845. RXF1C field descriptions

Field	Description
0 F1OM	FIFO 1 Operation Mode FIFO 1 can be operated in blocking or in overwrite mode. 0 FIFO 1 blocking mode 1 FIFO 1 overwrite mode
1:7 F1WM	Rx FIFO 1 Watermark 0x00: Watermark interrupt disabled 0x01: Level for Rx FIFO 1 watermark interrupt IR[RF1W] ... 0x40: Level for Rx FIFO 1 watermark interrupt IR[RF1W] 0x41: Watermark interrupt disabled ... 0x7F: Watermark interrupt disabled
9:15 F1S	F1S[6:0]: Rx FIFO 1 Size 0x00: No Rx FIFO 1 0x01: Number of Rx FIFO 1 elements ... 0x40: Number of Rx FIFO 1 elements 0x41: Values greater than 64 are interpreted as 64 ... 0x7F: Values greater than 64 are interpreted as 64 The Rx FIFO 1 elements are indexed from 0 to F1S - 1
16:29 F1SA	Rx FIFO 1 Start Address Start address of Rx FIFO 1 in Message RAM (32-bit word address, refer to *Figure 754*).

52.3.5.2.31 Rx FIFO 1 Status Register (RXF1S)

Offset: 0x00B4 Access: R

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	DMS		0	0	0	0	RF1L	F1F	0	0	F1PI					
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	F1GI						0	F1FL						
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 738. Rx FIFO 1 Status Register (RXF1S)

Table 846. RXF1S field descriptions

Field	Description
0:1 DMS	Debug Message Status 00 Idle state, wait for reception of debug messages, DMA request is cleared 01 Debug message A received 10 Debug messages A, B received 11 Debug messages A, B, C received, DMA request is set
6 RF1L	Rx FIFO 1 Message Lost This bit is a copy of interrupt flag IR[RF1L]. When IR[RF1L] is reset, this bit is also reset. 0 No Rx FIFO 1 message lost 1 Rx FIFO 1 message lost, also set after write attempt to Rx FIFO 1 of size zero **Note:** Overwriting the oldest message when RXF1C.F1OM = ‘1’ will not set this flag.
7 F1F	Rx FIFO 1 Full 0 Rx FIFO 1 not full 1 Rx FIFO 1 full
10:15 F1PI	Rx FIFO 1 Put Index Rx FIFO 1 write index pointer, range 0 to 63.
18:23 F1GI	Rx FIFO 1 Get Index Rx FIFO 1 read index pointer, range 0 to 63.
25:31 F1FL	Rx FIFO 1 Fill Level Number of elements stored in Rx FIFO 1, range 0 to 64.

52.3.5.2.32 Rx FIFO 1 Acknowledge Register (RXF1A)

Offset: 0x00B8 Access: R/W

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	F1AI					
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 739. Rx FIFO 1 Acknowledge Register (RXF1A)

Table 847. RXF1A field descriptions

Field	Description
26:31 F1AI	Rx FIFO 1 Acknowledge Index After the Host has read a message or a sequence of messages from Rx FIFO 1 it has to write the buffer index of the last element read from Rx FIFO 1 to F1AI. This will set the Rx FIFO 1 Get Index RXF1S[F1GI] to F1AI + 1 and update the FIFO 1 Fill Level RXF1S[F1FL].

52.3.5.2.33 Rx Buffer / FIFO Element Size Configuration (RXESC)

It configures the number of data bytes belonging to an Rx Buffer / Rx FIFO element. Data field sizes >8 bytes are intended for CAN FD operation only.

Offset: 0x00BC Access: RP

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	RBDS(1)				F1DS(1)				F0DS(1)		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. These are protected write (P) bits which means that write access by the bits is possible only when the bit 1 [CCE] and bit 0 [INIT] of CCCR register are set to '1'.

Figure 740. Rx Buffer / FIFO Element Size Configuration Register (RXESC)

Table 848. RXESC field descriptions

Field	Description
21:23 RBDS	Rx Buffer Data Field Size 000 8 byte data field 001 12 byte data field 010 16 byte data field 011 20 byte data field 100 24 byte data field 101 32 byte data field 110 48 byte data field 111 64 byte data field
25:27 F1DS	Rx FIFO 1 Data Field Size 000 8 byte data field 001 12 byte data field 010 16 byte data field 011 20 byte data field 100 24 byte data field 101 32 byte data field 110 48 byte data field 111 64 byte data field
29:31 F0DS	Rx FIFO 0 Data Field Size 000 8 byte data field 001 12 byte data field 010 16 byte data field 011 20 byte data field 100 24 byte data field 101 32 byte data field 110 48 byte data field 111 64 byte data field

Note: In case the data field size of an accepted CAN frame exceeds the data field size configured for the matching Rx Buffer or Rx FIFO, only the number of bytes as configured by RXESC is stored to the Rx Buffer respectively, Rx FIFO element. The rest of the frame's data field is ignored.

52.3.5.2.34 Tx Buffer Configuration Register (TXBC)

Offset: 0x00C0 Access: RP

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	TFQM(1)	TFQS(1)						0	0	NDTB(1)					
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	TBSA(1)														0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. These are protected write (P) bits which means that write access by the bits is possible only when the bit 1 [CCE] and bit 0 [INIT] of CCCR register are set to '1'.

Figure 741. Tx Buffer Configuration Register (TXBC)

Table 849. TXBC field descriptions

Field	Description
1 TFQM	Tx FIFO/Queue Mode 0 Tx FIFO operation 1 Tx Queue operation
2:7 TFQS	Tx FIFO/Queue Size 0x00: No Tx FIFO/Queue 0x01: Number of Tx Buffers used for Tx FIFO/Queue ... 0x20: Number of Tx Buffers used for Tx FIFO/Queue 0x21: Values greater than 32 are interpreted as 32 ... 0x3F: Values greater than 32 are interpreted as 32
10:15 NDTB	Number of Dedicated Transmit Buffers 0x00: No Dedicated Tx Buffers 0x01: Number of Dedicated Tx Buffers ... 0x20: Number of Dedicated Tx Buffers 0x21: Values greater than 32 are interpreted as 32 ... 0x3F: Values greater than 32 are interpreted as 32
16:29 TBSA	Tx Buffers Start Address Start address of Tx Buffers section in Message RAM (32-bit word address, refer to *Figure 754*).

Note: Be aware that the sum of TFQS and NDTB may be not greater than 32. There is no check for erroneous configurations. The Tx Buffers section in the Message RAM starts with the dedicated Tx Buffers.

52.3.5.2.35 Tx FIFO/Queue Status Register (TXFQS)

Offset: 0x00C4 Access: R

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	TFQF	TFQPI				
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	TFGI					0	0	TFFL					
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 742. Tx FIFO/Queue Status Register (TXFQS)

Table 850. TXFQS field descriptions

Field	Description
10 TFQF	Tx FIFO/Queue Full 0 Tx FIFO/Queue not full 1 Tx FIFO/Queue full
11:15 TFQPI	Tx FIFO/Queue Put Index Tx FIFO/Queue write index pointer, range 0 to 31.
19:23 TFGI	Tx FIFO Get Index Tx FIFO read index pointer, range 0 to 31. Read as zero when Tx Queue operation is configured (TXBC[TFQM] = ‘1’).
26:31 TFFL	Tx FIFO Free Level Number of consecutive free Tx FIFO elements starting from TFGI, range 0 to 32. Read as zero when Tx Queue operation is configured (TXBC[TFQM] = ‘1’).

Note: In case of mixed configurations where dedicated Tx Buffers are combined with a Tx FIFO or a Tx Queue, the Put and Get indexes indicate the number of the Tx Buffer starting with the first dedicated Tx Buffers. Example: For a configuration of 12 dedicated Tx Buffers and a Tx FIFO of 20 Buffers a Put Index of 15 points to the fourth buffer of the Tx FIFO.

52.3.5.2.36 Tx Buffer Element Size Configuration (TXESC)

It configures the number of data bytes belonging to a Tx Buffer element. Data field sizes > 8 bytes are intended for CAN FD operation only.

Offset: 0x00C8 Access: RP

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	TBDS(1)		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. These are protected write (P) bits which means that write access by the bits is possible only when the bit 1 [CCE] and bit 0 [INIT] of CCCR register are set to ‘1’.

Figure 743. Tx Buffer Element Size Configuration Register (TXESC)

Table 851. TXESC field descriptions

Field	Description
29:31 TBDS	Tx Buffer Data Field Size 000 8 byte data field 001 12 byte data field 010 16 byte data field 011 20 byte data field 100 24 byte data field 101 32 byte data field 110 48 byte data field 111 64 byte data field **Note:** In case the data length code DLC of a Tx Buffer element is configured to a value higher than the Tx Buffer data field size TXESC.TBDS, the bytes not defined by the Tx Buffer are transmitted as ‘0xCC’ (padding bytes).

52.3.5.2.37 Tx Buffer Request Pending Register (TXBRP)

Offset: 0x00CC Access: R

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	TRP31	TRP30	TRP29	TRP28	TRP27	TRP26	TRP25	TRP24	TRP23	TRP22	TRP21	TRP20	TRP19	TRP18	TRP17	TRP16
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	TRP15	TRP14	TRP13	TRP12	TRP11	TRP10	TRP9	TRP8	TRP7	TRP6	TRP5	TRP4	TRP3	TRP2	TRP1	TRP0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 744. Tx Buffer Request Pending Register (TXBRP)

Table 852. TXBRP field descriptions

Field	Description
0:31 TRP[31:0]	Transmission Request Pending Each Tx Buffer has its own Transmission Request Pending bit. The bits are set via register TXBAR. The bits are reset after a requested transmission has completed or has been cancelled via register TXBCR. TXBRP bits are set only for those Tx Buffers configured via TXBC. After a TXBRP bit has been set, a Tx scan (refer to *Section 52.4.7.16: Debug on CAN Support*) is started to check for the pending Tx request with the highest priority (Tx Buffer with lowest Message ID). A cancellation request resets the corresponding transmission request pending bit of register TXBRP. In case a transmission has already been started when a cancellation is requested, this is done at the end of the transmission, regardless whether the transmission was successful or not. The cancellation request bits are reset directly after the corresponding TXBRP bit has been reset. After a cancellation has been requested, a finished cancellation is signaled via TXBCF – after successful transmission together with the corresponding TXBTO bit – when the transmission has not yet been started at the point of cancellation – when the transmission has been aborted due to lost arbitration – when an error occurred during frame transmission In DAR mode all transmissions are automatically cancelled if they are not successful. The corresponding TXBCF bit is set for all unsuccessful transmissions. 0 No transmission request pending 1 Transmission request pending

Note: TXBRP bits which are set while a Tx scan is in progress are not considered during this particular Tx scan. In case a cancellation is requested for such a Tx Buffer, this Add Request is cancelled immediately, the corresponding TXBRP bit is reset.

52.3.5.2.38 Tx Buffer Add Request Register (TXBAR)

Offset: 0x00D0 Access: R/W

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	AR31	AR30	AR29	AR28	AR27	AR26	AR25	AR24	AR23	AR22	AR21	AR20	AR19	AR18	AR17	AR16
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	AR15	AR14	AR13	AR12	AR11	AR10	AR9	AR8	AR7	AR6	AR5	AR4	AR3	AR2	AR1	AR0
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 745. Tx Buffer Add Request Register (TXBAR)

Table 853. TXBAR field descriptions

Field	Description
0:31 AR[31:0]	Add Request Each Tx Buffer has its own Add Request bit. Writing a ‘1’ will set the corresponding Add Request bit; writing a ‘0’ has no impact. This enables the Host to set transmission requests for multiple Tx Buffers with one write to TXBAR. TXBAR bits are set only for those Tx Buffers configured via TXBC. When no Tx scan is running, the bits are reset immediately, else the bits remain set until the Tx scan process has completed. 0 No transmission request added 1 Transmission requested added

Note: If an add request is applied for a Tx Buffer with pending transmission request (corresponding TXBRP bit already set), this add request is ignored.

52.3.5.2.39 Tx Buffer Cancellation Request Register (TXBCR)

Offset: 0x00D4 Access: R/W

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	CR31	CR30	CR29	CR28	CR27	CR26	CR25	CR24	CR23	CR22	CR21	CR20	CR19	CR18	CR17	CR16
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	CR15	CR14	CR13	CR12	CR11	CR10	CR9	CR8	CR7	CR6	CR5	CR4	CR3	CR2	CR1	CR0
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 746. Tx Buffer Cancellation Request Register (TXBCR)

Table 854. TXBCR field descriptions

Field	Description
0:31 CR[31:0]	Cancellation Request Each Tx Buffer has its own Cancellation Request bit. Writing a ‘1’ will set the corresponding Cancellation Request bit; writing a ‘0’ has no impact. This enables the Host to set cancellation requests for multiple Tx Buffers with one write to TXBCR. TXBCR bits are set only for those Tx Buffers configured via TXBC. The bits remain set until the corresponding bit of TXBRP is reset. 0 No cancellation pending 1 Cancellation pending

52.3.5.2.40 Tx Buffer Transmission Occurred Register (TXBTO)

Offset: 0x00D8 Access: Read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	TO31	TO30	TO29	TO28	TO27	TO26	TO25	TO24	TO23	TO22	TO21	TO20	TO19	TO18	TO17	TO16
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	TO15	TO14	TO13	TO12	TO11	TO10	TO9	TO8	TO7	TO6	TO5	TO4	TO3	TO2	TO1	TO0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 747. Tx Buffer Transmission Occurred Register (TXBTO)

Table 855. TXBTO field descriptions

Field	Description
0:31 TO[31:0]	Transmission Occurred Each Tx Buffer has its own Transmission Occurred bit. The bits are set when the corresponding TXBRP bit is cleared after a successful transmission. The bits are reset when a new transmission is requested by writing a ‘1’ to the corresponding bit of register TXBAR. 0 No transmission occurred 1 Transmission occurred

52.3.5.2.41 Tx Buffer Cancellation Finished Register (TXBCF)

Offset: 0x00DC Access: Read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	CF31	CF30	CF29	CF28	CF27	CF26	CF25	CF24	CF23	CF22	CF21	CF20	CF19	CF18	CF17	CF16
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	CF15	CF14	CF13	CF12	CF11	CF10	CF9	CF8	CF7	CF6	CF5	CF4	CF3	CF2	CF1	CF0
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 748. Tx Buffer Cancellation Finished Register (TXBCF)

Table 856. TXBCF field descriptions

Field	Description
0:31 CF[31:0]	Cancellation Finished Each Tx Buffer has its own Cancellation Finished bit. The bits are set when the corresponding TXBRP bit is cleared after a cancellation was requested via TXBCR. In case the corresponding TXBRP bit was not set at the point of cancellation, CF is set immediately. The bits are reset when a new transmission is requested by writing a '1' to the corresponding bit of register TXBAR. 0 No transmit buffer cancellation 1 Transmit buffer cancellation finished

52.3.5.2.42 Tx Buffer Transmission Interrupt Enable Register (TXBTIE)

Offset: 0x00E0 Access: Read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	TIE31	TIE30	TIE29	TIE28	TIE27	TIE26	TIE25	TIE24	TIE23	TIE22	TIE21	TIE20	TIE19	TIE18	TIE17	TIE16
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	TIE15	TIE14	TIE13	TIE12	TIE11	TIE10	TIE9	TIE8	TIE7	TIE6	TIE5	TIE4	TIE3	TIE2	TIE1	TIE0
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 749. Tx Buffer Transmission Interrupt Enable Register (TXBTIE)

Table 857. TXBTIE field descriptions

Field	Description
0:31 TIE[31:0]	Transmission Interrupt Enable Each Tx Buffer has its own Transmission Interrupt Enable bit. 0 Transmission interrupt disabled 1 Transmission interrupt enable

52.3.5.2.43 Tx Buffer Cancellation Finished Interrupt Enable Register (TXBCIE)

Offset: 0x00E4 Access: R/W

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R/W	CFIE31	CFIE30	CFIE29	CFIE28	CFIE27	CFIE26	CFIE25	CFIE24	CFIE23	CFIE22	CFIE21	CFIE20	CFIE19	CFIE18	CFIE17	CFIE16
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R/W	CFIE15	CFIE14	CFIE13	CFIE12	CFIE11	CFIE10	CFIE9	CFIE8	CFIE7	CFIE6	CFIE5	CFIE4	CFIE3	CFIE2	CFIE1	CFIE0
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 750. Tx Buffer Cancellation Finished Interrupt Enable Register (TXBCIE)

Table 858. TXBCIE field description

Field	Description
0:31 CFIE[31:0]	Cancellation Finished Interrupt Enable Each Tx Buffer has its own Cancellation Finished Interrupt Enable bit. 0 Cancellation finished interrupt disabled 1 Cancellation finished interrupt enabled

52.3.5.2.44 Tx Event FIFO Configuration Register (TXEFC)

Offset: 0x00F0 Access: RP

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0		EFWM(1)						0		EFS(1)					
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	EFSA(1)														0	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. These are protected write (P) bits which means that write access by the bits is possible only when the bit 1 [CCE] and bit 0 [INIT] of CCCR register are set to ‘1’.

Figure 751. Tx Event FIFO Configuration Register (TXEFC)

Table 859. TXEFC field descriptions

Field	Description
2:7 EFWM	Event FIFO Watermark 0x00: Watermark interrupt disabled 0x01: Level for Tx Event FIFO watermark interrupt (IR[TEFW]) ... 0x20: Level for Tx Event FIFO watermark interrupt (IR[TEFW]) 0x21: Watermark interrupt disabled ... 0x3F: Watermark interrupt disabled
10:15 EFS	Event FIFO Size 0x00: Tx Event FIFO disabled 0x01: Number of Tx Event FIFO elements ... 0x20: Number of Tx Event FIFO elements 0x21: Values greater than 32 are interpreted as 32 ... 0x3F: Values greater than 32 are interpreted as 32 The Tx Event FIFO elements are indexed from 0 to EFS - 1
16:29 EFSA	Event FIFO Start Address. Start address of Tx Event FIFO in Message RAM (32-bit word address,*Figure 754*.)

52.3.5.2.45 Tx Event FIFO Status Register (TXEFS)

Offset: 0x00F4 Access: R

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	TEFL	EFF	0	0	0	EFPI				
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	EFGI					0	0	EFFL					
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 752. Tx Event FIFO Status Register (TXEFS)

Table 860. TXEFS field descriptions

Field	Description
6 TEFL	Tx Event FIFO Element Lost This bit is a copy of interrupt flag IR[TEFL]. When IR[TEFL] is reset, this bit is also reset. 0 No Tx Event FIFO element lost 1 Tx Event FIFO element lost, also set after write attempt to Tx Event FIFO of size zero.
7 EFF	Event FIFO Full 0 Tx Event FIFO not full 1 Tx Event FIFO full
11:15 EFPI	Event FIFO Put Index Tx Event FIFO write index pointer, range 0 to 31.
19:23 EFGI	Event FIFO Get Index Tx Event FIFO read index pointer, range 0 to 31.
26:31 EFFL	Event FIFO Fill Level Number of elements stored in Tx Event FIFO, range 0 to 32.

52.3.5.2.46 Tx Event FIFO Acknowledge Register (TXEFA)

Offset: 0x00F8 Access: R/W

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	EFAI				
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 753. Tx Event FIFO Acknowledge Register (TXEFA)

Table 861. TXEFA field descriptions

Field	Description
27:31 EFAI	Event FIFO Acknowledge Index After the Host has read an element or a sequence of elements from the Tx Event FIFO it has to write the index of the last element read from Tx Event FIFO to EFAI. This will set the Tx Event FIFO Get Index TXEFS[EFGI] to EFAI + 1 and update the FIFO 0 Fill Level TXEFS[EFFL].

52.3.6 Message RAM

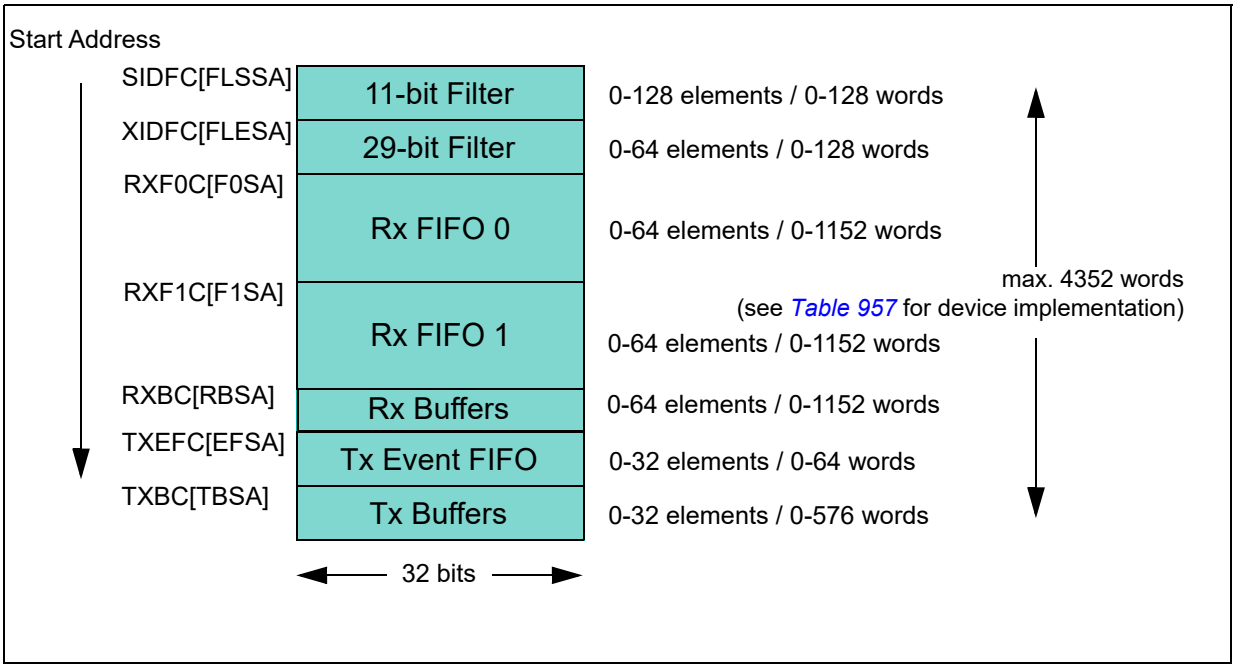
For storage of Rx/Tx messages and for storage of the filter configuration a single- or dual-ported Message RAM has to be connected to the M_CAN module.

Note: Because the Message RAM is equipped with ECC functionality, it is recommended to initialize the Message RAM after hardware reset by writing, for example, 0x00000000 to each Message RAM word to create valid parity/ECC checksums. This avoids that reading from uninitialized Message RAM sections will activate interrupt IR.BEC (Bit Error Corrected) or IR.BEU (Bit Error Uncorrected).

The Message RAM has a width of 32 bits. In case parity checking or ECC is used a respective number of bits has to be added to each word. The M_CAN module can be configured to allocate up to 4352 words in the Message RAM (for the chip-specific implementation details of each module's instance, refer to device configuration chapter). It is not necessary to configure each of the sections listed in *Figure 754*, nor is there any restriction with respect to the sequence of the sections.

When operated in CAN FD mode the required Message RAM size strongly depends on the element size configured for Rx FIFO0, Rx FIFO1, Rx Buffers, and Tx Buffers via RXESC.F0DS, RXESC.F1DS, RXESC.RBDS, and TXESC.TBDS.

Figure 754. Message RAM Configuration



When the M_CAN addresses the Message RAM it addresses 32-bit words, not single bytes. The configurable start addresses are 32-bit word addresses such as only bits 15 to 2 are evaluated, the two least significant bits are ignored.

Note: The M_CAN does not check for erroneous configuration of the Message RAM. Especially the configuration of the start addresses of the different sections and the number of elements of each section has to be done carefully to avoid falsification or loss of data.

52.3.6.1 Rx Buffer and FIFO element

Up to 64 Rx Buffers and two Rx FIFOs can be configured in the Message RAM. Each Rx FIFO section can be configured to store up to 64 received messages. The structure of a Rx Buffer / FIFO element is shown in the following figure. The element size can be configured for storage of CAN FD messages with up to 64 bytes data field via register RXESC.

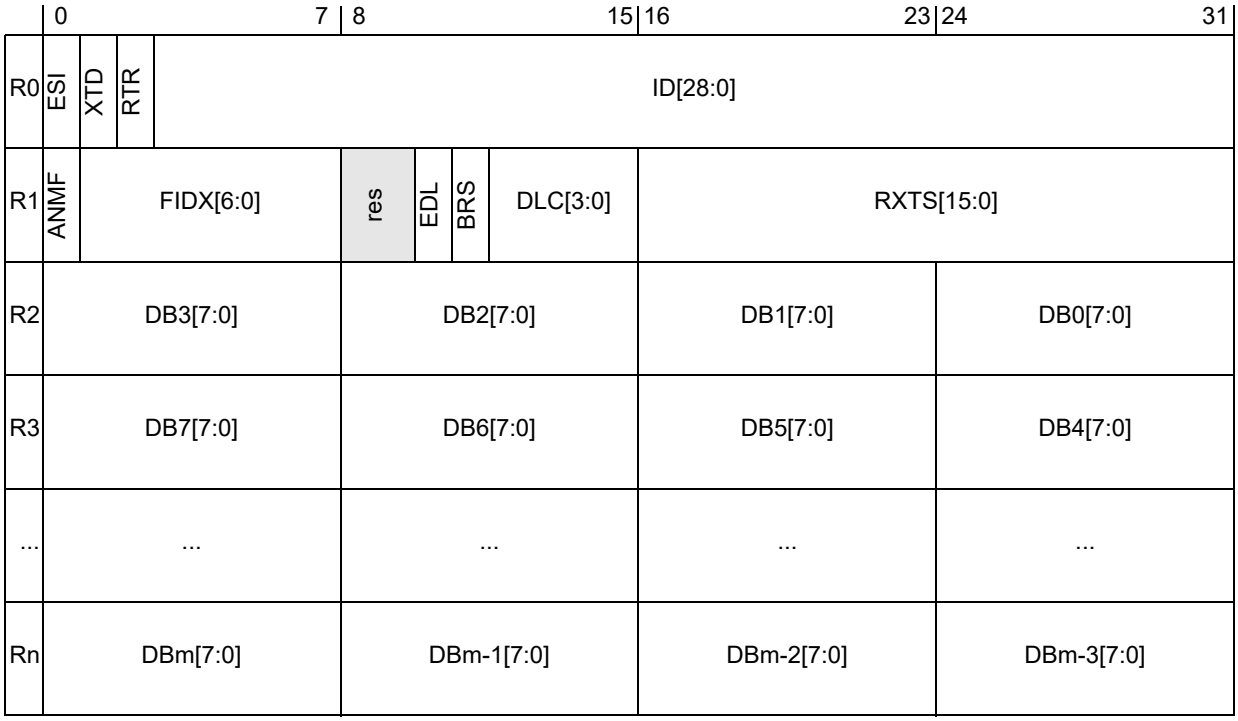


Figure 755. Rx Buffer and FIFO element

Table 862. Rx Buffer and FIFO element descriptions

Field	Description
R0 bit 0 ESI: Error State Indicator	0 Transmitting node is error active 1 Transmitting node is error passive
R0 bit 1 XTD: Extended Identifier	Signals to the Host whether the received frame has a standard or extended identifier. 0 11-bit standard identifier 1 29-bit extended identifier
R0 bit 2 RTR: Remote Transmission Request	Signals to the Host whether the received frame is a data frame or a remote frame. 0 Received frame is a data frame 1 Received frame is a remote frame **Note:** There are no remote frames in CAN FD format. In case a CAN FD frame was received (EDL = '1'), bit RTR reflects the state of the reserved bit r1.
R0 bits 3:31 ID[28:0]: Identifier	Standard or extended identifier depending on bit XTD. A standard identifier is stored into ID[28:18].
R1 bit 0 ANMF: Accepted Non-matching Frame	Acceptance of non-matching frames may be enabled via GFC[ANFS] and GFC[ANFE]. 0 Received frame matching filter index FIDX 1 Received frame did not match any Rx filter element

Table 862. Rx Buffer and FIFO element descriptions (continued)

Field	Description
R1 bits 1:7 FIDX[6:0]: Filter Index	0-127=Index of matching Rx acceptance filter element (invalid if ANMF = '1'). Range is 0 to SIDFC[LSS] - 1 resp. XIDFC[LSE] - 1.
R1 bit 10 EDL: Extended Data Length	0 Standard frame format 1 CAN FD frame format (new DLC-coding and CRC)
R1 bit 11 BRS: Bit Rate Switch	0 Frame received without bit rate switching 1 Frame received with bit rate switching
R1 bits 12:15 DLC[3:0]: Data Length Code	0-8 CAN + CAN FD: Received frame has 0-8 data bytes 9-15 CAN: Received frame has 8 data bytes 9-15 CAN FD: received frame has 12/16/20/24/32/48/64 data bytes
R1 bits 16:31 RXTS[15:0]: Rx Timestamp	Timestamp Counter value captured on start of frame reception. Resolution depending on configuration of the Timestamp Counter Prescaler TSCC[TCP].
R2 bits 0:7	DB3[7:0]: Data Byte 3
R2 bits 8:15	DB2[7:0]: Data Byte 2
R2 bits 16:23	DB1[7:0]: Data Byte 1
R2 bits 24:31	DB0[7:0]: Data Byte 0
R3 bits 0:7	DB7[7:0]: Data Byte 7
R3 bits 8:15	DB6[7:0]: Data Byte 6
R3 bits 16:23	DB5[7:0]: Data Byte 5
R3 bits 24:31	DB4[7:0]: Data Byte 4
...	...
Rn bits 0:7	DBm[7:0]: Data Byte m
Rn bits 8:15	DBm-1[7:0]: Data Byte m-1
Rn bits 16:23	DBm-2[7:0]: Data Byte m-2
Rn bits 24:31	DBm-3[7:0]: Data Byte m-3

Note: Depending on the configuration of the element size (RXESC), between two and sixteen 32-bit words (Rn = 3 to 17) are used for storage of a CAN message's data field.

52.3.6.2 Tx Buffer element

The Tx Buffers section can be configured to hold dedicated Tx Buffers as well as a Tx FIFO / Tx Queue. In case that the Tx Buffers section is shared by dedicated Tx buffers and a Tx FIFO / Tx Queue, the dedicated Tx Buffers start at the beginning of the Tx Buffers section followed by the buffers assigned to the Tx FIFO or Tx Queue. The Tx Handler distinguishes between dedicated Tx Buffers and Tx FIFO / Tx Queue by evaluating the Tx Buffer

configuration TXBC.TFQS and TXBC.NDTB. The element size can be configured for storage of CAN FD messages with up to 64 bytes data field via register TXESC.

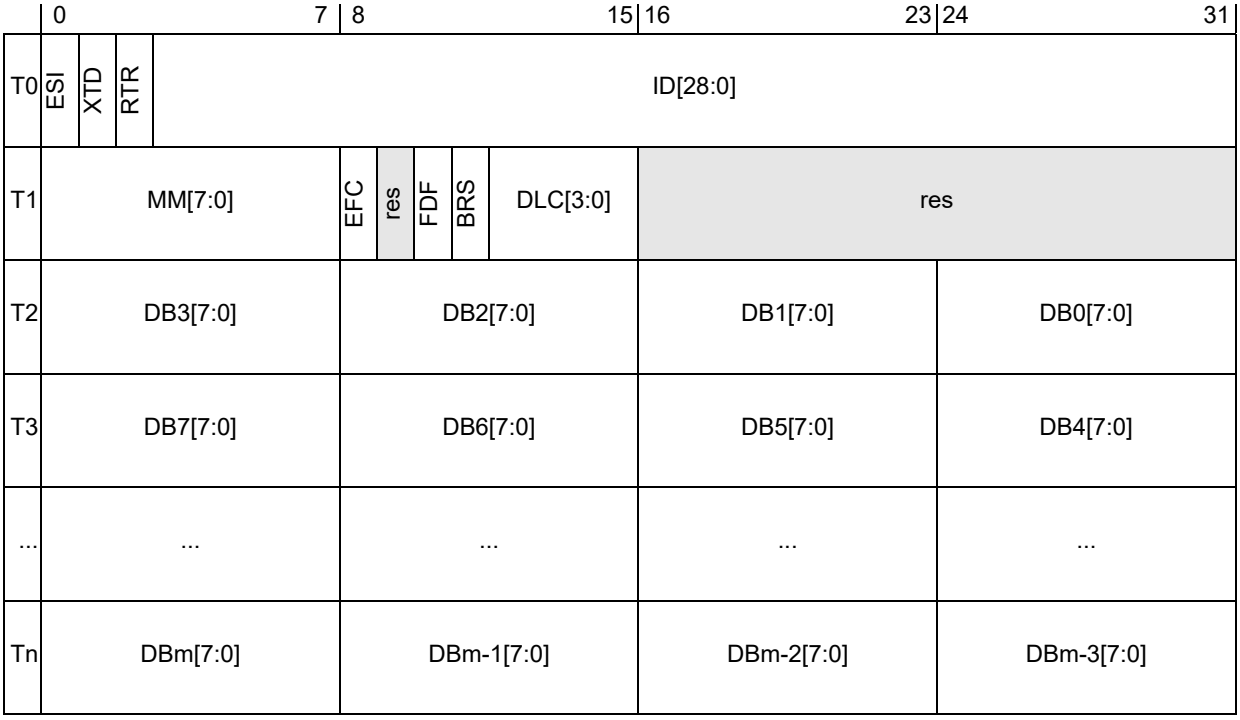


Figure 756. Tx Buffer element

Table 863. Tx Buffer element descriptions

Field	Description
T0 bit 0 ESI: Error State Indicator	0 ESI bit in CAN FD format depends only on error passive flag 1 ESI bit in CAN FD format transmitted recessive **Note:** The ESI bit of the transmit buffer is or'ed with the error passive flag to decide the value of the ESI bit in the transmitted FD frame. As required by the CAN FD protocol specification, an error active node may optionally transmit the ESI bit recessive, but an error passive node will always transmit the ESI bit recessive
T0 bit 1 XTD: Extended Identifier	0 11-bit standard identifier 129-bit extended identifier
T0 bit 2 RTR: Remote Transmission Request	0 Transmit data frame 1 Transmit remote frame **Note:** When RTR = 1, the M_CAN transmits a remote frame according to ISO11898-1, even if CCCR.FDOE enables the transmission in CAN FD format.
T0 bits 3:31 ID[28:0]: Identifier	Standard or extended identifier depending on bit XTD. A standard identifier has to be written to ID[28:18].
T1 bits 0:7 MM[7:0]: Message Marker	Written by CPU during Tx Buffer configuration. Copied into Tx Event FIFO element for identification of Tx message status.

Table 863. Tx Buffer element descriptions (continued)

Field	Description
T1 bit 8 EFC **Event FIFO Control**	0 Don't store Tx events 1 Store Tx events
T1 bit 10 FDF **FD Format**	0 Frame transmitted in Classic CAN format 1 Frame transmitted in CAN FD format
T1 bit 11 BRS **Bit Rat Switching**	0 CAN FD frames transmitted without bit rate switching 1 CAN FD frames transmitted with bit rate switching
T1 bits 12:15 DLC[3:0] **Data Length Code**	0-8 Transmit frame with 0-8 data bytes 9-15 Transmit frame with 8 data bytes 9-15 CAN FD: transmit frame has 12/16/20/24/32/48/64 data bytes
T2 bits 0:7	DB3[7:0]: Data Byte 3
T2 bits 8:15	DB2[7:0]: Data Byte 2
T2 bits 16:23	DB1[7:0]: Data Byte 1
T2 bits 24:31	DB0[7:0]: Data Byte 0
T3 bits 0:7	DB7[7:0]: Data Byte 7
T3 bits 8:15	DB6[7:0]: Data Byte 6
T3 bits 16:23	DB5[7:0]: Data Byte 5
T3 bits 24:31	DB4[7:0]: Data Byte 4
...	...
Tn bits 0:7	DBm[7:0]: Data Byte m
Tn bits 8:15	DBm-1[7:0]: Data Byte m-1
Tn bits 16:23	DBm-2[7:0]: Data Byte m-2
Tn bits 24:31	DBm-3[7:0]: Data Byte m-3

Note: Depending on the configuration of the element size (TXESC), between two and sixteen 32-bit words (Tn = 3 to 17) are used for storage of a CAN message's data field.

52.3.6.3 Tx Event FIFO element

Each element stores information about transmitted messages. By reading the Tx Event FIFO the Host CPU gets this information in the order the messages were transmitted. Status information about the Tx Event FIFO can be obtained from register TXEFS.

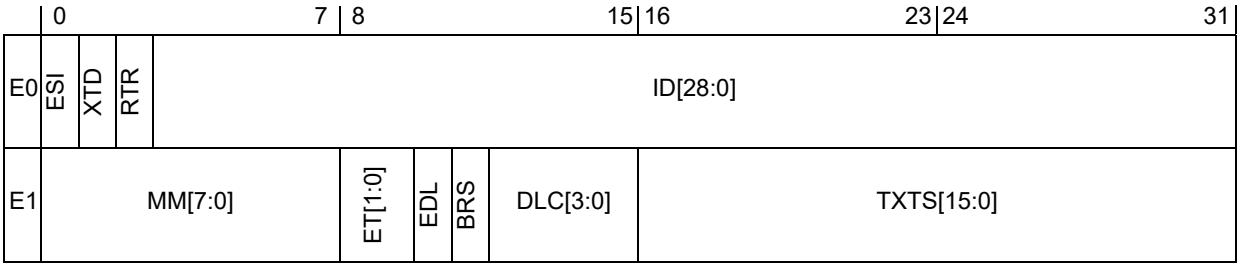


Figure 757. Tx Event FIFO element

Table 864. Tx Event FIFO element descriptions

Field	Description
E0 bit 0 ESI	Error State Indicator 0 Transmitting node is error active 1 Transmitting node is error passive
E0 bit 1 XTD	Extended Identifier 0 11-bit standard identifier 1 29-bit extended identifier
E0 bit 2 RTR	Remote Transmission Request 0 Data frame transmitted 1 Remote frame transmitted
E0 bits 3:31 ID[28:0]	Identifier Standard or extended identifier depending on bit XTD. A standard identifier has to be written to ID[28:18].
E1 bits 0:7 MM[7:0]	Message Marker Copied from Tx Buffer into Tx Event FIFO element for identification of Tx message status.
E1 bit 8:9 ET[1:0]	Event Type 00 Reserved 01 Tx event 10 Transmission in spite of cancellation (always set for transmissions in DAR mode) 11 Reserved
E1 bit 10 EDL	Extended Data Length 0 Standard frame format 1 CAN FD frame format (new DLC-coding and CRC)
E1 bit 11 BRS	Bit Rate Switch 0 Frame transmitted without bit rate switching 1 Frame transmitted with bit rate switching

Table 864. Tx Event FIFO element descriptions (continued)

Field	Description
E1 bits 12:15 DLC[3:0]	Data Length Code 0-8 CAN + CAN FD: frame with 0-8 data bytes transmitted 9-15 CAN: frame with 8 data bytes transmitted 9-15 CAN FD: frame with 12/16/20/24/32/48/64 data bytes transmitted
E1 bits 16:31 TXTS[15:0]	Tx Timestamp Timestamp Counter value captured on start of frame transmission. Resolution depending on configuration of the Timestamp Counter Prescaler TSCC[TCP].

52.3.6.4 Standard Message ID Filter element

Up to 128 filter elements can be configured for 11-bit standard IDs. When accessing a Standard Message ID Filter element, its address is the Filter List Standard Start Address SIDFC[FLSSA] plus the index of the filter element (0 to 127).

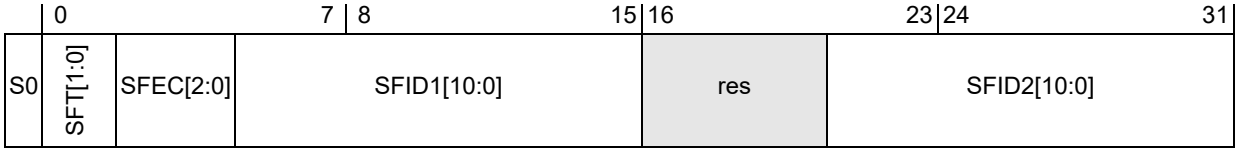


Figure 758. Standard Message ID Filter element

Table 865. Standard Message ID Filter element descriptions

Field	Description
Bits 0:1 SFT[1:0]	Standard Filter Type 00 Range filter from SFID1 to SFID2 (SFID2 ≥ SFID1) 01 Dual ID filter for SFID1 or SFID2 10 Classic filter: SFID1 = filter, SFID2 = mask 11 Reserved
Bit 2:4 SFEC[2:0]	Standard Filter Element Configuration All enabled filter elements are used for acceptance filtering of standard frames. Acceptance filtering stops at the first matching enabled filter element or when the end of the filter list is reached. If SFEC = ‘100’, ‘101’, or ‘110’ a match sets interrupt flag IR[HPM] and, if enabled, an interrupt is generated. In this case register HPMS is updated with the status of the priority match. 000 Disable filter element 001 Store in Rx FIFO 0 if filter matches 010 Store in Rx FIFO 1 if filter matches 011 Reject ID if filter matches 100 Set priority if filter matches 101 Set priority and store in FIFO 0 if filter matches 110 Set priority and store in FIFO 1 if filter matches 111 Store into Rx Buffer or as debug message, configuration of SFT[1:0] ignored

Table 865. Standard Message ID Filter element descriptions (continued)

Field	Description
Bits 5:15 SFID1[10:0]	Standard Filter ID 1 First ID of standard ID filter element. When filtering for Rx Buffers or for debug messages this field defines the ID of a standard message to be stored. The received identifiers must match exactly, no masking mechanism is used.
Bits 21:31 SFID2[10:0]	Standard Filter ID 2 This bit field has a different meaning depending on the configuration of SFEC: 1) SFEC = '001'...'110' Second ID of standard ID filter element 2) SFEC = '111' Filter for Rx Buffers or for debug messages
	SFID2[10:9] decides whether the received message is stored into an Rx Buffer or treated as message A, B, or C of the debug message sequence. 00 Store message into an Rx Buffer 01 Debug Message A 10 Debug Message B 11 Debug Message C
	SFID2[8:6] is used to control the filter event pins at the Extension Interface. A one at the respective bit position enables generation of a pulse at the related filter event pin with the duration of one M_CAN Host clock period in case the filter matches.
	SFID2[5:0] defines the offset to the Rx Buffer Start Address RXBC.RBSA for storage of a matching message.

52.3.6.5 Extended Message ID Filter element

Up to 64 filter elements can be configured for 29-bit extended IDs. When accessing an Extended Message ID Filter element, its address is the Filter List Extended Start Address XIDFC[FLESA] plus two times the index of the filter element (0 to 63).

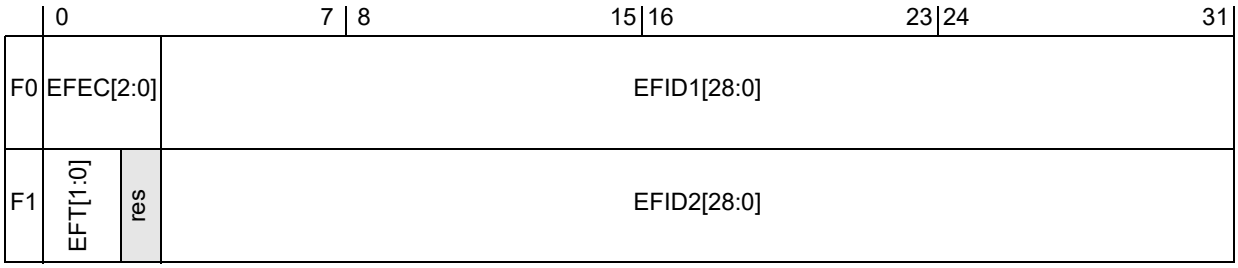


Figure 759. Extended Message ID Filter element

Table 866. Extended Message ID Filter element description

Field	Description
F0 bits 0:2 EFEC[2:0]	Extended Filter Element Configuration All enabled filter elements are used for acceptance filtering of extended frames. Acceptance filtering stops at the first matching enabled filter element or when the end of the filter list is reached. If EFEC = '100', '101', or '110' a match sets interrupt flag IR[HPM] and, if enabled, an interrupt is generated. In this case register HPMS is updated with the status of the priority match. 000 Disable filter element 001 Store in Rx FIFO 0 if filter matches 010 Store in Rx FIFO 1 if filter matches 011 Reject ID if filter matches 100 Set priority if filter matches 101 Set priority and store in FIFO 0 if filter matches 110 Set priority and store in FIFO 1 if filter matches 111 Store into Rx Buffer or as debug message, configuration of EFT[1:0] ignored
F0 bits 3:31 EFID1[28:0]	Extended Filter ID 1 First ID of extended ID filter element. When filtering for Rx Buffers or for debug messages this field defines the ID of an extended message to be stored. The received identifiers must match exactly, only XIDAM masking mechanism (refer to *Section 52.3.10.1.5*) is used.
F1 bits 0:1 EFT[1:0]	Extended Filter Type 00 Range filter from EFID1 to EFID1 (EFID2 ≥ EFID1) 01 Dual ID filter for EFID1 or EFID2 10 Classic filter: EFID1 = filter, EFID2 = mask 11 Range filter from EFID1 to EFID2 (EFID2 ≥ EFID1), XIDAM mask not applied
F1 bits 3:31 EFID2[28:0]	This bit field has a different meaning depending on the configuration of EFEC: 1) EFEC = '001'...'110' Second ID of extended ID filter element 2) EFEC = '111' Filter for Rx Buffers or for debug messages EFID2[10:9] decides whether the received message is stored into an Rx Buffer or treated as message A, B, or C of the debug message sequence. 00 Store message into an Rx Buffer 01 Debug Message A 10 Debug Message B 11 Debug Message C EFID2[8:6] is used to control the filter event pins at the Extension Interface. A 1 at the respective bit position enables generation of a pulse at the related filter event pin with the duration of one Host clock period in case the filter matches. EFID2[5:0] defines the offset to the Rx Buffer Start Address RXBC.RBSA for storage of a matching message.

52.3.7 M_CAN functional description

52.3.7.1 Operating modes

52.3.7.1.1 Software initialization

Software initialization is started by setting bit CCCR[INIT], either by software or by a hardware reset, when an uncorrected bit error was detected in the Message RAM, or by going Bus_Off. While CCCR[INIT] is set, message transfer from and to the CAN bus is stopped, the status of the CAN bus transmit output is recessive (HIGH). The counters of the Error Management Logic EML are unchanged. Setting CCCR[INIT] does not change any configuration register. Resetting CCCR[INIT] finishes the software initialization. Afterwards the Bit Stream Processor (BSP) synchronizes itself to the data transfer on the CAN bus by waiting for the occurrence of a sequence of 11 consecutive recessive bits (= Bus_Idle) before it can take part in bus activities and start the message transfer.

Access to the M_CAN configuration registers is only enabled when both bits CCCR[INIT] and CCCR[CCE] are set (protected write).

CCCR[CCE] can only be set/reset while CCCR[INIT] = ‘1’. CCCR[CCE] is automatically reset when CCCR[INIT] is reset.

The following registers are reset when CCCR[CCE] is set

- HPMS–High Priority Message Status
- RXF0S–Rx FIFO 0 Status
- RXF1S–Rx FIFO 1 Status
- TXFQS–Tx FIFO/Queue Status
- TXBRP–Tx Buffer Request Pending
- TXBTO–Tx Buffer Transmission Occurred
- TXBCF–Tx Buffer Cancellation Finished
- TXEFS–Tx Event FIFO Status

The Timeout Counter value TOCV[TOC] is preset to the value configured by TOCC[TOP] when CCCR[CCE] is set.

In addition the state machines of the Tx Handler and Rx Handler are held in idle state while CCCR[CCE] = ‘1’.

The following registers are only writable while CCCR[CCE] = ‘0’

- TXBAR–Tx Buffer Add Request
- TXBCR–Tx Buffer Cancellation Request

CCCR[TEST] and CCCR.MON can only be set by the Host while CCCR[INIT] = ‘1’ and CCCR[CCE] = ‘1’. Both bits may be reset at any time. CCCR.DAR can only be set/reset while CCCR[INIT] = ‘1’ and CCCR[CCE] = ‘1’.

The following sequence can be used as an example code for initializing the CAN module (M_CAN0 in example):

```
-- Setting the INIT bit high to start the initialization
     while (read_data != 0x003){
        write(M_CAN0,0xF,driver.regs.can0_cccr.addr,0x003);
           read_data = read(M_CAN0,0xF,driver.regs.can0_cccr.addr);
     };
```

```
-- Configuring the time stamp and timeout registers
     write(M_CAN0,0xF,driver.regs.can0_tscc.addr,0x000);
     write(M_CAN0,0xF,driver.regs.can0_tocc.addr,0x000);

-- Configuring the interrupt registers
     write(M_CAN0,0xF,driver.regs.can0_ie.addr,0xFFFFFFFF);
     write(M_CAN0,0xF,driver.regs.can0_ils.addr,0x00000002);
     write(M_CAN0,0xF,driver.regs.can0_ile.addr,0x00000003);

     write(M_CAN0,0xF,driver.regs.can0_gfc.addr,0x000);
     write(M_CAN0,0xF,driver.regs.can0_xidam.addr,0x00FFFFFF);

-- Writing the start addresses of all the M_CAN Buffers
     generate and drive M_CAN0 can0_rxf0c keeping {
                                          .f0sa == 0x100;
                                          .f0s  == 0x03;
                                          .f0wm == 0x01;
                                       };
     generate and drive M_CAN0 can0_rxf1c keeping {
                                             .f1sa == 0x200;
                                             .f1s  == 0x03;
                                             .f1wm == 0x01;
                                             };
     generate and drive M_CAN0 can0_sidfc keeping {
                                             .flssa == 0x000;
                                             .lss   == 0x80;
                                             };
     generate and drive M_CAN0 can0_xidfc keeping {
                                             .flesa == 0x080;
                                             .lse   == 0x40;
                                             };
     generate and drive M_CAN0 can0_txbc  keeping {
                                             .tbsa == 0x300;
                                             .ndtb == 0x020;
                                             .tfqs == 0x1D;
                                             };
     generate and drive M_CAN0 can0_txefc keeping {
                                          .efsa == 0x380;
                                          .efs == 0x020;
                                          .efwm == 0x00;
                                          };
-- Writing to the RAM for filter configuration
=================================================
-- Writing to TX Buffer
```

```
      write(RAM,0xF,0xC00,0x04040000);
      write(RAM,0xF,0xC04,0x10830000);
      write(RAM,0xF,0xC08,0x00010001);
      write(RAM,0xF,0xC10,0x50000002);
      write(RAM,0xF,0xC14,0x11830000);
      write(RAM,0xF,0xC18,0x00070102);

-- Writing to s-filter
      write(RAM,0xF,0x000,0x58020004);
      write(RAM,0xF,0x004,0x08000005);
      write(RAM,0xF,0x008,0x58060007);
      write(RAM,0xF,0x00C,0x08030008);
      write(RAM,0xF,0x010,0x880E07FB);
      write(RAM,0xF,0x014,0x980D07FE);
      write(RAM,0xF,0x018,0x68100100);
      write(RAM,0xF,0x01C,0x70120200);
      write(RAM,0xF,0x020,0x981F07F6);
      write(RAM,0xF,0x024,0x50140014);
      write(RAM,0xF,0x028,0x58160017);
      write(RAM,0xF,0x02C,0x08110019);
      write(RAM,0xF,0x030,0x4A22001A);
      write(RAM,0xF,0x034,0xA01A07FF);
      write(RAM,0xF,0x038,0x5BFF07E3);
      write(RAM,0xF,0x03C,0x080007FF);

-- Writing to x-filter
      write(RAM,0xF,0x200,0x201ABCD9);
      write(RAM,0xF,0x204,0x001ABCE5);
      write(RAM,0xF,0x208,0x403BCDEF);
      write(RAM,0xF,0x20C,0x003BCDEF);
      write(RAM,0xF,0x210,0x605CDEF2);
      write(RAM,0xF,0x214,0x405CDEF0);
      write(RAM,0xF,0x218,0x407DEEFF);
      write(RAM,0xF,0x21C,0x007DEF01);
      write(RAM,0xF,0x220,0x209ABCDE);
      write(RAM,0xF,0x224,0x009ABCDF);
      write(RAM,0xF,0x228,0xA0BBCDEF);
      write(RAM,0xF,0x22C,0x40BBCDEE);
      write(RAM,0xF,0x230,0x20DCDEF0);
      write(RAM,0xF,0x234,0x9FFFFFFF);
      write(RAM,0xF,0x238,0x40000000);
      write(RAM,0xF,0x23C,0x1FFFFFFF);
================================================
write(M_CAN0,0xF,driver.regs.can0_nbtp.addr,0x00002303);
```

```
print "Configuration done";
read_data = 0x01;
while (read_data != 0x00){
   write(M_CAN0,0xF,driver.regs.can0_cccr.addr, 0x000);
      wait [1];
      read_data = read(M_CAN0,0xF,driver.regs.can0_cccr.addr);
   };

print "Initialization done";
```

52.3.7.2 Normal operation

Once the M_CAN is initialized and CCCR[INIT] is reset to zero, the M_CAN synchronizes itself to the CAN bus and is ready for communication.

After passing the acceptance filtering, the received messages including Message ID and DLC are stored into Rx FIFO 0 or Rx FIFO 1.

For messages to be transmitted dedicated Tx Buffers or a Tx FIFO, or both, or a Tx Queue can be initialized or updated. Automated transmission on reception of remote frames is not implemented.

52.3.7.3 CAN FD Operation

There are two variants in the CAN FD frame transmission, first the CAN FD frame without bit rate switching. The second variant is the CAN FD frame where control field, data field, and CRC field are transmitted with a higher bit rate than the beginning and the end of the frame.

The previously reserved bit in CAN frames with 11-bit identifiers and the first previously reserved bit in CAN frames with 29-bit identifiers will now be decoded as FDF bit. FDF = *recessive* signifies a CAN FD frame, FDF = *dominant* signifies a Classic CAN frame. In a CAN FD frame, the two bits following FDF, res and BRS, decide whether the bit rate inside of this CAN FD frame is switched. A CAN FD bit rate switch is signified by res = *dominant* and BRS = *recessive*. The coding of res = *recessive* is reserved for future expansion of the protocol. In case the M_CAN receives a frame with FDF = *recessive* and res = *recessive*, it will signal a Protocol Exception Event by setting bit PSR.PXE. When Protocol Exception Handling is enabled (CCCR.PXHD = ‘0’), this causes the operation state to change from Receiver (PSR.ACT = “10”) to Integrating (PSR.ACT = “00”) at the next sample point. In case Protocol Exception Handling is disabled (CCCR.PXHD = ‘1’), the M_CAN will treat a *recessive* res bit as an form error and will respond with an error frame.

CAN FD operation is enabled by programming CCCR.FDOE. In case CCCR.FDOE = ‘1’, transmission and reception of CAN FD frames is enabled. Transmission and reception of Classic CAN frames is always possible. Whether a CAN FD frame or a Classic CAN frame is transmitted can be configured via bit FDF in the respective Tx Buffer element. With CCCR.FDOE = ‘0’, received frames are interpreted as Classic CAN frames, which leads to the transmission of an error frame when receiving a CAN FD frame. When CAN FD operation is disabled, no CAN FD frames are transmitted even if bit FDF of a Tx Buffer element is set. CCCR.FDOE and CCCR.BRSE can only be changed while CCCR.INIT and CCCR.CCE are both set. CCCR.FDOE and CCCR.BRSE can only be changed while CCCR.INIT and CCCR.CCE are both set.

With CCCR.FDOE = ‘0’, the setting of bits FDF and BRS is ignored and frames are transmitted in Classic CAN format. With CCCR.FDOE = ‘1’ and CCCR.BRSE = ‘0’, only bit FDF of a Tx Buffer element is evaluated. With CCCR.FDOE = ‘1’ and CCCR.BRSE = ‘1’, transmission of CAN FD frames with bit rate switching is enabled. All Tx Buffer elements with bits FDF and BRS set are transmitted in CAN FD format with bit rate switching.

A mode change during CAN operation is only recommended under the following conditions:

- The failure rate in the CAN FD data phase is significantly higher than in the CAN FD arbitration phase. In this case disable the CAN FD bit rate switching option for transmissions
- During system startup all nodes are transmitting Classic CAN messages until it is verified that they are able to communicate in CAN FD format. If this is true, all nodes switch to CAN FD operation
- Wake-up messages in CAN Partial Networking have to be transmitted in Classic CAN format
- End-of-line programming in case not all nodes are CAN FD capable. Non CAN FD nodes are held in Silent mode until programming has completed. Then all nodes switch back to Classic CAN communication

In the CAN FD format, the coding of the DLC differs from the standard CAN format. The DLC codes 0 to 8 have the same coding as in standard CAN, the codes 9 to 15, which in standard CAN all code a data field of 8 bytes, are coded according to *Table 867*.

Table 867. Coding of DLC in CAN FD

DLC	9	10	11	12	13	14	15
Number of data bytes	12	16	20	24	32	48	64

In CAN FD frames, the bit timing will be switched inside the frame, after the BRS (Bit Rate Switch) bit, if this bit is *recessive*. Before the BRS bit, in the CAN FD arbitration phase, the nominal CAN bit timing is used as defined by the Nominal Bit Timing & Prescaler Register NBTP. In the following CAN FD data phase, the data phase bit timing is used as defined by the Data Bit Timing & Prescaler Register DBTP. The bit timing is switched back from the data phase timing at the CRC delimiter or when an error is detected, whichever occurs first.

The maximum configurable bit rate in the CAN FD data phase depends on the CAN clock frequency (CAN clock). Example: with a CAN clock frequency of 20MHz and the shortest configurable bit time of 4 tq, the bit rate in the data phase is 5 Mbit/s.

In both data frame formats, CAN FD and CAN FD with bit rate switching, the value of the bit ESI (Error Status Indicator) is determined by the transmitter’s error state at the start of the transmission. If the transmitter is error passive, ESI is transmitted *recessive*, else it is transmitted *dominant*.

52.3.7.4 Transceiver Delay Compensation

During the data phase of a CAN FD transmission only one node is transmitting, all others are receivers. The length of the bus line has no impact. When transmitting via CAN Tx output pin the M_CAN the transmitted data from its local CAN transceiver via CAN Rx input pin. The received data is delayed by the transmitter delay. In case this delay is greater than NTSEG1 (time segment before sample point), a bit error is detected. In order to enable a data phase bit time that is even shorter than the transmitter delay, the delay compensation is

introduced. Without transmitter delay compensation, the bit rate in the data phase of a CAN FD frame is limited by the transmitter delay.

52.3.7.4.1 Description

The CAN FD protocol unit has implemented a delay compensation mechanism to compensate the CAN transceiver's loop delay, thereby enabling transmission with higher bit rates during the CAN FD data phase independent of the delay of a specific CAN transceiver.

To check for bit errors during the data phase of transmitting nodes, the delayed transmit data is compared against the received data at the Secondary Sample Point SSP. If a bit error is detected, the transmitter will react on this bit error at the next following regular sample point. During arbitration the transmitter delay compensation enables configurations where the data bit time is shorter than the transmitter delay, it is described in detail in the new ISO11898-1. It is enabled by setting bit DBTP.TDC.phase the delay compensation is always disabled.

The received bit is compared against the transmitted bit at the SSP. The SSP position is defined as the sum of the measured delay from the M_CAN's Tx output through the transceiver to the CAN Rx input plus the transmitter delay compensation offset as configured by TDCR.TDCO. The transmitter delay compensation offset is used to adjust the position of the SSP inside the received bit (for example half of the bit time in the data phase). The position of the secondary sample point is rounded down to the next integer number of mtq.

PSR.TDCV shows the actual transmitter delay compensation value. PSR.TDCV is cleared when CCCR.INIT is set and is updated at each transmission of an FD frame while DBTP.TDC is set.

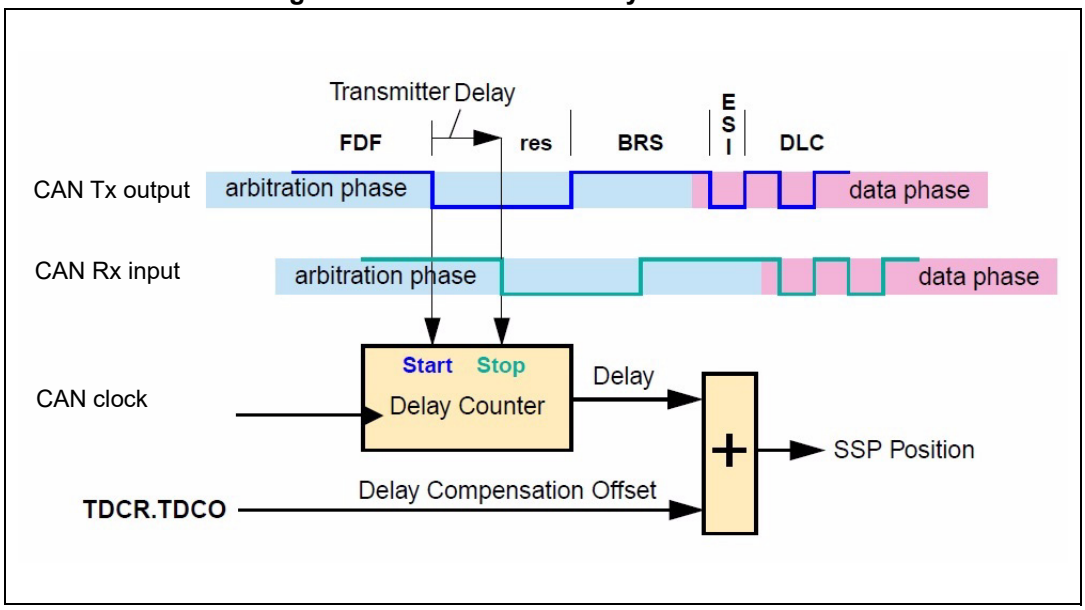
The following boundary conditions have to be considered for the transmitter delay compensation implemented in the M_CAN:

- The sum of the measured delay from CAN Tx output to CAN Rx input and the configured transmitter delay compensation offset TDCR.TDCO has to be less than 6 bit times in the data phase
- The sum of the measured delay from CAN Tx output to CAN Rx input and the configured transmitter delay compensation offset TDCR.TDCO has to be less than or equal to 127 mtq. In case this sum exceeds 127 mtq, the maximum value of 127 mtq is used for transmitter delay compensation
- The data phase ends at the sample point of the CRC delimiter, that stops checking of receive bits at the SSPs

Figure 760 below describes how the Transceiver Delay Compensation measurement.

If transmitter delay compensation is enabled by programming DBTP.TDC = '1', the measurement is started within each transmitted CAN FD frame at the falling edge of bit FDF to bit res. The measurement is stopped when this edge is seen at the receive input CAN Rx input of the transmitter. The resolution of this measurement is one mtq.

Figure 760. Transceiver Delay measurement



To avoid that a dominant glitch inside the received FDF bit ends the delay compensation measurement before the falling edge of the received res bit, resulting in a too early SSP position, the use of a transmitter delay compensation filter window can be enabled by programming TDCR.TDCF. This defines a minimum value for the SSP position. Dominant edges on CAN Rx input, that would result in an earlier SSP position are ignored for transmitter delay measurement. The measurement is stopped when the SSP position is at least TDCR.TDCF AND CAN Rx input is low.

52.3.7.5 Restricted Operation Mode

In Restricted Operation Mode the node is able to receive data and remote frames and to give acknowledge to valid frames, but it does not send data frames, remote frames, active error frames, or overload frames. In case of an error condition or overload condition, it does not send dominant bits, instead it waits for the occurrence of bus idle condition to resynchronize itself to the CAN communication. The error counters are not incremented. The Host can set the M_CAN into Restricted Operation mode by setting bit CCCR.ASM. The bit can only be set by the Host when both CCCR.CCE and CCCR.INIT are set to ‘1’. The bit can be reset by the Host at any time.

Restricted Operation Mode is automatically entered when the Tx Handler was not able to read data from the Message RAM in time. To leave Restricted Operation Mode, the Host CPU has to reset CCCR.ASM.

The Restricted Operation Mode can be used in applications that adapt themselves to different CAN bit rates. In this case the application tests different bit rates and leaves the Restricted Operation Mode after it has received a valid frame.

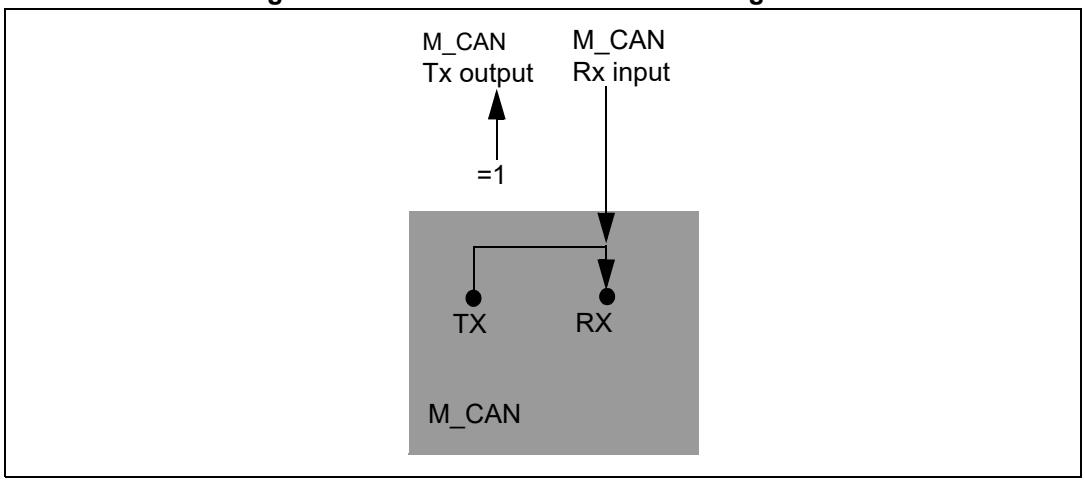
If the M_CAN is connected to a Clock Calibration on CAN Unit, CCCR.ASM is controlled by “calibration OK” input. In case “calibration OK” switches to ‘0’, bit CCCR.ASM is set. When “calibration OK” switches back to ‘1’, bit CCCR.ASM returns to the previously written value. The state of CCCR.ASM is the written value while “calibration OK” input is at ‘1’. The input is hardwired to ‘1’ when there is no Clock Calibration on CAN Unit connected.

52.3.7.6 Bus monitoring mode

The M_CAN is set in Bus Monitoring Mode by programming CCCR[MON] to one. In Bus Monitoring Mode (refer to ISO11898-1, 10.12 Bus monitoring), the M_CAN is able to receive valid data frames and valid remote frames, but cannot start a transmission. In this mode, it sends only recessive bits on the CAN bus. If the M_CAN is required to send a dominant bit (ACK bit, overload flag, active error flag), the bit is rerouted internally so that the M_CAN monitors this dominant bit, although the CAN bus may remain in recessive state. In Bus Monitoring Mode, register TXBRP is held in reset state. The Bus Monitoring Mode can be used to analyze the traffic on a CAN bus without affecting it by the transmission of dominant bits.

Figure 761 shows the connection of M_CAN Tx and Rx signals to the M_CAN in Bus Monitoring Mode.

Figure 761. Pin control in Bus Monitoring mode



52.3.7.7 Disabled automatic retransmission

According to the CAN Specification (refer to ISO11898-1, 6.3.3 Recovery Management), the M_CAN provides means for automatic retransmission of frames that have lost arbitration or that have been disturbed by errors during transmission. By default automatic retransmission is enabled. To support time-triggered communication as described in ISO 11898-1, chapter 9.2, the automatic retransmission may be disabled via CCCR[DAR].

52.3.7.7.1 Frame transmission in DAR mode

In DAR mode, all transmissions are automatically cancelled after they started on the CAN bus. A Tx Buffer's Tx Request Pending bit TXBRP[TRPx] is reset after successful

transmission, when a transmission has not yet been started at the point of cancellation, has been aborted due to lost arbitration, or when an error occurred during frame transmission.

- Successful transmission:
  - Corresponding Tx Buffer Transmission Occurred bit TXBTO[TOx] set
  - Corresponding Tx Buffer Cancellation Finished bit TXBCF[CFx] not set
- Successful transmission in spite of cancellation:
  - Corresponding Tx Buffer Transmission Occurred bit TXBTO[TOx] set
  - Corresponding Tx Buffer Cancellation Finished bit TXBCF[CFx] set
- Arbitration lost or frame transmission disturbed:
  - Corresponding Tx Buffer Transmission Occurred bit TXBTO[TOx] not set
  - Corresponding Tx Buffer Cancellation Finished bit TXBCF[CFx] set

In case of a successful frame transmission, and if storage of Tx events is enabled, a Tx Event FIFO element is written with Event Type ET = ‘10’ (transmission in spite of cancellation).

52.3.7.8 Power down (Sleep mode)

The M_CAN can be set into power down mode controlled by input signal clock stop request or via CC Control Register CCCR[CSR]. As long as the clock stop request signal is active, bit CCCR[CSR] is read as one. When all pending transmission requests have completed, the M_CAN waits until bus idle state is detected. Then the M_CAN sets then CCCR[INIT] to one to prevent any further CAN transfers. Now the M_CAN acknowledges that it is ready for power down by setting clock stop acknowledge output signal to one and CCCR[CSA] to one. In this state, before the clocks are switched off, further register accesses can be made. A write access to CCCR[INIT] will have no effect. Now the module clock inputs (CAN clock and Host clock) may be switched off. To leave power down mode, the application has to turn on the module clocks before resetting signal clock stop request signal resp. CC Control Register flag CCCR[CSR]. The M_CAN will acknowledge this by resetting clock stop acknowledge output signal and resetting CCCR[CSA]. Afterwards, the application can restart CAN communication by resetting bit CCCR[INIT].

52.3.7.9 Test modes

To enable write access to register TEST (refer to *Section 52.3.5.2.4: Test Register (TEST)*), bit CCCR[TEST] has to be set to one. This allows the configuration of the test modes and test functions.

Four output functions are available for the CAN Tx output pin by programming TEST.TX. Additionally to its default function – the serial data output – it can drive the CAN Sample Point signal to monitor the M_CAN’s bit timing and it can drive constant dominant or recessive values. The actual value at M_CAN Rx pin can be read from TEST[RX]. Both functions can be used to check the CAN bus physical layer.

Due to the synchronization mechanism between CAN clock and Host clock domain, there may be a delay of several Host clock periods between writing to TEST[TX] until the new configuration is visible at output Tx pin. This applies also when reading input Rx pin via TEST[RX].

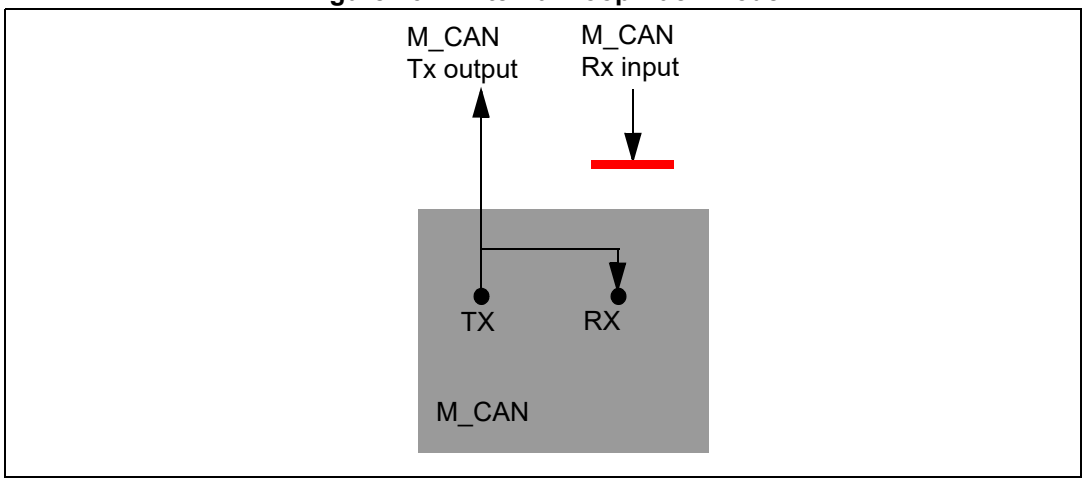
Note: Test modes should be used for production tests or self test only. The software control for Tx pin interferes with all CAN protocol functions. It is not recommended to use test modes for application.

52.3.7.9.1 External Loop Back mode

The M_CAN can be set in External Loop Back mode by programming TEST.LBCK to one. In Loop Back mode, the M_CAN treats its own transmitted messages as received messages and stores them (if they pass acceptance filtering) into Rx FIFOs. *Figure 762* shows the connection of Tx and Rx signals to the M_CAN in External Loop Back mode.

This mode is provided for hardware self-test. To be independent from external stimulation, the M_CAN ignores acknowledge errors (recessive bit sampled in the acknowledge slot of a data/remote frame) in Loop Back mode. In this mode the M_CAN performs an internal feedback from its Tx output to its Rx input. The actual value of the CAN Rx input pin is disregarded by the M_CAN. The transmitted messages can be monitored at the CAN Tx output pin.

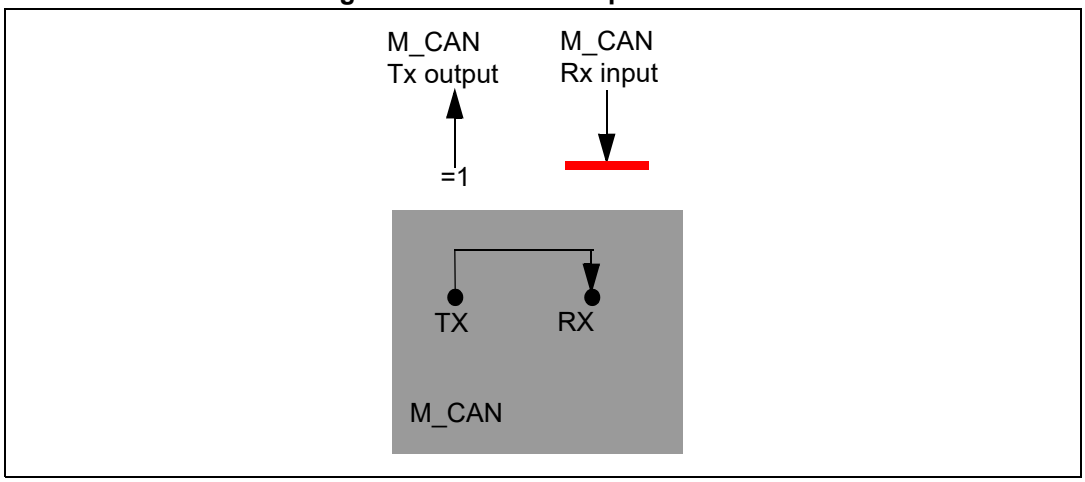
Figure 762. External Loop Back mode



52.3.7.9.2 Internal Loop Back mode

Internal Loop Back mode is entered by programming bits TEST[LBCK] and CCCR[MON] to one. This mode can be used for a “Hot Selftest”, meaning the M_CAN can be tested without affecting a running CAN system connected to the CAN Tx output and Rx input pins. In this mode M_CAN Tx pin is disconnected from the M_CAN and M_CAN Tx pin is held recessive. *Figure 763* shows the connection of M_CAN Tx pin and Rx to the M_CAN in case of Internal Loop Back mode.

Figure 763. Internal loop back mode



52.3.8 Timestamp generation

For timestamp generation the M_CAN supplies a 16-bit wrap-around counter. A prescaler TSCC[TCP] can be configured to clock the counter in multiples of CAN bit times (1…16). The counter is readable via TSCV[TSC]. A write access to register TSCV resets the counter to zero. When the timestamp counter wraps around interrupt flag IR[TSW] is set.

On start of frame reception / transmission the counter value is captured and stored into the timestamp section of an Rx Buffer /Rx FIFO (RXTS[15:0]) or Tx Event FIFO (TXTS[15:0]) element.

By programming bit TSCC.TSS, a 16-bit timestamp can be used.

52.3.9 Timeout counter

To signal timeout conditions for Rx FIFO 0, Rx FIFO 1, and the Tx Event FIFO the M_CAN supplies a 16-bit Timeout Counter. It operates as down-counter and uses the same prescaler controlled by TSCC.TCP as the Timestamp Counter. The Timeout Counter is configured via register TOCC. The actual counter value can be read from TOCV[TOC]. The Timeout Counter can only be started while CCCR[INIT] = ‘0’. It is stopped when CCCR[INIT] = ‘1’, for example when the M_CAN enters Bus_Off state.

The operation mode is selected by TOCC[TOS]. When operating in Continuous mode, the counter starts when CCCR[INIT] is reset. A write to TOCV presets the counter to the value configured by TOCC.TOP and continues down-counting. When the Timeout Counter is controlled by one of the FIFOs, an empty FIFO presets the counter to the value configured by TOCC[TOP]. Down-counting is started when the first FIFO element is stored. Writing to TOCV has no effect.

When the counter reaches zero, interrupt flag IR[TOO] is set. In Continuous mode, the counter is immediately restarted at TOCC[TOP].

Note: The clock signal for the Timeout Counter is derived from the CAN core’s sample point signal. Therefore the point in time where the Timeout Counter is decremented may vary due to the synchronization / re-synchronization mechanism of the CAN core. If the baud rate switch feature in CAN FD is used, the timeout counter is clocked differently in arbitration and data field.

52.3.10 Rx handling

The Rx Handler controls the acceptance filtering, the transfer of received messages to the Rx Buffers or to one of the two Rx FIFOs, as well as the Rx FIFO's Put and Get Indexes.

52.3.10.1 Acceptance filtering

The M_CAN offers the possibility to configure two sets of acceptance filters, one for standard identifiers and one for extended identifiers. These filters can be assigned to an Rx Buffer or to Rx FIFO 0,1. For acceptance filtering each list of filters is executed from element #0 until the first matching element. Acceptance filtering stops at the first matching element. The following filter elements are not evaluated for this message.

The main features are:

- Each filter element can be configured as
  - Range filter (from - to)
  - Filter for one or two dedicated IDs
  - Classic bit mask filter
- Each filter element is configurable for acceptance or rejection filtering
- Each filter element can be enabled / disabled individually
- Filters are checked sequentially, execution stops with the first matching filter element

Related configuration registers are:

- Global Filter Configuration (GFC)
- Standard ID Filter Configuration (SIDFC)
- Extended ID Filter Configuration (XIDFC)
- Extended ID AND Mask (XIDAM)

Depending on the configuration of the filter element (SFEC/EFEC) a match triggers one of the following actions:

- Store received frame in FIFO 0 or FIFO 1
- Store received frame in Rx Buffer
- Store received frame in Rx Buffer and generate pulse at filter event pin
- Reject received frame
- Set High Priority Message interrupt flag IR[HPM]
- Set High Priority Message interrupt flag IR[HPM] and store received frame in FIFO 0 or FIFO 1

Acceptance filtering is started after the complete identifier has been received. After acceptance filtering has completed, and if a matching Rx Buffer or Rx FIFO has been found, the Message Handler starts writing the received message data in portions of 32 bit to the matching Rx Buffer or Rx FIFO. If the CAN protocol controller has detected an error condition (such as CRC error), this message is discarded with the following impact on the affected Rx Buffer or Rx FIFO:

Rx Buffer

New Data flag of matching Rx Buffer is not set, but Rx Buffer (partly) overwritten with received data. For error type see PSR.LEC respectively PSR.DLEC.

Rx FIFO

Put index of matching Rx FIFO is not updated, but related Rx FIFO element (partly) overwritten with received data. For error type see PSR.LEC respectively PSR.DLEC. In case the matching Rx FIFO is operated in overwrite mode, the boundary conditions described in *Section 52.3.10.2.2: Rx FIFO Overwrite Mode* have to be considered.

Note: When an accepted message is written to one of the two Rx FIFOs, or into a Rx Buffer, the unmodified received identifier is stored independent of the filter(s) used. The result of the acceptance filter process strongly depends on the sequence of configured filter elements.

52.3.10.1.1 Range filter

The filter matches for all received frames with Message IDs in the range defined by SFID1/SFID2 resp. EFID1/EFID2. There are two possibilities when range filtering is used together with extended frames:

- EFT = '00': The Message ID of received frames is ANDed with the Extended ID AND Mask (XIDAM) before the range filter is applied
- EFT = '11': The Extended ID AND Mask (XIDAM) is not used for range filtering

52.3.10.1.2 Filter for specific IDs

A filter element can be configured to filter for one or two specific Message IDs. To filter for one specific Message ID, the filter element has to be configured with SFID1 = SFID2 resp. EFID1 =EFID2.

52.3.10.1.3 Classic Bit Mask Filter

Classic bit mask filtering is intended to filter groups of Message IDs by masking single bits of a received Message ID. With classic bit mask filtering SFID1/EFID2 is used as Message ID filter, while SFID2/EFID2 is used as filter mask.

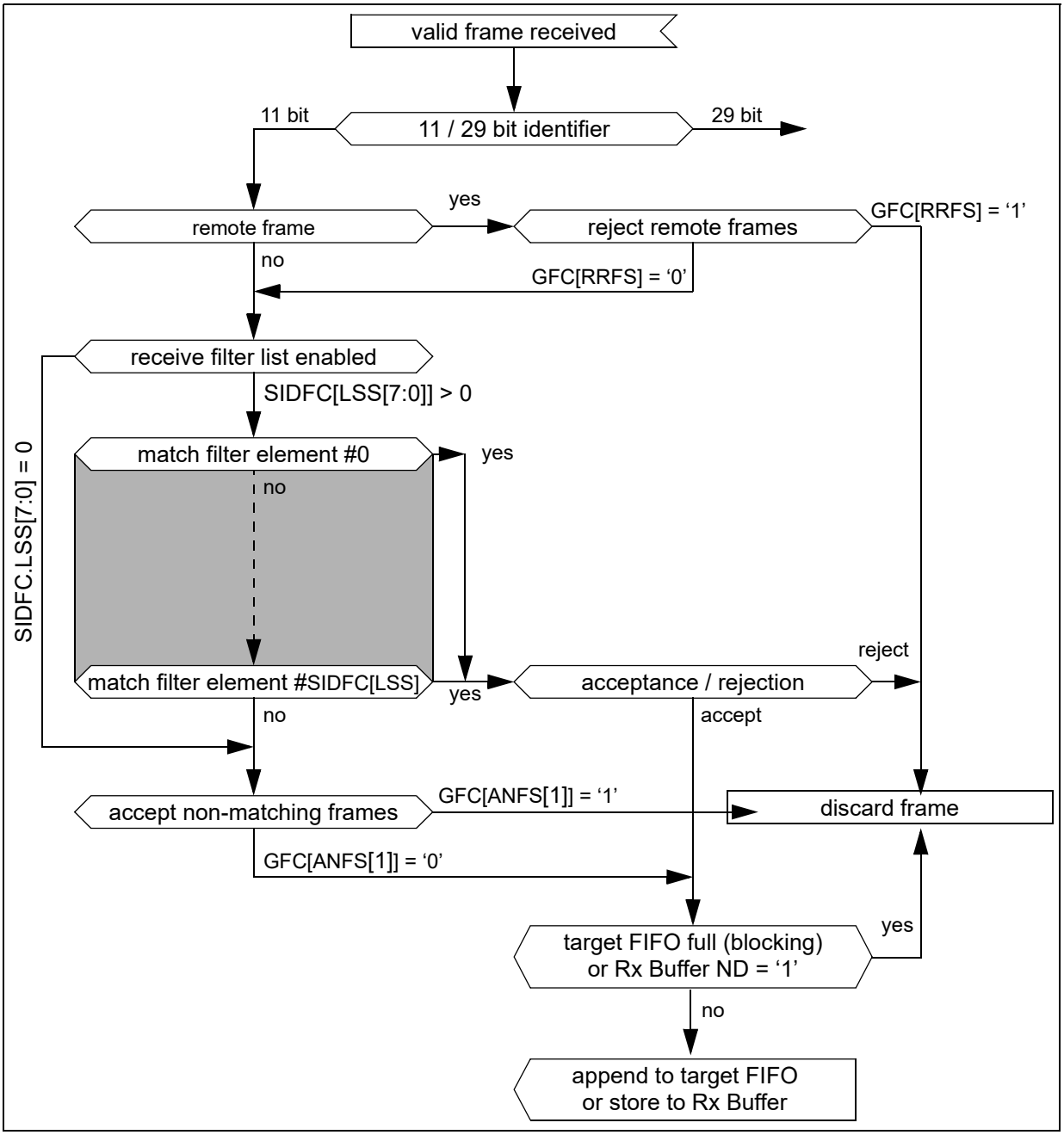
A zero bit at the filter mask will mask out the corresponding bit position of the configured ID filter, for example the value of the received Message ID at that bit position is not relevant for acceptance filtering. Only those bits of the received Message ID where the corresponding mask bits are one are relevant for acceptance filtering. In case all mask bits are one, a match occurs only when the received Message ID and the Message ID filter are identical. If all mask bits are zero, all Message IDs match.

52.3.10.1.4 Standard message ID filtering

Figure 758 shows the flow for standard message ID (11-bit Identifier) filtering. The Standard Message ID Filter element is described in *Section 52.4.6.4: Standard Message ID Filter element*.

Controlled by the Global Filter Configuration GFC and the Standard ID Filter Configuration SIDFC Message ID, Remote Transmission Request bit (RTR), and the Identifier Extension bit (IDE) of received frames are compared against the list of configured filter elements.

Figure 764. Standard Message ID Filter Path



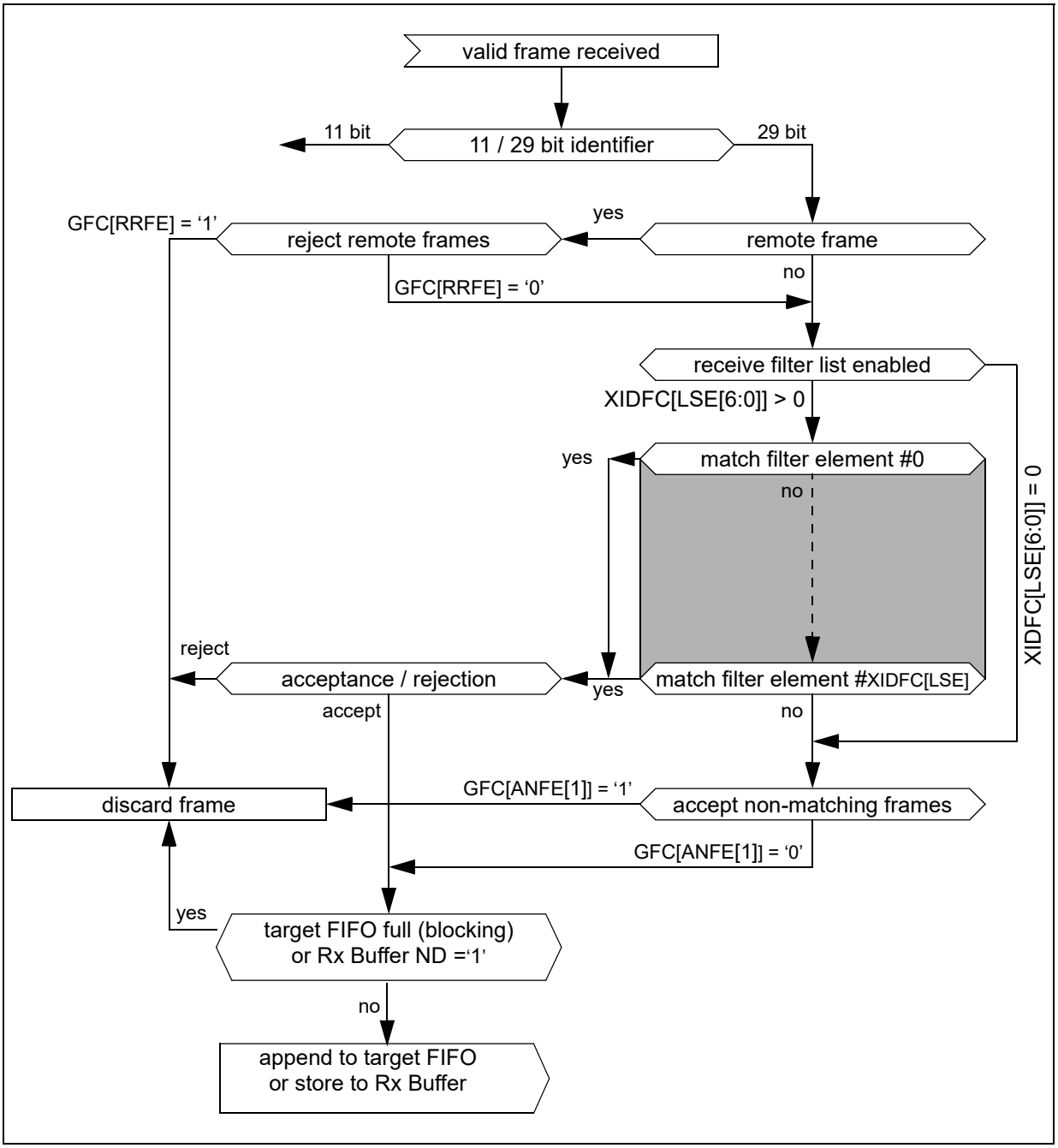
52.3.10.1.5 Extended message ID filtering

Figure 765 below shows the flow for extended Message ID (29-bit Identifier) filtering. The Extended Message ID Filter element is described in *Section 52.3.5.2.21: Extended ID Filter Configuration Register (XIDFC)*.

Remote Transmission Request bit (RTR), and the Identifier Extension bit (IDE) of received frames are compared against the list of configured filter elements which is controlled by the

Global Filter Configuration (GFC) and the Extended ID Filter Configuration (XIDFC) Message ID. The Extended ID AND Mask (XIDAM) is ANDed with the received identifier before the filter list is executed.

Figure 765. Extended Message ID Filter Path



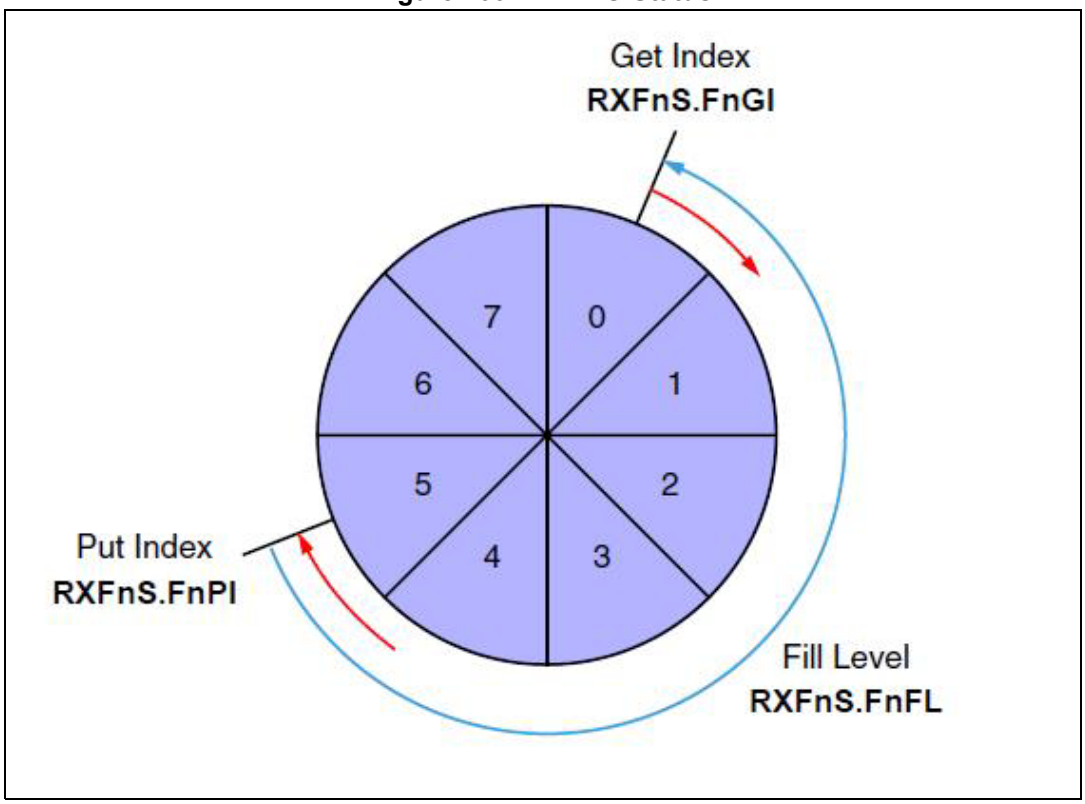
52.3.10.2 Rx FIFOs

Rx FIFO 0 and Rx FIFO 1 can be configured to hold up to 64 elements each. Configuration of the two Rx FIFOs is done via registers RXF0C and RXF1C.

Received messages that passed acceptance filtering are transferred to the Rx FIFO as configured by the matching filter element. For a description of the filter mechanisms available for Rx FIFO 0 and Rx FIFO 1 refer to *Section 52.3.10.1: Acceptance filtering*. The Rx FIFO element is described in *Section 52.3.6.1: Rx Buffer and FIFO element*.

To avoid an Rx FIFO overflow, the Rx FIFO watermark can be used. When the Rx FIFO fill level reaches the Rx FIFO watermark configured by RXFnC.FnWM, interrupt flag IR.RFnW is set. When the Rx FIFO Put Index reaches the Rx FIFO Get Index an Rx FIFO Full condition is signalled by RXFnS.FnF. In addition interrupt flag IR.RFnF is set.

Figure 766. Rx FIFO Status



When reading from an Rx FIFO, Rx FIFO Get Index RXFnS.FnGI × FIFO Element Size has to be added to the corresponding Rx FIFO start address RXFnC.FnSA.

Table 868. Rx Buffer / FIFO Element Size

RXESC.RBDS[2:0] RXESC.FnDS[2:0]	Data Field [bytes]	FIFO Element Size [RAM words]
000	8	4
001	12	5
010	16	6
011	20	7
100	24	8
101	32	10

Table 868. Rx Buffer / FIFO Element Size (continued)

RXESC.RBDS[2:0] RXESC.FnDS[2:0]	Data Field [bytes]	FIFO Element Size [RAM words]
110	48	14
111	64	18

52.3.10.2.1 Rx FIFO Blocking Mode

The Rx FIFO blocking mode is configured by RXFnC.FnOM = ‘0’. This is the default operation mode for the Rx FIFOs.

When an Rx FIFO full condition is reached (RXFnS.FnPI = RXFnS.FnGI), no further messages are written to the corresponding Rx FIFO until at least one message has been read out and the Rx FIFO Get Index has been incremented. An Rx FIFO full condition is signalled by RXFnS.FnF = ‘1’. In addition interrupt flag IR.RFnF is set.

In case a message is received while the corresponding Rx FIFO is full, this message is discarded and the message lost condition is signalled by RXFnS.RFnL = ‘1’. In addition interrupt flag IR.RFnL is set.

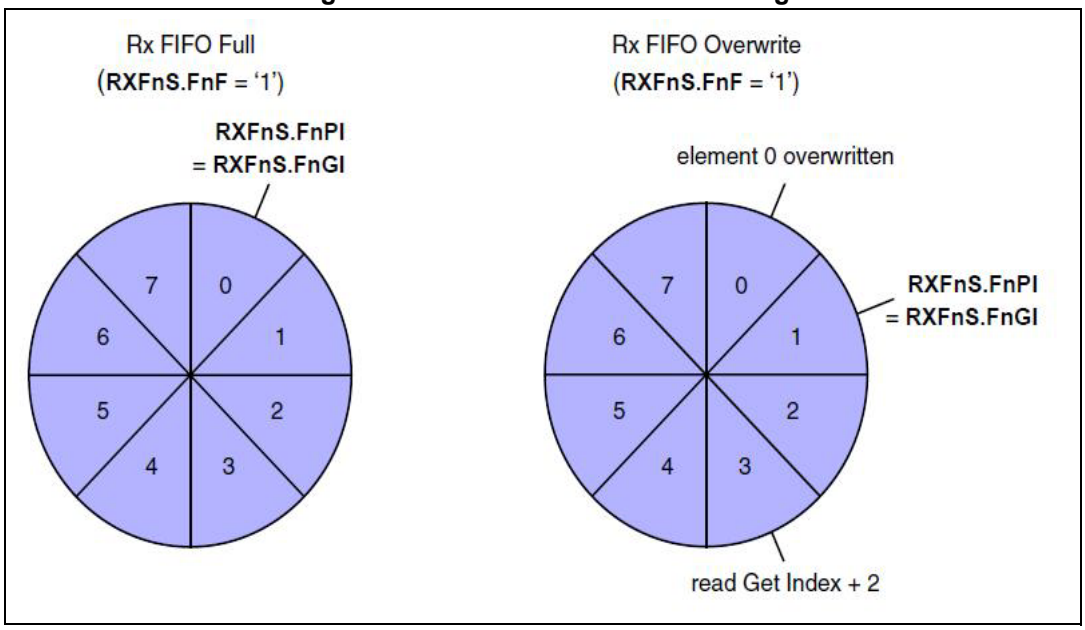
52.3.10.2.2 Rx FIFO Overwrite Mode

The Rx FIFO overwrite mode is configured by RXFnC.FnOM = ‘1’.

When an Rx FIFO full condition (RXFnS.FnPI = RXFnS.FnGI) is signalled by RXFnS.FnF = ‘1’, the next message accepted for the FIFO will overwrite the oldest FIFO message. Put and get index are both incremented by one.

When an Rx FIFO is operated in overwrite mode and an Rx FIFO full condition is signalled, reading of the Rx FIFO elements should start at least at get index + 1. The reason for that is, that it might happen, that a received message is written to the Message RAM (put index) while the CPU is reading from the Message RAM (get index). In this case inconsistent data may be read from the respective Rx FIFO element. Adding an offset to the get index when reading from the Rx FIFO avoids this problem. The offset depends on how fast the CPU accesses the Rx FIFO. *Figure 767* shows an offset of two with respect to the get index when reading the Rx FIFO. In this case the two messages stored in element 1 and 2 are lost.

Figure 767. Rx FIFO Overflow Handling



After reading from the Rx FIFO, the number of the last element read has to be written to the Rx FIFO Acknowledge Index RXFnA.FnA. This increments the get index to that element number. In case the put index has not been incremented to this Rx FIFO element, the Rx FIFO full condition is reset (RXFnS.FnF = '0').

52.3.11 Dedicated Rx Buffers

The M_CAN supports up to 64 dedicated Rx Buffers. The start address of the dedicated Rx Buffer section is configured via RXBC.RBSA.

For each Rx Buffer a Standard or Extended Message ID Filter Element with SFEC / EFEC = '111' and SFID2 / EFID2[10:9] = '00' has to be configured (refer to *Section 52.3.6.4: Standard Message ID Filter element* and *Section 52.3.6.5: Extended Message ID Filter element*).

After a received message has been accepted by a filter element, the message is stored into the Rx Buffer in the Message RAM referenced by the filter element. The format is the same as for an Rx FIFO element. In addition the flag IR.DRX (Message stored in Dedicated Rx Buffer) in the interrupt register is set.

Table 869. Example Filter Configuration for Rx buffers

Filter Element	SFID1[10:0] EFID1[28:0]	SFID2[10:9] EFID2[10:9]	SFID2[5:0] EFID2[5:0]
0	ID debug message 1	00	00 0000
1	ID debug message 2	00	00 0001
2	ID debug message 3	00	00 0010

After the last word of a matching received message has been written to the Message RAM, the respective New Data flag in register NDAT1,2 is set. As long as the New Data flag is set,

the respective Rx Buffer is locked against updates from received matching frames. The New Data flags have to be reset by the Host by writing a ‘1’ to the respective bit position.

While an Rx Buffer’s New Data flag is set, a Message ID Filter Element referencing this specific Rx Buffer will not match, causing the acceptance filtering to continue. Following Message ID Filter Elements may cause the received message to be stored into another Rx Buffer, or into an Rx FIFO, or the message may be rejected, depending on filter configuration.

52.3.11.1 Rx Buffer Handling

- Reset interrupt flag IR.DRX
- Read New Data registers
- Read messages from Message RAM
- Reset New Data flags of processed messages

52.3.12 Debug on CAN Support

Debug messages are stored into Rx Buffers. For debug handling three consecutive Rx buffers (for example #61, #62, #63) have to be used for storage of debug messages A, B, and C. The format is the same as for an Rx Buffer or an Rx FIFO element (refer to *Section 52.3.6.1: Rx Buffer and FIFO element*).

Advantage: Fixed start address for the DMA transfers (relative to RXBC.RBSA), no additional configuration required.

For filtering of debug messages Standard / Extended Filter Elements with SFEC / EFEC = ‘111’ have to be set up. Messages matching these filter elements are stored into the Rx Buffers addressed by SFID2 / EFID2[5:0].

After message C has been stored, the DMA request output is activated and the three messages can be read from the Message RAM under DMA control. The RAM words holding the debug messages will not be changed by the M_CAN while DMA request output is activated. The behavior is similar to that of an Rx Buffers with its New Data flag set.

After the DMA has completed the DMA unit sets the DMA acknowledge input . This resets the DMA request output. Now the M_CAN is prepared to receive the next set of debug messages.

Note: To use full ‘Debug on CAN Support’ feature on a M_(TT)CAN instance, a DMA channel is required. Refer to device DMA map, to see which M_(TT)CAN instances has got an assigned DMA channel.

52.3.12.1 Filtering for Debug Messages

Filtering for debug messages is done by configuring one Standard / Extended Message ID Filter Element for each of the three debug messages. To enable a filter element to filter for debug messages SFEC / EFEC has to be programmed to ‘111’. In this case fields SFID1 / SFID2 and EFID1 / EFID2 have a different meaning. While SFID2 /EFID2[10:9] controls the debug message handling state machine, SFID2 / EFID2[5:0] controls the location for storage of a received debug message.

When a debug message is stored, neither the respective New Data flag nor IR.DRX are set. The reception of debug messages can be monitored via RXF1S.DMS.

Table 870. Example Filter Configuration for debug messages

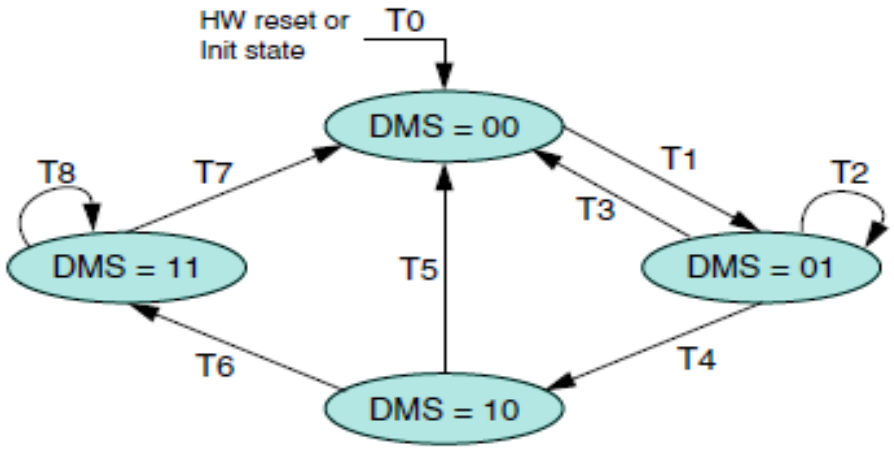
Filter Element	SFID1[10:0] EFID1[28:0]	SFID2[10:9] EFID2[10:9]	SFID2[5:0] EFID2[5:0]
0	ID debug message A	01	11 1101
1	ID debug message B	10	11 1110
2	ID debug message C	11	11 1111

52.3.12.2 Debug Message Handling

The debug message handling state machine assures that debug messages are stored to three consecutive Rx Buffers in correct order. In case of missing messages the process is restarted. The DMA request is activated only when all three debug messages A, B, C have been received in correct order.

The status of the debug message handling state machine is signaled via RXF1S.DMS.

Figure 768. Debug Message Handling State Machine



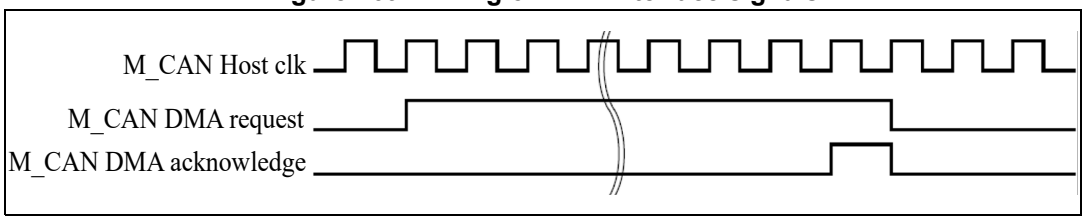
T0: reset m_cam_dma_req output, enable reception of debug messages A, B, and C
T1: reception of debug message A
T2: reception of debug message A
T3: reception of debug message C
T4: reception of debug message B
T5: reception of debug messages A, B
T6: reception of debug message C
T7: DMA transfer completed
T8: reception of debug message A,B,C (message rejected)

52.3.12.3 Interface to DMA Controller

When all three debug messages A, B, C have been received in the correct order, M_CAN DMA request signal is activated to trigger a DMA transfer. The RAM words holding debug messages A, B, C will not be changed by the M_CAN while M_CAN DMA request signal is active.

After the transfer of the received messages has completed the DMA unit activates the M_CAN DMA ACK signal. This resets M_CAN DMA request signal. The debug message handling state machine enters idle state (DMS = ‘00’) and waits for reception of the next debug messages.

Figure 769. Timing of DMA interface signals



Note: If the DMA unit activates input M_CAN DMA acknowledge signal before the DMA transfer has completed, the Rx Buffer elements holding debug messages A, B, C are unlocked and may be overwritten by received debug messages.

52.3.13 Tx handling

The Tx Handler handles transmission requests for the dedicated Tx Buffers, the Tx FIFO, and the Tx Queue. It controls the transfer of transmit messages to the CAN core, the Put and Get Indexes, and the Tx Event FIFO. Up to 32 Tx Buffers can be set up for message transmission. The Tx Buffer element is described in the *Note: on page 1472*.

The Tx Handler starts a Tx scan to check for the highest priority pending Tx request (Tx Buffer with lowest Message ID) when the Tx Buffer Request Pending register TXBRP is updated, or when a transmission has been started.

Note: AUTOSAR requires at least three Tx Queue Buffers and support of transmit cancellation.

52.3.13.1 Transmit Pause

The transmit pause feature is intended for use in CAN systems where the CAN message identifiers are (permanently) specified to specific values and cannot easily be changed. These message identifiers may have a higher CAN arbitration priority than other defined messages, while in a specific application their relative arbitration priority should be inverse. This may lead to a case where one ECU sends a burst of CAN messages that cause another ECU’s CAN messages to be delayed because that other messages have a lower CAN arbitration priority.

If for example CAN ECU-1 has the transmit pause feature enabled and is requested by its application software to transmit four messages, it will, after the first successful message transmission, wait for two CAN bit times of bus idle before it is allowed to start the next requested message. If there are other ECUs with pending messages, those messages are started in the idle time, they would not need to arbitrate with the next message of ECU-1. After having received a message, ECU-1 is allowed to start its next transmission as soon as the received message releases the CAN bus.

The transmit pause feature is controlled by bit CCCR.TXP. If the bit is set, the M_CAN will, each time it has successfully transmitted a message, pause for two CAN bit times before starting the next transmission. This enables other CAN nodes in the network to transmit messages even if their messages have lower prior identifiers. Default is transmit pause disabled (CCCR.TXP = ‘0’).

This feature looses up burst transmissions coming from a single node and it protects against “babbling idiot” scenarios where the application program erroneously requests too many transmissions.

52.3.13.2 Dedicated Tx Buffers

Dedicated Tx Buffers are intended for message transmission under complete control of the Host CPU. Each Dedicated Tx Buffer is configured with a specific Message ID. In case that multiple Tx Buffers are configured with the same Message ID, the Tx Buffer with the lowest buffer number is transmitted first.

If the data section has been updated, a transmission is requested by an Add Request via TXBAR[ARn]. The requested messages arbitrate internally with messages from an optional Tx FIFO or Tx Queue and externally with messages on the CAN bus, and are sent out according to their Message ID.

A Dedicated Tx Buffer allocates Element Size 32-bit words in the Message RAM (refer to *Table 871*). Therefore the start address of a dedicated Tx Buffer in the Message RAM is calculated by adding transmit buffer index (0…31) × Element Size to the Tx Buffer Start Address TXBC.TBSA.

Table 871. Tx Buffer / FIFO / Queue Element Size

TXESC.TBDS[2:0]	Data Field [bytes]	Element Size [RAM words]
000	8	4
001	12	5
010	16	6
011	20	7
100	24	8
101	32	10
110	48	14
111	64	18

52.3.13.3 Tx FIFO

Tx FIFO operation is configured by programming TXBC[TFQM] to ‘0’. Messages stored in the Tx FIFO are transmitted starting with the message referenced by the Get Index TXFQS[TFGI]. After each transmission the Get Index is incremented cyclically until the Tx FIFO is empty. The Tx FIFO enables transmission of messages with the same Message ID from different Tx Buffers in the order these messages have been written to the Tx FIFO. The M_CAN calculates the Tx FIFO Free Level TXFQS[TFFL] as difference between Get and Put Index. It indicates the number of available (free) Tx FIFO elements.

New transmit messages have to be written to the Tx FIFO starting with the Tx Buffer referenced by the Put Index TXFQS[TFQPI]. An Add Request increments the Put Index to the next free Tx FIFO element. When the Put Index reaches the Get Index, Tx FIFO Full (TXFQS[TFQF] = ‘1’) is signaled. In this case no further messages should be written to the Tx FIFO until the next message has been transmitted and the Get Index has been incremented.

When a single message is added to the Tx FIFO, the transmission is requested by writing a ‘1’ to the TXBAR bit related to the Tx Buffer referenced by the Tx FIFOs Put Index.

When multiple (n) messages are added to the Tx FIFO, they are written to n consecutive Tx Buffers starting with the Put Index. The transmissions are then requested via TXBAR. The Put Index is then cyclically incremented by n. The number of requested Tx buffers should not exceed the number of free Tx Buffers as indicated by the Tx FIFO Free Level.When a transmission request for the Tx Buffer referenced by the Get Index is cancelled, the Get Index is incremented to the next Tx Buffer with pending transmission request and the Tx FIFO Free Level is recalculated. When transmission cancellation is applied to any other Tx Buffer, the Get Index and the FIFO Free Level remain unchanged.

A Tx FIFO element allocates Element Size 32-bit words in the Message RAM (refer to *Table 871*). Therefore the start address of the next available (free) Tx FIFO Buffer is calculated by adding Tx FIFO/Queue Put Index TXFQS.TFQPI (0…31) × Element Size to the Tx Buffer Start Address TXBC.TBSA.

52.3.13.4 Tx Queue

Tx Queue operation is configured by programming TXBC[TFQM] to ‘1’. Messages stored in the Tx Queue are transmitted starting with the message with the lowest Message ID (highest priority). In case that multiple Queue Buffers are configured with the same Message ID, the Queue Buffer with the lowest buffer number is transmitted first. New messages have to be written to the Tx Buffer referenced by the Put Index TXFQS[TFQPI]. An Add Request cyclically increments the Put Index to the next free Tx Buffer. In case that the Tx Queue is full (TXFQS[TFQF] = ‘1’), the Put Index is not valid and no further message should be written to the Tx Queue until at least one of the requested messages has been sent out or a pending transmission request has been cancelled.

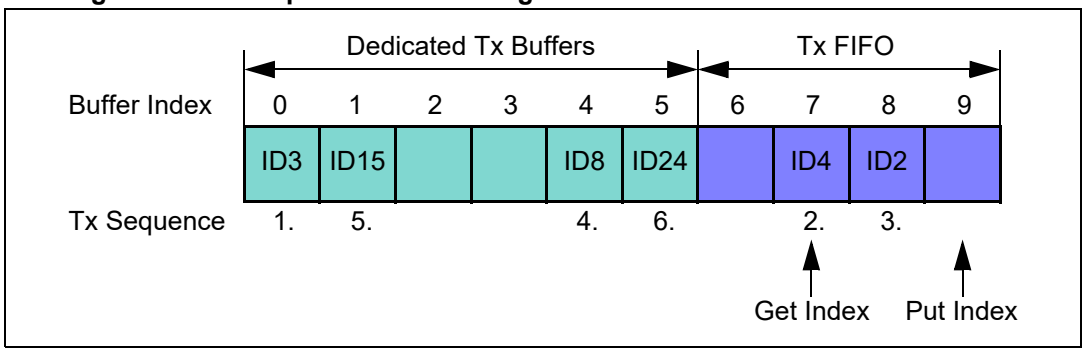
The application may use register TXBRP instead of the Put Index and may place messages to any Tx Buffer without pending transmission request.

A Tx Queue Buffer allocates Element Size 32-bit words in the Message RAM (refer to *Table 871*). Therefore the start address of the next available (free) Tx Queue Buffer is calculated by adding Tx FIFO/Queue Put Index TXFQS.TFQPI (0…31) × Element Size to the Tx Buffer Start Address TXBC.TBSA.

52.3.13.5 Mixed dedicated Tx Buffers / Tx FIFO

In this case the Tx Buffers section in the Message RAM is subdivided into a set of Dedicated Tx Buffers and a Tx FIFO. The number of Dedicated Tx Buffers is configured by TXBC[NDTB]. The number of Tx Buffers assigned to the Tx FIFO is configured by TXBC[TFQS]. In case TXBC[TFQS] is programmed to zero, only Dedicated Tx Buffers are used.

Figure 770. Example of mixed Configuration Dedicated Tx Buffers / Tx FIFO



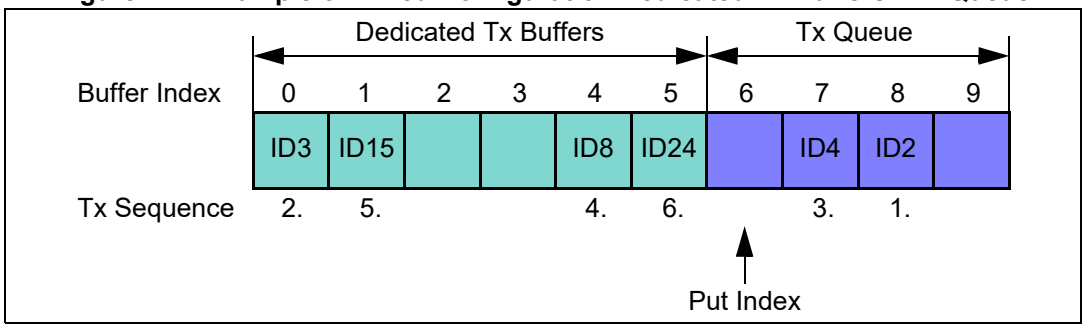
Tx prioritization:

- Scan Dedicated Tx Buffers and oldest pending Tx FIFO Buffer (referenced by TXFS[TFGI])
- Buffer with lowest Message ID gets highest priority and is transmitted next

52.3.13.6 Mixed dedicated Tx Buffers / Tx Queue

In this case the Tx Buffers section in the Message RAM is subdivided into a set of Dedicated Tx Buffers and a Tx Queue. The number of Dedicated Tx Buffers is configured by TXBC[NDTB]. The number of Tx Queue Buffers is configured by TXBC[TFQS]. In case TXBC[TFQS] is programmed to zero, only Dedicated Tx Buffers are used.

Figure 771. Example of mixed Configuration Dedicated Tx Buffers / Tx Queue



Tx prioritization:

- Scan all Tx Buffers with activated transmission request
- Tx Buffer with lowest Message ID gets highest priority and is transmitted next

52.3.13.7 Transmit cancellation

The M_CAN supports transmit cancellation. This feature is especially intended for gateway applications and AUTOSAR based applications. To cancel a requested transmission from a dedicated Tx Buffer or a Tx Queue Buffer the Host has to write a ‘1’ to the corresponding bit position (=number of Tx Buffer) of register TXBCR. Transmit cancellation is not intended for Tx FIFO operation.

Successful cancellation is signaled by setting the corresponding bit of register TXBCF to ‘1’.

In case a transmit cancellation is requested while a transmission from a Tx Buffer is already ongoing, the corresponding TXBRP bit remains set as long as the transmission is in progress. If the transmission was successful, the corresponding TXBTO and TXBCF bits

are set. If the transmission was not successful, it is not repeated and only the corresponding TXBCF bit is set.

Note: In case a pending transmission is cancelled immediately before this transmission could have been started, there follows a short time window where no transmission is started even if another message is also pending in this node. This may enable another node to transmit a message which may have a lower priority than the second message in this node.

52.3.13.8 Tx Event handling

To support Tx event handling the M_CAN has implemented a Tx Event FIFO. After the M_CAN has transmitted a message on the CAN bus, Message ID and timestamp are stored in a Tx Event FIFO element. To link a Tx event to a Tx Event FIFO element, the Message Marker from the transmitted Tx Buffer is copied into the Tx Event FIFO element.

The Tx Event FIFO can be configured to a maximum of 32 elements. The Tx Event FIFO element is described in the Note *Depending on the configuration of the element size (TXESC), between two and sixteen 32-bit words (Tn = 3 to 17) are used for storage of a CAN message's data field.*. When a Tx Event FIFO full condition is signaled by IR.TEFF, no further elements are written to the Tx Event FIFO until at least one element has been read out and the Tx Event FIFO Get Index has been incremented. In case a Tx event occurs while the Tx Event FIFO is full, this event is discarded and interrupt flag IR.TEFL is set.

To avoid a Tx Event FIFO overflow, the Tx Event FIFO watermark can be used. When the Tx Event FIFO fill level reaches the Tx Event FIFO watermark configured by TXEFC[EFWM], interrupt flag IR.TEFW is set.

When reading from the Tx Event FIFO, two times the Tx Event FIFO Get Index TXEFS[EFGI] has to be added to the Tx Event FIFO start address TXEFC[EFSA].

52.3.14 FIFO acknowledge handling

The Get Indexes of Rx FIFO 0, Rx FIFO 1, and the Tx Event FIFO are controlled by writing to the corresponding FIFO Acknowledge Index (refer to *Section 52.3.5.2.28: Rx FIFO 0 Acknowledge Register (RXF0A)* and *Section 52.3.5.2.46: Tx Event FIFO Acknowledge Register (TXEFA)*). Writing to the FIFO Acknowledge Index will set the FIFO Get Index to the FIFO Acknowledge Index plus one and thereby updates the FIFO Fill Level. There are two use cases:

- When only a single element has been read from the FIFO (the one being pointed to by the Get Index), this Get Index value is written to the FIFO Acknowledge Index.
- When a sequence of elements has been read from the FIFO, it is sufficient to write the FIFO Acknowledge Index only once at the end of that read sequence (value: Index of the last element read), to update the FIFOs Get Index.

Due to the fact that the CPU has free access to the M_CAN's Message RAM, special care has to be taken when reading FIFO elements in an arbitrary order (Get Index not considered). This might be useful when reading a High Priority Message from one of the two Rx FIFOs. In this case the FIFOs Acknowledge Index should not be written because this would set the Get Index to a wrong position and also alters the FIFOs Fill Level. In this case some of the older FIFO elements would be lost.

Note: The application has to ensure that a valid value is written to the FIFO Acknowledge Index. The M_CAN does not check for erroneous values.

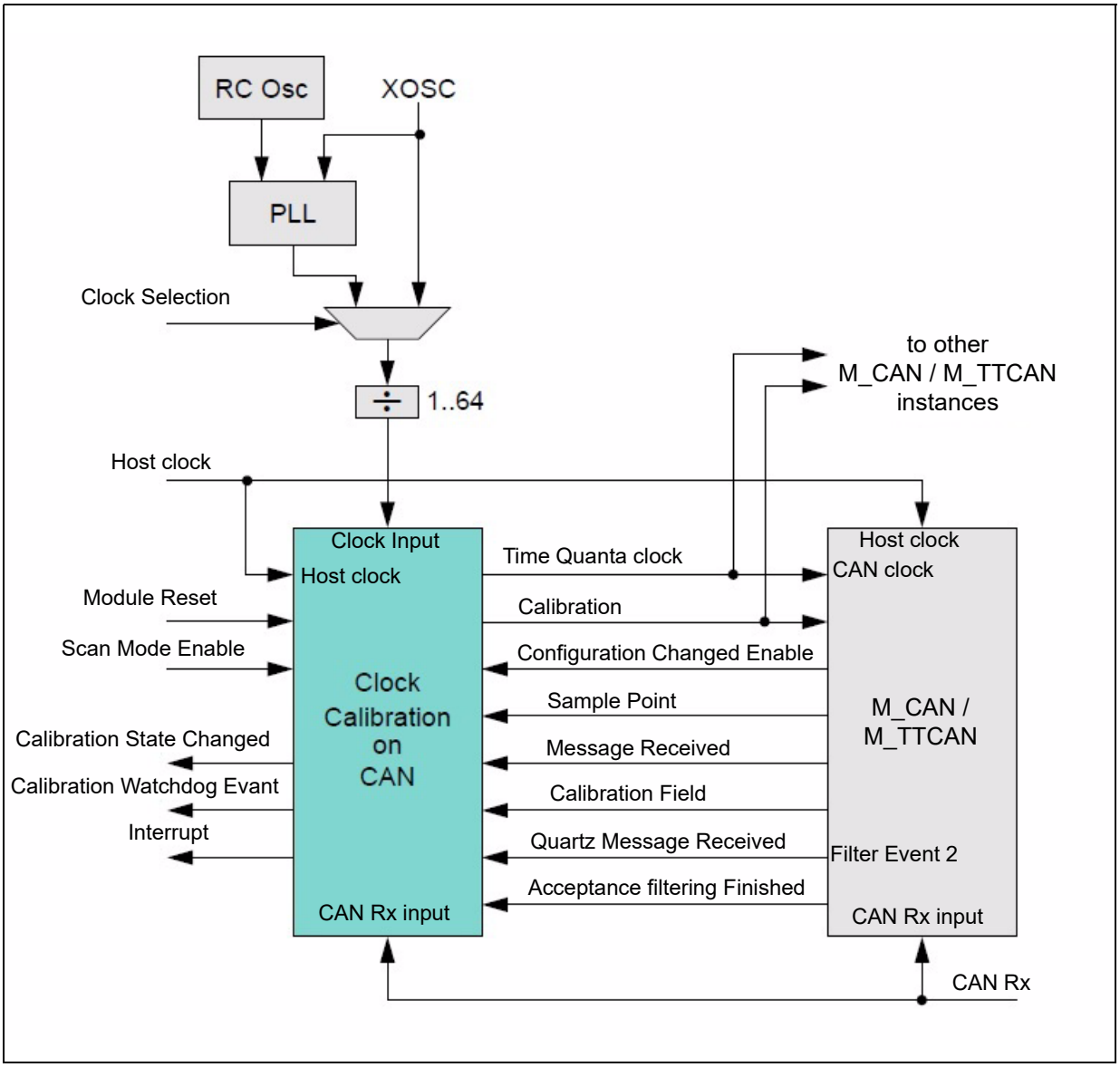
52.3.15 Clock Calibration on CAN Unit (CCCU)

This section describes an add-on for the M_CAN modules that generates a calibrated CAN Time Quanta Clock from an on-chip RC-oscillator clock by evaluating CAN messages received by the attached M_CAN. Calibration on CAN is only possible when the M_CAN is operated in standard CAN mode.

Note: If calibration unit is intended to generate the protocol clock, then it is mandatory to configure calibration unit and MCAN1 before use the other CAN controller like MCAN2 and MCAN3, because the calibrated clock is generated through MCAN1.

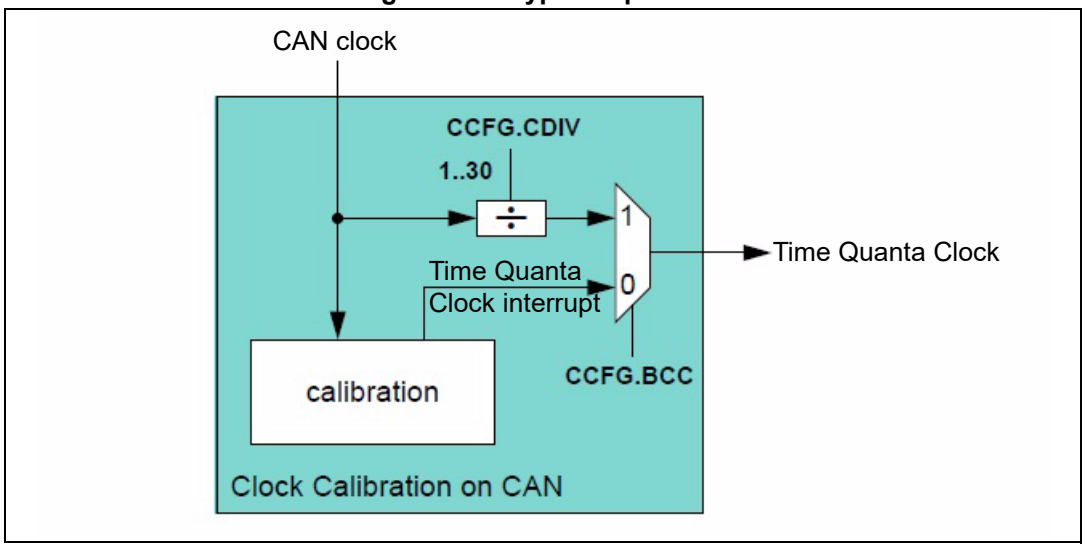
Figure 772 shows the interface of the Clock Calibration on CAN Unit. The clock multiplexer controlled by clock input from Clock Select multiplexer controls whether the clock from the PLL or XOSC is routed to Clock Select multiplexer input. The calibrated Time Quanta Clock is routed to all CAN instances. In case of a successful calibration this is signaled via Calibration.

Figure 772. Connection of Clock Calibration on CAN Unit to M_CAN



CPU accesses to the Clock Calibration on CAN Unit use the Host clock connected to Host clock input from Clock Select Multiplexer. Clock calibration is bypassed when CCFG.BCC = ‘1’. In this case, the clock from Clock Select multiplexer is routed through a clock divider to Time Quanta Clock output (refer to *Figure 773*).

Figure 773. Bypass operation



52.3.15.1 Operating Conditions

The Clock Calibration on CAN Unit is designed to operate under the following conditions:

- Clock input from PLL (clock input from Clock Select multiplexer) between 80 MHz and 500 MHz
- M_CAN bit rates between 125 kbit/s and 1 Mbit/s
- Clock for Host accesses (Host clock) is 40/50 MHz

The Clock Calibration on CAN Unit generates a calibrated Time Quanta clock in the range from 0.5 MHz to 25 MHz.

Note: The M_CAN requires that the CAN clock is always below or equal to the Host clock (CAN clock ≤ Host clock). This has to be considered when the Clock Calibration on CAN Unit is bypassed (CCFG.BCC = '1').

52.3.15.2 Calibration Accuracy

The calibration accuracy in state Precision_Calibrated depends on:

- The dynamic clock tolerance at the clock input from clock select multiplexer
- The measurement error. For each bit sequence used for calibration measurement, there is a maximum error of one clock input from Clock Select multiplexer period. The number of bits used for measurement of the bit time is 32 or 64-bit, depending on configuration of CCFG.CFL.
- Tolerable error in calibration mechanism

The distance between two calibration messages has to be chosen to fit the clock tolerance requirements of the attached M_CANs.

Note: Dynamic clock tolerance is the clock frequency variation between two calibration messages, such as, caused by change of temperature or operating voltage.

52.3.15.3 Memory map and register description

52.3.15.3.1 Hardware Reset Description

After hardware reset, the registers of the Clock Calibration on CAN Unit hold the reset values listed in *Table 872*.

The Clock Calibration on CAN Unit allocates an address space of 16 bytes. All registers are organized as 32-bit registers. The Clock Calibration on CAN Unit is accessible by the Host CPU using a data width of 8-bit (byte access), 16-bit (half-word access), or 32-bit (word access). Write access by the Host CPU to registers/bits marked with “P=Protected Write” is possible only when Configuration Change Enable input = ‘1’. Configuration Change Enable signal is activated when the M_CAN control bits CCCR.CCE = ’1’ AND CCCR.INIT = ‘1’.

Table 872. CCCU memory map

Address Offset	Register Name	Access(1)	Reset value	Location
0x0000	Core Release Register (CREL)	R	0x1114_0729	*Section 52.3.15.3.2*
0x0004	Calibration Configuration Register (CCFG)	RP	0x0000_0404	*Section 52.3.15.3.3*
0x0008	Calibration Status Register (CSTAT)	R	0x0203_FFFF	*Section 52.3.15.3.4*
0x000C	Calibration Watchdog (CWD)	RP	0x0000_0000	*Section 52.3.15.3.5*
0x0010	CU Interrupt Register (CUIR)	R/W	0x0000_0000	*Section 52.3.15.3.6*
0x0014	CU Interrupt Enable (CUIE)	R/W	0x0000_0000	*Section 52.3.15.3.7*

1. R = Read, W = Write, P = Protected Write

52.3.15.3.2 Core Release Register (CREL)

Offset: 0x0000 Access: Read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	REL				STEP				SUBSTEP				YEAR			
W																
Reset(1)	-r	-r	-r	-r	-r	-r	-r	-r	-r	-r	-r	-r	-d	-d	-d	-d

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	MON								DAY							
W																
Reset(1)	-d	-d	-d	-d	-d	-d	-d	-d	-d	-d	-d	-d	-d	-d	-d	-d

1. -r = release, -d = time stamp, the coding of revisions depends on the module version used in the device. For the device reset value refer to *Table 872: CCCU memory map*.

Figure 774. Core Release Register (CREL)

Table 873. CREL field descriptions

Field	Description
0:3 REL	Core Release One digit, BCD-coded.
4:7 STEP	Step of Core Release One digit, BCD-coded.
8:11 SUBSTEP	Substep of Core Release One digit, BCD-coded.
12:15 YEAR	Time Stamp Year One digit, BCD-coded.
16:23 MON	Time Stamp Month Two digits, BCD-coded.
24:31 DAY	Time Stamp Day Two digits, BCD-coded.

Table 874. Example for Coding of Revisions

Release	Step	SubStep	Year	Month	Day	Name
1	0	0	2	02	20	Revision 1.0.0, Date 2012/02/20

52.3.15.3.3 Calibration Configuration Register (CCFG)

Offset: 0x0004 Access: RP

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	SWR	0	0	0	0	0	0	0	0	0	0	0	CDIV(1)			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	OCPM(1)								CFL(1)	BCC(1)	0	TQBT(1)				
W																
Reset	0	0	0	0	0	1	0	0	0	0/1	0	0	0	1	0	0

1. Write access by the Host CPU to registers/bits marked with “P=Protected Write” is possible only when Configuration Change Enable input = ‘1’. Configuration Change Enable signal is activated when the M_CAN control bits CCCR.CCE = ‘1’ AND CCCR.INIT = ‘1’.

Figure 775. Calibration Configuration Register (CCFG)

Table 875. CCFG field descriptions

Field	Description
0 SWR	Software Reset Writing a '1' to this bit will reset the calibration FSM to state Not_Calibrated (CSTAT.CALS = '00'). The Calibration Watchdog value CWD.WDV is also reset. Registers CCFG, CSTAT and the Calibration Watchdog configuration CWD.WDC are unchanged. The bit remains set until reset has completed.
12:15 CDIV	Clock Divider The clock divider has to be configured when the clock calibration is bypassed (BCC = '1') to assure that the M_CAN requirement CAN clock <= Host clock is fulfilled. 0000 Divide by 1 0001 Divide by 2 0010 Divide by 4 0011 Divide by 6 0100 Divide by 8 0101 Divide by 10 0110 Divide by 12 0111 Divide by 14 1000 Divide by 16 1001 Divide by 18 1010 Divide by 20 1011 Divide by 22 1100 Divide by 24 1101 Divide by 26 1110 Divide by 28 1111 Divide by 30
16:23 OCPM	Oscillator Clock Periods Minimum (0x00–0xFF) Configures the minimum number of clock input from Clock Select multiplexer periods in two CAN bit times. OCPM is used in Basic Calibration to avoid false measurements in case of glitches on the bus line. The configured number of clock input from Clock Select multiplexer periods is OCPM × 32. The configuration depends on the clock input from Clock Select multiplexer frequency (80 MHz to 500 MHz) and the bit rate configured in the attached M_CANs (125 kbit/s to 1 Mbit/s). It is recommended to configure a value slightly below two CAN bit times. The reset value is 1.6 bit times at 80 MHz clock input from Clock Select multiplexer and 1 Mbit/s CAN bit rate.
24 CFL	Calibration Field Length 0 Calibragation field length is 32 bits. 1 Calibragation field length is 64 bits.

Table 875. CCFG field descriptions (continued)

Field	Description
25 BCC	Bypass Clock Calibration If this bit is set, the clock input clock input from Clock Select multiplexer is routed to the Time Quanta clock output through a clock divider configurable via CDIV, Calibration is always ‘1’. In this case the baud rate prescaler of the connected M_CANs has to be configured to generate the M_CAN internal time quanta clock. 0 Clock calibration unit generates Time Quanta clock 1 Clock calibration unit bypassed **Note:** As long as clock input from Clock Select multiplexer is equal or above 80 MHz the Clock Calibration on CAN Unit is functional, even when BCC = ‘1’. The calibration state can be read from register CSTAT.
27:31 TQBT	Time Quanta per Bit Time (0x04–0x19) Configures the number of time quanta per bit time. Same value as configured in the attached M_CANs. The range of the resulting Time Quanta clock is from 0,5 MHz (bit rate of 125 kbit/s with 4 tq per bit time) to 25 MHz (bit rate of 1 Mbit/s with 25 tq per bit time). Valid values are 4 to 25. Configured values below 4 are interpreted as 4, values above 25 are interpreted as 25.

52.3.15.3.4 Calibration Status Register (CSTAT)

Offset: 0x0008 Access: Read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	CALS		0		TQC										OCPC	
W																
Reset	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1	1

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	OCPC															
W																
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Figure 776. Calibration Status Register (CSTAT)

Table 876. CSTAT field descriptions

Field	Description
0:1 CALS	Calibration State 00 Not_Calibrated 01 Basic_Calibrated 10 Precision_Calibrated 11 Reserved

Table 876. CSTAT field descriptions (continued)

Field	Description
3:13 TQC	Time Quanta Counter (0x000–0x3FF) Captured number of time quanta in calibration field (32 or 64 bits). Only valid when the clock calibration unit is in state Precision_Calibrated.
14:31 OCPC	Oscillator Clock Period Counter (0x00000–0x3FFFF) Captured number of oscillator clock periods in calibration field (32 or 64 bits). Only valid when the clock calibration unit is in state Precision_Calibrated.

52.3.15.3.5 Calibration Watchdog Register (CWD)

The calibration watchdog is started after the first falling edge when the calibration FSM is in state Not_Calibrated (CSTAT.CALS = ‘00’). In this state the calibration watchdog monitors the Message Received input. In case no message was received until the calibration watchdog has counted down to zero, the calibration FSM stays in state Not_Calibrated (CSTAT.CALS = ‘00’), the counter is reloaded with RWD.WDC and basic calibration is restarted after the next falling edge.

When in state Basic_Calibrated (CSTAT.CALS = ‘01’), the calibration watchdog is restarted with each Message Received = ‘1’). In case no message was received until the calibration watchdog has counted down to zero, the calibration FSM returns to state Not_Calibrated (CSTAT.CALS = ‘00’), the counter is reloaded with RWD.WDC and basic calibration is restarted after the next falling edge.

When a quartz message was received, state Precision_Calibrated (CSTAT.CALS = ‘10’) is entered and the calibration watchdog is restarted. In this state the calibration watchdog monitors the Quartz Message Received input. In case no message from a quartz controlled node was received by the attached M_CAN until the calibration watchdog has counted down to zero, the calibration FSM transits back to state Basic_Calibrated (CSTAT.CALS = ‘01’).

The Calibration Watchdog Counter is clocked by the Sample Point input. The duration of a cu_spt pulse is one Time Quanta clock period. The signal is active when the CAN protocol engine on the attached M_CAN is started, that is when the INIT bit is reset.

A calibration watchdog event is signaled by Calibration Watchdog Event output. The duration of aCalibration Watchdog Event pulse is one Host clock period.

A calibration watchdog event also sets interrupt flag **CUIR.CWE**. If enabled by **CUIE.CWEE**, Interrupt line is activated (set to high). Interrupt line remains active until interrupt flag **CUIR.CWE** is reset.

Offset: 0x000C Access: RP

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	WDV															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	WDC(1)															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. Write access by the Host CPU to registers/bits marked with “P=Protected Write” is possible only when input Configuration Change Enable input = ‘1’. Configuration Change Enable inputsignal is activated when the M_CAN control bits CCCR.CCE = ‘1’ AND CCCR.INIT = ‘1’.

Figure 777. Calibration Watchdog (CWD)

Table 877. CWD field descriptions

Field	Description
0:15 WDV	Watchdog Value Actual Calibration Watchdog Counter Value.
16:31 WDC	Watchdog Configuration Start value of the CalibrationWatchdog Counter.With the reset value of ‘00’ the counter is disabled.

52.3.15.3.6 CU Interrupt Register (CUIR)

The flags are set when one of the listed conditions is detected (edge-sensitive). The flags remain set until the Host clears them. A flag is cleared by writing a ‘1’ to the corresponding bit position. Writing a ’0’ has no effect. A hard reset will clear the register. The configuration of CUIE controls whether an interrupt is generated.

Offset: 0x0010 Access: R/W

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	CSC	CWE
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 778. CU Interrupt Register (CUIR)

Table 878. CUIR field descriptions

Field	Description
30 CSC	Calibration State Changed 0 Calibration State unchanged 1 Calibration State has changed
31 CWE	Calibration Watchdog Event 0 No Calibration Watchdog Event 1 Calibration Watchdog Event occurred

52.3.15.3.7 CU Interrupt Enable Register (CUIE)

The settings in the CU Interrupt Enable register determine whether a status change in the CU Interrupt Register will be signalled on an interrupt line.

Offset: 0x0014 Access: R/W

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	CSCE	CWEE
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 779. CU Interrupt Enable Register (CUIE)

Table 879. CUIE field descriptions

Field	Description
30 CSCE	Calibration State Changed Enable 0 Interrupt disabled 1 Interrupt enabled
31 CWEE	Calibration Watchdog Event Enable 0 Interrupt disabled 1 Interrupt enabled

52.3.15.4 Functional Description

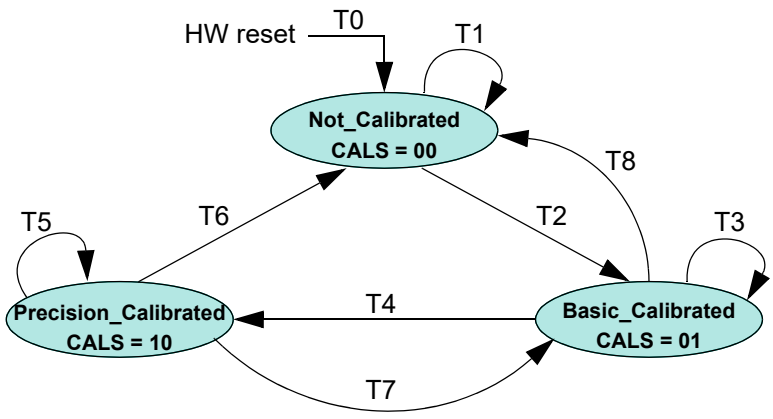
Calibration of the Time Quanta clock via CAN messages is performed by adapting a clock divider that generates the CAN protocol time quantum tq from the clock input from Clock Select multiplexer .

- First step: Basic Calibration–The minimum distance between two edges from recessive to dominant is measured, this time to be assumed two CAN bit times, counted in PLL clock periods. The clock divider is updated each time a new measurement finds a smaller distance between edges. Basic calibration is achieved when the CAN protocol

controller detects a valid CAN message. This is signaled to the Calibration on CAN unit by Message Received input signal.

- Second step: Precision Calibration – The calibration state machine measures the length of a longer bit sequence inside a CAN frame by counting the number of clock input from Clock Select multiplexer periods. The length of this bit sequence can be configured to 32 or 64 bits via CCFG.CLF. For a calibration field length of 32/64 bit a calibration message with at least 2/6 byte data field is required. The counter is controlled by Calibration Field and Sample Point input signals. The counter is enabled when Calibration Field is set and stops after the configured number of bits has been received. Received bits are signaled by the M_CAN by Sample Point signal. Precision calibration is based on the new clock divider value calculated from the measurement of the longer bit sequence.

Figure 780. Calibration FSM



T0: HW reset

T1: check for minimum

T2: Message Received = ‘1’

T3: evaluate messages

T4: Quartz Message Received = ‘1’

T5: evaluate quartz messages

T6: Configuration Change Enabled = ‘1’

T7: watchdog event: no quartz message received

T8: watchdog event: no message received
or Configuration Change Enabled = ‘1’

A change in the calibration state is signaled via by Calibration State Changed output. The duration of a Calibration State Changed output pulse is one Host clock period.

A change in the calibration state also sets interrupt flag CUIR.CSC. If enabled by CUIE.CSCE, Interrupt line is activated (set to high). Interrupt line remains active until interrupt flag CUIR.CSC is reset.

Until precision calibration is achieved (Calibration=‘1’), the attached M_CANs operate in a restricted mode (no frame transmission, no error or overload flag transmission, no error counting). In case calibration of the PLL is done by software by evaluating the calibration status from register CSTAT, the attached M_CANs have to be set to restricted operation

mode (CCCR.ASM = ‘1’) until the Calibration on CAN unit is in state Precision_Calibrated (refer to *Section 52.3.15.4.3: Software Calibration*).

Precision calibration may be performed only on valid CAN frames transmitted by a node with a stable, quartz-controlled clock. Calibration frames are detected by the M_CAN’s acceptance filtering and are signaled to the Clock Calibration on CAN Unit by Quartz Message Received, to be connected to one of the M_CAN’s Filter Event outputs, signal. A filter element and an Rx Buffer have to be configured in the attached M_CAN to identify and store calibration messages. After reception of a calibration message the Rx Buffer’s new data flag has to be reset to enable signaling of the next calibration message.

In case there is only one CAN transmitter with a quartz clock in the network, this node has to transmit its first message after startup with at least one ‘1010’ sequence in the data field or in the identifier. This assures that the non-quartz nodes can enter state Basic_Calibrated and then acknowledge the quartz node’s messages.

Precision calibration must be repeated in predefined maximum intervals supervised by the calibration watchdog.

Note: When the Clock Calibration on CAN Unit transits from state Precision_Calibrated back to Basic_Calibrated, the calibration OK signal is deasserted, the attached M_CANs complete ongoing transmissions, and then enter restricted operation (no frame transmission, no error or overload flag transmission, no error counting).

52.3.15.4.1 Configuration

The Clock Calibration on CAN Unit is configured via register CCFG. The register is only writeable while Configuration Change Enable = ‘1’, that is when the attached M_CAN has CCCR.CCE and CCCR.INIT set.

For basic calibration the minimum number of oscillator periods between two consecutive falling edges at CAN receive input pin is measured. The number of clock periods depends on the clock frequency applied at clock input from clock select multiplexer. In case the measured number of clock periods is below the minimum configured by CCFG.OCPM, for example due to a glitch on CAN receive input, the value is discarded and measurement continues.

It is recommended to configure CCFG.OCPM slightly below two CAN bit times:

CCFG.OCPM < ((2 x CAN bit time) / Clock input from Clock Select multiplexer period) / 32

The length of the bit field used for precision calibration can be configured to 32 or 64 bits via CCFG.CFL. The number of bits used for precision calibration has an impact on calibration accuracy and the maximum distance between two calibration messages. The reception of the bits is signaled by the attached M_CAN by Sample Point input.

The number of time quanta per bit time configured by CCFG.TQBT is used together with the measured number of oscillator clock periods CSTAT.OCPC to define the number of oscillator clocks per bit time.

When the clock calibration is bypassed by configuring CCFG.BCC = ‘1’, the internal clock divider has to be configured via CCFG.CDIV to fulfill the condition CAN clock $\leq$ Host clock.

Note: When clock calibration on CAN is active (CCFG.BCC = ‘0’), the baud rate prescaler of the attached M_CANs has to be configured to inactive.

52.3.15.4.2 Status signaling

The status of the Clock Calibration on CAN Unit can be monitored by reading register CSTAT. When in state Precision_Calibrated the oscillator clock period counter CSTAT.OCPC signals the number of oscillator clock periods in the calibration field while CSTAT.TQC signals the number of time quanta in the calibration field.

The calibration state is monitored by CSTAT.CALS. A change in the calibration state is signalled at the module interface by Calibration State Changed. The duration of a Calibration State Changed pulse is one Host clock period. A change in the calibration state also sets interrupt flag CUIR.CSC. If enabled by CUIE.CSCE, Interrupt line is activated (set to high). Interrupt line remains active until interrupt flag CUIR.CSC is reset.

A calibration watchdog event is signalled by Calibration Watchdog Event output. The duration of a Calibration Watchdog Event output pulse is one Host clock period. A calibration watchdog event also sets interrupt flag CUIR.CWE. If enabled by CUIE.CWEE, Interrupt line is activated (set to high). Interrupt line remains active until interrupt flag CUIR.CWE is reset

52.3.15.4.3 Software Calibration

The Clock Calibration on CAN Unit also supports software calibration of the clock generated from an adjustable PLL, connected to the clock input, from Clock Select multiplexer. For this purpose the clock from Clock Select multiplexer has to be routed to Time Quanta clock output by programming CCFG.BCC = ‘1’. The Clock Calibration on CAN Unit’s internal clock divider has to be configured via CCFG.CDIV to meet the clock requirements of the attached M_CANs. All other configuration parameters have to be set via CCFG as if the Clock Calibration on CAN Unit is generating Time Quanta clock.

For startup the M_CANs attached to the Clock Calibration on CAN Unit have to be configured for Restricted Operation (CCCR.ASM = ‘1’). The PLL clock has to be adjusted until the Clock Calibration on CAN Unit has reached state Precision_Calibrated. Now the software has to reset CCCR.ASM and the M_CANs can start normal operation.

During operation the software has to check regularly whether the Clock Calibration on CAN Unit is still in state Precision_Calibrated. In case the Clock Calibration on CAN Unit has left state Precision_Calibrated due to drift of the PLL clock, the attached M_CANs have to be set into Restricted Operation Mode by programming CCCR.INIT, CCCR.CCE, and CCCR.ASM to ‘1’.

After the PLL clock has been adjusted successfully (Clock Calibration on CAN Unit is in state Precision_Calibrated), the M_CANs can be set to normal operation again.

52.4 Time Triggered Modular CAN (M_TTCAN) core

The M_TTCAN performs communication according to ISO 11898-1 (identical to Bosch CAN protocol specification 2.0 part A,B) and according to ISO 11898-4 (time-triggered communication on CAN). The M_TTCAN provides all features of time-triggered communication specified in ISO 11898-4, including event synchronized time-triggered communication, global system time, and clock drift compensation.

In addition the M_TTCAN supports communication according to CAN FD protocol specification 1.0. The CAN FD option can be used together with event-triggered and time-triggered CAN communication.

Note: Implemented TTCAN nodes are fully compliant to ISO CAN FD specification ISO 11898-1: 2015.

The message storage is intended to be a single-ported or dual-ported Message RAM outside of the module. It is connected to the M_TTCAN via the Generic Master Interface. Depending on the chosen ASIC integration, multiple M_TTCAN controllers can share the same Message RAM.

All functions concerning the handling of messages are implemented by the Rx Handler and the Tx Handler. The Rx Handler manages message acceptance filtering, the transfer of received messages from the CAN core to the Message RAM as well as providing receive message status information. The Tx Handler is responsible for the transfer of transmit messages from the Message RAM to the CAN core as well as providing transmit status information. It implements all functions concerning the time schedule and the global system time.

Acceptance filtering is implemented by a combination of up to 128 filter elements where each one can be configured as a range, as a bit mask, or as a dedicated ID filter.

The M_TTCAN can be connected to a wide range of Host CPUs via its 8/16/32-bit Generic Slave Interface. The M_TTCAN’s clock domain concept allows the separation between the high precision CAN clock and the Host clock, which may be generated by an FMPLL.

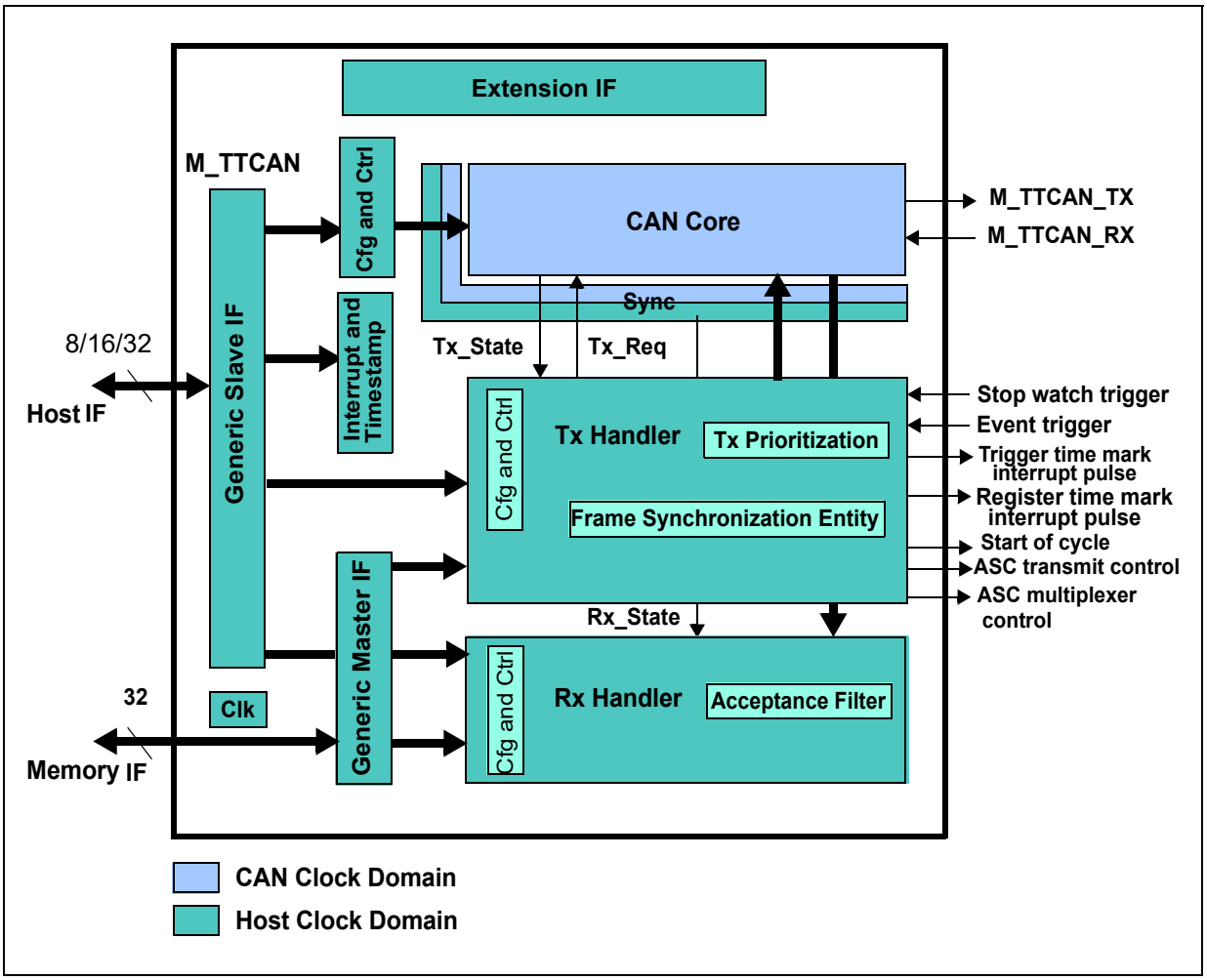
52.4.1 Features

The following are the features of M_TTCAN:

- M_TTCAN protocol level 1 and level 2 completely in hardware
- AUTOSAR optimized
- SAE J1939 optimized
- Conform with CAN protocol version 2.0 part A, B and ISO 11898-1, -4
- ISO CAN FD with max. 64 data bytes supported
- TTCAN protocol level 1 and level 2 completely in hardware
- Event synchronized time-triggered communication supported
- CAN Error Logging
- AUTOSAR optimized
- SAE J1939 optimized
- Improved acceptance filtering
- Two configurable Receive FIFOs
- Separate signaling on reception of High Priority Messages
- Up to 64 dedicated Receive Buffers
- Up to 32 Transmit Buffers
- Configurable Transmit FIFO / Queue
- Configurable Transmit Event FIFO
- Direct Message RAM access for Host CPU
- Multiple M_(TT)CAN share the same Message RAM
- Programmable loop-back test mode
- Maskable module interrupts
- 8/16/32 bit Generic Slave Interface for connection customer-specific Host CPUs
- Two clock domains (CAN clock and Host clock)
- Power-down support
- Debug on CAN support

52.4.2 Block Diagram

Figure 781. M_TTCAN core block diagram



- CAN core: CAN Protocol Controller and Rx/Tx Shift Register. Handles all ISO 11898-1 protocol functions. Supports 11-bit and 29-bit identifiers.
- Sync: Synchronizes signals from the Host clock domain to the CAN clock domain and vice versa.
- Clk: Synchronizes reset signal to the Host clock domain and to the CAN clock domain.
- Cfg and Ctrl: CAN core related configuration and control bits.
- Interrupt and Timestamp: Interrupt control and 16-bit CAN bit time counter for receive and transmit timestamp generation. An externally generated 16-bit vector may substitute the integrated 16-bit CAN bit time counter for receive and transmit timestamp generation.
- Tx Handler: Controls the message transfer from the external Message RAM to the CAN core. A maximum of 32 Tx Buffers can be configured for transmission. FIFO stores Tx timestamps together with the corresponding Message ID. Transmit cancellation is also supported. The Tx Handler also implements the Frame Synchronization Entity FSE which controls time-triggered communication according to ISO11898-4. It synchronizes itself to the reference messages on the CAN bus, controls cycle time and global time,

and handles transmissions according to the predefined message schedule, the system matrix. It also handles the time marks of the system matrix that are linked to the messages in the Message RAM. Stop Watch Trigger, Event Trigger, and Time Mark Interrupt are synchronization interfaces.

- Rx Handler: Controls the transfer of received messages from the CAN core to the external Message RAM. The Rx Handler supports two Receive FIFOs, each of configurable size, and up to 64 dedicated Rx Buffers for storage of all messages that have passed acceptance filtering. A dedicated Rx Buffer, in contrast to a Receive FIFO, is used to store only messages with a specific identifier. An Rx timestamp is stored together with each message. Up to 128 filters can be defined for 11-bit IDs and up to 64 filters for 29-bit IDs
- Generic Slave Interface: Connects the M_TTCAN to a customer specific Host CPU. The Generic Slave Interface is capable to connect to an 8-/16-/32-bit bus to support a wide range of interconnection structures.
- Generic Master Interface: Connects the M_TTCAN access to an external 32-bit Message RAM. The maximum Message RAM size is 16 KB $\times$ 32 bit.
- Extension Interface: All flags from the Interrupt Register IR and TT Interrupt Register TTIR as well as selected internal status and control signals are routed to this interface. The interface is intended for connection of the M_TTCAN to a module-external interrupt unit or to other module-external components. The connection of these signals is optional.

52.4.3 Dual clock sources

To improve the EMC behavior, a spread spectrum clock can be used for the Host clock domain. Due to the high precision clocking requirements of the CAN core, a separate clock without any modulation has to be provided as M_TTCAN clock. The CAN core should be programmed to have at least 8 clocks per bit time, this is, for example, 1 Mbaud at M_TTCAN clock $\geq$ 8 MHz. Even in case of a very high Host clock frequency, the clock frequency of the CAN core needs not to be higher than 8 MHz.

Within the M_TTCAN module there is a synchronization mechanism implemented to ensure save data transfer between the two clock domains.

Note: In order to achieve a stable function of the M_TTCAN, the Host clock must always be faster than or equal to the CAN clock. Also the modulation depth of the spread spectrum clock has to be regarded.

Caution: Stop this module when the device is in STOP mode. If CAN message reception is mandatory, check that the PLL is unlocked before transmitting any message. Use the external oscillator (XOSC) instead of the internal RC oscillator (IRCOSC) during the STOP mode.

52.4.4 Dual interrupt lines

The module provides two interrupt lines. Interrupts can be routed either to TTCAN interrupt 0 or TTCAN interrupt 1. By default all interrupts are routed to interrupt line TTCAN interrupt 0. By programming ILE.EINT0 and ILE.EINT1 the interrupt lines can be enabled or disabled separately.

52.4.5 Memory map and registers description

52.4.5.1 Hardware reset description

After hardware reset, the registers of the M_TTCAN hold the reset values listed in *Table 880*. Additionally the Bus_Off state is reset and the TTCAN Tx output is set to recessive (HIGH). The value 0x0001 (CCCR[INIT] = '1') in the CC Control Register enables software initialization. The M_TTCAN does not influence the CAN bus until the CPU resets CCCR[INIT] to '0'.

52.4.5.2 Register map

The M_TTCAN module allocates an address space of 512 bytes. All registers are organized as 32-bit registers. The M_TTCAN is accessible by the Host CPU via the Generic Slave Interface using a data width of 8-bit (byte access), 16-bit (half-word access), or 32-bit (word access). Write access by the Host CPU to registers/bits marked with "P=Protected Write" is possible only with CCCR[CCE]= '1' AND CCCR[INIT]= '1'. There is a delay from writing to a command register until the update of the related status register bits due to clock domain crossing.

Table 880. M_TTCAN memory map

Address Offset	Register Name	Access(1)	Reset value	Location
0x0000	Core Release Register (CREL)	R	0x3215_0323	*Section 52.4.5.2.1*
0x0004	Endian Register (ENDN)	R	0x8765_4321	*Section 52.4.5.2.2*
0x0008–0x000B	Reserved			
0x000C	Data Bit Timing and Prescaler Register (DBTP)	RP	0x0000_0A33	*Section 52.4.5.2.3*
0x0010	Test Register (TEST)	RP	0x0000_0000	*Section 52.4.5.2.4*
0x0014	RAM Watchdog Register (RWD)	RP	0x0000_0000	*Section 52.4.5.2.5*
0x0018	CC Control Register (CCCR)	RP	0x0000_0001	*Section 52.4.5.2.6*
0x001C	Nominal Bit Timing and Prescaler Register (NBTP)	RP	0x0600_0A03	*Section 52.4.5.2.7*
0x0020	Timestamp Counter Configuration Register (TSCC)	RP	0x0000_0000	*Section 52.4.5.2.8*
0x0024	Timestamp Counter Value Register (TSCV)	RC	0x0000_0000	*Section 52.4.5.2.9*
0x0028	Timeout Counter Configuration Register (TOCC)	RP	0xFFFF_0000	*Section 52.4.5.2.10*
0x002C	Timeout Counter Value Register (TOCV)	RC	0x0000_FFFF	*Section 52.4.5.2.11*
0x0030–0x003F	Reserved			
0x0040	Error Counter Register (ECR)	R	0x0000_0000	*Section 52.4.5.2.12*
0x0044	Protocol Status Register (PSR)	RS	0x0000_0707	*Section 52.4.5.2.13*
0x0048	Transmitter Delay Compensation Register (TDCR)	RP	0x0000_0000	*Section 52.4.5.2.14*
0x004C–0x004F	Reserved			

Table 880. M_TTCAN memory map (continued)

Address Offset	Register Name	Access(1)	Reset value	Location
0x0050	Interrupt Register (IR)	R/W	0x0000_0000	Section 52.4.5.2.15
0x0054	Interrupt Enable Register (IE)	R/W	0x0000_0000	Section 52.4.5.2.16
0x0058	IInterrupt Line Select Register (LS)	R/W	0x0000_0000	Section 52.4.5.2.17
0x005C	Interrupt Line Enable Register (ILE)	R/W	0x0000_0000	Section 52.4.5.2.18
0x0060–0x007F	Reserved			
0x0080	Global Filter Configuration Register (GFC)	RP	0x0000_0000	Section 52.4.5.2.19
0x0084	Standard ID Filter Configuration Register (SIDFC)	RP	0x0000_0000	Section 52.4.5.2.20
0x0088	Extended ID Filter Configuration Register (XIDFC)	RP	0x0000_0000	Section 52.4.5.2.21
0x008C–0x008F	Reserved			
0x0090	Extended ID AND Mask Register (XIDAM)	RP	0x1FFF_FFFF	Section 52.4.5.2.22
0x0094	High Priority Message Status Register (HPMS)	R	0x0000_0000	Section 52.4.5.2.23
0x0098	New Data 1 Register (NDAT1)	R/W	0x0000_0000	Section 52.4.5.2.24
0x009C	New Data 2 Register (NDAT2)	R/W	0x0000_0000	Section 52.4.5.2.25
0x00A0	Rx FIFO 0 Configuration Register (RXF0C)	RP	0x0000_0000	Section 52.4.5.2.26
0x00A4	Rx FIFO 0 Status Register (RXF0S)	R	0x0000_0000	Section 52.4.5.2.27
0x00A8	Rx FIFO 0 Acknowledge Register (RXF0A)	R/W	0x0000_0000	Section 52.4.5.2.28
0x00AC	Rx Buffer Configuration Register (RXBC)	R/W	0x0000_0000	Section 52.4.5.2.29
0x00B0	Rx FIFO 1 Configuration Register (RXF1C)	RP	0x0000_0000	Section 52.4.5.2.30
0x00B4	Rx FIFO 1 Status Register (RXF1S)	R	0x0000_0000	Section 52.4.5.2.31
0x00B8	Rx FIFO 1 Acknowledge Register (RXF1A)	R/W	0x0000_0000	Section 52.4.5.2.32
0x00BC	Rx Buffer / FIFO Element Size Configuration (RXESC)	RP	0x0000_0000	Section 52.4.5.2.33
0x00C0	Tx Buffer Configuration Register (TXBC)	RP	0x0000_0000	Section 52.4.5.2.34
0x00C4	Tx FIFO/Queue Status Register (TXFQS)	R	0x0000_0000	Section 52.4.5.2.35
0x00C8	Tx Buffer Element Size Configuration (TXESC)	RP	0x0000_0000	Section 52.4.5.2.36
0x00CC	Tx Buffer Request Pending Register (TXBRP)	R	0x0000_0000	Section 52.4.5.2.37
0x00D0	Tx Buffer Add Request Register (TXBAR)	R/W	0x0000_0000	Section 52.4.5.2.38
0x00D4	Tx Buffer Cancellation Register (TXBCR)	R/W	0x0000_0000	Section 52.4.5.2.39
0x00D8	Tx Buffer Transmission Occurred Register (TXBTO)	R	0x0000_0000	Section 52.4.5.2.40
0x00DC	Tx Buffer Cancellation Finished Register (TXBCF)	R	0x0000_0000	Section 52.4.5.2.41

Table 880. M_TTCAN memory map (continued)

Address Offset	Register Name	Access[1]	Reset value	Location
0x00E0	Tx Buffer Transmission Interrupt Enable (TXBTIE)	R/W	0x0000_0000	*Section 52.4.5.2.42*
0x00E4	Tx Buffer Cancellation Finished Interrupt Enable Register (TXBCIE)	R/W	0x0000_0000	*Section 52.4.5.2.43*
0x00E8–0x00EF	Reserved			
0x00F0	Tx Event FIFO Configuration Register (TXEFC)	RP	0x0000_0000	*Section 52.4.5.2.44*
0x00F4	Tx Event FIFO Status Register (TXEFS)	R	0x0000_0000	*Section 52.4.5.2.45*
0x00F8	Tx Event FIFO Acknowledge Register (TXEFA)	R/W	0x0000_0000	*Section 52.4.5.2.46*
0x00FC–0x00FF	Reserved			
0x0100	TT Trigger Memory Configuration Register (TTTMC)	RP	0x0000_0000	*Section 52.4.5.2.47*
0x0104	TT Reference Message Configuration Register (TTRMC)	RP	0x0000 0000	*Section 52.4.5.2.48*
0x0108	TT Operation Configuration Register (TTOCF)	RP	0x0001_0000	*Section 52.4.5.2.49*
0x010C	TT Matrix Limits Register (TTMLM)	RP	0x0000_0000	*Section 52.4.5.2.50*
0x0110	TUR Configuration Register (TURCF)	RP	0x1000_0000	*Section 52.4.5.2.51*
0x0114	TT Operation Control Register (TTOCN)	R/W	0x0000_0000	*Section 52.4.5.2.52*
0x0118	TT Global Time Preset Register (TTGTP)	R/W	0x0000_0000	*Section 52.4.5.2.53*
0x011C	TT Time Mark Register (TTTMK)	R/W	0x0000_0000	*Section 52.4.5.2.54*
0x0120	TT Interrupt Register (TTIR)	R/W	0x0000_0000	*Section 52.4.5.2.55*
0x0124	TT Interrupt Enable Register (TTIE)	R/W	0x0000_0000	*Section 52.4.5.2.56*
0x0128	TT Interrupt Line Select Register (TTILS)	R/W	0x0000_0000	*Section 52.4.5.2.57*
0x012C	TT Operation Status Register (TTOST)	R	0x0000_0080	*Section 52.4.5.2.58*
0x0130	TUR Numerator Actual Register (TURNA)	R	0x0001_0000	*Section 52.4.5.2.59*
0x0134	TT Local and Global Time Register (TTLGT)	R	0x0000_0000	*Section 52.4.5.2.60*
0x0138	TT Cycle Time and Count Register (TTCTC)	R	0x003F_0000	*Section 52.4.5.2.61*
0x013C	TT Capture Time Register (TTCPT)	R	0x0000_0000	*Section 52.4.5.2.62*
0x0140	TT Cycle Sync Mark Register (TTCSM)	R	0x0000_0000	*Section 52.4.5.2.63*
0x0144–0x01FF	Reserved			

1. R = Read, S = Set on read, X = Reset on read, W = Write, P = Protected Write, C = Clear/preset on write

52.4.5.2.1 Core Release Register (CREL)

Offset: 0x0000 Access: Read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	REL				STEP				SUBSTEP				YEAR			
W																
Reset(1)	-r	-r	-r	-r	-r	-r	-r	-r	-r	-r	-r	-r	-d	-d	-d	-d

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	MON								DAY							
W																
Reset(1)	-d	-d	-d	-d	-d	-d	-d	-d	-d	-d	-d	-d	-d	-d	-d	-d

1. -r = release, -d = time stamp. The coding of revisions depends on the module version used in the device. For the device reset value refer to *Table 880: M_TTCAN memory map*.

Figure 782. Core Release Register (CREL)

Table 881. CREL field descriptions

Field	Description
0:3 REL	Core Release One digit, BCD.
4:7 STEP	Step of Core Release One digit, BCD.
8:11 SUBSTEP	Sub-step of Core Release One digit, BCD.
12:15 YEAR	Time Stamp Year One digit, BCD..
16:23 MON	Time Stamp Month Two digits, BCD.
24:31 DAY	Time Stamp Day Two digits, BCD.

52.4.5.2.2 Endian Register (ENDN)

Offset: 0x0004 Access: R

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	ETV															
W																
Reset	1	0	0	0	0	1	1	1	0	1	1	0	0	1	0	1

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	ETV															
W																
Reset	0	1	0	0	0	0	1	1	0	0	1	0	0	0	0	1

Figure 783. Endian Register (ENDN)

Table 882. ENDN field descriptions

Field	Description
0:31 ETV	Endianness Test Value The endianness test value is 0x87654321.

52.4.5.2.3 Data Bit Timing and Prescaler register (DBTP)

This register is only writable if bits CCCR.CCE and CCCR.INIT are set. The CAN bit time may be programmed in the range of 4 to 49 time quanta. The CAN time quantum may be programmed in the range of 1 to 32 M_TTCAN clock periods.

Equation 31 tq = (DBRP + 1) M_TTCAN clock period

DTSEG1 is the sum of Prop_Seg and Phase_Seg1. DTSEG2 is Phase_Seg2.

Therefore the length of the bit time is (programmed values) [DTSEG1 + DTSEG2 + 3] tq or (functional values) [Sync_Seg + Prop_Seg + Phase_Seg1 + Phase_Seg2] tq.

The Information Processing Time (IPT) is zero, meaning the data for the next bit is available at the first clock edge after the sample point.

Offset: 0x000C Access: Read Protected

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	TDC(1)	0	0	DBRP(1)				
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	DTSEG1(1)					DTSEG2(1)				DSJW(1)			
W																
Reset	0	0	0	0	1	0	1	0	0	0	1	1	0	0	1	1

1. These are protected write (P) bits which means that write access by the bits is possible only when the bit 1 [CCE] and bit 0 [INIT] of CCCR register are set to ‘1’.

Figure 784. Data Bit Timing and Prescaler Register (DBTP)

Table 883. DBTP field descriptions

Field	Description
8 TDC	Transmitter Delay Compensation 0 Transmitter Delay Compensation disabled 1 Transmitter Delay Compensation enabled
11:15 DBRP	Data Bit Rate Prescaler (0x00–0x1F) The value by which the oscillator frequency is divided for generating the bit time quanta. The bit time is built up from a multiple of this quanta. Valid values for the Bit Rate Prescaler are 0 to 31. The actual interpretation by the hardware of this value is such that one more than the value programmed here is used.
19:23 DTSEG1	Data time segment before sample point (0x1–0xF) Valid values are 1 to 15. The actual interpretation by the hardware of this value is such that one more than the programmed value is used.
24:27 DTSEG2	Data time segment after sample point (0x0–0xF) Valid values are 0 to 7. The actual interpretation by the hardware of this value is such that one more than the programmed value is used.
28:31 DSJW	Data (Re) Synchronization Jump Width (0x0–0xF) Valid values are 0 to 3. The actual interpretation by the hardware of this value is such that one more than the value programmed here is used.

Note: With a M_TTCAN clock of 8 MHz, the reset value of 0x00000A33 configures the M_TTCAN for a fast bit rate of 500 kbit/s.

The bit rate configured for the CAN FD data phase via DBTP must be superior or equal to the bit rate configured for the arbitration phase via NBTP.

52.4.5.2.4 Test Register (TEST)

Write access to the Test Register has to be enabled by setting bit CCCR[TEST] to ‘1’. All Test Register functions are set to their reset values when bit CCCR[TEST] is reset.

Loop Back mode and software control of TTCAN Tx pin are hardware test modes. Programming of TX ≠ ‘00’ may disturb the message transfer on the CAN bus.

Offset: 0x0010 Access: RP

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	RX	TX(1)		LBCK(1)	CAT	CAM	TAT(1)	TAM(1)
W																
Reset	0	0	0	0	0	0	0	0	—	0	0	0	0	0	0	0

1. These are protected write (P) bits which means that write access by the bits is possible only when the bit 1 [CCE] and bit 0 [INIT] of CCCR register are set to ‘1’.

Figure 785. Test Register (TEST)

Table 884. TEST field descriptions

Field	Description
24 RX	Receive Pin Monitors the actual value of TTCAN Rx input pin 0 The CAN bus is dominant 1 The CAN bus is recessive
25:26 TX	Control of Transmit Pin 00 Reset value, TTCAN Tx output pinis controlled by the CAN core, updated at the end of the CAN bit time 01 Sample Point can be monitored at TTCAN Tx output pin 10 Dominant (‘0’) level at TTCAN Tx output pin 11 Recessive (‘1’) at TTCAN Tx output pin
27 LBCK	Loop Back mode 0 Reset value, Loop Back mode is disabled 1 Loop Back mode is enabled (refer to *Section 52.4.7.10: Test modes*)
28 CAT	Check ASC Transmit Control Monitors level at M_TTCAN ASC transmit control. 0 Output M_TTCAN ASC transmit control pin = ‘0’ 1 Output M_TTCAN ASC transmit control pin = ‘1’
29 CAM	Check ASC Multiplexer Control Monitors level at M_TTCAN ASC multiplexer control. 0 Output M_TTCAN ASC multiplexer control pin = ‘0’ 1 Output M_TTCAN ASC multiplexer control pin = ‘1’

Table 884. TEST field descriptions (continued)

Field	Description
30 TAT	Test ASC Transmit Control Controls output ASC Transmit Control pin in test mode, ORed with the signal from the FSE. 0 Level at M_TTCAN ASC transmit control pin controlled by FSE 1 Level at M_TTCAN ASC transmit control pin = '1'
31 TAM	Test ASC Multiplexer Control Controls output M_TTCAN ASC multiplexer control in test mode, ORed with the signal from the FSE. 0 Level at M_TTCAN ASC multiplexer control pin controlled by FSE 1 Level at M_TTCAN ASC multiplexer control pin = '1'

52.4.5.2.5 RAM Watchdog Register (RWD)

The RAM Watchdog monitors the READY output of the Message RAM .A Message RAM access via the M_CAN's Generic Master Interface starts the Message RAM Watchdog Counter with the value configured by the RWD[WDC] bits. The counter is reloaded with RWD[WDC] bits when the Message RAM signals successful completion by activating its READY output. In case there is no response from the Message RAM until the counter has counted down to zero, the counter stops and interrupt flag IR[WDI] bit is set. The RAM Watchdog Counter is clocked by the Host clock.

Offset: 0x0014 Access: RP

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	WDV								WDC(1)							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 786. RAM Watchdog Register (RWD)

1. These are protected write (P) bits which means that write access by the bit is possible only when the bit 1 [CCE] and bit 0 [INIT] of CCCR register are set to '1'.

Table 885. RWD field descriptions

Field	Description
16:23 WDV	Watchdog Value Actual Message RAM Watchdog Counter Value.
24:31 WDC	Watchdog Configuration Start value of the Message RAM Watchdog Counter. With the reset value of '00' the counter is disabled.

52.4.5.2.6 CC Control Register (CCCR)

For details about setting and resetting of single bits refer to *Section 52.4.7.1.1: Software initialization*.

Offset: 0x0018 Access: Read

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	NISO(1)	TXP(1)	EFBI(1)	PXHD(1)	0	0	BRSE(1)	FDOE(1)	TEST(1)	DAR(1)	MON(1)	CSR	CSA	ASM(1)	CCE(1)	INIT
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

1. These are protected write (P) bits which means that write access by the bits is possible only when the bit 1 [CCE] and bit 0 [INIT] of CCCR register are set to '1'.

Figure 787. CC Control Register (CCCR)

Table 886. CCCR field descriptions

Field	Description
16 NISO	Non ISO Operation If this bit is set, the M_TTCAN uses the CAN FD frame format as specified by the Bosch CAN FD Specification V1.0. 0 CAN FD frame format according to ISO11898-1 1 CAN FD frame format according to Bosch CAN FD Specification V1.0
17 TXP	TXP Transmit Pause If this bit is set, the M_TTCAN pauses for two CAN bit times before starting the next transmission after itself has successfully transmitted a frame (refer to *Section 52.4.7.19: Transmit Pause*). 0 Transmit pause disabled 1 Transmit pause enabled
18 EFBI	CAN FD Bit Rate Switching 0 This node transmits no frames with bit rate switching 1 This node transmits all frames (excluding remote frames) with bit rate switching
19 PXHD	CAN FD Operation 0 This node transmits all frames in CAN format according to ISO11898-1 1 This node transmits all frames (excluding remote frames) in CAN FD format **Note:** When protocol exception handling is disabled, the M_TTCAN will transmit an error frame when it detects a protocol exception condition.
22 BRSE	Bit Rate Switch Enable 0 Bit rate switching for transmissions disabled 1 Bit rate switching for transmissions enabled **Note:** When CAN FD operation is disabled FDOE = '0', BRSE is not evaluated.

Table 886. CCCR field descriptions (continued)

Field	Description
23 FDOE	FD Operation Enable 0 FD operation disabled 1 FD operation enabled
24 TEST	Test Mode Enable 0 Normal operation, register TEST holds reset values 1 Test Mode, write access to register TEST enabled
25 DAR	Disable Automatic Retransmission 0 Automatic retransmission of messages not transmitted successfully enabled 1 Automatic retransmission disabled
26 MON	Bus Monitoring Mode Bit MON can only be set by the Host when both CCE and INIT are set to ‘1’. The bit can be reset by the Host at any time. 0 Bus Monitoring Mode is disabled 1 Bus Monitoring Mode is enabled
27 CSR	Clock Stop Request 0 No clock stop is requested 1 Clock stop requested. When clock stop is requested, first INIT and then CSA will be set after all pending transfer requests have been completed and the CAN bus reached idle.
28 CSA	Clock Stop Acknowledge 0 No clock stop acknowledged 1 M_TTCAN may be set in power down by stopping Host clock and Core CAN clock
29 ASM	ASM Restricted Operation Mode The Restricted Operation Mode is intended for applications that adapt themselves to different CAN bit rates. The application tests different bit rates and leaves the Restricted Operation Mode after it has received a valid frame. In the optional Restricted Operation Mode the node is able to transmit and receive data and remote frames and it gives acknowledge to valid frames, but it does not send active error frames or overload frames. In case of an error condition or overload condition, it does not send dominant bits, instead it waits for the occurrence of bus idle condition to resynchronize itself to the CAN communication. The error counters are not incremented. Bit ASM can only be set by the Host when both CCE and INIT are set to ‘1’. The bit can be reset by the Host at any time. If the M_TTCAN is connected to a Clock Calibration on CAN Unit, ASM is controlled by calibration input signal. In case calibration ” switches to ‘0’, bit ASM is set. When calibration ” switches back to ‘1’, bit ASM returns to the previously written value. The state of ASM is the written value while input calibration” is at ‘1’. The input is hardwired to ‘1’ when there is no Clock Calibration on CAN Unit connected. 0 Normal CAN operation 1 Restricted Operation Mode active

Table 886. CCCR field descriptions (continued)

Field	Description
30 CCE	Configuration Change Enable 0 The CPU has no write access to the protected configuration registers 1 The CPU has write access to the protected configuration registers (while CCCR.INIT = '1')
31 INIT	Initialization 0 Normal Operation 1 Initialization is started

Note: Due to the synchronization mechanism between the two clock domains, there may be a delay until the value written to INIT can be read back. Therefore the programmer has to assure that the previous value written to INIT has been accepted by reading INIT before setting INIT to a new value.

52.4.5.2.7 Nominal Bit Timing and Prescaler Register (NBTP)

This register is only writable if bits CCCR[CCE] and CCCR[INIT] are set. The CAN bit time may be programed in the range of 4 to 385 time quanta (tq). The CAN time quantum may be programmed in the range of 1 to 512 core CAN clock periods.

Equation 32 tq = (NBRP + 1) core CAN clock period

NTSEG1 is the sum of Prop_Seg and Phase_Seg1. NTSEG2 is Phase_Seg2.

Therefore the length of the bit time is (programmed values) [NTSEG1 + NTSEG2 + 3] tq or (functional values) [Sync_Seg + Prop_Seg + Phase_Seg1 + Phase_Seg2] tq.

The Information Processing Time (IPT) is zero, meaning the data for the next bit is available at the first clock edge after the sample point.

Offset: 0x001C Access: Read Protected

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	NSJW(1)								NBRP(1)							
Reset	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	NTSEG1(1)								0	NTSEG2(1)						
Reset	0	0	0	0	1	0	1	0	0	0	0	0	0	0	1	1

1. These are protected write (P) bits which means that write access by the bits is possible only when the bit 1 [CCE] and bit 0 [INIT] of CCCR register are set to '1'.

Figure 788. Nominal Bit Timing and Prescaler Register (NBTP)

Table 887. NBTP field descriptions

Field	Description
0:6 NSJW	Nominal (Re) Synchronization Jump Width (0x00–0x7F) Valid values are 0 to 127. The actual interpretation by the hardware of this value is such that one more than the value programmed here is used.
7:15 NBRP	Nominal Baud Rate Prescaler (0x000–0x1FF) The value by which the oscillator frequency is divided for generating the bit time quanta. The bit time is built up from a multiple of this quanta. Valid values for the Baud Rate Prescaler are 0 to 511. The actual interpretation by the hardware of this value is such that one more than the value programmed here is used.
16:23 NTSEG1	Nominal Time segment before sample point (0x1–0xFF) Valid values are 1 to 255. The actual interpretation by the hardware of this value is such that one more than the programmed value is used.
25:31 NTSEG2	Nominal Time segment after sample point (0x00–0x7F) Valid values are 0 to 127. The actual interpretation by the hardware of this value is such that one more than the programmed value is used.

Note: With a CAN clock of 8 MHz, the reset value of 0x06000A33 configures the M_TTCAN for a bit rate of 500 kbit/s.

52.4.5.2.8 Timestamp Counter Configuration Register (TSCC)

Offset: 0x0020 Access: RP

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	TCP(1)			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	TSS(1)	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. These are protected write (P) bits which means that write access by the bit is possible only when the bit 1 [CCE] and bit 0 [INIT] of CCCR register are set to ‘1’.

Figure 789. Timestamp Counter Configuration Register (TSCC)

Table 888. TSCC field descriptions

Field	Description
12:15 TCP	Timestamp Counter Prescaler (0x0–0xF) Configures the timestamp and timeout counters time unit in multiples of CAN bit times [1…16]. The actual interpretation by the hardware of this value is such that one more than the value programmed here is used. **Note:** .
30:31 TSS	Timestamp Select 00 Timestamp counter value always 0x0000 01 Timestamp counter value incremented according to TCP 10 Reserved for both SPC58xEx/ SPC58xGx 11 Same as ‘00’ **Note:** In case of CAN FD mode the absolute duration of a bit time does change in one CAN message due to the different baud rates for arbitration phase and data phase. Therefore an external counter (available through GTM supported only by SPC58xEx) is request for timestamp generation. Take care on SPC58xGx where the internal timestamp counter does still reflect increasing number on bit times but the fix relationship to absolute time is not given any more. It is possible to establish in which order the frames are received.

52.4.5.2.9 Timestamp Counter Value Register (TSCV)

Offset: 0x0024 Access: R/W

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	TSC															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 790. Timestamp Counter Value Register (TSCV)

Table 889. TSCV field descriptions

Field	Description
16:31 TSC	Timestamp Counter The internal Counter value is captured on start of frame (both Rx and Tx). When TSCC[TSS] = ‘01’, the Timestamp Counter is incremented in multiples of CAN bit times [1…16] depending on the configuration of TSCC[TCP]. A wrap around sets interrupt flag IR[TSW]. Write access resets the counter to *zero*.

Note: A "wrap around" is a change of the Timestamp Counter value from non-zero to zero not caused by write access to TSCV.

52.4.5.2.10 Timeout Counter Configuration Register (TOCC)

For a description of the Timeout Counter, refer to *Section 52.4.7.12: Timeout counter*.

Offset: 0x0028 Access: RP

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	TOP(1)															
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	TOS(1)		ETOC(1)
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. These are protected write (P) bits which means that write access by the bit is possible only when the bit 1 [CCE] and bit 0 [INIT] of CCCR register are set to '1'.

Figure 791. Timeout Counter Configuration Register (TOCC)

Table 890. TOCC field descriptions

Field	Description
0:15 TOP	Timeout Period Start value of the Timeout Counter (down-counter). Configures the Timeout Period.
29:30 TOS	Timeout Select When operating in Continuous mode, a write to **TOCV** presets the counter to the value configured by **TOCC[TOP]** and continues down-counting. When the Timeout Counter is controlled by one of the FIFOs, an empty FIFO presets the counter to the value configured by **TOCC[TOP]**. Down-counting is started when the first FIFO element is stored. 00 Continuous operation 01 Timeout controlled by Tx Event FIFO 10 Timeout controlled by Rx FIFO 0 11 Timeout controlled by Rx FIFO 1
31 ETOC	Enable Timeout Counter 0 Timeout Counter disabled 1 Timeout Counter enabled **Note:** For use of timeout function with CAN FD, refer to *Section 52.4.7.12: Timeout counter*".

52.4.5.2.11 Timeout Counter Value Register (TOCV)

Offset: 0x002C Access: R/W

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	TOC															
W																
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Figure 792. Timeout Counter Value Register (TOCV)

Table 891. TOCV field descriptions

Field	Description
16:31 TOC	Timeout Counter The Timeout Counter is decremented in multiples of CAN bit times [1…16] depending on the configuration of TSCC.TCP. When decremented to zero, interrupt flag IR.TOO is set and the Timeout Counter is stopped. Start and reset/restart conditions are configured via TOCC.TOS.

52.4.5.2.12 Error Counter Register (ECR)

Offset: 0x0040 Access: R

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	CEL(1)							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	RP	REC							TEC							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. Access type is RX: reset on read.

Figure 793. Error Counter Register (ECR)

Table 892. ECR field descriptions

Field	Description
8:15 CEL	CAN Error Logging The counter is incremented each time when a CAN protocol error causes the Transmit Error Counter or the Receive Error Counter to be incremented. It is reset by read access to CEL. The counter stops at 0xFF; the next increment of TEC or REC sets interrupt flag IR[ELO].
16 RP	Receive Error Passive 0 The Receive Error Counter is below the *error passive* level of 128 1 The Receive Error Counter has reached the *error passive* level of 128
17:23 REC	Receive Error Counter Actual state of the Receive Error Counter, values between 0 and 127.
24:31 TEC	Transmit Error Counter Actual state of the Transmit Error Counter, values between 0 and 255.

Note: When CCCR[ASM] is set, the CAN protocol controller does not increment TEC and REC when a CAN protocol error is detected, but CEL is still incremented. This enables monitoring of collisions between CAN frames and ASC frames.

52.4.5.2.13 Protocol Status Register (PSR)

Offset: 0x0044 Access: RS

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	TDCV						
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	PXE	RFDF(1)	RBRS(1)	RESI(1)	DLEC(2)			BO	EW	EP	ACT		LEC(2)		
W																
Reset	0	0	0	0	0	1	1	1	0	0	0	0	0	1	1	1

1. Access type is RX: reset on read.
2. Access type is RS: set on read.

Figure 794. Protocol Status Register (PSR)

Table 893. PSR field descriptions

Field	Description
9:15 TDCV	Transmitter Delay Compensation Value (0x00-0x7F) Position of the secondary sample point, defined by the sum of the measured delay from TTCAN Tx output to TTCAN Rx input and TDCR.TDCO. The SSP position is, in the data phase, the number of mtq between the start of the transmitted bit and the secondary sample point Valid values are 0 to 127 mtq.
17 PXE	Protocol Exception Event 0 No protocol exception event occurred since last read access 1 Protocol exception event occurred
18 RFDF	Received CAN FD Message This bit is set independent of acceptance filtering. 0 Since this bit was reset by the CPU, no CAN FD message has been received 1 Message in CAN FD format with EDL flag set has been received
19 RBRS	BRS flag of last received CAN FD Message This bit is set together with REDL, independent of acceptance filtering. 0 Last received CAN FD message did not have its BRS flag set. 1 Last received CAN FD message had its BRS flag set.
20 RESI	ESI flag of last received CAN FD Message This bit is set together with REDL, independent of acceptance filtering. 0 Last received CAN FD message did not have its ESI flag set. 1 Last received CAN FD message had its ESI flag set
21:23 DLEC	Data phase Last Error Code Type of last error that occurred in the data phase of a CAN FD format frame with its BRS flag set. Coding is the same as for LEC. This field will be cleared to zero when a CAN FD format frame with its BRS flag set has been transferred (reception or transmission) without error.
24 BO	Bus_Off Status 0 The M_TTCAN is not Bus_Off 1 The M_TTCAN is in Bus_Off state
25 EW	Warning Status 0 Both error counters are below the Error_Warning limit of 96 1 At least one of error counter has reached the Error_Warning limit of 96
26 EP	Error Passive 0 The M_TTCAN is in the Error_Active state. It normally takes part in bus communication and sends an active error flag when an error has been detected 1 The M_TTCAN is in the Error_Passive state

Table 893. PSR field descriptions (continued)

Field	Description
27:28 ACT	Activity Monitors the module's CAN communication state. 00 Synchronizing: node is synchronizing on CAN communication 01 Idle: node is neither receiver nor transmitter 10 Receiver: node is operating as receiver 11 Transmitter: node is operating as transmitter **Note:** ACT is set to "00" by a Protocol Exception Event.
29:31 LEC	Last Error Code The LEC indicates the type of the last error to occur on the CAN bus. This field will be cleared to '0' when a message has been transferred (reception or transmission) without error. 000 No Error: No error occurred since LEC has been reset by successful reception or transmission. 001 Stuff Error: More than 5 equal bits in a sequence have occurred in a part of a received message where this is not allowed. 010 Form Error: A fixed format part of a received frame has the wrong format. 011 AckError: The message transmitted by the M_TTCAN was not acknowledged by another node. 100 Bit1Error: During the transmission of a message (with the exception of the arbitration field), the device wanted to send a recessive level (bit of logical value '1'), but the monitored bus value was dominant. 101 Bit0Error: During the transmission of a message (or acknowledge bit, or active error flag, or overload flag), the device wanted to send a dominant level (data or identifier bit logical value '0'), but the monitored bus value was recessive. During Bus_Off recovery this status is set each time a sequence of 11 recessive bits has been monitored. This enables the CPU to monitor the proceeding of the Bus_Off recovery sequence (indicating the bus is not stuck at dominant or continuously disturbed). 110 CRCError: The CRC check sum of a received message was incorrect. The CRC of an incoming message does not match with the CRC calculated from the received data. 111 NoChange: Any read access to the Protocol Status Register re-initializes the LEC to '7'. When the LEC shows the value '7', no CAN bus event was detected since the last CPU read access to the Protocol Status Register.

Note: When a frame in CAN FD format has reached the data phase with BRS flag set, the next CAN event (error or valid frame) will be shown in DLEC instead of LEC. An error in a fixed stuff bit of a CAN FD CRC sequence will be shown as a Form Error, not Stuff Error

The Bus_Off recovery sequence (refer to CAN Specification Rev. 2.0 or ISO11898-1) cannot be shortened by setting or resetting CCCR[INIT]. If the device goes Bus_Off, it will set CCCR.INIT of its own accord, stopping all bus activities. Once CCCR[INIT] has been cleared by the CPU, the device will then wait for 129 occurrences of Bus Idle (129 $\times$ 11 consecutive recessive bits) before resuming normal operation. At the end of the Bus_Off recovery sequence, the Error Management Counters will be reset. During the waiting time after the resetting of CCCR[INIT], each time a sequence of 11 recessive bits has been monitored, a Bit0 Error code is written to PSR[LEC], enabling the CPU to readily check up whether the CAN bus is stuck at dominant or continuously disturbed and to monitor the Bus_Off recovery sequence. ECR[REC] is used to count these sequences.

52.4.5.2.14 Transmitter Delay Compensation Register (TDCR)

Offset: 0x0048 Access: RP

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	TDCO(1)							0	TDCF(1)						
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. These are protected write (P) bits which means that write access by the bit is possible only when the bit 1 [CCE] and bit 0 [INIT] of CCCR register are set to ‘1’.

Figure 795. Transmitter Delay Compensation Register (TDCR)

Table 894. TDCR field descriptions

Field	Description
17:23 TDCO	Transmitter Delay Compensation Offset (0x00-0x7F) Offset value defining the distance between the measured delay from TTCAN Tx output to M_TTCAN Rx input and the secondary sample point. Valid values are 0 to 127 mtq.
25:31 TDCF	Transmitter Delay Compensation Filter Window Length (0x00-0x7F) Defines the minimum value for the SSP position, dominant edges on TTCAN Rx input that would result in an earlier SSP position are ignored for transmitter delay measurement. The feature is enabled when TDCF is configured to a value greater than TDCO. Valid values are 0 to 127 mtq.

52.4.5.2.15 Interrupt Register (IR)

The flags are set when one of the listed conditions is detected (edge-sensitive). The flags remain set until the Host clears them. A flag is cleared by writing a ‘1’ to the corresponding bit position. Writing a ‘0’ has no effect. A hard reset will clear the register. The configuration of **IE** controls whether an interrupt is generated. The configuration of **ILS** controls on which interrupt line an interrupt is signaled.

Offset: 0x0050 Access: R/W

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	ARA	PED	PEA	WDI	BO	EW	EP	ELO	BEU	BEC	DRX	TOO	MRAF	TSW
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	TEFL	TEFF	TEFW	TEFN	TFE	TCF	TC	HPM	RF1L	RF1F	RF1W	RF1N	RF0L	RF0F	RF0W	RF0N
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 796. Interrupt Register (IR)

Table 895. IR field descriptions

Field	Description
2 ARA	Access to Reserved Address 0 No access to reserved address occurred 1 Access to reserved address occurred
3 PED	Protocol Error in Data phase (Data Bit Time is used) 0 No protocol error in data phase 1 Protocol error in data phase detected (PSR.DLEC ≠ 0,7)
4 PEA	Protocol Error in Arbitration Phase (Nominal Bit Time is used) 0= No protocol error in arbitration phase 1= Protocol error in arbitration phase detected (PSR.LEC ≠ 0,7)
5 WDI	Watchdog Interrupt 0 No Message RAM Watchdog event occurred 1 Message RAM Watchdog event due to missing READY
6 BO	Bus_Off Status 0 Bus_Off status unchanged 1 Bus_Off status changed
7 EW	Warning Status 0 Error_Warning status unchanged 1 Error_Warning status changed
8 EP	Error Passive 0 Error_Passive status unchanged 1 Error_Passive status changed
9 ELO	Error Logging Overflow 0 CAN Error Logging Counter did not overflow 1 Overflow of CAN Error Logging Counter occurred
10 BEU	Bit Error Uncorrected Message RAM bit error detected, uncorrected. Controlled by Message RAM bit error input signal generated by an optional external parity / ECC logic attached to the Message RAM. An uncorrected Message RAM bit error sets CCCR[INIT] to ‘1’. This is done to avoid transmission of corrupted data. 0 No bit error detected when reading from Message RAM 1 Bit error detected, uncorrected (for example parity logic)

Table 895. IR field descriptions (continued)

Field	Description
11 BEC	Bit Error Corrected Message RAM bit error detected and corrected. Controlled by Message RAM bit error input signal generated by an optional external parity / ECC logic attached to the Message RAM. 0 No bit error detected when reading from Message RAM 1 Bit error detected and corrected (for example ECC)
12 DRX	Message stored to Dedicated Rx Buffer The flag is set whenever a received message has been stored into a dedicated Rx Buffer. 0 No Rx Buffer updated 1 At least one received message stored into a Rx Buffer
13 TOO	Timeout Occurred 0 No timeout 1 Timeout reached
14 MRAF	Message RAM Access Failure The flag is set, when the Rx Handler – has not completed acceptance filtering or storage of an accepted message until the arbitration field of the following message has been received. In this case acceptance filtering or message storage is aborted and the Rx Handler starts processing of the following message. – was not able to write a message to the Message RAM. In this case message storage is aborted. In both cases the FIFO put index is not updated resp. the New Data flag for a dedicated Rx Buffer is not set, a partly stored message is overwritten when the next message is stored to this location. The flag is also set when the Tx Handler was not able to read a message from the Message RAM in time. In this case message transmission is aborted. In case of a Tx Handler access failure the M_TTCAN is switched into Restricted Operation Mode. To leave Restricted Operation Mode, the Host CPU has to reset CCCR.ASM. 0 No Message RAM access failure occurred 1 Message RAM access failure occurred
15 TSW	Timestamp Wraparound 0 No timestamp counter wrap-around 1 Timestamp counter wrapped around
16 TEFL	Tx Event FIFO Element Lost 0 No Tx Event FIFO element lost 1 Tx Event FIFO element lost, also set after write attempt to Tx Event FIFO of size zero
17 TEFF	Tx Event FIFO Full 0 Tx Event FIFO not full 1 Tx Event FIFO full
18 TEFW	Tx Event FIFO Watermark Reached 0 Tx Event FIFO fill level below watermark 1 Tx Event FIFO fill level reached watermark

Table 895. IR field descriptions (continued)

Field	Description
19 TEFN	Tx Event FIFO New Entry 0 Tx Event FIFO unchanged 1 Tx Handler wrote Tx Event FIFO element
20 TFE	Tx FIFO Empty 0 Tx FIFO non-empty 1 Tx FIFO empty
21 TCF	Transmission Cancellation Finished 0 No transmission cancellation finished 1 Transmission cancellation finished
22 TC	Transmission Completed 0 No transmission completed 1 Transmission completed
23 HPM	High Priority Message 0 No high priority message received 1 High priority message received
24 RF1L	Rx FIFO 1 Message Lost 0 No Rx FIFO 1 message lost 1 Rx FIFO 1 message lost, also set after write attempt to Rx FIFO 1 of size zero
25 RF1F	Rx FIFO 1 Full 0 Rx FIFO 1 not full 1 Rx FIFO 1 full
26 RF1W	Rx FIFO 1 Watermark Reached 0 Rx FIFO 1 fill level below watermark 1 Rx FIFO 1 fill level reached watermark
27 RF1N	Rx FIFO 1 New Message 0 No new message written to Rx FIFO 1 1 New message written to Rx FIFO 1
28 RF0L	Rx FIFO 0 Message Lost 0 No Rx FIFO 0 message lost 1 Rx FIFO 0 message lost, also set after write attempt to Rx FIFO 0 of size zero
29 RF0F	Rx FIFO 0 Full 0 Rx FIFO 0 not full 1 Rx FIFO 0 full
30 RF0W	Rx FIFO 0 Watermark Reached 0 Rx FIFO 0 fill level below watermark 1 Rx FIFO 0 fill level reached watermark
31 RF0N	Rx FIFO 0 New Message 0 No new message written to Rx FIFO 0 1 New message written to Rx FIFO 0

52.4.5.2.16 Interrupt Enable Register (IE)

The settings in the Interrupt Enable register determine which status changes in the Interrupt Register will be signaled on an interrupt line.

Offset: 0x0054 Access: R/W

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	ARAE	PEDE	PEAE	WDIE	BOE	EWE	EPE	ELOE	BEUE	BECE	DRXE	TOOE	MRAFE	TSWE
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	TEFLE	TEFFE	TEFWE	TEFNE	TFEE	TCFE	TCE	HPME	RF1LE	RF1FE	RF1WE	RF1NE	RF0LE	RF0FE	RF0WE	RF0NE
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 797. Interrupt Enable Register (IE)

Table 896. IE field descriptions

Field	Description
2 ARAE	Access to Reserved Address Enable 0 Interrupt disabled 1 Interrupt enabled
3 PEDE	Protocol Error in Data phase Enable 0 Interrupt disabled 1 Interrupt enabled
4 PEAE	Protocol Error in Arbitration Phase Enable 0 Interrupt disabled 1 Interrupt enabled
5 WDIE	Watchdog Interrupt Enable 0 Interrupt disabled 1 Interrupt enabled
6 BOE	Bus_Off Status Interrupt Enable 0 Interrupt disabled 1 Interrupt enabled
7 EWE	Warning Status Interrupt Enable 0 Interrupt disabled 1 Interrupt enabled
8 EPE	Error Passive Interrupt Enable 0 Interrupt disabled 1 Interrupt enabled
9 ELOE	Error Logging Overflow Interrupt Enable 0 Interrupt disabled 1 Interrupt enabled
10 BEUE	Bit Error Uncorrected Interrupt Enable 0 Interrupt disabled 1 Interrupt enabled

Table 896. IE field descriptions (continued)

Field	Description
11 BECE	Bit Error Corrected Interrupt Enable 0 Interrupt disabled 1 Interrupt enabled
12 DRXE	Message stored to Dedicated Rx Buffer Interrupt Enable 0 Interrupt disabled 1 Interrupt enabled
13 TOOE	Timeout Occurred Interrupt Enable 0 Interrupt disabled 1 Interrupt enabled
14 MRAFE	Message RAM Access Failure Interrupt Enable 0 Interrupt disabled 1 Interrupt enabled
15 TSWE	Timestamp Wraparound Interrupt Enable 0 Interrupt disabled 1 Interrupt enabled
16 TEFLE	Tx Event FIFO Event Lost Interrupt Enable 0 Interrupt disabled 1 Interrupt enabled
17 TEFFE	Tx Event FIFO Full Interrupt Enable 0 Interrupt disabled 1 Interrupt enabled
18 TEFWE	Tx Event FIFO Watermark Reached Interrupt Enable 0 Interrupt disabled 1 Interrupt enabled
19 TEFNE	Tx Event FIFO New Entry Interrupt Enable 0 Interrupt disabled 1 Interrupt enabled
20 TFEE	Tx FIFO Empty Interrupt Enable 0 Interrupt disabled 1 Interrupt enabled
21 TCFE	Transmission Cancellation Finished Interrupt Enable 0 Interrupt disabled 1 Interrupt enabled
22 TCE	Transmission Completed Interrupt Enable 0 Interrupt disabled 1 Interrupt enabled
23 HPME	High Priority Message Interrupt Enable 0 Interrupt disabled 1 Interrupt enabled
24 RF1LE	Rx FIFO 1 Message Lost Interrupt Enable 0 Interrupt disabled 1 Interrupt enabled

Table 896. IE field descriptions (continued)

Field	Description
25 RF1FE	Rx FIFO 1 Full Interrupt Enable 0 Interrupt disabled 1 Interrupt enabled
26 RF1WE	Rx FIFO 1 Watermark Reached Interrupt Enable 0 Interrupt disabled 1 Interrupt enabled
27 RF1NE	Rx FIFO 1 New Message Interrupt Enable 0 Interrupt disabled 1 Interrupt enabled
28 RF0LE	Rx FIFO 0 Message Lost Interrupt Enable 0 Interrupt disabled 1 Interrupt enabled
29 RF0FE	Rx FIFO 0 Full Interrupt Enable 0 Interrupt disabled 1 Interrupt enabled
30 RF0WE	Rx FIFO 0 Watermark Reached Interrupt Enable 0 Interrupt disabled 1 Interrupt enabled
31 RF0NE	Rx FIFO 0 New Message Interrupt Enable 0 Interrupt disabled 1 Interrupt enabled

52.4.5.2.17 Interrupt Line Select Register (ILS)

The Interrupt Line Select register assigns an interrupt generated by a specific interrupt flag from the Interrupt Register to one of the two module interrupt lines. For interrupt generation the respective interrupt line has to be enabled via ILE[EINT0] and ILE[EINT1].

Offset: 0x0058 Access: R/W

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	ARAL	PEDL	PEAL	WDIL	BOL	EWL	EPL	ELOL	BEUL	BECL	DRXL	TOOL	MRAFL	TSWL
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	TEFLL	TEFFL	TEFWL	TEFNL	TFEL	TCFL	TCL	HPML	RF1LL	RF1FL	RF1WL	RF1NL	RF0LL	RF0FL	RF0WL	RF0NL
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 798. Interrupt Line Select Register (ILS)

Table 897. ILS field descriptions

Field	Description
2 ARAL	Access to Reserved Address Line 0 Interrupt assigned to interrupt line M_TTCAN INT0 1 Interrupt assigned to interrupt line M_TTCAN INT1
3 PEDL	Protocol Error in Data phase Line 0 Interrupt assigned to interrupt line M_TTCAN INT0 1 Interrupt assigned to interrupt line M_TTCAN INT1
4 PEAL	Protocol Error in Arbitration Phase Line 0 Interrupt assigned to interrupt line M_TTCAN INT0 1 Interrupt assigned to interrupt line M_TTCAN INT1
5 WDIL	Watchdog Interrupt Line 0 Interrupt assigned to interrupt line M_TTCAN INT0 1 Interrupt assigned to interrupt line M_TTCAN INT1
6 BOL	Bus_Off Status Interrupt Line 0 Interrupt assigned to interrupt line M_TTCAN INT0 1 Interrupt assigned to interrupt line M_TTCAN INT1
7 EWL	Warning Status Interrupt Line 0 Interrupt assigned to interrupt line M_TTCAN INT0 1 Interrupt assigned to interrupt line M_TTCAN INT1
8 EPL	Error Passive Interrupt Line 0 Interrupt assigned to interrupt line M_TTCAN INT0 1 Interrupt assigned to interrupt line M_TTCAN INT1
9 ELOL	Error Logging Overflow Interrupt Line 0 Interrupt assigned to interrupt line M_TTCAN INT0 1 Interrupt assigned to interrupt line M_TTCAN INT1
10 BEUL	Bit Error Uncorrected Interrupt Line 0 Interrupt assigned to interrupt line M_TTCAN INT0 1 Interrupt assigned to interrupt line M_TTCAN INT1
11 BECL	Bit Error Corrected Interrupt Line 0 Interrupt assigned to interrupt line M_TTCAN INT0 1 Interrupt assigned to interrupt line M_TTCAN INT1
12 DRXL	Message stored to Dedicated Rx Buffer Interrupt Line 0 Interrupt assigned to interrupt line M_TTCAN INT0 1 Interrupt assigned to interrupt line M_TTCAN INT1
13 TOOL	Timeout Occurred Interrupt Line 0 Interrupt assigned to interrupt line M_TTCAN INT0 1 Interrupt assigned to interrupt line M_TTCAN INT1
14 MRAFL	Message RAM Access Failure Interrupt Line 0 Interrupt assigned to interrupt line M_TTCAN INT0 1 Interrupt assigned to interrupt line M_TTCAN INT1
15 TSWL	TSWL: Timestamp Wraparound Interrupt Line 0 Interrupt assigned to interrupt line M_TTCAN INT0 1 Interrupt assigned to interrupt line M_TTCAN INT1

Table 897. ILS field descriptions (continued)

Field	Description
16 TEFLL	Tx Event FIFO Event Lost Interrupt Line 0 Interrupt assigned to interrupt line M_TTCAN INT0 1 Interrupt assigned to interrupt line M_TTCAN INT1
17 TEFFL	Tx Event FIFO Full Interrupt Line 0 Interrupt assigned to interrupt line M_TTCAN INT0 1 Interrupt assigned to interrupt line M_TTCAN INT1
18 TEFWL	Tx Event FIFO Watermark Reached Interrupt Line 0 Interrupt assigned to interrupt line M_TTCAN INT0 1 Interrupt assigned to interrupt line M_TTCAN INT1
19 TEFNL	Tx Event FIFO New Entry Interrupt Line 0 Interrupt assigned to interrupt line M_TTCAN INT0 1 Interrupt assigned to interrupt line M_TTCAN INT1
20 TFEL	Tx FIFO Empty Interrupt Line 0 Interrupt assigned to interrupt line M_TTCAN INT0 1 Interrupt assigned to interrupt line M_TTCAN INT1
21 TCFL	Transmission Cancellation Finished Interrupt Line 0 Interrupt assigned to interrupt line M_TTCAN INT0 1 Interrupt assigned to interrupt line M_TTCAN INT1
22 TCL	Transmission Completed Interrupt Line 0 Interrupt assigned to interrupt line M_TTCAN INT0 1 Interrupt assigned to interrupt line M_TTCAN INT1
23 HPML	High Priority Message Interrupt Line 0 Interrupt assigned to interrupt line M_TTCAN INT0 1 Interrupt assigned to interrupt line M_TTCAN INT1
24 RF1LL	Rx FIFO 1 Message Lost Interrupt Line 0 Interrupt assigned to interrupt line M_TTCAN INT0 1 Interrupt assigned to interrupt line M_TTCAN INT1
25 RF1FL	Rx FIFO 1 Full Interrupt Line 0 Interrupt assigned to interrupt line M_TTCAN INT0 1 Interrupt assigned to interrupt line M_TTCAN INT1
26 RF1WL	Rx FIFO 1 Watermark Reached Interrupt Line 0 Interrupt assigned to interrupt line M_TTCAN INT0 1 Interrupt assigned to interrupt line M_TTCAN INT1
27 RF1NL	Rx FIFO 1 New Message Interrupt Line 0 Interrupt assigned to interrupt line M_TTCAN INT0 1 Interrupt assigned to interrupt line M_TTCAN INT1
28 RF0LL	Rx FIFO 0 Message Lost Interrupt Line 0 Interrupt assigned to interrupt line M_TTCAN INT0 1 Interrupt assigned to interrupt line M_TTCAN INT1
29 RF0FL	Rx FIFO 0 Full Interrupt Line 0 Interrupt assigned to interrupt line M_TTCAN INT0 1 Interrupt assigned to interrupt line M_TTCAN INT1

Table 897. ILS field descriptions (continued)

Field	Description
30 RF0WL	Rx FIFO 0 Watermark Reached Interrupt Line 0 Interrupt assigned to interrupt line M_TTCAN INT0 1 Interrupt assigned to interrupt line M_TTCAN INT1
31 RF0NL	Rx FIFO 0 New Message Interrupt Line 0 Interrupt assigned to interrupt line M_TTCAN INT0 1 Interrupt assigned to interrupt line M_TTCAN INT1

52.4.5.2.18 Interrupt Line Enable Register (ILE)

Each of the two interrupt lines to the CPU can be enabled / disabled separately by programming bits EINT0 and EINT1.

Offset: 0x005C Access: R/W

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	EINT1	EINT0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 799. Interrupt Line Enable Register (ILE)

Table 898. ILE field descriptions

Field	Description
30 EINT1	Enable Interrupt Line 1 0 Interrupt line disabled 1 Interrupt line enabled
31 EINT0	Enable Interrupt Line 0 0 Interrupt line disabled 1 Interrupt line enabled

52.4.5.2.19 Global Filter Configuration Register (GFC)

Global settings for Message ID filtering. The Global Filter Configuration controls the filter path for standard and extended messages as described in *Figure 855* and *Figure 856*.

Offset: 0x0080 Access: RP

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	ANFS(1)		ANFE(1)		RRFS(1)	RRFE(1)
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. These are protected write (P) bits which means that write access by the bits is possible only when the bit 1 [CCE] and bit 0 [INIT] of CCCR register are set to ‘1’.

Figure 800. Global Filter Configuration Register (GFC)

Table 899. GFC field descriptions

Field	Description
26:27 ANFS	Accept Non-matching Frames Standard Defines how received messages with 11-bit IDs that do not match any element of the filter list are treated. 00 Accept in Rx FIFO 0 01 Accept in Rx FIFO 1 10 Reject 11 Reject
28:29 ANFE	Accept Non-matching Frames Extended Defines how received messages with 29-bit IDs that do not match any element of the filter list are treated. 00 Accept in Rx FIFO 0 01 Accept in Rx FIFO 1 10 Reject 11 Reject
30 RRFS	Reject Remote Frames Standard 0 Filter remote frames with 11-bit standard IDs 1 Reject all remote frames with 11-bit standard IDs
31 RRFE	Reject Remote Frames Extended 0 Filter remote frames with 29-bit extended IDs 1 Reject all remote frames with 29-bit extended IDs

52.4.5.2.20 Standard ID Filter Configuration Register (SIDFC)

Settings for 11-bit standard Message ID filtering.The Standard ID Filter Configuration controls the filter path for standard messages as described in *Figure 855*.

Offset: 0x0084 Access: RP

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	LSS[1]							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	FLSSA[1]														0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. These are protected write (P) bits which means that write access by the bits is possible only when the bit 1 [CCE] and bit 0 [INIT] of CCCR register are set to ‘1’.

Figure 801. Standard ID Filter Configuration Register (SIDFC)

Table 900. SIDFC field descriptions

Field	Description
8:15 LSS	List Size Standard 0x00: No standard Message ID filter 0x01: Number of standard Message ID filter elements ... 0x80: Number of standard Message ID filter elements 0x81: Values greater than 128 are interpreted as 128 ... 0xFF: Values greater than 128 are interpreted as 128
16:29 FLSSA	Filter List Standard Start Address Start address of standard Message ID filter list (32-bit word address, refer to *Figure 849*).

52.4.5.2.21 Extended ID Filter Configuration Register (XIDFC)

Settings for 29-bit extended Message ID filtering. The Extended ID Filter Configuration controls the filter path for standard messages as described in *Figure 856*.

Offset: 0x0088 Access: RP

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	LSE[1]						
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	FLESA[1]														0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. These are protected write (P) bits which means that write access by the bits is possible only when the bit 1 [CCE] and bit 0 [INIT] of CCCR register are set to ‘1’.

Figure 802. Extended ID Filter Configuration Register (XIDFC)

Table 901. XIDFC field descriptions

Field	Description
9:15 LSE	List Size Extended 0x00: No extended Message ID filter 0x01: Number of extended Message ID filter elements ... 0x40: Number of extended Message ID filter elements 0x41: Values greater than 64 are interpreted as 64 ... 0x7F: Values greater than 64 are interpreted as 640 No extended Message ID filter
16:29 FLESA	Filter List Extended Start Address Start address of extended Message ID filter list (32-bit word address, refer to *Figure 850*).

52.4.5.2.22 Extended ID and Mask Register (XIDAM)

Offset: 0x0090 Access: RP

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	EIDM(1)												
W																
Reset	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	EIDM(1)															
W																
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

1. These are protected write (P) bits which means that write access by the bits is possible only when the bit 1 [CCE] and bit 0 [INIT] of CCCR register are set to '1'.

Figure 803. Extended ID and Mask Register (XIDAM)

Table 902. XIDAM field descriptions

Field	Description
3:31 EIDM	Extended ID Mask For acceptance filtering of extended frames the Extended ID AND Mask is ANDed with the Message ID of a received frame. Intended for masking of 29-bit IDs in SAE J1939. With the reset value of all bits set to one the mask is not active.

52.4.5.2.23 High Priority Message Status Register (HPMS)

This register is updated every time a Message ID filter element configured to generate a priority event match. This can be used to monitor the status of incoming high priority messages and to enable fast access to these messages.

Offset: 0x0094 Access: R

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	FLST	FIDX							MSI		BIDX					
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 804. High Priority Message Status Register (HPMS)

Table 903. HPMS field descriptions

Field	Description
16 FLST	Filter List Indicates the filter list of the matching filter element. 0 Standard Filter List 1 Extended Filter List
17:23 FIDX	Filter Index Index of matching filter element. Range is 0 to SIDFC[LSS] - 1 resp. XIDFC[LSE] - 1.
24:25 MSI	Message Storage Indicator 00 No FIFO selected 01 FIFO overrun 10 Message stored in FIFO 0 11 Message stored in FIFO 1
26:31 BIDX	Buffer Index Index of Rx FIFO element to which the message was stored. Only valid when MSI[1] = '1'.

52.4.5.2.24 New Data 1 Register (NDAT1)

Offset: 0x0098 Access: R/W

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	ND31	ND30	ND29	ND28	ND27	ND26	ND25	ND24	ND23	ND22	ND21	ND20	ND19	ND18	ND17	ND16
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	ND15	ND14	ND13	ND12	ND11	ND10	ND9	ND8	ND7	ND6	ND5	ND4	ND3	ND2	ND1	ND0
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 805. New Data 1 Register (NDAT1)

Table 904. NDAT1 field descriptions

Field	Description
0:31 ND[31:0]	New Data [31:0] The register holds the New Data flags of Rx Buffers 0 to 31. The flags are set when the respective Rx Buffer has been updated from a received frame. The flags remain set until the Host clears them. A flag is cleared by writing a ’1’ to the corresponding bit position. Writing a ’0’ has no effect. A hard reset will clear the register. 0 Rx Buffer not updated 1 Rx Buffer updated from new message

52.4.5.2.25 New Data 2 Register (NDAT2)

Offset: 0x009C Access: R/W

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	ND63	ND62	ND61	ND60	ND59	ND58	ND57	ND56	ND55	ND54	ND53	ND52	ND51	ND50	ND49	ND48
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	ND47	ND46	ND45	ND44	ND43	ND42	ND41	ND40	ND39	ND38	ND37	ND36	ND35	ND34	ND33	ND32
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 806. New Data 2 Register (NDAT2)

Table 905. NDAT2 field descriptions

Field	Description
0:31 ND[63:32]	New Data [63:32] The register holds the New Data flags of Rx Buffers 32 to 63. The flags are set when the respective Rx Buffer has been updated from a received frame. The flags remain set until the Host clears them. A flag is cleared by writing a ‘1’ to the corresponding bit position. Writing a ‘0’ has no effect. A hard reset will clear the register. 0 Rx Buffer not updated 1 Rx Buffer updated from new message

52.4.5.2.26 Rx FIFO 0 Configuration Register (RXF0C)

Offset: 0x00A0 Access: RP

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	F0OM	F0WM(1)							0	F0S(1)						
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	F0SA(1)														0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. These are protected write (P) bits which means that write access by the bits is possible only when the bit 1 [CCE] and bit 0 [INIT] of CCCR register are set to '1'.

Figure 807. Rx FIFO 0 Configuration Register (RXF0C)

Table 906. RXF0C field descriptions

Field	Description
0 F0OM	FIFO 0 Operation Mode FIFO 0 can be operated in blocking or in overwrite mode (refer to *Section 52.3.10.2*). 0 FIFO 0 blocking mode 1 FIFO 0 overwrite mode
1:7 F0WM	Rx FIFO 0 Watermark 0x00: Watermark interrupt disabled 0x01: Level for Rx FIFO 0 watermark interrupt (**IR[RF0W]**) ... 0x40: Level for Rx FIFO 0 watermark interrupt (**IR[RF0W]**) 0x41: Watermark interrupt disabled ... 0x7F: Watermark interrupt disabled
9:15 F0S	Rx FIFO 0 Size 0x00: No Rx FIFO 0 0x01: Number of Rx FIFO 0 elements 0x40: Number of Rx FIFO 0 elements 0x41: Values greater than 64 are interpreted as 64 0x7F: Values greater than 64 are interpreted as 64 The Rx FIFO 0 elements are indexed from 0 to **F0S**-1.
16:29 F0SA	Rx FIFO 0 Start Address Start address of Rx FIFO 0 in Message RAM (32-bit word address, *Figure 845*).

52.4.5.2.27 Rx FIFO 0 Status Register (RXF0S)

Offset: 0x0A4 Access: R

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	RF0L	F0F	0	0	F0PI					
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	F0GI						0	F0FL						
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 808. Rx FIFO 0 Status Register (RXF0S)

Table 907. RXF0S field descriptions

Field	Description
6 RF0L	Rx FIFO 0 Message Lost This bit is a copy of interrupt flag IR[RF0L]. When IR[RF0L] is reset, this bit is also reset. 0 No Rx FIFO 0 message lost 1 Rx FIFO 0 message lost, also set after write attempt to Rx FIFO 0 of size zero
7 F0F	Rx FIFO 0 Full 0 Rx FIFO 0 not full 1 Rx FIFO 0 full
10:15 F0PI	Rx FIFO 0 Put Index Rx FIFO 0 write index pointer, range 0 to 63.
18:23 F0GI	Rx FIFO 0 Get Index Rx FIFO 0 read index pointer, range 0 to 63.
25:31 F0FL	Rx FIFO 0 Fill Level Number of elements stored in Rx FIFO 0, range 0 to 64.

52.4.5.2.28 Rx FIFO 0 Acknowledge Register (RXF0A)

Offset: 0x0A8 Access: R/W

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	F0AI					
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 809. Rx FIFO 0 Acknowledge Register (RXF0A)

Table 908. RXF0A field descriptions

Field	Description
26:31 F0AI	Rx FIFO 0 Acknowledge Index After the Host has read a message or a sequence of messages from Rx FIFO 0 it has to write the buffer index of the last element read from Rx FIFO 0 to F0AI. This will set the Rx FIFO 0 Get Index. RXF0S[F0GI] to F0AI + 1 and update the FIFO 0 Fill Level RXF0S[F0FL].

52.4.5.2.29 Rx Buffer Configuration Register (RXBC)

Offset: 0x00AC Access: R/W

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	RBSA(1)														0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. These are protected write (P) bits which means that write access by the bits is possible only when the bit 1 [CCE] and bit 0 [INIT] of CCCR register are set to ‘1’.

Figure 810. Rx FIFO 0 Configuration Register (RXBC)

Table 909. RXBC field descriptions

Field	Description
16:29 RBSA	Rx Buffer Start Address Configures the start address of the Rx Buffers section in the Message RAM (32-bit word address). Also used to reference debug messages A,B,C.

52.4.5.2.30 Rx FIFO 1 Configuration Register (RXF1C)

Offset: 0x0B0 Access: RP

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	F1WM(1)							0	F1S(1)						
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	F1SA(1)														0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. These are protected write (P) bits which means that write access by the bits is possible only when the bit 1 [CCE] and bit 0 [INIT] of CCCR register are set to '1'.

Figure 811. Rx FIFO 1 Configuration Register (RXF1C)

Table 910. RXF1C field descriptions

Field	Description
1:7 F1WM	Rx FIFO 1 Watermark 0x00: Watermark interrupt disabled 0x01: Level for Rx FIFO 1 watermark interrupt (IR[RF1W]) ... 0x40: Level for Rx FIFO 1 watermark interrupt (IR[RF1W]) 0x41: Watermark interrupt disabled ... 0x7F: Watermark interrupt disabled
9:15 F1S	Rx FIFO 1 Size 0x00: No Rx FIFO 1 0x01: Number of Rx FIFO 1 elements ... 0x40: Number of Rx FIFO 1 elements 0x41: Values greater than 64 are interpreted as 64 ... 0x7F: Values greater than 64 are interpreted as 64 The Rx FIFO 1 elements are indexed from 0 to F1S - 1.
16:29 F1SA	Rx FIFO 1 Start Address Start address of Rx FIFO 1 in Message RAM (32-bit word address, refer to *Figure 845*).

52.4.5.2.31 Rx FIFO 1 Status Register (RXF1S)

Offset: 0x0B4 Access: R

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	DMS		0	0	0	0	RF1L	F1F	0	0	F1PI					
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	F1GI						0	F1FL						
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 812. Rx FIFO 1 Status Register (RXF1S)

Table 911. RXF1S field descriptions

Field	Description
0:1 DMS	Debug Message Status 00 Idle state, wait for reception of debug messages, DMA request is cleared 01 Debug message A received 10 Debug messages A, B received 11 Debug messages A, B, C received, DMA request is set
6 RF1L	Rx FIFO 1 Message Lost This bit is a copy of interrupt flag IR[RF1L]. When IR[RF1L] is reset, this bit is also reset. 0 No Rx FIFO 1 message lost 1 Rx FIFO 1 message lost, also set after write attempt to Rx FIFO 1 of size zero
7 F1F	Rx FIFO 1 Full 0 Rx FIFO 1 not full 1 Rx FIFO 1 full
10:15 F1PI	Rx FIFO 1 Put Index Rx FIFO 1 write index pointer, range 0 to 63.
18:23 F1GI	Rx FIFO 1 Get Index Rx FIFO 1 read index pointer, range 0 to 63.
25:31 F1FL	Rx FIFO 1 Fill Level Number of elements stored in Rx FIFO 1, range 0 to 64.

52.4.5.2.32 Rx FIFO 1 Acknowledge Register (RXF1A)

Offset: 0x0B8 Access: R/W

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	F1AI					
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 813. Rx FIFO 1 Acknowledge Register (RXF1A)

Table 912. RXF1A field descriptions

Field	Description
26:31 F1AI	Rx FIFO 1 Acknowledge Index After the Host has read a message or a sequence of messages from Rx FIFO 1 it has to write the buffer index of the last element read from Rx FIFO 1 to F1AI. This will set the Rx FIFO 1 Get Index RXF1S[F1GI] to F1AI + 1 and update the FIFO 1 Fill Level RXF1S[F1FL].

52.4.5.2.33 Rx Buffer / FIFO element size configuration (RXESC)

Offset: 0x0BC Access: R/W

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	RBDS(1)			0	F1DS(1)			0	F0DS(1)		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. These are protected write (P) bits which means that write access by the bits is possible only when the bit 1 [CCE] and bit 0 [INIT] of CCCR register are set to ‘1’.

Figure 814. Rx Buffer / FIFO element size configuration (RXESC)

Table 913. RXESC field descriptions

Field	Description
21:23 RBDS	Rx Buffer Data Field Size 000 8 byte data field 001 12 byte data field 010 16 byte data field 011 20 byte data field 100 24 byte data field 101 32 byte data field 110 48 byte data field 111 64 byte data field
25:27 F1DS	Rx FIFO 1 Data Field Size 000 8 byte data field 001 12 byte data field 010 16 byte data field 011 20 byte data field 100 24 byte data field 101 32 byte data field 110 48 byte data field 111 64 byte data field
29:31 F0DS	Rx FIFO 0 Data Field Size 000 8 byte data field 001 12 byte data field 010 16 byte data field 011 20 byte data field 100 24 byte data field 101 32 byte data field 110 48 byte data field 111 64 byte data field

Note: Be aware that the sum of $TFQS$ and $NDTB$ may be not greater than 32. There is no check for erroneous configurations. The Tx Buffers section in the Message RAM starts with the dedicated Tx Buffers.

52.4.5.2.34 Tx Buffer Configuration Register (TXBC)

Offset: 0x0C0 Access: RP

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	TFQM(1)	TFQS(1)						0	0	NDTB(1)					
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	TBSA(1)														0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. These are protected write (P) bits which means that write access by the bits is possible only when the bit 1 [CCE] and bit 0 [INIT] of CCCR register are set to ‘1’.

Figure 815. Tx Buffer Configuration Register (TXBC)

Table 914. TXBC field descriptions

Field	Description
1 TFQM	Tx FIFO/Queue Mode 0 Tx FIFO operation 1 Tx Queue operation
2:7 TFQS	Transmit FIFO/Queue Size 0x00: No Tx FIFO/Queue 0x01: Number of Tx Buffers used for Tx FIFO/Queue ... 0x20: Number of Tx Buffers used for Tx FIFO/Queue 0x21: Values greater than 32 are interpreted as 32 ... 0x3F: Values greater than 32 are interpreted as 32
10:15 NDTB	Number of Dedicated Transmit Buffers 0x00: No Dedicated Tx Buffers 0x01: Number of Dedicated Tx Buffers ... 0x20: Number of Dedicated Tx Buffers 0x21: Values greater than 32 are interpreted as 32 ... 0x3F: Values greater than 32 are interpreted as 32
16:29 TBSA	Tx Buffers Start Address Start address of Tx Buffers section in Message RAM (32-bit word address, refer to *Figure 845*).

Note: Be aware that the sum of TFQS and NDTB may be not greater than 32. There is no check for erroneous configurations. The Tx Buffers section in the Message RAM starts with the dedicated Tx Buffers.

52.4.5.2.35 Tx FIFO/Queue Status Register (TXFQS)

The Tx FIFO/Queue status is related to the pending Tx requests listed in register **TXBRP**. Therefore the effect of Add/Cancellation requests may be delayed due to a running Tx scan (**TXBRP** not yet updated).

Offset: 0xC4 Access: R

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	TFQF	TFQPI				
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	TFGI					0	0	TFFL					
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 816. Tx FIFO/Queue Status Register (TXFQS)

Table 915. TXFQS field descriptions

Field	Description
10 TFQF	Tx FIFO/Queue Full 0 Tx FIFO/Queue not full 1 Tx FIFO/Queue full
11:15 TFQPI	Tx FIFO/Queue Put Index Tx FIFO/Queue write index pointer, range 0 to 31
19:23 TFGI	Tx FIFO Get Index Tx FIFO read index pointer, range 0 to 31. Read as zero when Tx Queue operation is configured (TXBC.TFQM = ‘1’).
26:31 TFFL	Tx FIFO Free Level Number of consecutive free Tx FIFO elements starting from TFGI, range 0 to 32. Read as zero when Tx Queue operation is configured (TXBC[TFQM] = ‘1’).

Note: In case of mixed configurations where dedicated Tx Buffers are combined with a Tx FIFO or a Tx Queue, the Put and Get Index indicate the number of the Tx Buffer starting with the first dedicated Tx Buffers.

For example: For a configuration of 12 dedicated Tx Buffers and a Tx FIFO of 20 Buffers a Put Index of 15 points to the fourth buffer of the Tx FIFO.

52.4.5.2.36 Tx buffer element size configuration (TXESC)

Offset: 0x0C8 Access: R/W

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	TBDS(1)		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. These are protected write (P) bits which means that write access by the bits is possible only when the bit 1 [CCE] and bit 0 [INIT] of CCCR register are set to '1'.

Figure 817. Tx Buffer element size configuration (TXESC)

Table 916. TXESC field descriptions

Field	Description
29:31 TBDS	Tx Buffer Data Field Size 000 8 byte data field 001 12 byte data field 010 16 byte data field 011 20 byte data field 100 24 byte data field 101 32 byte data field 110 48 byte data field 111 64 byte data field

Note: In case the data length code DLC of a Tx Buffer element is configured to a value higher than the Tx Buffer data field size TXESC.TBDS, the bytes not defined by the Tx Buffer are transmitted as "0xCC" (padding bytes).

52.4.5.2.37 Tx Buffer Request Pending Register (TXBRP)

Offset: 0xCC Access: R

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	TRP31	TRP30	TRP29	TRP28	TRP27	TRP26	TRP25	TRP24	TRP23	TRP22	TRP21	TRP20	TRP19	TRP18	TRP17	TRP16
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	TRP15	TRP14	TRP13	TRP12	TRP11	TRP10	TRP9	TRP8	TRP7	TRP6	TRP5	TRP4	TRP3	TRP2	TRP1	TRP0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 818. Tx Buffer Request Pending Register (TXBRP)

Table 917. TXBRP field descriptions

Field	Description
0:31 TRP[31:0]	Transmission Request Pending Each Tx Buffer has its own Transmission Request Pending bit. The bits are set via register TXBAR. The bits are reset after a requested transmission has completed or has been cancelled via register TXBCR. TXBRP bits are set only for those Tx Buffers configured via TXBC. After a TXBRP bit has been set, a Tx scan (refer to *Section 52.4.7.16: Debug on CAN Support*) is started to check for the pending Tx request with the highest priority (Tx Buffer with lowest Message ID). A cancellation request resets the corresponding transmission request pending bit of register TXBRP. In case a transmission has already been started when a cancellation is requested, this is done at the end of the transmission, regardless whether the transmission was successful or not. The cancellation request bits are reset directly after the corresponding TXBRP bit has been reset. After a cancellation has been requested, a finished cancellation is signaled via TXBCF – after successful transmission together with the corresponding TXBTO bit – when the transmission has not yet been started at the point of cancellation – when the transmission has been aborted due to lost arbitration – when an error occurred during frame transmission In DAR mode all transmissions are automatically cancelled if they are not successful. The corresponding TXBCF bit is set for all unsuccessful transmissions. 0 No transmission request pending 1 Transmission request pending

Note: TXBRP bits which are set while a Tx scan is in progress are not considered during this particular Tx scan. In case a cancellation is requested for such a Tx Buffer, this Add Request is cancelled immediately, the corresponding TXBRP bit is reset.

52.4.5.2.38 Tx Buffer Add Request Register (TXBAR)

Offset: 0x0D0 Access: R/W

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	AR31	AR30	AR29	AR28	AR27	AR26	AR25	AR24	AR23	AR22	AR21	AR20	AR19	AR18	AR17	AR16
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	AR15	AR14	AR13	AR12	AR11	AR10	AR9	AR8	AR7	AR6	AR5	AR4	AR3	AR2	AR1	AR0
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 819. Tx Buffer Add Request Register (TXBAR)

Table 918. TXBAR field descriptions

Field	Description
0:31 AR[31:0]	Add Request Each Tx Buffer has its own Add Request bit. Writing a '1' will set the corresponding Add Request bit; writing a '0' has no impact. This enables the Host to set transmission requests for multiple Tx Buffers with one write to TXBAR. TXBAR bits are set only for those Tx Buffers configured via TXBC. When no Tx scan is running, the bits are reset immediately, else the bits remain set until the Tx scan process has completed. 0 No transmission request added 1 Transmission requested added

Note: If an add request is applied for a Tx Buffer with pending transmission request (corresponding TXBRP bit already set), this add request is ignored.

52.4.5.2.39 Tx Buffer Cancellation Request Register (TXBCR)

Offset: 0x0D4 Access: R/W

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	CR31	CR30	CR29	CR28	CR27	CR26	CR25	CR24	CR23	CR22	CR21	CR20	CR19	CR18	CR17	CR16
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	CR15	CR14	CR13	CR12	CR11	CR10	CR9	CR8	CR7	CR6	CR5	CR4	CR3	CR2	CR1	CR0
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 820. Tx Buffer Cancellation Request Register (TXBCR)

Table 919. TXBCR field descriptions

Field	Description
0:31 CR[31:0]	Cancellation Request Each Tx Buffer has its own Cancellation Request bit. Writing a '1' will set the corresponding Cancellation Request bit; writing a '0' has no impact. This enables the Host to set cancellation requests for multiple Tx Buffers with one write to TXBCR. TXBCR bits are set only for those Tx Buffers configured via TXBC. The bits remain set until the corresponding bit of TXBRP is reset. 0 No cancellation pending 1 Cancellation pending

52.4.5.2.40 Tx Buffer Transmission Occurred Register (TXBTO)

Offset: 0x0D8 Access: R

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	TO31	TO30	TO29	TO28	TO27	TO26	TO25	TO24	TO23	TO22	TO21	TO20	TO19	TO18	TO17	TO16
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	TO15	TO14	TO13	TO12	TO11	TO10	TO9	TO8	TO7	TO6	TO5	TO4	TO3	TO2	TO1	TO0
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 821. Tx Buffer Transmission Occurred Register (TXBTO)

Table 920. TXBTO field descriptions

Field	Description
0:31 TO[31:0]	Transmission Occurred Each Tx Buffer has its own Transmission Occurred bit. The bits are set when the corresponding TXBRP bit is cleared after a successful transmission. The bits are reset when a new transmission is requested by writing a ‘1’ to the corresponding bit of register TXBAR. 0 No transmission occurred 1 Transmission occurred

52.4.5.2.41 Tx Buffer Cancellation Finished Register (TXBCF)

Offset: 0x0DC Access: R

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	CF31	CF30	CF29	CF28	CF27	CF26	CF25	CF24	CF23	CF22	CF21	CF20	CF19	CF18	CF17	CF16
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	CF15	CF14	CF13	CF12	CF11	CF10	CF9	CF8	CF7	CF6	CF5	CF4	CF3	CF2	CF1	CF0
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 822. Tx Buffer Cancellation Finished Register (TXBCF)

Table 921. TXBCF field descriptions

Field	Description
0:31 CF[31:0]	Cancellation Finished Each Tx Buffer has its own Cancellation Finished bit. The bits are set when the corresponding TXBRP bit is cleared after a cancellation was requested via TXBCR. In case the corresponding TXBRP bit was not set at the point of cancellation, CF is set immediately. The bits are reset when a new transmission is requested by writing a '1' to the corresponding bit of register TXBAR. 0 No transmit buffer cancellation 1 Transmit buffer cancellation finished

52.4.5.2.42 Tx Buffer Transmission Interrupt Enable Register (TXBTIE)

Offset: 0x0E0 Access: R/W

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	TIE31	TIE30	TIE29	TIE28	TIE27	TIE26	TIE25	TIE24	TIE23	TIE22	TIE21	TIE20	TIE19	TIE18	TIE17	TIE16
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	TIE15	TIE14	TIE13	TIE12	TIE11	TIE10	TIE9	TIE8	TIE7	TIE6	TIE5	TIE4	TIE3	TIE2	TIE1	TIE0
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 823. Tx Buffer Transmission Interrupt Enable Register (TXBTIE)

Table 922. TXBTIE field descriptions

Field	Description
0:31 TIE[31:0]	Transmission Interrupt Enable Each Tx Buffer has its own Transmission Interrupt Enable bit. 0 Transmission interrupt disabled 1 Transmission interrupt enable

52.4.5.2.43 Tx Buffer Cancellation Finished Interrupt Enable Register (TXBCIE)

Offset: 0x0E4 Access: R/W

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	CFIE31	CFIE30	CFIE29	CFIE28	CFIE27	CFIE26	CFIE25	CFIE24	CFIE23	CFIE22	CFIE21	CFIE20	CFIE19	CFIE18	CFIE17	CFIE16
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	CFIE15	CFIE14	CFIE13	CFIE12	CFIE11	CFIE10	CFIE9	CFIE8	CFIE7	CFIE6	CFIE5	CFIE4	CFIE3	CFIE2	CFIE1	CFIE0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 824. Tx Buffer Cancellation Finished Interrupt Enable Register (TXBCIE)

Table 923. TXBCIE field descriptions

Field	Description
0:31 CFIE[31:0]	Cancellation Finished Interrupt Enable. Each Tx Buffer has its own Cancellation Finished Interrupt Enable bit. 0 Cancellation finished interrupt disabled 1 Cancellation finished interrupt enabled

52.4.5.2.44 Tx Event FIFO Configuration Register (TXEFC)

Offset: 0x0F0 Access: RP

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	EFWM(1)						0	0	EFS(1)					
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	EFSA(1)														0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. These are protected write (P) bits which means that write access by the bits is possible only when the bit 1 [CCE] and bit 0 [INIT] of CCCR register are set to ‘1’.

Figure 825. Tx Event FIFO Configuration Register (TXEFC)

Table 924. TXEFC field descriptions

Field	Description
2:7 EFWM	Event FIFO Watermark 0x00: Watermark interrupt disabled 0x01: Level for Tx Event FIFO watermark interrupt (IR[TEFW]) ... 0x20: Level for Tx Event FIFO watermark interrupt (IR[TEFW]) 0x21: Watermark interrupt disabled ... 0x3F: Watermark interrupt disabled
10:15 EFS	Event FIFO Size. 0x00: x Event FIFO disabled 0x01: Number of Tx Event FIFO elements ... 0x20: Number of Tx Event FIFO elements 0x21: Values greater than 32 are interpreted as 32 ... 0x3F: Values greater than 32 are interpreted as 32 The Tx Event FIFO elements are indexed from 0 to **EFS**-1.
16:29 EFSA	Event FIFO Start Address Start address of Tx Event FIFO in Message RAM (32-bit word address, *Figure 845*).

52.4.5.2.45 Tx Event FIFO Status Register (TXEFS)

Offset: 0x0F4 Access: R

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	TEFL	EFF	0	0	0	EFPI				
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	EFGI					0	0	EFFL					
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 826. Tx Event FIFO Status Register (TXEFS)

Table 925. TXEFS field descriptions

Field	Description
6 TEFL	Tx Event FIFO Element Lost. This bit is a copy of interrupt flag IR[TEFL]. When IR[TEFL] is reset, this bit is also reset. 0 No Tx Event FIFO element lost 1 Tx Event FIFO element lost, also set after write attempt to Tx Event FIFO of size zero
7 EFF	Event FIFO Full. 0 Tx Event FIFO not full 1 Tx Event FIFO full
11:15 EFPI	Event FIFO Put Index. Tx Event FIFO write index pointer, range 0 to 31.
19:23 EFGI	Event FIFO Get Index. Tx Event FIFO read index pointer, range 0 to 31.
26:31 EFFL	Event FIFO Fill Level. Number of elements stored in Tx Event FIFO, range 0 to 32.

52.4.5.2.46 Tx Event FIFO Acknowledge Register (TXEFA)

Offset: 0x0F8 Access: R/W

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	EFAI				
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 827. Tx Event FIFO Acknowledge Register (TXEFA)

Table 926. TXEFA field descriptions

Field	Description
27:31 EFAI	Event FIFO Acknowledge Index. After the Host has read an element or a sequence of elements from the Tx Event FIFO it has to write the index of the last element read from Tx Event FIFO to EFAI. This will set the Tx Event FIFO Get Index TXEFS[EFGI] to EFAI + 1 and update the FIFO 0 Fill Level TXEFS[EFFL].

52.4.5.2.47 TT Trigger Memory Configuration Register (TTTMC)

Offset: 0x100 Access: RP

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	TME(1)						
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	TMSA(1)														0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. These are protected write (P) bits which means that write access by the bits is possible only when the bit 1 [CCE] and bit 0 [INIT] of CCCR register are set to '1'.

Figure 828. TT Trigger Memory Configuration Register (TTTMC)

Table 927. TTTMC field descriptions

Field	Description
9:15 TME	Trigger Memory Elements. 0x00: No Trigger Memory 0x01: Number of Trigger Memory elements ... 0x40: Number of Trigger Memory elements 0x41: Values greater than 64 are interpreted as 64 ... 0x7F: Values greater than 64 are interpreted as 64
16:29 TMSA	Trigger Memory Start Address. Start address of Trigger Memory in Message RAM (32-bit word address, *Figure 845*).

52.4.5.2.48 TT Reference Message Configuration Register (TTRMC)

Offset: 0x104 Access: RP

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	RMPS(1)	XTD(1)	0	RID(1)												
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	RID(1)															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. These are protected write (P) bits which means that write access by the bits is possible only when the bit 1 [CCE] and bit 0 [INIT] of CCCR register are set to '1'.

Figure 829. TT Reference Message Configuration Register (TTRMC)

Table 928. TTRMC field descriptions

Field	Description
0 RMPS	Reference Message Payload Select. Ignored in case of time slaves. 0 Reference message has no additional payload 1 The following elements are taken from Tx Buffer 0: Message Marker MM, Event FIFO Control EFC, Data Length Code DLC, Data Bytes DB (Level 1: bytes 2-8, Level 0, 2: bytes 5-8)
1 XTD	Extended Identifier 0 11-bit standard identifier 1 29-bit extended identifier
3:31 RID	Reference Identifier. Identifier transmitted with Reference message and used for Reference message filtering. Standard or extended reference identifier depending on bit XTD. A standard identifier has to be written to ID[28:18].

52.4.5.2.49 TT Operation Configuration Register (TTOCF)

Offset: 0x108 Access: RP

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	EVTP(1)	ECC(1)	EGTF(1)	AWL(1)							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	EECS(1)	IRTO(1)							LDSDL(1)			TM(1)	GEN(1)	0	OM(1)	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. These are protected write (P) bits which means that write access by the bits is possible only when the bit 1 [CCE] and bit 0 [INIT] of CCCR register are set to ‘1’.

Figure 830. TT Operation Configuration Register (TTOCF)

Table 929. TTOCF field descriptions

Field	Description
5 EVTP	Event Trigger Polarity 0 Rising edge trigger 1 Falling edge trigger
6 ECC	Enable Clock Calibration 0 Automatic clock calibration in M_TTCAN Level 0, 2 is disabled 1 Automatic clock calibration in M_TTCAN Level 0, 2 is enabled
7 EGTF	Enable Global Time Filtering 0 Global time filtering in M_TTCAN Level 0, 2 is disabled 1 Global time filtering in M_TTCAN Level 0, 2 is enabled

Table 929. TTOCF field descriptions (continued)

Field	Description
8:15 AWL	Application Watchdog Limit The application watchdog can be disabled by programming **AWL** to 0x00. (0x00–0xFF) Maximum time after which the application has to serve the application watchdog. The application watchdog is incremented once each 256 NTUs.
16 EECS	Enable External Clock Synchronization If enabled, TUR configuration (TURCF[NCL] only) may be updated during M_TTCAN operation. 0 External clock synchronization in M_TTCAN Level 0,2 disabled 1 External clock synchronization in M_TTCAN Level 0,2 enabled
17:23 IRTO	Initial Reference Trigger Offset (0x00–0x7F) Positive offset, range from 0 to 127.
24:26 LDSDL	LD of Synchronization Deviation Limit The Synchronization Deviation Limit SDL is configured by its dual logarithm LDSDL with SDL = $2^{(LDSDL + 5)}$. It should not exceed the clock tolerance given by the CAN bit timing configuration. (0x0–0x7) LD of Synchronization Deviation Limit (SDL ≤ 32…4096)
27 TM	Time Master 0 Time Master function disabled 1 Potential Time Master
28 GEN	Gap Enable 0 Strictly time-triggered operation 1 External event-synchronized time-triggered operation
30:31 OM	Operation Mode 00 Event-driven CAN communication, default 01 M_TTCAN level 1 10 M_TTCAN level 2 11 M_TTCAN level 0

52.4.5.2.50 TT Matrix Limits Register (TTMLM)

Offset: 0x10C Access: RP

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	ENTT(1)											
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	TXEW(1)				CSS		CCM(1)					
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. These are protected write (P) bits which means that write access by the bits is possible only when the bit 1 [CCE] and bit 0 [INIT] of CCCR register are set to ‘1’.

Figure 831. TT Matrix Limits Register (TTMLM)

Table 930. TTMLM field descriptions

Field	Description
4:15 ENTT	Expected Number of Tx Triggers (0x000–0xFFF) Expected number of Tx Triggers in one Matrix Cycle.
20:23 TXEW	Tx Enable Window (0x0–0xF) Length of Tx enable window, 1-16 NTU cycles.
24:25 CSS	Cycle Start Synchronization Enables sync pulse output at Start of Cycle pin. 00 No sync pulse 01 Sync pulse at start of basic cycle 10 Sync pulse at start of matrix cycle 11 Reserved
26:31 CCM	Cycle Count Max 0x00: 1 Basic Cycle per Matrix Cycle 0x01: 2 Basic Cycles per Matrix Cycle 0x03: 4 Basic Cycles per Matrix Cycle 0x07: 8 Basic Cycles per Matrix Cycle 0x0F: 16 Basic Cycles per Matrix Cycle 0x1F: 32 Basic Cycles per Matrix Cycle 0x3F: 64 Basic Cycles per Matrix Cycle Others reserved.

Note: ISO 11898-4, Section 5.2.1 requires, that only the listed cycle count values are configured. Other values are possible but may lead to inconsistent matrix cycles.

52.4.5.2.51 TUR Configuration Register (TURCF)

The length of the NTU is given by: NTU = CAN Clock Period x NC/DC

NC is an 18-bit value. Its high part, NCH[17:16] is hard wired to 0b01. Therefore the range of NC is 0x10000…0x1FFFF. The value configured by NCL is the initial value for TURNA[NAV[15:0]]. DC is set to 0x1000 by hardware reset and it may not be written to 0x0000.

Level 1: NC >= 4 × DC and NTU = CAN bit time

Level 0, 2: NC >= 8 × DC

The actual value of TUR may be changed by the clock drift compensation function of M_TTCAN Level 0 and Level 2 in order to adjust the node's local view of the NTU to the time master's view of the NTU. DC will not be changed by the automatic drift compensation, TURNA[NAV] may be adjusted around NC in the range of the Synchronization Deviation Limit given by TTOCF[LDSDL]. NC and DC should be programmed to the largest suitable values in order to allow the best computational accuracy for the drift compensation process.

Offset: 0x110 Access: RP

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	ELT(1)	0	DC(1)													
W																
Reset	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	NCL(1)															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. These are protected write (P) bits which means that write access by the bits is possible only when the bit 1 [CCE] and bit 0 [INIT] of CCCR register are set to ‘1’.

Figure 832. TUR Configuration Register (TURCF)

Table 931. TURCF field descriptions

Field	Description
0 ELT	Enable Local Time 0 Local time is stopped, default 1 Local time is enabled **Note:** The local time is started by setting ELT. It remains active until ELT is reset or until the next hardware reset. TURCF[DC] is locked when TURCF[ELT] = ‘1’. If ELT is written to ‘0’, the readable value will stay at ‘1’ until the new value has been synchronized into the CAN clock domain. During this time write access to the other bits of the register remains locked.
2:15 DC	Denominator Configuration 0x0000: Illegal value (0x0001–0x3FFF) Denominator Configuration
16:31 NCL	Numerator Configuration Low Write access to the TUR Numerator Configuration Low is only possible during configuration with TURCF[ELT] = ‘0’ or if TTOCF[EECS] (external clock synchronization enabled) is set. When a new value for NCL is written outside TT Configuration Mode, the new value takes effect when TTOST.WECS is cleared to ‘0’. NCL is locked TTOST[WECS] is ‘1’. (0x0000–0xFFFF) Numerator Configuration Low

Note: If NC < 7 × DC in M_TTCAN Level 1, then it is required that subsequent Time Marks in the Trigger Memory must differ by at least two NTU.

52.4.5.2.52 TT Operation Control Register (TTOCN)

Offset: 0x114 Access: R/W

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	LCKC	0	ESCN	NIG	TMG	FGP	GCS	TTIE	TMC		RTIE	SWS		SWP	ECS	SGT
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 833. TT Operation Control Register (TTOCN)

Table 932. TTOCN field descriptions

Field	Description
16 LCKC	TT Operation Control Register Locked Set by a write access to register TTOCN. Reset when the updated configuration has been synchronized into the CAN clock domain. 0 Write access to TTOCN enabled 1 Write access to TTOCN locked
18 ESCN	External Synchronization Control If enabled the M_TTCAN synchronizes its cycle time phase to an external event signaled by a rising edge at Event Trigger pin(*Section 52.4.19: Synchronization to external time schedule*). 0 External synchronization disabled 1 External synchronization enabled
19 NIG	Next is Gap This bit can only be set when the M_TTCAN is the actual Time Master and when it is configured for external event-synchronized time-triggered operation (TTOCF[GEN] = ‘1’). 0 No action, reset by reception of any Reference message 1 Transmit next Reference Message with Next_is_Gap = ‘1’
20 TMG	Time Mark Gap 0 Reset by each Reference message 1 Next Reference message started when Register Time Mark interrupt TTIR[RTMI] is activated
21 FGP	Finish Gap Set by the CPU, reset by each reference message 0 No reference message requested 1 Application requested start of reference message
22 GCS	Gap Control Select 0 Gap control independent from Event trigger 1 Gap control by input Event trigger pin

Table 932. TTOCN field descriptions (continued)

Field	Description
23 TTIE	Trigger Time Mark Interrupt Pulse Enable External time mark events are configured by trigger memory element TMEX. A trigger time mark interrupt pulse is generated when the trigger memory element becomes active, and the M_TTCAN is in synchronization state In_Schedule or In_Gap. 0 Trigger Time Mark Interrupt output disabled 1 Trigger Time Mark Interrupt output enabled
24:25 TMC	Register Time Mark Compare 00 No Register Time Mark Interrupt generated 01 Register Time Mark Interrupt if Time Mark = cycle time 10 Register Time Mark Interrupt if Time Mark = local time 11 Register Time Mark Interrupt if Time Mark = global time **Note:** When changing the time mark reference (cycle, local, global time), it is recommended to first write TMC = ‘00’, then reconfigure TTTMK, and finally set TMC to the intended time reference.
26 RTIE	Register Time Mark Interrupt Pulse Enable Register time mark interrupts are configured by register **TTTMK**. A register time mark interrupt pulse with the length of one CAN clock period is generated when time referenced by TTOCN[TMC] (cycle, local, or global) equals TTTMK[TM], independent of the synchronization state. 0 Register Time Mark Interrupt output disabled 1 Register Time Mark Interrupt output enabled
27:28 SWS	Stop Watch Source 00 Stop Watch disabled 01 Actual value of cycle time is copied to TTCPT[SWV] 10 Actual value of local time is copied to TTCPT[SWV] 11 Actual value of global time is copied to TTCPT[SWV]
29 SWP	Stop Watch Polarity 0 Rising edge trigger 1 Falling edge trigger
30 ECS	External Clock Synchronization Writing a ‘1’ to **ECS** sets TTOST[WECS] if the node is the actual Time Master. ECS is reset after one Host clock period. The external clock synchronization takes effect at the start of the next basic cycle.
31 SGT	Set Global time Writing a ‘1’ to SGT sets TTOST[WGDT] if the node is the actual Time Master. SGT is reset after one Host clock period. The global time preset takes effect when the node transmits the next Reference message with the Master_Ref_Mark modified by the preset value written to TTGTP.

52.4.5.2.53 TT Global Time Preset Register (TTGTP)

If TTOST.WGDT is set, the next Reference message will be transmitted with the Master_Ref_Mark modified by the preset value and with Disc_Bit = ‘1’, presetting the global time in all nodes simultaneously.

TP is reset to 0x0000 each time a Reference message with Disc_Bit = ‘1’ becomes valid or if the node is not the current Time Master. TP is locked while TTOST[WGTD] = ‘1’ after

setting TTOCN[SGT] until the Reference message with Disc_Bit = ‘1’ becomes valid or until the node is no longer the current Time Master.

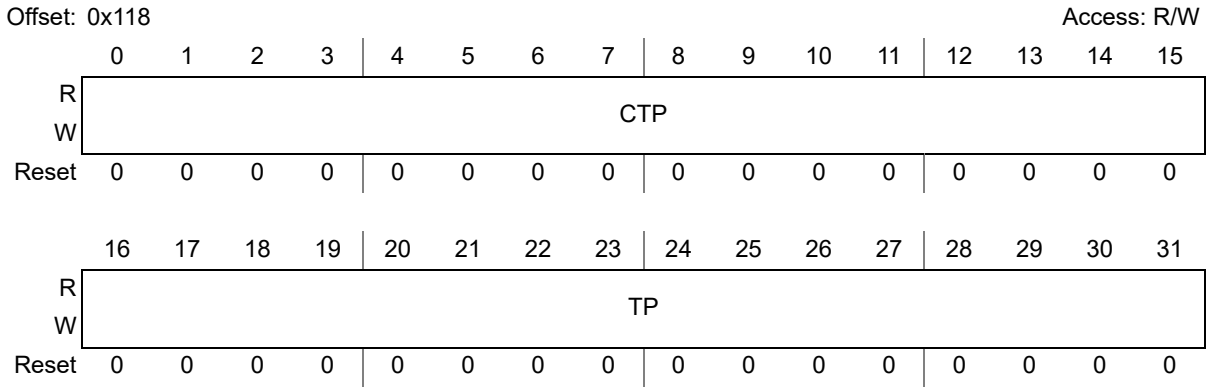


Figure 834. TT Global Time Preset Register (TTGTP)

Table 933. TTGTP field descriptions

Field	Description
0:15 CTP	Cycle Time Target Phase CTP is write-protected while TTOCN[ESCN] or TTOST[SPL] are set (refer to *Section 52.4.19*). (0x0000–0xFFFF) Defines target value of cycle time when a rising edge of event trigger is expected
16:31 TP	Time Preset TP is write-protected while TTOST[WGTD] is set. (0x0000–0x7FFF) Next Master Reference Mark = Master Reference Mark + TP 0x8000 Reserved (0x8001–0xFFFF) Next Master Reference Mark = Master Reference Mark - (0x10000 - TP)

52.4.5.2.54 TT Time Mark Register (TTTMK)

A time mark interrupt (**TTIR[TMI]** = ‘1’) is generated when the time base indicated by **TTOCN[TMC]** (cycle time, local time, or global time) has the same value as **TM**.

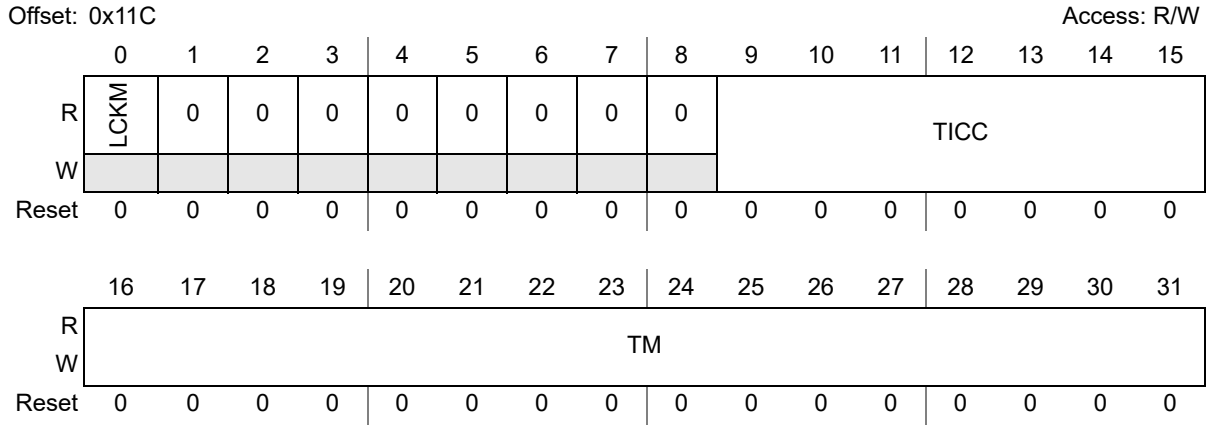


Figure 835. TT Time Mark Register (TTTMK)

Table 934. TTTMK field descriptions

Field	Description
0 LCKM	TT Time Mark Register Locked Always set by a write access to registers TTOCN. Set by write access to register TTTMK when TTOCN[TMC] ≠ '00'. Reset when the registers have been synchronized into the CAN clock domain. 0 Write access to TTTMK enabled 1 Write access to TTTMK locked
9:15 TICC	Time Mark Cycle Code Cycle count for which the time mark is valid. 000000x valid for all cycles 000001c valid every second cycle at cycle count mod2 = c 00001cc valid every fourth cycle at cycle count mod4 = cc 0001ccc valid every eighth cycle at cycle count mod8 = ccc 001cccc valid every sixteenth cycle at cycle count mod16 = cccc 01ccccc valid every thirty-second cycle at cycle count mod32 = ccccc 1cccccc valid every sixty-fourth cycle at cycle count mod64 = cccccc
16:31 TM	Time Mark (0x0000–0xFFFF) Time Mark

Note: When using byte access to register TTTMK it is recommended to first disable the time mark compare function (TTOCN[TMC] = '00') to avoid comparisons with inconsistent register values.

52.4.5.2.55 TT Interrupt Register (TTIR)

The flags are set when one of the listed conditions is detected (edge-sensitive). The flags remain set until the Host clears them. A flag is cleared by writing a '1' to the corresponding bit position. Writing a '0' has no effect. A hard reset will clear the register.

Offset: 0x120 Access: R/W

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	CER	AW	WT
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	IWT	ELC	SE2	SE1	TXO	TXU	GTE	GTD	GTW	SWE	TTMI	RTMI	SOG	CSM	SMC	SBC
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 836. TT Interrupt Register (TTIR)

Table 935. TTIR field descriptions

Field	Description
13 CER	Configuration Error Trigger out of order. 0 No error found in trigger list 1 Error found in trigger list
14 AW	Application Watchdog 0 Application watchdog served in time 1 Application watchdog not served in time
15 WT	Watch Trigger 0 No missing Reference message 1 Missing Reference message (Level 0: cycle time 0xFF00)
16 IWT	Initialization Watch Trigger The initialization is restarted by resetting **IWT**. 0 No missing Reference message during system startup 1 No system startup due to missing Reference message
17 ELC	Error Level Changed Not set when error level changed during initialization. 0 No change in error level 1 Error level changed
18 SE2	Scheduling Error 2 0 No scheduling error 2 1 Scheduling error 2 occurred
19 SE1	Scheduling Error 1 0 No scheduling error 1 1 Scheduling error 1 occurred
20 TXO	Tx Count Overflow 0 Number of Tx Trigger as expected 1 More Tx trigger than expected in one cycle
21 TXU	Tx Count Underflow 0 Number of Tx Trigger as expected 1 Less Tx trigger than expected in one cycle

Table 935. TTIR field descriptions (continued)

Field	Description
22 GTE	Global Time Error Synchronization deviation SD exceeds limit specified by TTOCF[LDSDL], M_TTCAN Level 0, 2 only. 0 Synchronization deviation within limit 1 Synchronization deviation exceeded limit
23 GTD	Global Time Discontinuity 0 No discontinuity of global time 1 Discontinuity of global time
24 GTW	Global Time Wrap 0 No global time wrap occurred 1 Global time wrap from 0xFFFF to 0x0000 occurred
25 SWE	Stop Watch Event 0 No rising/falling edge at stop watch trigger pin detected 1 Rising/falling edge at stop watch trigger pin detected
26 TTMI	Trigger Time Mark Event Internal Internal time mark events are configured by trigger memory element TMIN (refer to *Section 52.4.6.6: Trigger Memory element*). Set when the trigger memory element becomes active, and the M_TTCAN is in synchronization state In_Gap or In_Schedule. 0 Time mark not reached 1 Time mark reached (Level 0: cycle time TTOCF[RTO] x 0x200)
27 RTMI	Register Time Mark Interrupt Set when time referenced by TTOCN[TMC] (cycle, local, or global) equals TTTMK[TM], independent of the synchronization state. 0 Time mark not reached 1 Time mark reached
28 SOG	Start of Gap 0 No reference message seen with Next_is_Gap bit set 1 Reference message with Next_is_Gap bit set becomes valid
29 CSM	Change of Synchronization Mode 0 No change in master to slave relation or schedule synchronization 1 Master to slave relation or schedule synchronization changed, also set when TTOST[SPL] is reset.
30 SMC	Start of Matrix Cycle 0 No Matrix Cycle started since bit has been reset 1 Matrix Cycle started
31 SBC	Start of Basic Cycle 0 No Basic Cycle started since bit has been reset 1 Basic Cycle started

52.4.5.2.56 TT Interrupt Enable Register (TTIE)

The settings in the TT Interrupt Enable register determine which status changes in the TT Interrupt Register will result in an interrupt.

Offset: 0x124 Access: R/W

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	CERE	AWE	WTE
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	IWTE	ELCE	SE2E	SE1E	TXOE	TXUE	GTEE	GTDE	GTWE	SWEE	TTMIE	RTMIE	SOGE	CSME	SMCE	SBCE
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 837. TT Interrupt Enable Register (TTIE)

Table 936. TTIE field descriptions

Field	Description
13 CERE	Configuration Error Interrupt Enable 0 TT interrupt disabled 1 TT interrupt enabled
14 AWE	Application Watchdog Interrupt Enable 0 TT interrupt disabled 1 TT interrupt enabled
15 WTE	Watch Trigger Interrupt Enable 0 TT interrupt disabled 1 TT interrupt enabled
16 IWTE	Initialization Watch Trigger Interrupt Enable 0 TT interrupt disabled 1 TT interrupt enabled
17 ELCE	Change Error Level Interrupt Enable 0 TT interrupt disabled 1 TT interrupt enabled
18 SE2E	Scheduling Error 2 Interrupt Enable 0 TT interrupt disabled 1 TT interrupt enabled
19 SE1E	Scheduling Error 1 Interrupt Enable 0 TT interrupt disabled 1 TT interrupt enabled
20 TXOE	Tx Count Overflow Interrupt Enable 0 TT interrupt disabled 1 TT interrupt enabled
21 TXUE	Tx Count Underflow Interrupt Enable 0 TT interrupt disabled 1 TT interrupt enabled
22 GTEE	Global Time Error Interrupt Enable 0 TT interrupt disabled 1 TT interrupt enabled

Table 936. TTIE field descriptions (continued)

Field	Description
23 GTDE	Global Time Discontinuity Interrupt Enable 0 TT interrupt disabled 1 TT interrupt enabled
24 GTWE	Global Time Wrap Interrupt Enable 0 TT interrupt disabled 1 TT interrupt enabled
25 SWEE	Stop Watch Event Interrupt Enable 0 TT interrupt disabled 1 TT interrupt enabled
26 TTMIE	Trigger Time Mark Event Internal Enable 0 TT interrupt disabled 1 TT interrupt enabled
27 RTMIE	Register Time Mark Interrupt Enable 0 TT interrupt disabled 1 TT interrupt enabled
28 SOGE	Start of Gap Interrupt Enable 0 TT interrupt disabled 1 TT interrupt enabled
29 CSME	Change of Synchronization Mode Interrupt Enable 0 TT interrupt disabled 1 TT interrupt enabled
30 SMCE	Start of Matrix Cycle Interrupt Enable 0 TT interrupt disabled 1 TT interrupt enabled
31 SBCE	Start of Basic Cycle Interrupt Enable 0 TT interrupt disabled 1 TT interrupt enabled

52.4.5.2.57 TT Interrupt Line Select Register (TTILS)

The TT Interrupt Line Select register assigns an interrupt generated by a specific interrupt flag from the TT Interrupt Register to one of the two module interrupt lines. For interrupt generation the respective interrupt line has to be enabled via ILE[EINT0] and ILE[EINT1].

- 0 TT interrupt assigned to interrupt line 0
- 1 TT interrupt assigned to interrupt line 1

Offset: 0x128 Access: R/W

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	CERL	AWL	WTL
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	IWTL	ELCL	SE2L	SE1L	TXOL	TXUL	GTEL	GTDL	GTWL	SWEL	TTMIL	RTMIL	SOGL	CSML	SMCL	SBCL
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 838. TT Interrupt Line Select Register (TTILS)

Table 937. TTILS field descriptions

Field	Description
13 CERL	Configuration Error Interrupt Line 0 TT interrupt assigned to interrupt line 0 1 TT interrupt assigned to interrupt line 1
14 AWL	Application Watchdog Interrupt Line 0 TT interrupt assigned to interrupt line 0 1 TT interrupt assigned to interrupt line 1
15 WTL	Watch Trigger Interrupt Line 0 TT interrupt assigned to interrupt line 0 1 TT interrupt assigned to interrupt line 1
16 IWTL	Initialization Watch Trigger Interrupt Line 0 TT interrupt assigned to interrupt line 0 1 TT interrupt assigned to interrupt line 1
17 ELCL	Change Error Level Interrupt Line 0 TT interrupt assigned to interrupt line 0 1 TT interrupt assigned to interrupt line 1
18 SE2L	Scheduling Error 2 Interrupt Line 0 TT interrupt assigned to interrupt line 0 1 TT interrupt assigned to interrupt line 1
19 SE1L	Scheduling Error 1 Interrupt Line 0 TT interrupt assigned to interrupt line 0 1 TT interrupt assigned to interrupt line 1
20 TXOL	Tx Count Overflow Interrupt Line 0 TT interrupt assigned to interrupt line 0 1 TT interrupt assigned to interrupt line 1
21 TXUL	Tx Count Underflow Interrupt Line 0 TT interrupt assigned to interrupt line 0 1 TT interrupt assigned to interrupt line 1
22 GTEL	Global Time Error Interrupt Line 0 TT interrupt assigned to interrupt line 0 1 TT interrupt assigned to interrupt line 1

Table 937. TTILS field descriptions (continued)

Field	Description
23 GTDL	Global Time Discontinuity Interrupt Line 0 TT interrupt assigned to interrupt line 0 1 TT interrupt assigned to interrupt line 1
24 GTWL	Global Time Wrap Interrupt Line 0 TT interrupt assigned to interrupt line 0 1 TT interrupt assigned to interrupt line 1
25 SWEL	Stop Watch Event Interrupt Line 0 TT interrupt assigned to interrupt line 0 1 TT interrupt assigned to interrupt line 1
26 TTMIL	Trigger Time Mark Event Internal Line 0 TT interrupt assigned to interrupt line 0 1 TT interrupt assigned to interrupt line 1
27 RTMIL	Register Time Mark Interrupt Line 0 TT interrupt assigned to interrupt line 0 1 TT interrupt assigned to interrupt line 1
28 SOGL	Start of Gap Interrupt Line 0 TT interrupt assigned to interrupt line 0 1 TT interrupt assigned to interrupt line 1
29 CSML	Change of Synchronization Mode Interrupt Line 0 TT interrupt assigned to interrupt line 0 1 TT interrupt assigned to interrupt line 1
30 SMCL	Start of Matrix Cycle Interrupt Line 0 TT interrupt assigned to interrupt line 0 1 TT interrupt assigned to interrupt line 1
31 SBCL	Start of Basic Cycle Interrupt Line 0 TT interrupt assigned to interrupt line 0 1 TT interrupt assigned to interrupt line 1

52.4.5.2.58 TT Operation Status Register (TTOST)

Offset: 0x12C Access: R

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	SPL	WECS	AWE	WFE	GSI	TMP			GFI	WGTD	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	RTO								QCS	QGTP	SYS		MS		EL	
W																
Reset	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0

Figure 839. TT Operation Status Register (TTOST)

Table 938. TTOST field descriptions

Field	Description
0 SPL	Schedule Phase Lock The bit is valid only when external synchronization is enabled (TTOCN[ESCN] = ‘1’). In this case it signals that the difference between cycle time configured by TTGTP[CTP] and the cycle time at the rising edge at Event Trigger pin is less than or equal to 9 NTU (refer to *Section 52.4.19*). 0 Phase outside range 1 Phase inside range
1 WECS	Wait for External Clock Synchronization 0 No external clock synchronization pending 1 Node waits for external clock synchronization to take effect. The bit is reset at the start of the next basic cycle.
2 AWE	Application Watchdog Event The application watchdog is served by reading TTOST. When the watchdog is not served in time, bit AWE is set, all M_TTCAN communication is stopped, and the M_TTCAN is set into Bus Monitoring Mode. 0 Application Watchdog served in time 1 Failed to serve Application Watchdog in time
3 WFE	Wait for Event 0 No Gap announced, reset by a Reference Message with Next_is_Gap = ‘0’ 1 Reference Message with Next_is_Gap = ‘1’ received
4 GSI	Gap Started Indicator 0 No Gap in schedule, reset by each reference message and for all time slaves 1 Gap time after Basic Cycle has started
5:7 TMP	Time Master Priority (0x0-0x7) Priority of actual Time Master

Table 938. TTOST field descriptions (continued)

Field	Description
8 GFI	Gap Finished Indicator Set when the CPU writes TTOCN[FGP], or by a Time Mark Interrupt if TMG = '1', or via input Event Trigger pin if TTOCN[GCS] = '1'. Not set by Ref_Trigger_Gap or when Gap is finished by another node sending a reference message. 0 Reset at the end of each reference message 1 Gap finished by M_TTCAN
9 WGTD	Wait for Global Time Discontinuity 0 No global time preset pending 1 Node waits for the global time preset to take effect. The bit is reset when the node has transmitted a Reference message with **Disc_Bit** = '1' or after it received a Reference message.
16:23 RTO	Reference Trigger Offset The Reference Trigger Offset value is a signed integer with a range from -127 (0x81) to 127 (0x7F). There is no notification when the lower limit of-127 is reached. In case the M_TTCAN becomes Time Master (MS[1:0] = '11'), the reset of RTO is delayed due to synchronization between Host and CAN clock domain. For time slaves the value configured by TTOCF[IRTO] is read. (0x00-0xFF) Actual Reference Trigger offset value
24 QCS	Quality of Clock Speed Only relevant in M_TTCAN Level 0 and Level 2, otherwise fixed to '1'. 0 Local clock speed not synchronized to Time Master clock speed 1 Synchronization Deviation $\leq$ **SDL**
25 QGTP	Quality of Global Time Phase Only relevant in M_TTCAN Level 0 and Level 2, otherwise fixed to '0'. 0 Global time not valid 1 Global time in phase with Time Master
26:27 SYS	Synchronization State 00 Out of Synchronization 01 Synchronizing to M_TTCAN communication 10 Schedule suspended by Gap (In_Gap) 11 Synchronized to schedule (In_Schedule)
28:29 MS	Master State 00 Master_Off, no master properties relevant 01 Operating as Time Slave 10 Operating as Backup Time Master 11 Operating as current Time Master
30:31 EL	Error Level 00 Severity 0: No Error 01 Severity 1: Warning 10 Severity 2: Error 11 Severity 3: Severe Error

52.4.5.2.59 TUR Numerator Actual Register (TURNA)

There is no drift compensation in M_TTCAN Level 1 (NAV = NC). In M_TTCAN Level 0 and Level 2, the drift between the node's local clock and the time master's local clock is calculated.

The drift is compensated when the Synchronization Deviation (difference between NC and the calculated NAV) is not more than $2^{(TTOCF[LDSDL] + 5)}$. With TTOCF[LDSDL] <= 7, this results in a maximum range for NAV of (NC - 0x1000) <= NAV <= (NC + 0x1000).

Offset: 0x130 Access: R

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	NAV	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	NAV															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 840. TUR Numerator Actual Register (TURNA)

Table 939. TURNA field descriptions

Field	Description
14:31 NAV	Numerator Actual Value <= 0x0EFFF Illegal value (0x0F000–0x20FFF) Actual numerator value >= 0x21000 Illegal value

52.4.5.2.60 TT Local and Global Time Register (TTLGT)

Offset: 0x134 Access: R

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	GT															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	LT															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 841. TT Local and Global Time Register (TTLGT)

Table 940. TTLGT field descriptions

Field	Description
0:15 GT	Global Time Non-fractional part of the sum of the node's local time and its local offset (refer to *Section 52.4.14*). (0x0000–0xFFFF) Global time value of M_TTCAN network
16:31 LT	Local Time Non-fractional part of local time, incremented once each local NTU (refer to *Section 52.4.14*). (0x0000–0xFFFF) Local time value of M_TTCAN node

52.4.5.2.61 TT Cycle Time and Count Register (TTCTC)

Offset: 0x138 Access: R

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	CC					
W																
Reset	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	CT															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 842. TT Cycle Time and Count Register (TTCTC)

Table 941. TTCTC field descriptions

Field	Description
10:15 CC	Cycle Count (0x00–0x3F) Number of actual Basic Cycle in the System Matrix
16:31 CT	Cycle Time Non-fractional part of the difference of the node's local time and Ref_Mark (refer to *Section 52.4.14*). (0x0000–0xFFFF) Cycle time value of M_TTCAN Basic Cycle

52.4.5.2.62 TT Capture Time Register (TTCPT)

Offset: 0x13C Access: R

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	SWV															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	CCV					
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 843. TT Capture Time Register (TTCPT)

Table 942. TTCPT field descriptions

Field	Description
0:15 SWV	Stop Watch Value On a rising/falling edge (as configured via TTOCN[SWP]) at the Stop Watch Trigger pin, when TTOCN[SWS] is ≠ '00' and TTIR[SWE] is '0', the actual time value as selected by TTOCN[SWS] (cycle, local, global) is copied to SWV and TTIR[SWE] will be set to '1'. Capturing of the next stop watch value is enabled by resetting TTIR[SWE]. (0x0000–0xFFFF) Captured Stop Watch value
26:31 CCV	Cycle Count Value Cycle count value captured together with SWV. (0x00–0x3F) Captured cycle count value

52.4.5.2.63 TT Cycle Sync Mark Register (TTCSM)

Offset: 0x140 Access: R

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	CSM															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 844. TT Cycle Sync Mark Register (TTCSM)

Table 943. TTCSM field descriptions

Field	Description
16:31 CSM	Cycle Sync Mark The Cycle Sync Mark is measured in cycle time. It is updated when the reference message becomes valid and retains its value until the next reference message becomes valid. (0x0000–0xFFFF) Captured cycle time

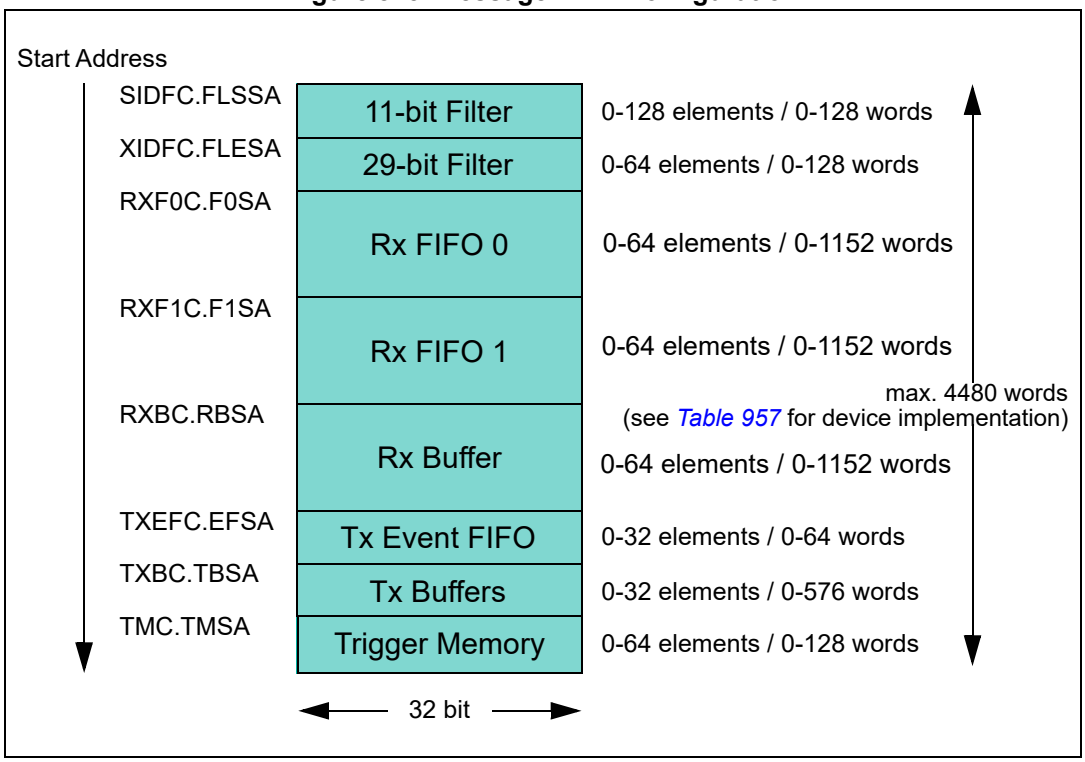
52.4.6 Message RAM

For storage of Rx/Tx messages and for storage of the filter configuration a single- or dual-ported Message RAM has to be connected to the M_TTCAN module.

The Message RAM has a width of 32 bits plus ECC checking bit. The M_TTCAN module can be configured to allocate up to 4480 words in the Message RAM (for the chip-specific implementation details of each module's instance, refer to device configuration chapter). It is not necessary to configure each of the sections listed in *Figure 845*, nor is there any restriction with respect to the sequence of the sections.

When operated in CAN FD mode the required Message RAM size strongly depends on the element size configured for Rx FIFO0, Rx FIFO1, Rx Buffers, and Tx Buffers via RXESC.F0DS, RXESC.F1DS, RXESC.RBDS, and TXESC.TBDS.

Figure 845. Message RAM Configuration



When the M_TTCAN addresses the Message RAM it addresses 32-bit words, not single bytes. The configured start addresses are 32-bit word addresses that is only bits 15 to 2 are evaluated, the two least significant bits are ignored.

Note: The M_TTCAN does not check for erroneous configuration of the Message RAM. Especially the configuration of the start addresses of the different sections and the number of elements of each section has to be done carefully to avoid falsification or loss of data.

52.4.6.1 Rx Buffer and FIFO element

Up to 64 Rx Buffers and two Rx FIFOs can be configured in the Message RAM. Each Rx FIFO section can be configured to store up to 64 received messages.The structure of a Rx Buffer/ FIFO element is shown below.

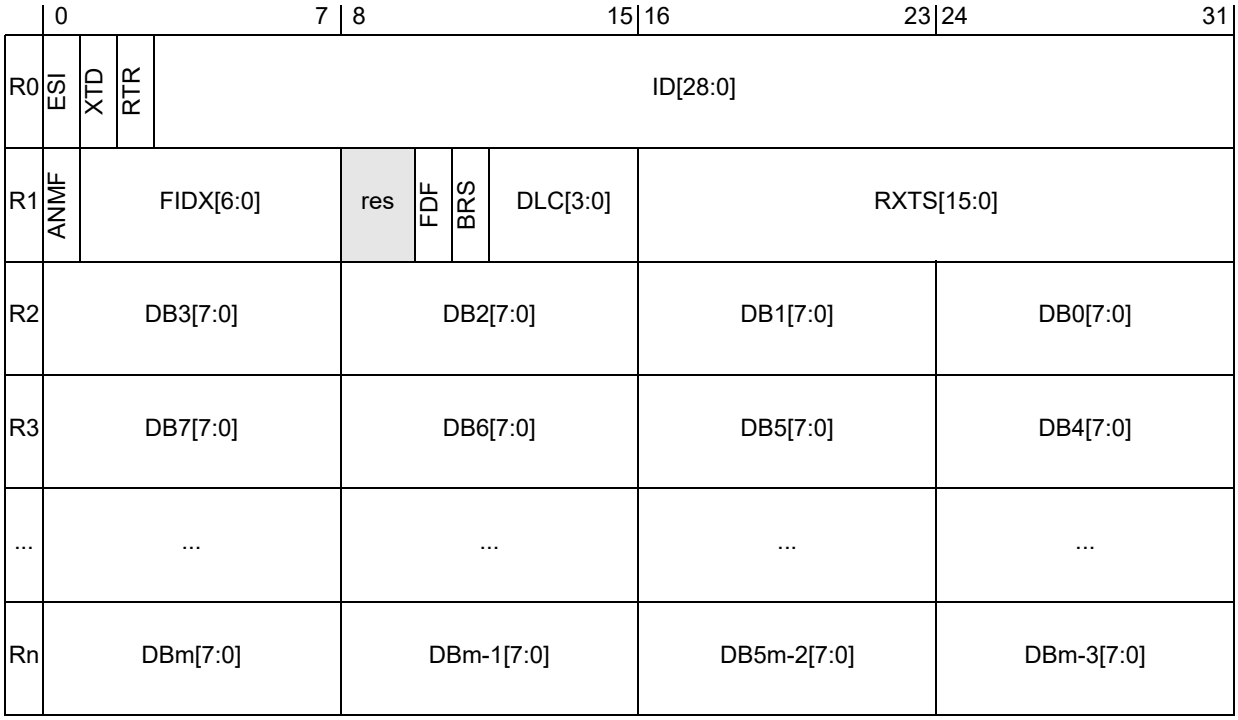


Figure 846. Rx Buffer and FIFO element

Table 944. Rx Buffer and FIFO element descriptions

Field	Description
R0 bit 0 ESI: Error State Indicator	0 Transmitting node is error active 1 Transmitting node is error passive
R0 bit 1 XTD: Extended Identifier	Signals to the Host whether the received frame has a standard or extended identifier. 0 11-bit standard identifier 1 29-bit extended identifier
R0 bit 2 RTR: Remote Transmission Request	Signals to the Host whether the received frame is a data frame or a remote frame. 0 Received frame is a data frame 1 Received frame is a remote frame

Table 944. Rx Buffer and FIFO element descriptions (continued)

Field	Description
R0 bits 3:31 ID[28:0]: Identifier	Standard or extended identifier depending on bit XTD. A standard identifier is stored into ID[28:18].
R1 bit 0 ANMF: Accepted Non-matching Frame	Acceptance of non-matching frames may be enabled via GFC[ANFS] and GFC[ANFE]. 0 Received frame matching filter index FIDX 1 Received frame did not match any Rx filter element
R1 bits 1:7 FIDX[6:0]: Filter Index	0-127 = Index of matching Rx acceptance filter element (invalid if ANMF = '1'). Range is 0 to SIDFC[LSS] - 1 resp. XIDFC[LSE] - 1.
R1 bit 10 FDF: FD Format	0 Standard frame format 1 CAN FD frame format (new DLC-coding and CRC)
R1 bit 11 BRS: Bit Rate Switch	0 Frame received without bit rate switching 1 Frame received with bit rate switching
R1 bits 12:15 DLC[3:0]: Data Length Code	0-8 Received frame has 0-8 data bytes 9-1 Received frame has 8 data bytes
R1 bits 16:31 RXTS[15:0]: Rx Timestamp	Timestamp Counter value captured on start of frame reception. Resolution depending on configuration of the Timestamp Counter Prescaler TSCC[TCP].
R2 bits 0:7	DB3[7:0]: Data Byte 3
R2 bits 8:15	DB2[7:0]: Data Byte 2
R2 bits 16:23	DB1[7:0]: Data Byte 1
R2 bits 24:31	DB0[7:0]: Data Byte 0
R3 bits 0:7	DB7[7:0]: Data Byte 7
R3 bits 8:15	DB6[7:0]: Data Byte 6
R3 bits 16:23	DB5[7:0]: Data Byte 5
R3 bits 24:31	DB4[7:0]: Data Byte 4
Rn bits 0:7	DBm[7:0]: Data Byte m
Rn bits 8:15	DBm-1[7:0]: Data Byte m-1
Rn bits 16:23	DBm-2[7:0]: Data Byte m-2
Rn bits 24:31	DBm-3[7:0]: Data Byte m-3

Note: Depending on the configuration of the element size (RXESC), between two and sixteen 32-bit words (Rn = 3...17) are used for storage of a CAN message's data field.

52.4.6.2 Tx Buffer element

The Tx Buffers section can be configured to hold dedicated Tx Buffers as well as a Tx FIFO / Tx Queue. In case that the Tx Buffers section is shared by dedicated Tx buffers and a Tx FIFO / Tx Queue, the dedicated Tx Buffers start at the beginning of the Tx Buffers section followed by the buffers assigned to the Tx FIFO or Tx Queue. The Tx Handler distinguishes between dedicated Tx Buffers and Tx FIFO / Tx Queue by evaluating the Tx Buffer configuration TXBC[TFQS] and TXBC[NDTB].

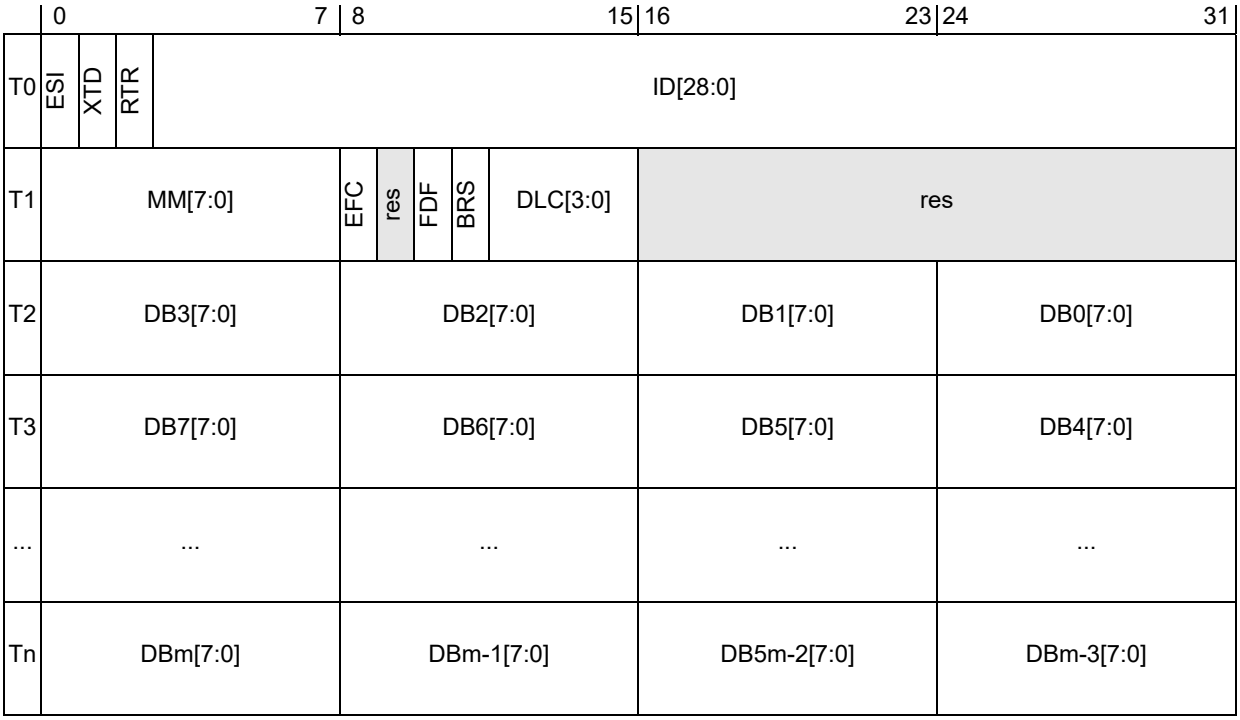


Figure 847. Tx Buffer element

Table 945. Tx Buffer element descriptions

Field	Description
T0 bit 0 ESI: Error State Indicator	0 ESI bit in CAN FD format depends only on error passive flag 1 ESI bit in CAN FD format transmitted recessive
T0 bit 1 XTD: Extended Identifier	0 11-bit standard identifier 1 29-bit extended identifier
T0 bit 2 RTR: Remote Transmission Request	0 Transmit data frame 1 Transmit remote frame
T0 bits 3:31 ID[28:0]: Identifier	Standard or extended identifier depending on bit XTD. A standard identifier has to be written to ID[28:18].
T1 bits 0:7 MM[7:0]: Message Marker	Written by CPU during Tx Buffer configuration. Copied into Tx Event FIFO element for identification of Tx message status.

Table 945. Tx Buffer element descriptions (continued)

Field	Description
T1 bit 8 EFC **Event FIFO Control**	0 Don't store Tx events 1 Store Tx events
T1 bit 10 FDF **FD Format**	0 Frame transmitted in Classic CAN format 1 Frame transmitted in CAN FD format
T1 bit 11 BRS **Bit Rat Switching**	0 CAN FD frames transmitted without bit rate switching 1 CAN FD frames transmitted with bit rate switching
T1 bits 12:15 DLC[3:0]	Data Length Code 0-8 Transmit frame with 0-8 data bytes 9-15 Transmit frame with 8 data bytes
T2 bits 0:7	DB3[7:0]: Data Byte 3
T2 bits 8:15	DB2[7:0]: Data Byte 2
T2 bits 16:23	DB1[7:0]: Data Byte 1
T2 bits 24:31	DB0[7:0]: Data Byte 0
T3 bits 0:7	DB7[7:0]: Data Byte 7
T3 bits 8:15	DB6[7:0]: Data Byte 6
T3 bits 16:23	DB5[7:0]: Data Byte 5
T3 bits 24:31	DB4[7:0]: Data Byte 4
Tn bits 0:7	DBm[7:0]: Data Byte m
Tn bits 8:15	DBm-1[7:0]: Data Byte m-1
Tn bits 16:23	DBm-2[7:0]: Data Byte m-2
Tn bits 24:31	DBm-3[7:0]: Data Byte m-3

Note: Depending on the configuration of the element size (TXESC), between two and sixteen 32-bit words (Tn = 3 ..17) are used for storage of a CAN message's data field.

52.4.6.3 Tx Event FIFO element

Each element stores information about transmitted messages. By reading the Tx Event FIFO the Host CPU gets this information in the order the messages were transmitted. Status information about the Tx Event FIFO can be obtained from register TXEFS.

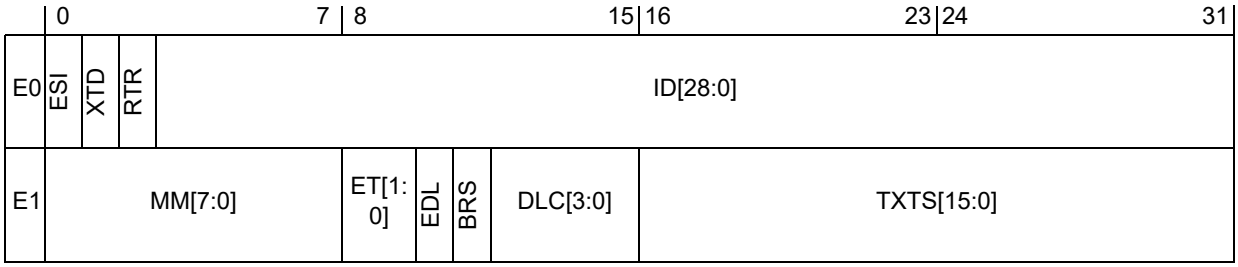


Figure 848. Tx Event FIFO element

Table 946. Tx Event FIFO element

Field	Description
E0 bit 0 ESI	Error State Indicator 0 Transmitting node is error active 1 Transmitting node is error passive
E0 bit 1 XTD	Extended Identifier 0 11-bit standard identifier 1 29-bit extended identifier
E0 bit 2 RTR	Remote Transmission Request 0 Data frame transmitted 1 Remote frame transmitted
E0 bits 3:31 ID[28:0]	Identifier Standard or extended identifier depending on bit XTD. A standard identifier has to be written to ID[28:18].
E1 bits 0:7 MM[7:0]	Message Marker Copied from Tx Buffer into Tx Event FIFO element for identification of Tx message status.
E1 bit 8:9 ET[1:0]	Event Type 00 Reserved 01 Tx event 10 Transmission in spite of cancellation (always set for transmissions in DAR mode) 11 Reserved
E1 bit 10 EDL	Extended Data Length 0 Standard frame format 1 CAN FD frame format (new DLC-coding and CRC)
E1 bit 11 BRS	Bit Rate Switch 0 Frame transmitted without bit rate switching 1 Frame transmitted with bit rate switching
E1 bits 12:15 DLC[3:0]	Data Length Code 0-8 Frame with 0-8 data bytes transmitted 9-1 Frame with 8 data bytes transmitted
E1 bits 16:31 TXTS[15:0]	Tx Timestamp Timestamp Counter value captured on start of frame transmission. Resolution depending on configuration of the Timestamp Counter Prescaler TSCC[TCP].

52.4.6.4 Standard Message ID Filter element

Up to 128 filter elements can be configured for 11-bit standard IDs. When accessing a Standard Message ID Filter element, its address is the Filter List Standard Start Address SIDFC.FLSSA plus the index of the filter element (0…127).

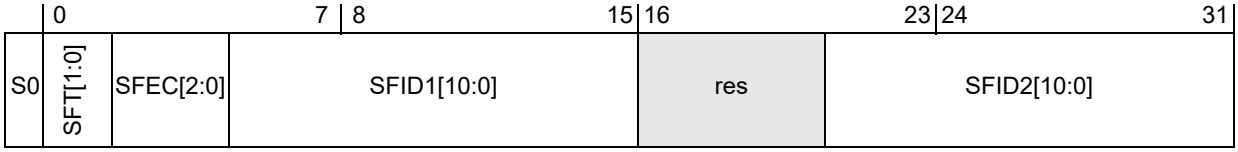


Figure 849. Standard Message ID Filter element

Table 947. Standard Message ID Filter element description

Field	Description
Bits 0:1 SFT[1:0]	Standard Filter Type 00 Range filter from SFID1 to SFID2 (SFID2 ≥ SFID1) 01 Dual ID filter for SFID1 or SFID2 10 Classic filter: SFID1 = filter, SFID2 = mask 11 Reserved
Bit 2:4 SFEC[2:0]	Standard Filter Element Configuration All enabled filter elements are used for acceptance filtering of standard frames. Acceptance filtering stops at the first matching enabled filter element or when the end of the filter list is reached. If SFEC = ‘100’, ‘101’, or ‘110’ a match sets interrupt flag IR[HPM] and, if enabled, an interrupt is generated. In this case register HPMS is updated with the status of the priority match. 000 Disable filter element 001 Store in Rx FIFO 0 if filter matches 010 Store in Rx FIFO 1 if filter matches 011 Reject ID if filter matches 100 Set priority if filter matches 101 Set priority and store in FIFO 0 if filter matches 110 Set priority and store in FIFO 1 if filter matches 111 Store into Rx Buffer, configuration of SFT[1:0] ignored
Bits 5:15 SFID1[10:0]	Standard Filter ID 1 First ID of standard ID filter element. When filtering for Rx Buffers or for debug messages this field defines the ID of a standard message to be stored. The received identifiers must match exactly, no masking mechanism is used.

Table 947. Standard Message ID Filter element description (continued)

Field		Description
Bits 21:31	**SFID2[10:0]**	Standard Filter ID 2 This bit field has a different meaning depending on the configuration of SFEC: – SFEC = '001'...'110' Second ID of standard ID filter element – SFEC = '111' Filter for Rx Buffers or for debug messages
	SFID2[10:9]	Decides whether the received message is stored into an Rx Buffer or treated as message A, B, or C of the debug message sequence. 00 Store message into an Rx Buffer 01 Debug Message A 10 Debug Message B 11 Debug Message C
	SFID2[8:6]	Is used to control the Filter Event pins at the Extension Interface. A one at the respective bit position enables generation of a pulse at the related filter event pin with the duration of one Host clock period in case the filter matches.
	SFID2[5:0]	Defines the offset to the Rx Buffer Start Address RXBC.RBSA for storage of a matching message.

Note: In case a reserved value is configured, the filter element is considered disabled.

52.4.6.5 Extended Message ID Filter element

Up to 64 filter elements can be configured for 29-bit extended IDs. When accessing an Extended Message ID Filter element, its address is the Filter List Extended Start Address XIDFC[FLESA] plus two times the index of the filter element (0…63).

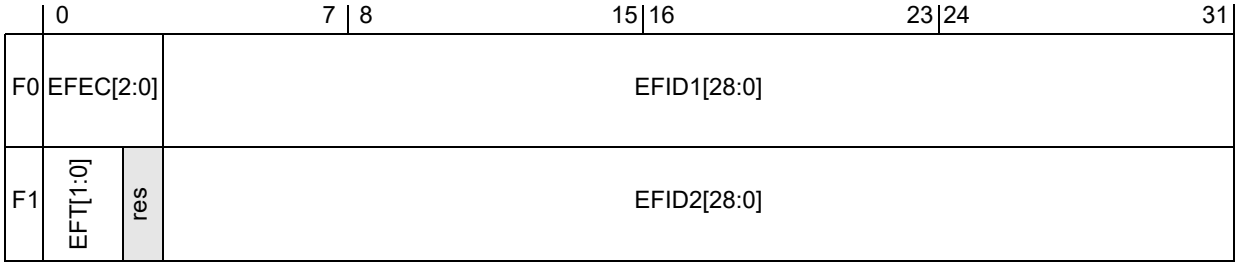


Figure 850. Extended Message ID Filter element

Table 948. Extended Message ID Filter element description

Field		Description
F0 bits 0:2 EFEC[2:0]		Extended Filter Element Configuration All enabled filter elements are used for acceptance filtering of extended frames. Acceptance filtering stops at the first matching enabled filter element or when the end of the filter list is reached. If EFEC = '100', '101', or '110' a match sets interrupt flag IR[HPM] and, if enabled, an interrupt is generated. In this case register HPMS is updated with the status of the priority match. 000 Disable filter element 001 Store in Rx FIFO 0 if filter matches 010 Store in Rx FIFO 1 if filter matches 011 Reject ID if filter matches 100 Set priority if filter matches 101 Set priority and store in FIFO 0 if filter matches 110 Set priority and store in FIFO 1 if filter matches 111 Store into Rx Buffer, configuration of EFT[1:0] ignored
F0 bits 3:31 EFID1[28:0]		Extended Filter ID 1 First ID of extended ID filter element. When filtering for Rx Buffers or for debug messages this field defines the ID of an extended message to be stored. The received identifiers must match exactly, only XIDAM masking mechanism (refer to *Section 52.4.7.13.5*) is used.
F1 bits 0:1 EFT[1:0]		Extended Filter Type 00 Range filter from EFID1 to EFID2 (EFID2 ≥ EFID1) 01 Dual ID filter for EFID1 or EFID2 10 Classic filter: EFID1 = filter, EFID2 = mask 11 Range filter from EFID1 to EFID2 (EFID2 ≥ EFID1), XIDAM mask not applied
F1 bits 3:31	**EFID2[28:0]**	Extended Filter ID 2 This bit field has a different meaning depending on the configuration of EFEC: – EFEC = '001'...'110' Second ID of extended ID filter element – EFEC = '111' Filter for Rx Buffers or for debug messages
	EFID2[10:9]	Decides whether the received message is stored into an Rx Buffer or treated as message A, B, or C of the debug message sequence. 00 Store message into an Rx Buffer 01 Debug Message A 10 Debug Message B 11 Debug Message C
	EFID2[8:6]	Is used to control the Filter Event pins at the Extension Interface. A 1 at the respective bit position enables generation of a pulse at the related filter event pin with the duration of one Host clock period in case the filter matches.
	EFID2[5:0]	Defines the offset to the Rx Buffer Start Address RXBC.RBSA for storage of a matching message.

52.4.6.6 Trigger Memory element

Up to 64 trigger memory elements can be configured. When accessing a Trigger Memory element, its address is the Trigger Memory Start Address TTTMC[TMSA] plus the index of the trigger memory element (0…63).

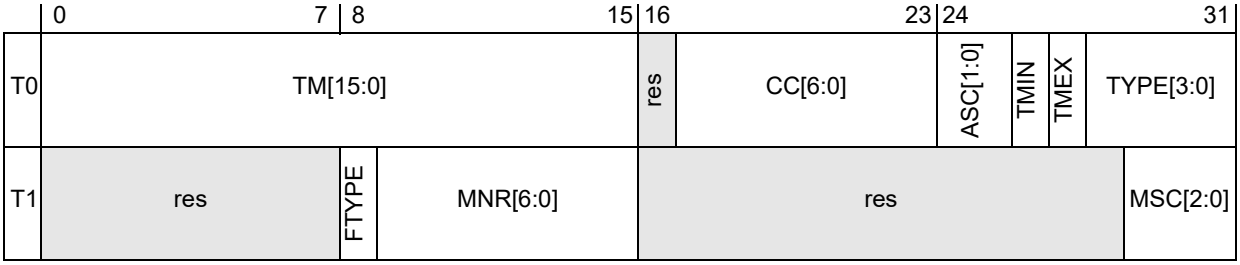


Figure 851. Trigger Memory element

Table 949. Trigger Memory element descriptions

Field	Description
T0 bit 0:15 TM[15:0]	Time Mark Cycle time for which the trigger becomes active.
T0 bit 17:23 CC[6:0]	Cycle Code Cycle count for which the trigger is valid. Ignored for trigger types Tx_Ref_Trigger, Tx_Ref_Trigger_Gap, Watch_Trigger, Watch_Trigger_Gap, End_of_List. 0b000000x valid for all cycles 0b000001c valid every 2nd cycle at cycle count mod2 = c 0b00001cc valid every 4th cycle at cycle count mod4 = cc 0b0001ccc valid every 8th cycle at cycle count mod8 = ccc 0b001cccc valid every 16th cycle at cycle count mod16 = cccc 0b01ccccc valid every 32nd cycle at cycle count mod32 = ccccc 0b1cccccc valid every 64th cycle at cycle count mod64 = cccccc
T0 bit 24:25 ASC[1:0]	Asynchronous Serial Communication 00 No ASC operation 01 Reserved, do not use 10 Node is ASC receiver 11 Node is ASC transmitter
T0 bit 26 TMIN	Time Mark Event Internal 0 No action 1 TTIR.TTMI is set when trigger memory element becomes active
T0 bit 27 TMEX	Time Mark Event External 0 No action 1 Pulse at Trigger Time Mark Interrupt Pulse output with the length of one CAN clock period is generated when the time mark of the trigger memory element becomes active and TTOCN.TTMIE = ‘1’

Table 949. Trigger Memory element descriptions (continued)

Field	Description
T0 bit 28:31 TYPE[3:0]	Trigger Type 0000 Tx_Ref_Trigger - valid when not in Gap 0001 Tx_Ref_Trigger_Gap - valid when in Gap 0010 Tx_Trigger_Single - starts a single transmission in an exclusive time window 0011 Tx_Trigger_Continuous - starts continuous transmission in an exclusive time window 0100 Tx_Trigger_Arbitration - starts a transmission in an arbitrating time window 0101 Tx_Trigger_Merged - starts a merged arbitration window 0110 Watch_Trigger - valid when not in Gap 0111 Watch_Trigger_Gap - valid when in Gap 1000 Rx_Trigger - check for reception 1001 Time_Base_Trigger - only control TMIN, TMEX, and ASC 1010…1111 = End_of_List - illegal type, causes configuration error **Note:** For ASC operation (ASC = '10', '11') only trigger types Rx_Trigger and Time_Base_Trigger should be used.
T1 bit 8 FTYPE	Filter Type 0 11-bit standard message ID 1 29-bit extended message ID
T1 bit 9:15 MNR[6:0]	Message Number Transmission: Trigger is valid for configured Tx Buffer number. Valid values are 0 to 31. Reception: Trigger is valid for standard / extended message ID filter element number. Valid values are 0 to 63 resp. 0 to 127.
T1 bits 29:31 MSC[2:0]	Message Status Count Counts scheduling errors for periodic messages in exclusive time windows. It has no function for arbitrating messages and in event-driven CAN communication (ISO11898-1). 0-7= Actual status **Note:** The trigger memory elements have to be written when the M_TTCAN is in INIT state.Write access to the trigger memory elements outside INIT state is not allowed.There is an exception for TMIN and TMEX when they are defined as part of a trigger memory element of TYPE Tx_Ref_Trigger. In this case they become active at the time mark modified by the actual Reference Trigger Offset (TTOST[RTO]).

52.4.7 M_TTCAN functional description

52.4.7.1 Operating modes

52.4.7.1.1 Software initialization

Software initialization is started by setting bit CCCR[INIT], either by software or by a hardware reset, when an uncorrected bit error was detected in the Message RAM, or by going Bus_Off. While CCCR[INIT] is set, message transfer from and to the CAN bus is stopped, the status of the CAN bus output is recessive (HIGH). The counters of the Error Management Logic EML are unchanged. Setting CCCR[INIT] does not change any

configuration register. Resetting CCCR[INIT] finishes the software initialization. Afterwards the Bit Stream Processor BSP synchronizes itself to the data transfer on the CAN bus by waiting for the occurrence of a sequence of 11 consecutive recessive bits (=Bus_Idle) before it can take part in bus activities and start the message transfer.

Access to the M_TTCAN configuration registers is only enabled when both bits CCCR[INIT] and CCCR[CCE] are set (protected write).

CCCR[CCE] can only be set/reset while CCCR[INIT] = ‘1’. CCCR[CCE] is automatically reset when CCCR[INIT] is reset.

The following registers are reset when CCCR[CCE] is set:

- HPMS - High Priority Message Status
- RXF0S - Rx FIFO 0 Status
- RXF1S - Rx FIFO 1 Status
- TXFQS - Tx FIFO/Queue Status
- TXBRP - Tx Buffer Request Pending
- TXBTO - Tx Buffer Transmission Occurred
- TXBCF - Tx Buffer Cancellation Finished
- TXEFS - Tx Event FIFO Status
- TTOST - TT Operation Status
- TTLGT - TT Local and Global Time, only Global Time TTLGT[GT] is reset
- TTCTC - TT Cycle Time and Count
- TTCSM - TT Cycle Sync Mark

The Timeout Counter value TOCV[TOC] is preset to the value configured by TOCC[TOP] when CCCR[CCE] is set.

In addition the state machines of the Tx Handler and Rx Handler are held in idle state while CCCR[CCE] = ‘1’.

The following registers are only writeable while CCCR[CCE] = ‘0’

- TXBAR - Tx Buffer Add Request
- TXBCR - Tx Buffer Cancellation Request

CCCR[TEST] and CCCR[MON] can only be set by the Host while CCCR[INIT] = ‘1’ and CCCR[CCE] =‘1’. Both bits may be reset at any time. CCCR[DAR] can only be set/reset while CCCR[INIT] = ‘1’ and CCCR[CCE] = ‘1’.

Note: In case the Message RAM is equipped with parity or ECC functionality, it is recommended to initialize the Message RAM after hardware reset by writing such as 0x00000000 to each Message RAM word to create valid parity/ECC checksums. This avoids that reading from uninitialized Message RAM sections will activate interrupt IR.BEC (Bit Error Corrected) or IR.BEU (Bit Error Uncorrected).

52.4.7.2 Normal operation

The M_TTCAN’s default operating mode after hardware reset is event-driven CAN communication without time triggers (TTOCF[OM] = ‘00’). It is required that both CCCR[INIT] and CCCR[CCE] are set before the TT Operation Mode can be changed.

Once the M_TTCAN is initialized and CCCR[INIT] is reset to zero, the M_TTCAN synchronizes itself to the CAN bus and is ready for communication.

After passing the acceptance filtering, received messages including Message ID and DLC are stored into a dedicated Rx Buffer or into the Rx FIFO 0 or Rx FIFO 1.

For messages to be transmitted dedicated Tx Buffers or a Tx FIFO, or both, or a Tx Queue can be initialized or updated. Automated transmission on reception of remote frames is not implemented.

52.4.7.3 CAN FD Operation

There are two variants in the CAN FD frame format, first the CAN FD frame without bit rate switching. The second variant is the CAN FD frame where control field, data field, and CRC field of a CAN frame are transmitted with a higher bit rate than the beginning and the end of the frame.

The previously reserved bit in CAN frames with 11-bit identifiers and the first previously reserved bit in CAN frames with 29-bit identifiers will now be decoded as FDF bit. FDF = recessive signifies a CAN FD frame, FDF = dominant signifies a Classic CAN frame. In a CAN FD frame, the two bits following FDF, res and BRS, decide whether the bit rate inside of this CAN FD frame is switched. A CAN FD bit rate switch is signified by res = dominant and BRS = recessive. The coding of res = recessive is reserved for future expansion of the protocol. In case the M_TTCAN receives a frame with FDF = recessive and res = recessive, it will signal a Protocol Exception Event by setting bit PSR.PXE. When Protocol Exception Handling is enabled (CCCR.PXHD = ‘0’), this causes the operation state to change from Receiver (PSR.ACT = “10”) to Integrating (PSR.ACT = “00”) at the next sample point. In case Protocol Exception Handling is disabled (CCCR.PXHD = ‘1’), the M_TTCAN will treat a recessive res bit as an form error and will respond with an error frame.

CAN FD operation is enabled by programming CCCR.FDOE. In case CCCR.FDOE = ‘1’, transmission and reception of CAN FD frames is enabled. Transmission and reception of Classic CAN frames is always possible. Whether a CAN FD frame or a Classic CAN frame is transmitted can be configured via bit FDF in the respective Tx Buffer element. With CCCR.FDOE = ‘0’, received frames are interpreted as Classic CAN frames, which leads to the transmission of an error frame when receiving a CAN FD frame. When CAN FD operation is disabled, no CAN FD frames are transmitted even if bit FDF of a Tx Buffer element is set. CCCR.FDOE and CCCR.BRSE can only be changed while CCCR.INIT and CCCR.CCE are both set.

With CCCR.FDOE = ‘0’, the setting of bits FDF and BRS is ignored and frames are transmitted in Classic CAN format. With CCCR.FDOE = ‘1’ and CCCR.BRSE = ‘0’, only bit FDF of a Tx Buffer element is evaluated. With CCCR.FDOE = ‘1’ and CCCR.BRSE = ‘1’, transmission of CAN FD frames with bit rate switching is enabled. All Tx Buffer elements with bits FDF and BRS set are transmitted in CAN FD format with bit rate switching.It is not necessary to change the CAN operation mode after system startup.

A mode change during CAN operation is only recommended under the following conditions:

- The failure rate in the CAN FD data phase is significant higher than in the CAN FD arbitration phase. In this case disable the CAN FD bit rate switching option for transmissions
- During system startup all nodes are transmitting Classic CAN messages until it is verified that they are able to communicate in CAN FD format. If this is true, all nodes switch to CAN FD operation
- Wake-up messages in CAN Partial Networking have to be transmitted in Classic CAN format
- End-of-line programming in case not all nodes are CAN FD capable. Non CAN FD nodes are held in Silent mode until programming has completed. Then all nodes switch back to Classic CAN communication

In the CAN FD format, the coding of the DLC differs from the standard CAN format. The DLC codes 0 to 8 have the same coding as in standard CAN, the codes 9 to 15, which in standard CAN all code a data field of 8 bytes, are coded according to *Table 950*:

Table 950. Coding of DLC in CAN FD

DLC	9	10	11	12	13	14	15
Number of Data Bytes	12	16	20	24	32	48	64

In CAN FD frames, the bit timing will be switched inside the frame, after the BRS (Bit Rate Switch) bit, if this bit is recessive. Before the BRS bit, in the CAN FD arbitration phase, the nominal CAN bit timing is used as defined by the Nominal Bit Timing & Prescaler Register NBTP. In the following CAN FD data phase, the data phase bit timing is used as defined by the Data Bit Timing & Prescaler Register DBTP. The bit timing is switched back from the data phase timing at the CRC delimiter or when an error is detected, whichever occurs first.

The maximum configurable bit rate in the CAN FD data phase depends on the CAN clock frequency. For example, with a CAN clock frequency of 20MHz and the shortest configurable bit time of 4 tq, the bit rate in the data phase is 5 Mbit/s.

In both data frame formats, CAN FD long and CAN FD fast, the value of the bit ESI (Error Status Indicator) is determined by the transmitter's error state at the start of the transmission. If the transmitter is error passive, ESI is transmitted recessive, else it is transmitted dominant.

52.4.7.4 Transceiver Delay Compensation

During the data phase of a CAN FD transmission only one node is transmitting, all others are receivers. The length of the bus line has no impact. When transmitting via M_TTCAN Tx output pin the TTCAN receives the transmitted data from its local CAN transceiver via M_TTCAN Rx input pin. The received data is delayed by the transmitter delay. In case this delay is greater than TSEG1 (time segment before sample point), a bit error is detected. In order to enable a data phase bit time that is even shorter than the transmitter delay, the delay compensation is introduced. Without transmitter delay compensation, the bit rate in the data phase of a CAN FD frame is limited by the transmitter delay. The CAN FD protocol unit has implemented a delay compensation mechanism to compensate the transmitter delay, thereby enabling transmission with higher bit rates during the CAN FD data phase independent of the delay of a specific CAN transceiver.

The M_TTCAN's protocol unit has implemented a delay compensation mechanism to compensate the transmitter delay, thereby enabling transmission with higher bit rates during the CAN FD data phase independent of the delay of a specific CAN transceiver.

To check for bit errors during the data phase of transmitting nodes, the delayed transmit data is compared against the received data at the Secondary Sample Point SSP. If a bit error is detected, the transmitter will react on this bit error at the next following regular sample point. During arbitration phase the delay compensation is always disabled.

The transmitter delay compensation enables configurations where the data bit time is shorter than the transmitter delay, it is described in detail in the new ISO11898-1. It is enabled by setting bit DBTP.TDC.

The received bit is compared against the transmitted bit at the SSP. The SSP position is defined as the sum of the measured delay from the M_TTCAN Tx output through the transceiver to the M_TTCAN Rx input plus the transmitter delay compensation offset as configured by TDCR.TDCO. The transmitter delay compensation offset is used to adjust the position of the SSP inside the received bit (such as half of the bit time in the data phase). The position of the secondary sample point is rounded down to the next integer number of mtq.

PSR.TDCV shows the actual transmitter delay compensation value. PSR.TDCV is cleared when CCCR.INIT is set and is updated at each transmission of an FD frame while DBTP.TDC is set.

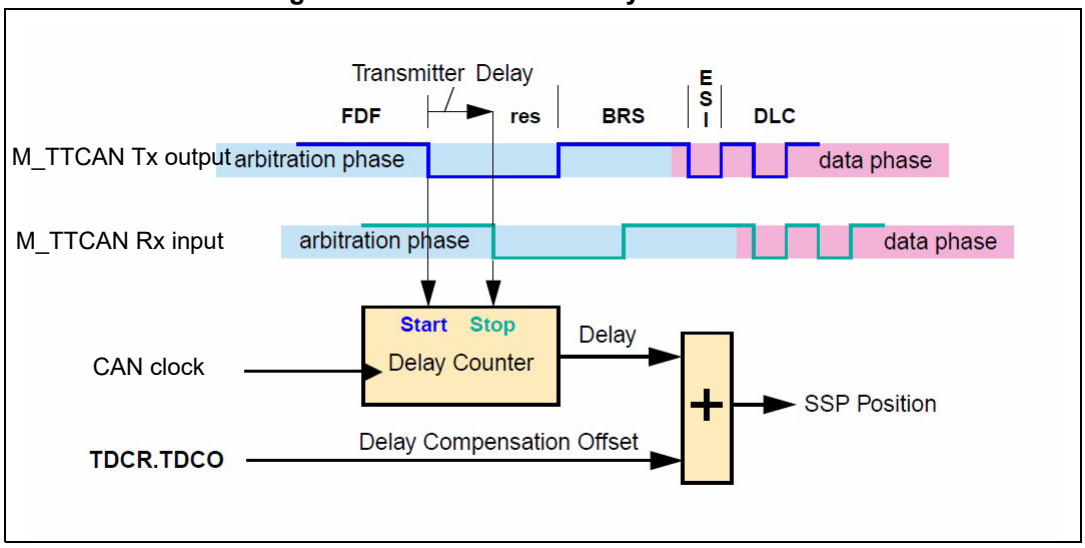
The following boundary conditions have to be considered for the transmitter delay compensation implemented in the M_TTCAN:

- The sum of the measured delay from the M_TTCAN Tx to the M_TTCAN Rx and the configured transmitter delay compensation offset TDCR.TDCO has to be less than 6 bit times in the data phase.
- The sum of the measured delay from the M_TTCAN Tx output to the M_TTCAN Rx input and the configured transmitter delay compensation offset TDCR.TDCO has to be less or equal 127 mtq. In case this sum exceeds 127 mtq, the maximum value of 127 mtq is used for transmitter delay compensation.
- The data phase ends at the sample point of the CRC delimiter, that stops checking of receive bits at the SSPs

If transmitter delay compensation is enabled by programming DBTP.TDC = '1', the measurement is started within each transmitted CAN FD frame at the falling edge of bit FDF to bit res. The measurement is stopped when this edge is seen at the M_TTCAN Rx input of the transmitter. The resolution of this measurement is one mtq.

Figure 852 describes how the Transmitter Delay Compensation is measured.

Figure 852. Transceiver Delay measurement



To avoid that a dominant glitch inside the received FDF bit ends the delay compensation measurement before the falling edge of the received res bit, resulting in a to early SSP position, the use of a transmitter delay compensation filter window can be enabled by programming TDCR.TDCF. This defines a minimum value for the SSP position. Dominant edges on M_TTCAN Rx input, that would result in an earlier SSP position are ignored for transmitter delay measurement. The measurement is stopped when the SSP position is at least TDCR.TDCF AND M_TTCAN Rx input is low.

52.4.7.5 Restricted Operation Mode

In Restricted Operation Mode the node is able to receive data and remote frames and to give acknowledge to valid frames, but it does not send data frames, remote frames, active error frames, or overload frames. In case of an error condition or overload condition, it does not send dominant bits, instead it waits for the occurrence of bus idle condition to resynchronize itself to the CAN communication. The error counters (ECR.REC, ECR.TEC) are frozen while Error Logging (ECR.CEL) is active. The Host can set the M_TTCAN into Restricted Operation mode by setting bit CCCR.ASM. The bit can only be set by the Host when both CCCR.CCE and CCCR.INIT are set to ‘1’. The bit can be reset by the Host at any time.

Restricted Operation Mode is automatically entered when the Tx Handler was not able to read data from the Message RAM in time. To leave Restricted Operation Mode, the Host CPU has to reset CCCR.ASM.

The Restricted Operation Mode can be used in applications that adapt themselves to different CAN bit rates. In this case the application tests different bit rates and leaves the Restricted Operation Mode after it has received a valid frame.

When the M_TTCAN is configured for Asynchronous Serial Communication, the Host has to set CCCR.ASM during initialization to start with an ASC window. The bit is reset at the end of the first ASC window after the M_TTCAN has finished initializing. During time-triggered operation CCCR.ASM is set by the M_TTCAN at the beginning of an ASC window (trigger memory element with T0.ASC = “10”, “11”). It is reset by each trigger memory element with T0.ASC = “00”.

If the M_TTCAN is connected to a Clock Calibration on CAN Unit, CCCR.ASM is controlled by calibration input. In case calibration input switches to ‘0’, bit CCCR.ASM is set. When m_ttcan_cok switches back to ‘1’, bit CCCR.ASM returns to the previously written value. When there is no Clock Calibration on CAN unit connected calibration input is hardwired to ‘1’.

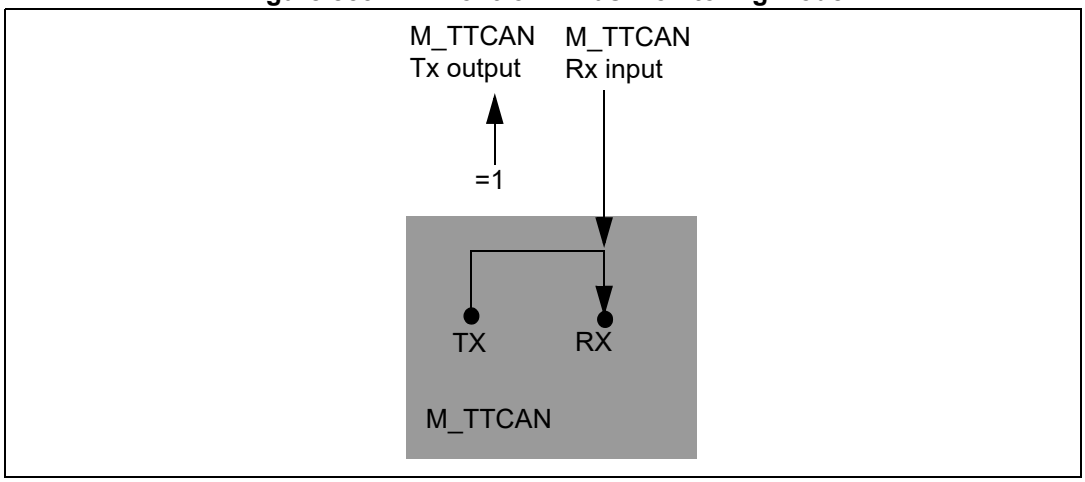
Note: The Restricted Operation Mode must not be combined with the Loop Back Mode (internal or external).

52.4.7.6 Bus Monitoring mode

The M_TTCAN is set in Bus Monitoring Mode by programming CCCR.MON to one or when error level S3 (TTOST[EL] = ‘11’) is entered. In Bus Monitoring Mode (refer to ISO11898-1, 10.12 Bus monitoring), the M_TTCAN is able to receive valid data frames and valid remote frames, but cannot start a transmission. In this mode, it sends only recessive bits on the CAN bus, if the M_TTCAN is required to send a dominant bit (ACK bit, overload flag, active error flag), the bit is rerouted internally so that the M_TTCAN monitors this dominant bit, although the CAN bus may remain in recessive state. In Bus Monitoring Mode register TXBRP is held in reset state.

The Bus Monitoring Mode can be used to analyze the traffic on a CAN bus without affecting it by the transmission of dominant bits. *Figure 853* shows the connection of signals M_TTCAN Tx output and M_TTCAN Rx output to the M_TTCAN in Bus Monitoring Mode.

Figure 853. Pin Control in Bus Monitoring Mode



52.4.7.7 Disabled automatic retransmission

According to the CAN Specification (refer to ISO11898-1, 6.3.3 Recovery Management), the M_TTCAN provides means for automatic retransmission of frames that have lost arbitration or that have been disturbed by errors during transmission. By default automatic retransmission is enabled. To support time-triggered communication as described in ISO 11898-1, chapter 9.2, the automatic retransmission may be disabled via CCCR[DAR].

52.4.7.8 Frame transmission in DAR mode

In DAR mode all transmissions are automatically cancelled after they started on the CAN bus. A Tx Buffer’s Tx Request Pending bit TXBRP.TRPx is reset after successful

transmission, when a transmission has not yet been started at the point of cancellation, has been aborted due to lost arbitration, or when an error occurred during frame transmission.

- Successful transmission:
  - Corresponding Tx Buffer Transmission Occurred bit TXBTO[TOx] set
  - Corresponding Tx Buffer Cancellation Finished bit TXBCF[CFx] not set
- Successful transmission in spite of cancellation:
  - Corresponding Tx Buffer Transmission Occurred bit TXBTO[TOx] set
  - Corresponding Tx Buffer Cancellation Finished bit TXBCF[CFx] set
- Arbitration lost or frame transmission disturbed:
  - Corresponding Tx Buffer Transmission Occurred bit TXBTO[TOx] not set
  - Corresponding Tx Buffer Cancellation Finished bit TXBCF[CFx] set

In case of a successful frame transmission, and if storage of Tx events is enabled, a Tx Event FIFO element is written with Event Type ET = ‘10’ (transmission in spite of cancellation).

52.4.7.9 Power down (Sleep mode)

The M_TTCAN can be set into power down mode controlled by clock stop request input signal or via CC Control Register CCCR[CSR]. As long as the clock stop request signal is active, bit CCCR[CSR] is read as one.

When all pending transmission requests have completed, the M_TTCAN waits until bus idle state is detected. Then the M_TTCAN sets then CCCR[INIT] to one to prevent any further CAN transfers. Now the M_TTCAN acknowledges that it is ready for power down by setting clock stop acknowledge output signal to one and CCCR[CSA] to one. In this state, before the clocks are switched off, further register accesses can be made. A write access to CCCR[INIT] will have no effect. Now the module clock inputs M_TTCAN Host clock and CAN clock may be switched off.

To leave power down mode, the application has to turn on the module clocks before resetting clock stop request signal respectively CC Control Register flag CCCR.CSR. The M_TTCAN will acknowledge this by resetting output clock stop acknowledge signal and resetting CCCR[CSA]. Afterwards, the application can restart CAN communication by resetting bit CCCR[INIT].

52.4.7.10 Test modes

To enable write access to register TEST (refer to *Section 52.4.5.2.4: Test Register (TEST)*), bit CCCR.TEST has to be set to one. This allows the configuration of the test modes and test functions.

Four output functions are available for the M_TTCAN Tx output pin by programming TEST.TX. Additionally to its default function – the serial data output – it can drive the CAN Sample Point signal to monitor the M_TTCAN’s bit timing and it can drive constant dominant or recessive values. The actual value at M_TTCAN Rx input pin can be read from TEST.RX. Both functions can be used to check the CAN bus’ physical layer.

Due to the synchronization mechanism between CAN clock and Host clock domain, there may be a delay of several Host clock periods between writing to TEST.TX until the new configuration is visible at M_TTCAN Tx output pin. This applies also when reading M_TTCAN Rx input pin via TEST.RX.

Note: Test modes should be used for production tests or self test only. The software control for M_TTCAN Tx output pin interferes with all CAN protocol functions. It is not recommended to use test modes for application.

52.4.7.10.1 Application Watchdog

The application watchdog is served by reading register TTOST. When the application watchdog is not served in time, bit TTOST.AWE is set, all M_TTCAN communication is stopped, and the M_TTCAN is set into Bus Monitoring Mode.

The TT Application Watchdog can be disabled by programming the Application Watchdog Limit TTOCF[AWL] to 0x00. The TT Application Watchdog should not be disabled in a M_TTCAN application program.

52.4.7.10.2 External Loop Back mode

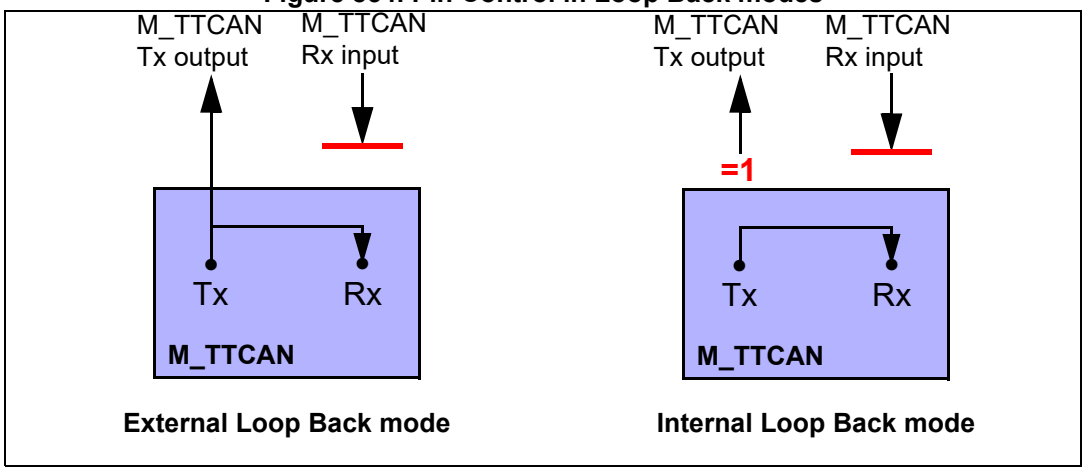
The M_TTCAN can be set in External Loop Back mode by programming TEST.LBCK to one. In Loop Back mode, the M_TTCAN treats its own transmitted messages as received messages and stores them (if they pass acceptance filtering) into Rx FIFOs. *Figure 854* shows the connection of transmit and receive signals to the M_TTCAN in External Loop Back mode.

This mode is provided for hardware self-test. To be independent from external stimulation, the M_TTCAN ignores acknowledge errors (recessive bit sampled in the acknowledge slot of a data/ remote frame) in Loop Back mode. In this mode the M_TTCAN performs an internal feedback from its Tx output to its Rx input. The actual value of the M_TTCAN Rx input pin is disregarded by the M_TTCAN. The transmitted messages can be monitored at the M_TTCAN Tx output pin.

52.4.7.10.3 Internal Loop Back mode

Internal Loop Back mode is entered by programming bits TEST.LBCK and CCCR.MON to one. This mode can be used for a “Hot Selftest”, meaning the M_TTCAN can be tested without affecting a running CAN system connected to the M_TTCAN Tx output and Rx input pins. In this mode M_TTCAN Rx input pin is disconnected from the M_TTCAN and the M_TTCAN Tx output pin is held recessive. *Figure 854* shows the connection of the M_TTCAN Tx output and Rx input to the M_TTCAN in case of Internal Loop Back Mode.

Figure 854. Pin Control in Loop Back modes



52.4.7.11 Timestamp generation

For timestamp generation the M_TTCAN supplies a 16-bit wrap-around counter. A prescaler TSCC.TCP can be configured to clock the counter in multiples of CAN bit times (1…16). The counter is readable via TSCV[TCV]. A write access to register TSCV resets the counter to zero. When the timestamp counter wraps around interrupt flag IR[TSW] is set.

On start of frame reception / transmission the counter value is captured and stored into the timestamp section of a Rx Buffer/ Rx FIFO (RXTS[15:0]) or Tx Event FIFO (TXTS[15:0]) element.

By programming bit TSCC.TSS, a 16-bit timestamp can be used.

52.4.7.12 Timeout counter

To signal timeout conditions for Rx FIFO 0, Rx FIFO 1, and the Tx Event FIFO the M_TTCAN supplies a 16-bit Timeout Counter. It operates as down-counter and uses the same prescaler controlled by TSCC[TCP] as the Timestamp Counter. The Timeout Counter is configured via register TOCC. The actual counter value can be read from TOCV[TOC].

The Timeout Counter can only be started while CCCR[INIT] = ‘0’. It is stopped when CCCR[INIT] = ‘1’, for example when the M_TTCAN enters Bus_Off state.

The operation mode is selected by TOCC[TOS]. When operating in Continuous mode, the counter starts when CCCR[INIT] is reset. A write to TOCV presets the counter to the value configured by TOCC[TOP] and continues down-counting.

When the Timeout Counter is controlled by one of the FIFOsFIFOs, an empty FIFO presets the counter to the value configured by TOCC[TOP]. Down-counting is started when the first FIFO element is stored. Writing to TOCV has no effect.

When the counter reaches zero, interrupt flag IR[TOO] is set. In Continuous mode, the counter is immediately restarted at TOCC[TOP].

Note: The clock signal for the Timeout Counter is derived from the CAN core's sample point signal. Therefore the point in time where the Timeout Counter is decremented may vary due to the synchronization / re-synchronization mechanism of the CAN core. If the baud rate switch feature in CAN FD is used, the timeout counter is clocked differently in arbitration and data field.

52.4.7.13 Rx handling

The Rx Handler controls the acceptance filtering, the transfer of received messages to Rx Buffers or to one of the two Rx FIFOs, as well as the Rx FIFO's Put and Get Indexes.

52.4.7.13.1 Acceptance filtering

The M_TTCAN offers the possibility to configure two sets of acceptance filters, one for standard identifiers and one for extended identifiers. These filters can be assigned to Rx buffer, Rx FIFO 0 or Rx FIFO 1. For acceptance filtering each list of filters is executed from element #0 until the first matching element. Acceptance filtering stops at the first matching element. The following filter elements are not evaluated for this message.

The main features are:

- Each filter element can be configured as
  - range filter (from - to)
  - filter for one or two dedicated IDs
  - classic bit mask filter
- Each filter element is configurable for acceptance or rejection filtering
- Each filter element can be enabled / disabled individually
- Filters are checked sequentially, execution stops with the first matching filter element

Related configuration registers are:

- Global Filter Configuration (GFC)
- Standard ID Filter Configuration (SIDFC)
- Extended ID Filter Configuration (XIDFC)
- Extended ID AND Mask (XIDAM)

Depending on the configuration of the filter element (SFEC/EFEC) a match triggers one of the following actions:

- Store received frame in FIFO 0 or FIFO 1
- Store received frame in Rx Buffer
- Reject received frame
- Set High Priority Message interrupt flag IR[HPM]
- Set High Priority Message interrupt flag IR[HPM] and store received frame in FIFO 0 or FIFO 1

Note: When an accepted message is written to one of the two Rx FIFOs, or into an Rx Buffer, the unmodified received identifier is stored independent of the filter(s) used. The result of the acceptance filter process strongly depends on the sequence of configured filter elements.

52.4.7.13.2 Range filter

The filter matches for all received frames with Message IDs in the range defined by SFID1/SFID2 resp. EFID1/EFID2.

There are two possibilities when range filtering is used together with extended frames:

- EFT = ‘00’: The Message ID of received frames is ANDed with the Extended ID AND Mask (XIDAM) before the range filter is applied
- EFT = ‘11’: The Extended ID AND Mask (XIDAM) is not used for range filtering

52.4.7.13.3 Filter for dedicated IDs

A filter element can be configured to filter for one or two specific Message IDs. To filter for one specific Message ID, the filter element has to be configured with SFID1=SFID2 resp. EFID1=EFID2.

52.4.7.13.4 Classic bit mask filter

Classic bit mask filtering is intended to filter groups of Message IDs by masking single bits of a received Message ID. With classic bit mask filtering SFID1 /EFID1 is used as Message ID filter, while SFID2/EFID2 is used as filter mask.

A zero bit at the filter mask will mask out the corresponding bit position of the configured ID filter, for example the value of the received Message ID at that bit position is not relevant for acceptance filtering. Only those bits of the received Message ID where the corresponding mask bits are one are relevant for acceptance filtering.

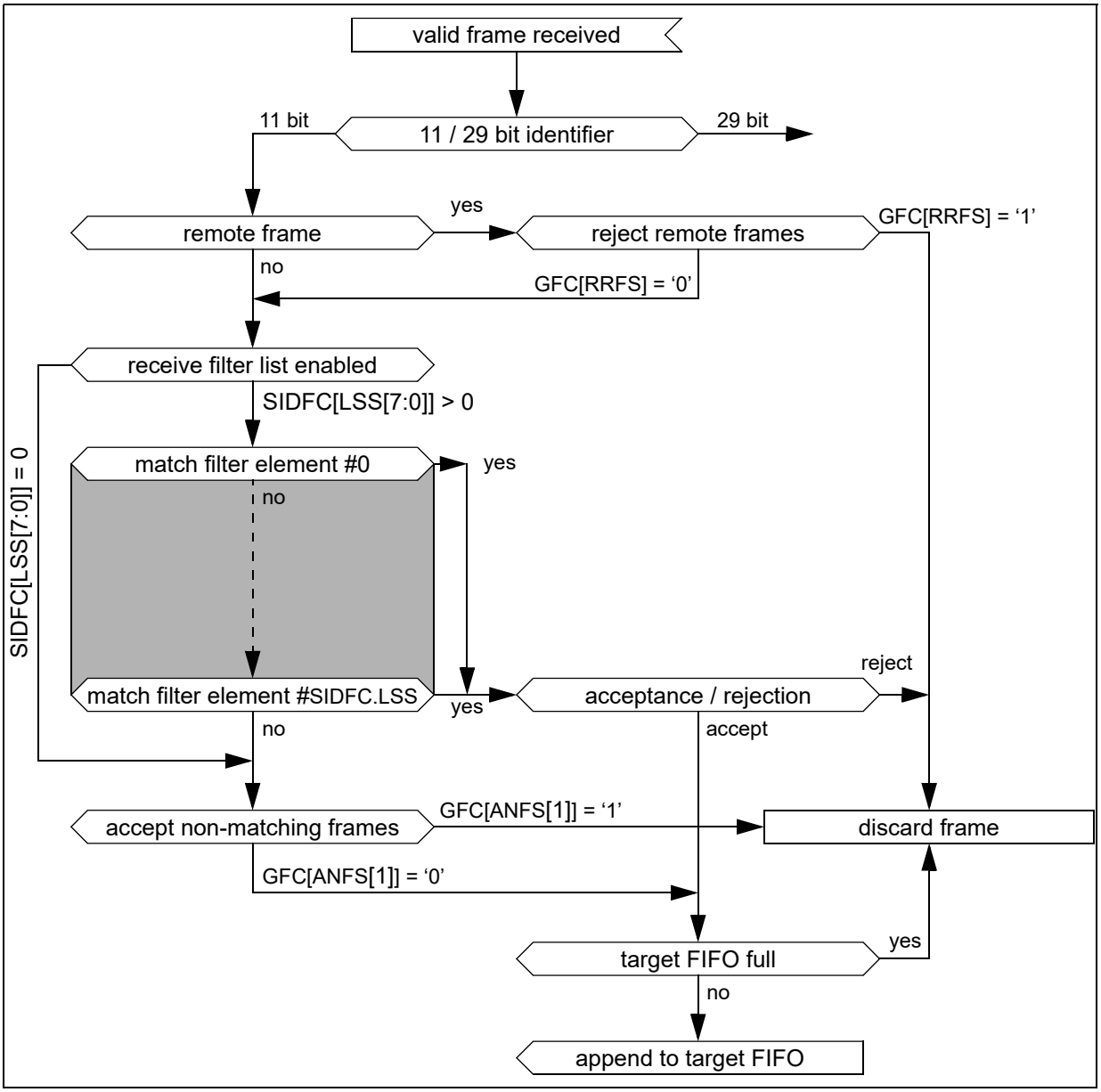
In case all mask bits are one, a match occurs only when the received Message ID and the Message ID filter are identical. If all mask bits are zero, all Message IDs match.

52.4.7.13.5 Standard message ID filtering

Figure 855 below shows the ow for standard Message ID (11-bit Identifier) filtering. The Standard Message ID Filter element is described in *Section 52.4.6.4: Standard Message ID Filter element*.

Controlled by the Global Filter Configuration (GFC) and the Standard ID Filter Configuration (SIDFC) Message ID, Remote Transmission Request bit (RTR), and the Identifier Extension bit (IDE) of received frames are compared against the list of configured filter elements.

Figure 855. Standard Message ID Filter Path



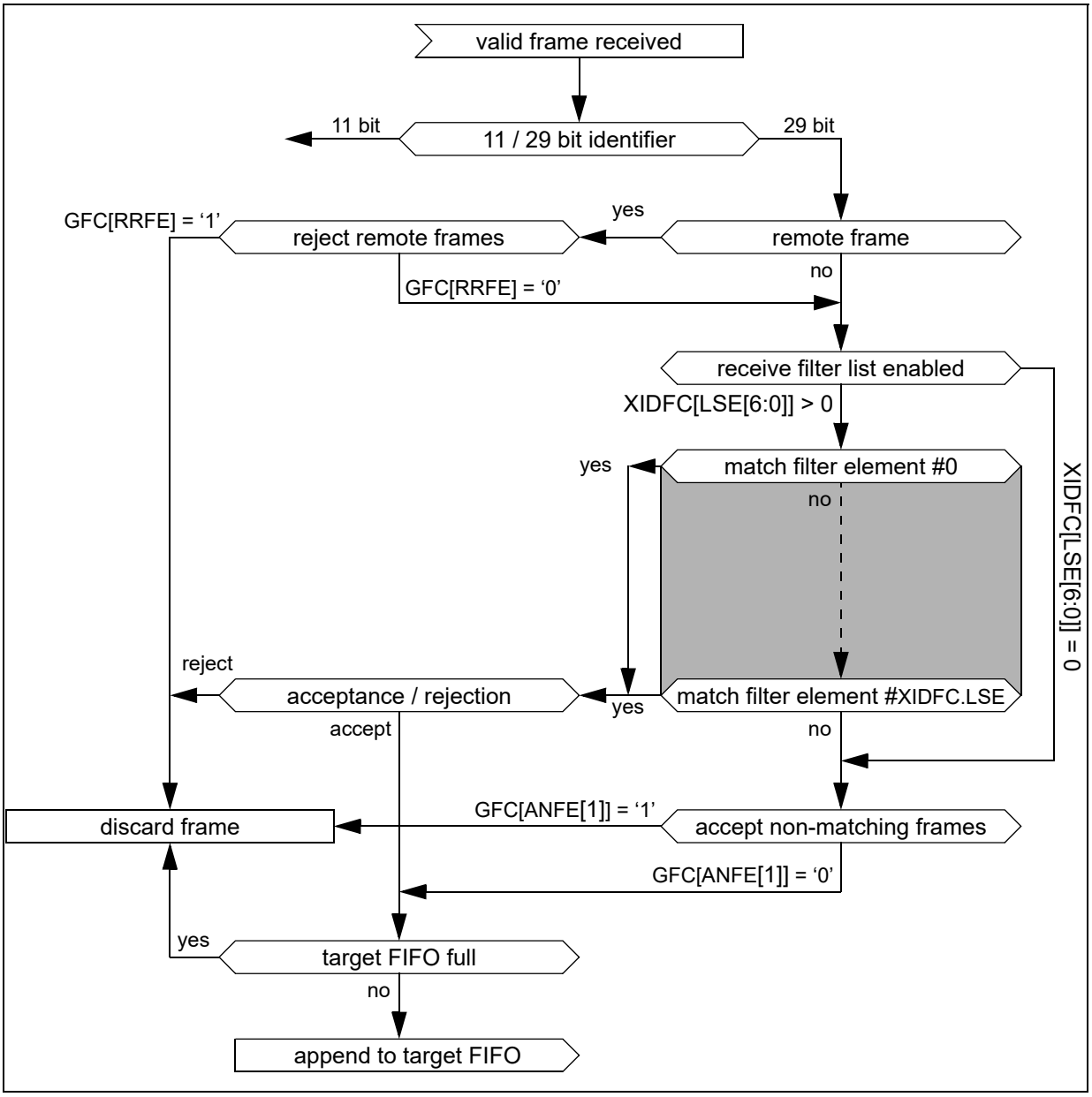
52.4.7.13.6 Extended message ID filtering

Figure 856 below shows the flow for extended Message ID (29-bit Identifier) filtering. The Extended Message ID Filter element is described in *Section 52.4.6.5: Extended Message ID Filter element*.

Controlled by the Global Filter Configuration GFC and the Extended ID Filter Configuration XIDFC Message ID, Remote Transmission Request bit (RTR), and the Identifier Extension bit (IDE) of received frames are compared against the list of configured filter elements.

The Extended ID AND Mask (XIDAM) is ANDed with the received identifier before the filter list is executed.

Figure 856. Extended Message ID Filter Path



52.4.7.14 Rx FIFOs

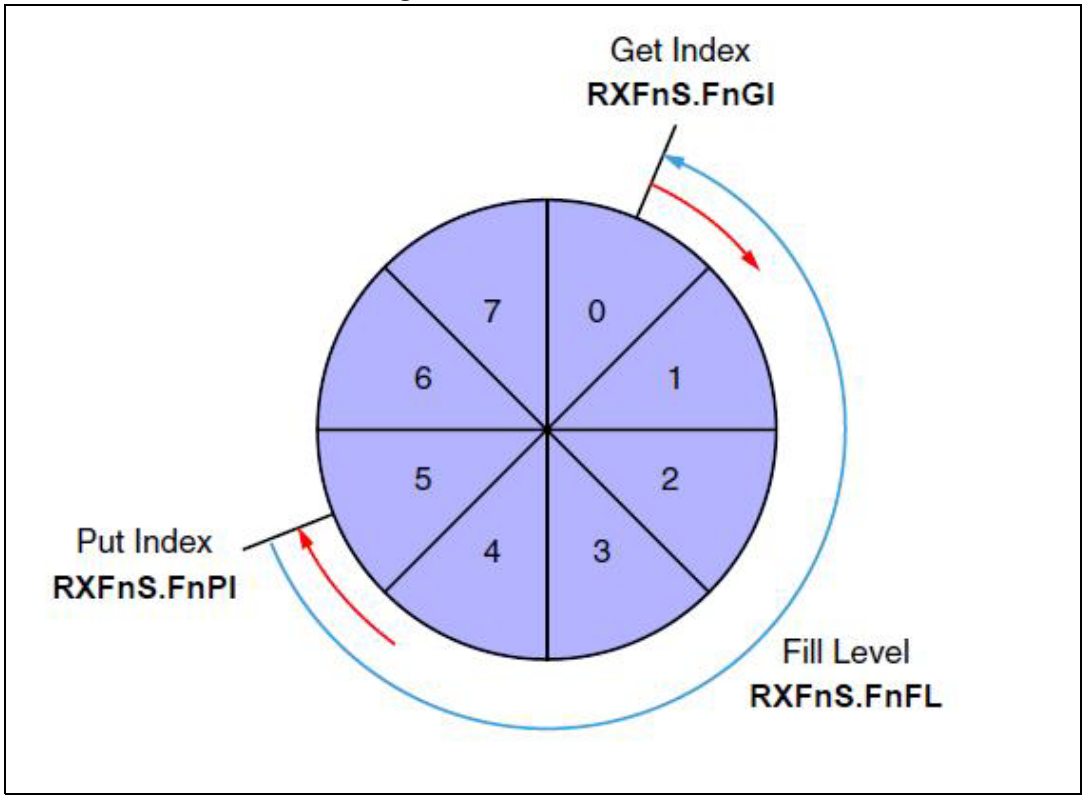
Rx FIFO 0 and Rx FIFO 1 can be configured to hold up to 64 elements each. Configuration of the two Rx FIFOs is done via registers RXF0C and RXF1C.

Received messages that passed acceptance filtering are transferred to the Rx FIFO as configured by the matching filter element. For a description of the filter mechanisms available for Rx FIFO 0 and Rx FIFO 1, refer to *Section 52.4.7.13.1: Acceptance filtering*. The Rx buffer and FIFO element is described in *Section 52.4.6.1: Rx Buffer and FIFO element*.

To avoid an Rx FIFO over ow, the Rx FIFO watermark can be used. When the Rx FIFO fill level reaches the Rx FIFO watermark configured by RXFnC.FnWM, interrupt flag IR.RFnW

is set. When the Rx FIFO Put Index reaches the Rx FIFO Get Index an Rx FIFO Full condition is signalled by RXFnS.FnF. In addition interrupt flag IR.RFnF is set.

Figure 857. Rx FIFO Status



When reading from an Rx FIFO, Rx FIFO Get Index RXFnS.FnGI × FIFO Element Size has to be added to the corresponding Rx FIFO start address RXFnC.FnSA.

Table 951. Rx Buffer / FIFO Element Size

RXESC.RBDS[2:0] RXESC.FnDS[2:0]	Data Field [bytes]	FIFO Element Size [RAM words]
000	8	4
001	12	5
010	16	6
011	20	7
100	24	8
101	32	10
110	48	14
111	64	18

52.4.7.14.1 Rx FIFO Blocking Mode

The Rx FIFO blocking mode is configured by RXFnC.FnOM = ‘0’. This is the default operation mode for the Rx FIFOs.

When an Rx FIFO full condition is reached (RXFnS.FnPI = RXFnS.FnGI), no further messages are written to the corresponding Rx FIFO until at least one message has been read out and the Rx FIFO Get Index has been incremented. An Rx FIFO full condition is signalled by RXFnS.FnF = ‘1’. In addition interrupt flag IR.RFnF is set.

In case a message is received while the corresponding Rx FIFO is full, this message is discarded and the message lost condition is signalled by RXFnS.RFnL = ‘1’. In addition interrupt flag IR.RFnL is set.

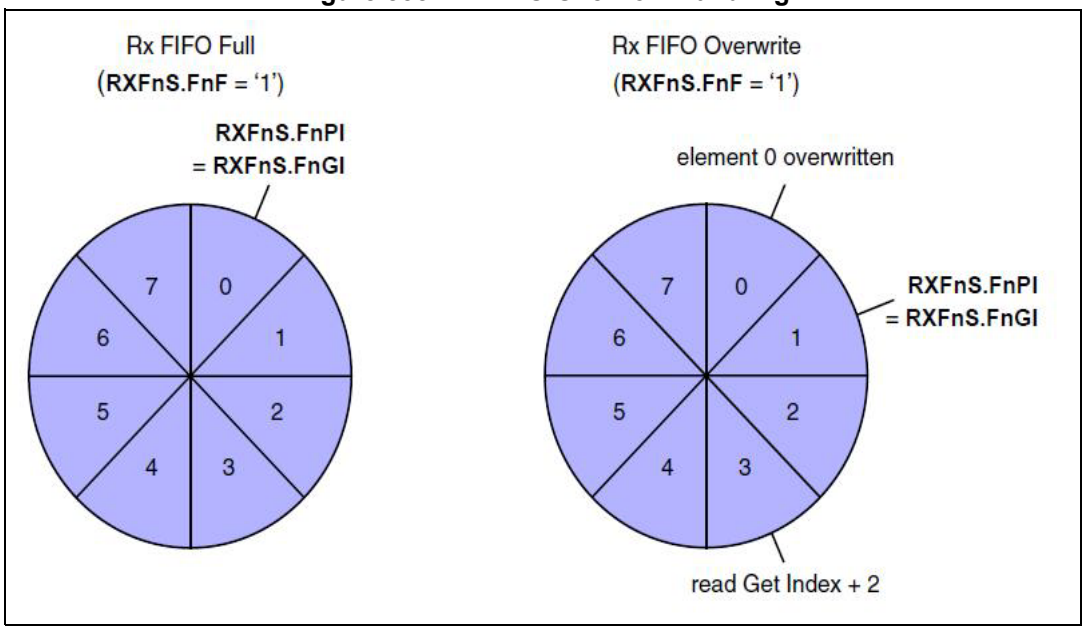
52.4.7.14.2 Rx FIFO Overwrite Mode

The Rx FIFO overwrite mode is configured by RXFnC.FnOM = ‘1’.

When an Rx FIFO full condition (RXFnS.FnPI = RXFnS.FnGI) is signalled by RXFnS.FnF = ‘1’, the next message accepted for the FIFO will overwrite the oldest FIFO message. Put and get index are both incremented by one.

When an Rx FIFO is operated in overwrite mode and an Rx FIFO full condition is signalled, reading of the Rx FIFO elements should start at least at get index + 1. The reason for that is, that it might happen, that a received message is written to the Message RAM (put index) while the CPU is reading from the Message RAM (get index). In this case inconsistent data may be read from the respective Rx FIFO element. Adding an offset to the get index when reading from the Rx FIFO avoids this problem. The offset depends on how fast the CPU accesses the Rx FIFO. Figure 9 shows an offset of two with respect to the get index when reading the Rx FIFO. In this case the two messages stored in element 1 and 2 are lost.

Figure 858. Rx FIFO Overflow Handling



After reading from the Rx FIFO, the number of the last element read has to be written to the Rx FIFO Acknowledge Index RXFnA.FnA. This increments the get index to that element number. In case the put index has not been incremented to this Rx FIFO element, the Rx FIFO full condition is reset (RXFnS.FnF = ‘0’).

52.4.7.15 Dedicated Rx Buffers

The M_TTCAN supports up to 64 dedicated Rx Buffers. The start address of the dedicated Rx Buffer section is configured via RXBC.RBSA.

For each Rx Buffer a Standard or Extended Message ID Filter Element with SFEC / EFEC = '111' and SFID2 / EFID2[10:9] = '00' has to be configured (refer to *Section 52.4.6.4: Standard Message ID Filter element* and *Section 52.4.6.5: Extended Message ID Filter element*).

After a received message has been accepted by a filter element, the message is stored into the Rx Buffer in the Message RAM referenced by the filter element. The format is the same as for an Rx FIFO element. In addition the flag IR.DRX (Message stored in Dedicated Rx Buffer) in the interrupt register is set.

Table 952. Example Filter Configuration for Rx Buffers

Filter Element	SFID1[10:0] EFID1[28:0]	SFID2[10:9] EFID2[10:9]	SFID2[5:0] EFID2[5:0]
0	ID message 1	00	00 0000
1	ID message 2	00	00 0001
2	ID message 3	00	00 0010

After the last word of a matching received message has been written to the Message RAM, the respective New Data flag in register NDAT1,2 is set. As long as the New Data flag is set, the respective Rx Buffer is locked against updates from received matching frames. The New Data flags have to be reset by the Host by writing a '1' to the respective bit position.

While an Rx Buffer's New Data flag is set, a Message ID Filter Element referencing this specific Rx Buffer will not match, causing the acceptance filtering to continue. The following Message ID Filter Elements may cause the received message to be stored into another Rx Buffer, or into an Rx FIFO, or the message may be rejected, depending on filter configuration.

52.4.7.15.1 Rx Buffer Handling

- Reset interrupt flag IR.DRX
- Read New Data registers
- Read messages from Message RAM
- Reset New Data flags of processed messages

52.4.7.16 Debug on CAN Support

Debug messages are stored into Rx Buffers. For debug handling three consecutive Rx buffers (for example, #61, #62, #63) have to be used for storage of debug messages A, B, and C. The format is the same as for an Rx Buffer or an Rx FIFO element (refer to *Section 52.4.6.1*).

Advantage: Fixed start address for the DMA transfers (relative to RXBC.RBSA), no additional configuration required.

For filtering of debug messages Standard / Extended Filter Elements with SFEC / EFEC = ‘111’ have to be set up. Messages matching these filter elements are stored into the Rx Buffers addressed by SFID2 / EFID2[5:0].

After message C has been stored, the DMA request output is activated and the three messages can be read from the Message RAM under DMA control. The RAM words holding the debug messages will not be changed by the M_TTCAN while DMA request output is activated. The behaviour is similar to that of an Rx Buffers with its New Data flag set.

After the DMA has completed the DMA unit sets DMA acknowledge input . This resets DMA request output. Now the M_TTCAN is prepared to receive the next set of debug messages.

Note: To use full ‘Debug on CAN Support’ feature on a M_(TT)CAN instance a DMA channel is required. Please refer to device DMA map to see which M_(TT)CAN instances has got an assigned DMA channel.

52.4.7.16.1 Filtering for Debug Messages

Filtering for debug messages is done by configuring one Standard / Extended Message ID Filter Element for each of the three debug messages. To enable a filter element to filter for debug messages SFEC / EFEC has to be programmed to ‘111’. In this case fields SFID1 / SFID2 and EFID1 / EFID2 have a different meaning (refer to *Section 52.4.6.4: Standard Message ID Filter element* and *Section 52.4.6.5: Extended Message ID Filter element*). While SFID2 / EFID2[10:9] controls the debug message handling state machine, SFID2 / EFID2[5:0] controls the location for storage of a received debug message.

When a debug message is stored, neither the respective New Data flag nor IR.DRX are set. The reception of debug messages can be monitored via RXF1S.DMS.

Table 953. Example Filter Configuration for Debug Messages

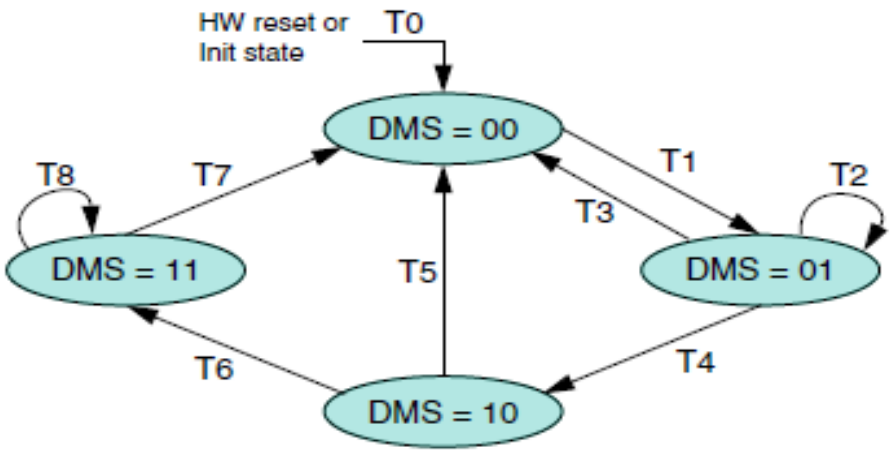
Filter Element	SFID1[10:0] EFID1[28:0]	SFID2[10:9] EFID2[10:9]	SFID2[5:0] EFID2[5:0]
0	ID debug message A	01	11 1101
1	ID debug message B	10	11 1110
2	ID debug message C	11	11 1111

52.4.7.16.2 Debug Message Handling

The debug message handling state machine assures that debug messages are stored to three consecutive Rx Buffers in correct order. In case of missing messages the process is restarted. The DMA request is activated only when all three debug messages A, B, C have been received in correct order.

The status of the debug message handling state machine is signalled via RXF1S.DMS.

Figure 859. Debug Message Handling State Machine



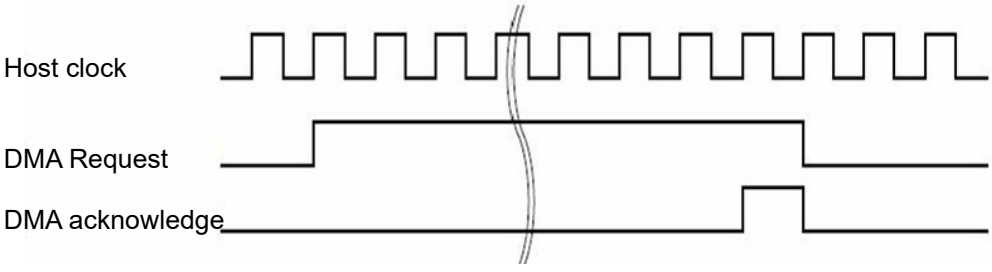
T0: reset m_cam_dma_req output, enable reception of debug messages A, B, and C
T1: reception of debug message A
T2: reception of debug message A
T3: reception of debug message C
T4: reception of debug message B
T5: reception of debug messages A, B
T6: reception of debug message C
T7: DMA transfer completed
T8: reception of debug message A,B,C (message rejected)

52.4.7.17 Interface to DMA Controller

When all three debug messages A, B, C have been received in the correct order, M_TTCAN DMA request output signal is activated to trigger a DMA transfer. The RAM words holding debug messages A, B, C will not be changed by the M_TTCAN while M_TTCAN DMA request output is active.

After the transfer of the received messages has completed the DMA unit activates DMA acknowledge input signal. This resets M_TTCAN DMA request output. The debug message handling state machine enters idle state (DMS = “00”) and waits for reception of the next debug messages.

Figure 860. Timing of DMA Interface Signals



Note: If the DMA unit activates TT_CAN DMA acknowledge input signal before the DMA transfer has completed, the Rx Buffer elements holding debug messages A, B, C are unlocked and may be overwritten by received debug messages.

52.4.7.18 Tx handling

The Tx Handler handles transmission requests for the dedicated Tx Buffers, the Tx FIFO, and the Tx Queue. It controls the transfer of transmit messages to the CAN core, the Put and Get Indexes, and the Tx Event FIFO. Up to 32 Tx Buffers can be set up for message transmission. The Tx Buffer element is described in the *Note:: Depending on the configuration of the element size (RXESC), between two and sixteen 32-bit words (Rn = 3 to 17) are used for storage of a CAN message's data field. on page 1472*

Note: AUTOSAR requires at least three Tx Queue Buffers and support of transmit cancellation.

The Tx Handler starts a Tx scan to check for the highest priority pending Tx request (Tx Buffer with lowest Message ID) when the Tx Buffer Request Pending register TXBRP is updated, or when a transmission has been started.

52.4.7.19 Transmit Pause

The transmit pause feature is intended for use in CAN systems where the CAN message identifiers are (permanently) specified to specific values and cannot easily be changed. These message identifiers may have a higher CAN arbitration priority than other defined messages, while in a specific application their relative arbitration priority should be inverse. This may lead to a case where one ECU sends a burst of CAN messages that cause another ECU's CAN messages to be delayed because that other messages have a lower CAN arbitration priority.

If, for example, CAN ECU-1 has the transmit pause feature enabled and is requested by its application software to transmit four messages, it will, after the first successful message transmission, wait for two CAN bit times of bus idle before it is allowed to start the next requested message. If there are other ECUs with pending messages, those messages are started in the idle time, they would not need to arbitrate with the next message of ECU-1. After having received a message, ECU-1 is allowed to start its next transmission as soon as the received message releases the CAN bus.

The transmit pause feature is controlled by bit **CCCR.TXP**. If the bit is set, the M_TTCAN will, each time it has successfully transmitted a message, pause for two CAN bit times before starting the next transmission. This enables other CAN nodes in the network to transmit messages even if their messages have lower prior identifiers. Default is transmit pause disabled (**CCCR.TXP** = '0').

This feature looses up burst transmissions coming from a single node and it protects against "babbling idiot" scenarios where the application program erroneously requests too many transmissions.

52.4.7.20 Dedicated Tx buffers

Dedicated Tx Buffers are intended for message transmission under complete control of the Host CPU. Each Dedicated Tx Buffer is configured with a specific Message ID. In case that multiple Tx Buffers are configured with the same Message ID, the Tx Buffer with the lowest buffer number is transmitted first.

If the data section has been updated, a transmission is requested by an Add Request via TXBAR[ARn]. The requested messages arbitrate internally with messages from an optional

Tx FIFO or Tx Queue and externally with messages on the CAN bus, and are sent out according to their Message ID.

A Dedicated Tx Buffer allocates four 32-bit words in the Message RAM. Therefore the start address of a Dedicated Tx Buffer in the Message RAM is calculated by adding four times the transmit buffer index (0…31) to the Tx Buffer Start Address TXBC[TBSA].

52.4.7.21 Tx FIFO

Tx FIFO operation is configured by programming TXBC[TFQM] to ‘0’. Messages stored in the Tx FIFO are transmitted starting with the message referenced by the Get Index TXFQS[TFGI]. After each transmission the Get Index is incremented cyclically until the Tx FIFO is empty. The Tx FIFO enables transmission of messages with the same Message ID from different Tx Buffers in the order these messages have been written to the Tx FIFO. The M_TTCAN calculates the Tx FIFO Free Level TXFQS[TFFL] as difference between Get and Put Index. It indicates the number of available (free) Tx FIFO elements.

New transmit messages have to be written to the Tx FIFO starting with the Tx Buffer referenced by the Put Index TXFQS[TFQPI]. An Add Request increments the Put Index to the next free Tx FIFO element. When the Put Index reaches the Get Index, Tx FIFO Full (TXFQS[TFQF]= ‘1’) is signaled. In this case no further messages should be written to the Tx FIFO until the next message has been transmitted and the Get Index has been incremented.

When a single message is added to the Tx FIFO, the transmission is requested by writing a ‘1’ to the TXBAR bit related to the Tx Buffer referenced by the Tx FIFO's Put Index.

When multiple (n) messages are added to the Tx FIFO, they are written to n consecutive Tx Buffers starting with the Put Index. The transmissions are then requested via TXBAR. The Put Index is then cyclically incremented by n. The number of requested Tx buffers should not exceed the number of free Tx Buffers as indicated by the Tx FIFO Free Level.

When a transmission request for the Tx Buffer referenced by the Get Index is cancelled, the Get Index is incremented to the next Tx Buffer with pending transmission request and the Tx FIFO Free Level is recalculated. When transmission cancellation is applied to any other Tx Buffer, the Get Index and the FIFO Free Level remain unchanged.

A Tx FIFO element allocates four 32-bit words in the Message RAM. Therefore the start address of the next available (free) Tx FIFO Buffer is calculated by adding four times the Put Index TXFQS[TFQPI] (0 to 31) to the Tx Buffer Start Address TXBC[TBSA].

52.4.7.22 Tx Queue

Tx Queue operation is configured by programming TXBC[TFQM] to ‘1’. Messages stored in the Tx Queue are transmitted starting with the message with the lowest Message ID (highest priority). In case that multiple Queue Buffers are configured with the same Message ID, the Queue Buffer with the lowest buffer number is transmitted first.

New messages have to be written to the Tx Buffer referenced by the Put Index TXFQS[TFQPI]. An Add Request cyclically increments the Put Index to the next free Tx Buffer. In case that the Tx Queue is full (TXFQS[TFQF]= ‘1’), the Put Index is not valid and no further message should be written to the Tx Queue until at least one of the requested messages has been sent out or a pending transmission request has been cancelled.

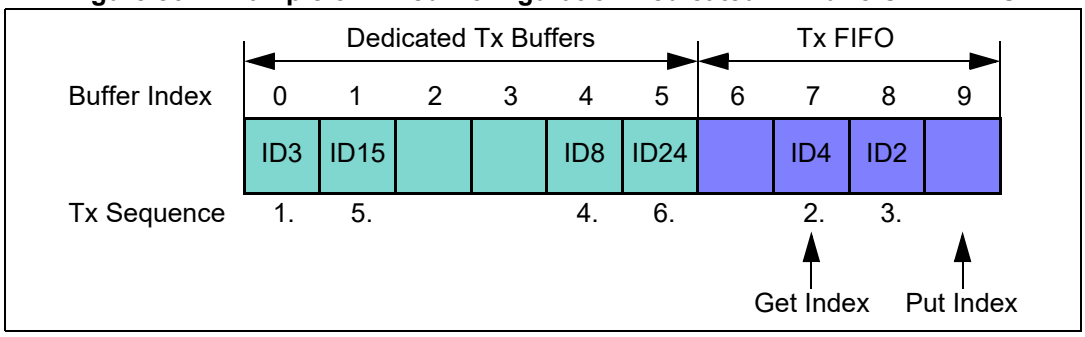
The application may use register TXBRP instead of the Put Index and may place messages to any Tx Buffer without pending transmission request.

A Tx Queue Buffer allocates four 32-bit words in the Message RAM. Therefore the start address of the next available (free) Tx Queue Buffer is calculated by adding four times the Tx Queue Put Index TXFQS[TFQPI] (0 to 31) to the Tx Buffer Start Address TXBC[TBSA].

52.4.7.23 Mixed dedicated Tx Buffers / Tx FIFO

In this case the Tx Buffers section in the Message RAM is subdivided into a set of Dedicated Tx Buffers and a Tx FIFO. The number of Dedicated Tx Buffers is configured by TXBC[NDTB]. The number of Tx Buffers assigned to the Tx FIFO is configured by TXBC[TFQS]. In case, TXBC[TFQS] is programmed to zero, only Dedicated Tx Buffers are used.

Figure 861. Example of mixed Configuration Dedicated Tx Buffers / Tx FIFO



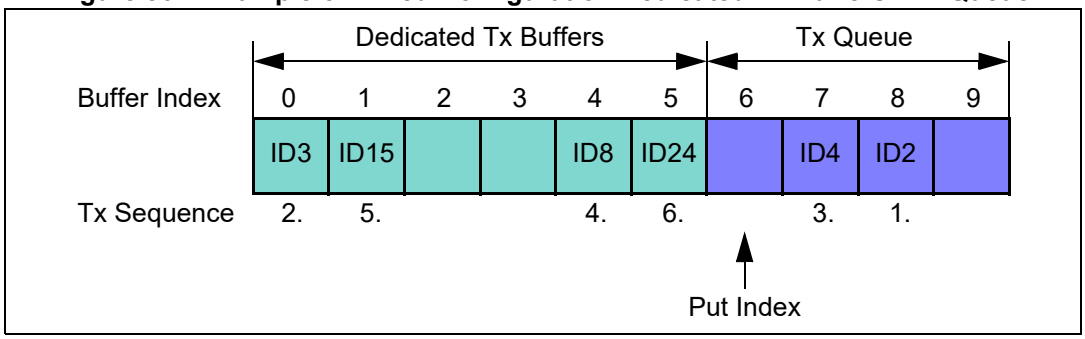
Tx prioritization:

- Scan Dedicated Tx Buffers and oldest pending Tx FIFO Buffer
- Buffer with lowest Message ID gets highest priority and is transmitted next

52.4.7.24 Mixed dedicated Tx Buffers / Tx Queue

In this case the Tx Buffers section in the Message RAM is subdivided into a set of Dedicated Tx Buffers and a Tx Queue. The number of Dedicated Tx Buffers is configured by TXBC[NDTB]. The number of Tx Queue Buffers is configured by TXBC[TFQS]. In case TXBC[TFQS] is programmed to zero, only dedicated Tx Buffers are used.

Figure 862. Example of mixed Configuration Dedicated Tx Buffers / Tx Queue



Tx prioritization:

- Scan all Tx Buffers with activated transmission request
- Tx Buffer with lowest Message ID gets highest priority and is transmitted next

52.4.7.25 Transmit cancellation

The M_TTCAN supports transmit cancellation. This feature is especially intended for gateway applications and AUTOSAR based applications. To cancel a requested transmission from a dedicated Tx Buffer or a Tx Queue Buffer the Host has to write a ‘1’ to the corresponding bit position (=number of Tx Buffer) of register TXBCR. Transmit cancellation is not intended for Tx FIFO operation.

Successful cancellation is signaled by setting the corresponding bit of register TXBCF to ‘1’.

In case a transmit cancellation is requested while a transmission from a Tx Buffer is already ongoing, the corresponding TXBRP bit remains set as long as the transmission is in progress. If the transmission was successful, the corresponding TXBTO and TXBCF bits are set. If the transmission was not successful, it is not repeated and only the corresponding TXBCF bit is set.

Note: In case a pending transmission is cancelled immediately before this transmission could have been started, there follows a short time window where no transmission is star ed even if another message is also pending in this node. This may enable another node to transmit a message which may have a lower priority than the second message in this node.

52.4.7.26 Tx Event handling

To support Tx event handling the M_TTCAN has implemented a Tx Event FIFO. After the M_TTCAN has transmitted a message on the CAN bus, Message ID and timestamp are stored in a Tx Event FIFO element. To link a Tx event to a Tx Event FIFO element, the Message Marker from the transmitted Tx Buffer is copied into the Tx Event FIFO element.

The Tx Event FIFO can be configured to a maximum of 32 elements. The Tx Event FIFO element is described in the Note *Depending on the configuration of the element size (TXESC), between two and sixteen 32-bit words (Tn = 3 to 17) are used for storage of a CAN message’s data field.*

When a Tx Event FIFO full condition is signaled by IR[TEFF], no further elements are written to the Tx Event FIFO until at least one element has been read out and the Tx Event FIFO Get Index has been incremented. In case a Tx event occurs while the Tx Event FIFO is full, this event is discarded and interrupt flag IR[TEFL] is set.

To avoid a Tx Event FIFO over ow, the Tx Event FIFO watermark can be used. When the Tx Event FIFO fill level reaches the Tx Event FIFO watermark configured by TXEFC[EFWM], interrupt flag IR[TEFW] is set.

When reading from the Tx Event FIFO, two times the Tx Event FIFO Get Index TXEFS[EFGI] has to be added to the Tx Event FIFO start address TXEFC[EFSA].

52.4.8 FIFO acknowledge handling

The Get Indexes of Rx FIFO 0, Rx FIFO 1, and the Tx Event FIFO are controlled by writing to the corresponding FIFO Acknowledge Index (refer to *Section 52.4.5.2.28: Rx FIFO 0 Acknowledge Register (RXF0A)*, *Section 52.4.5.2.33: Rx Buffer / FIFO element size configuration (RXESC)*, and *Section 52.4.5.2.44: Tx Event FIFO Configuration Register (TXEFC)*). Writing to the FIFO Acknowledge Index will set the FIFO Get Index to the FIFO Acknowledge Index plus one and thereby updates the FIFO Fill Level. There are two use cases:

When only a single element has been read from the FIFO (the one being pointed to by the Get Index), this Get Index value is written to the FIFO Acknowledge Index.

When a sequence of elements has been read from the FIFO, it is sufficient to write the FIFO Acknowledge Index only once at the end of that read sequence (value: Index of the last element read), to update the FIFO's Get Index.

Due to the fact that the CPU has free access to the M_TTCAN's Message RAM, special care has to be taken when reading FIFO elements in an arbitrary order (Get Index not considered). This might be useful when reading a High Priority Message from one of the two Rx FIFOs. In this case the FIFO's Acknowledge Index should not be written because this would set the Get Index to a wrong position and also alters the FIFO's Fill Level. In this case some of the older FIFO elements would be lost.

Note: The application has to ensure that a valid value is written to the FIFO Acknowledge Index. The M_CAN does not check for erroneous values.

52.4.9 M_TTCAN operations

52.4.9.1 Reference message

A reference message is a data frame characterized by a specific CAN identifier. It is received and accepted by all nodes except the Time Master (sender of the reference message).

For Level 1 the data length must be at least one; for Level 0, 2 the data length must be at least four; otherwise, the message is not accepted as reference message. The reference message may be extended by other data up to the sum of eight CAN data bytes. All bits of the identifier except the three LSBs characterize the message as a reference message. The last three bits specify the priorities of up to eight potential time masters. Reserved bits are transmitted as logical 0 and are ignored by the receivers. The reference message is configured via register TTRMC.

The time master transmits the reference message. If the reference message is disturbed by an error, it is retransmitted immediately. In case of a retransmission, the transmitted Master_Ref_Mark is updated. The reference message is sent periodically, but is allowed to stop the periodic transmission (Next_is_Gap bit) and to initiate transmission event-synchronized at the start of the next basic cycle by the current time master or by one of the other potential time masters.

The node transmitting the reference message is the current time master. The time master is allowed to transmit other messages. If the current time master fails, its function is replicated by the potential time master with the highest priority. Nodes that are neither time master nor potential time master are time-receiving nodes.

52.4.9.2 Level 1

Level 1 operation is configured via TTOCF[OM] = ‘01’ and TTOCF[GEN]. External clock synchronization is not available in Level 1. The information related to the reference message is stored in the first data byte as shown below. Cycle_Count is optional.

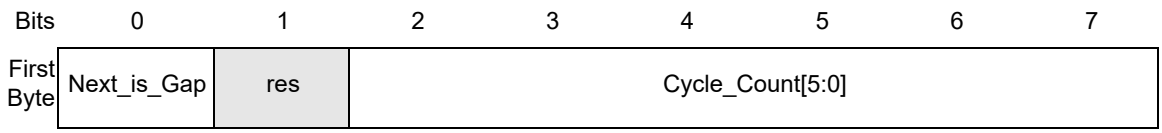


Figure 863. First byte of Level 1 reference message

52.4.9.3 Level 2

Level 2 operation is configured via TTOCF[OM] = ‘10’ and TTOCF[GEN].The information related to the reference message is stored in the first four data bytes as shown below. Cycle_Count and the lower four bits of NTU_Res are optional. The M_TTCAN does not evaluate NTU_Res[3:0] from received reference messages, it always transmits these bits as zero.

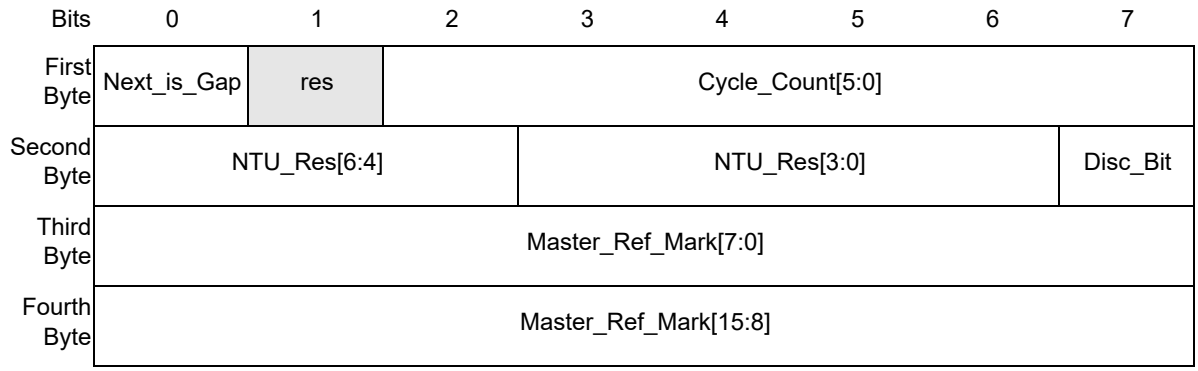


Figure 864. First four bytes of Level 2 reference message

52.4.9.4 Level 0

Level 0 operation is configured via TTOCF[OM] = ‘11’. External event-synchronized time-triggered operation is not available in Level 0. The information related to the reference message is stored in the first four data bytes as shown in the table below. In Level 0 Next_is_Gap is always zero. Cycle_Count and the lower four bits of NTU_Res are optional. The M_TTCAN does not evaluate NTU_Res[3:0] from received reference messages, it always transmits these bits as zero.

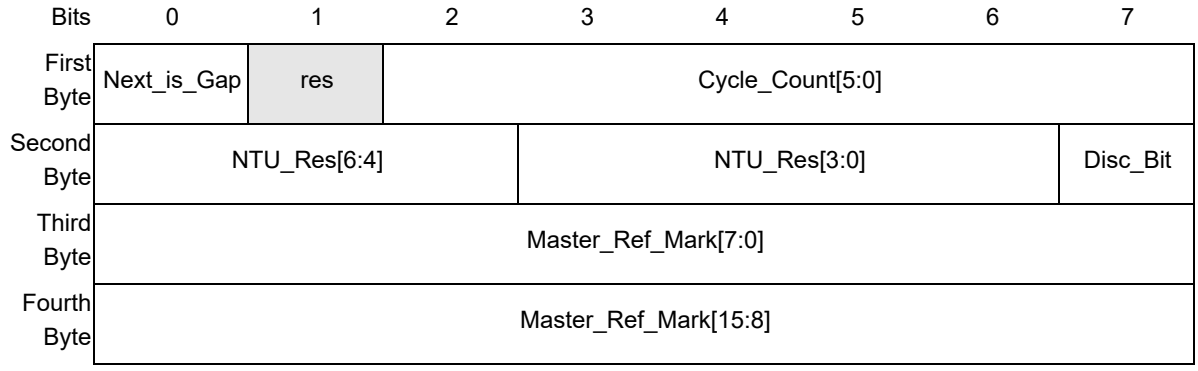


Figure 865. First four bytes of Level 0 reference message

52.4.10 M_TTCAN configuration

52.4.10.1 M_TTCAN timing

The Network Time Unit NTU is the unit in which all times are measured. The NTU is a constant of the whole network and is defined a priority by the network system designer. In M_TTCAN Level 1 the NTU is the nominal CAN bit time. In M_TTCAN Level 0 and Level 2 the NTU is a fraction of the physical second.

The NTU is the time base for the local time. The integer part of the local time (16-bit value) is incremented once each NTU. Cycle time and global time are both derived from local time. The fractional part (3-bit value) of local time, cycle time, and global time is not readable.

In M_TTCAN Level 0 and Level 2 the length of the NTU is defined by the Time Unit Ratio TUR. The TUR is in general a non-integer number, given by the formula TUR = TURNA[NAV]/TURCF[DC]. The length of the NTU is given by the formula NTU = CAN Clock Period $\times$ TUR.

The TUR Numerator Configuration NC is an 18-bit number, TURCF[NCL[15:0]] can be programmed in the range 0x0000–0xFFFF. TURCF[NCH[17:16]] is hard wired to 0b01. When the number 0xnnnn is written to TURCF[NCL[15:0]], TURNA[NAV] starts with the value 0x10000 + 0x0nnnn = 0x1nnnn. The TUR Denominator Configuration TURCF[DC] is a 14-bit number. TURCF[DC] may be programmed in the range 0x0001 - 0x3FFF, 0x0000 is an illegal value.

In Level 1, NC must be $\geq 4 \times$ TURCF**[**DC]. In Level 0, 2 NC must be $\geq 8 \times$ TURCF**[**DC] to allow the 3-bit resolution for the internal fractional part of the NTU.

A hardware reset presets TURCF[DC] to 0x1000 and TURCF[NCL] to 0x10000, resulting in an NTU consisting of 16 CAN clock periods. Local time and application watchdog are not started before either the CCCR[INIT] is reset, or TURCF[ELT is set. TURCF[ELT] may not be set before the NTU is configured. Setting TURCF[ELT] to e1f also locks the write access to register TURCF.

At startup TURNA[NAV] is updated from NC (= TURCF[NCL] + 0x10000) when TURCF[ELT] is set. In M_TTCAN Level 1 there is no drift compensation. TURNA.NAV does not change during operation, it always equals NC.

In M_TTCAN Level 0 and Level 2 there are two possibilities for TURNA[NAV] to change. When operating as time slave or backup time master, and when TTOCF[ECC] is set, TURNA[NAV] is updated automatically to the value calculated from the monitored global time speed, as long as the M_TTCAN is in synchronization state In_Schedule or In_Gap. When it loses synchronization it returns to NC. When operating as the actual time master, and when TTOCF[EECS] is set, the Host may update TURCF[NCL]. When the Host sets TTOCN[ECS], TURNA[NAV] will be updated from the new value of NC at the next reference message. The status flag TTOST[WECS] as is set when TTOCN[ECS] is set and is cleared when TURNA[NAV] is updated. TURCF[NCL] is write locked while TTOST[WECS] is set.

In M_TTCAN Level 0 and Level 2 the clock calibration process adapts TURNA[NAV] in the range of the Synchronization Deviation Limit SDL of NC+/- $2^{(TTOCF[LDSDL]+5)}$. TURCF[NCL] should be programmed to the largest applicable numerical value in order to achieve the best accuracy in the calculation of TURNA[NAV].

The synchronization deviation SD is the difference between NC and TURNA[NAV] (SD = |NC - TURNA[NAV]|). It is limited by the Synchronization Deviation Limit SDL, which is configured by its dual logarithm TTOCF[LDSDL] (SDL = $2^{(TTOCF[LDSDL]+5)}$) and should not exceed the clock tolerance given by the CAN bit timing configuration. SD is calculated at each new Basic Cycle. When the calculated TURNA[NAV] deviates by more than SDL from NC, or if the Disc_Bit in the Reference Message is set, the drift compensation is suspended and TTIR[GTE] is set and TTOSC[QCS] is reset, or in case of the Disc_Bit = ‘1’, TTIR[GTD] is set.

Figure 866. TUR configuration examples

TUR	8	10	24	50	510	125000	32.5	100/12	529/17
NC	0x1FFF8	0x1FFFE	0x1FFF8	0x1FFEA	0x1FFFE	0x1E848	0x1FFE0	0x19000	0x10880
TURCF.DC	0x3FFF	0x3333	0x1555	0x0A3D	0x0101	0x0001	0x0FC0	0x3000	0x0880

TTOCN[ECS] schedules NC for activation by the next reference message. TTOCN[SGT] schedules TTGTP[TP] for activation by the next reference message. Setting of TTOCN[ECS] and TTOCN[SGT] requires TTOCF[EECS] to be set (external clock synchronization enabled) while the M_TTCAN is actual time master.

The M_TTCAN module provides an application watchdog to verity the function of the application program. The Host has to serve this watchdog regularly, else all CAN bus activity is stopped. The Application Watchdog Limit TTOCF[AWL] specifies the number of NTUs between two times the watchdog has to be served. The maximum number of NTUs is 256. The Application Watchdog is served by reading register TTOST. TTOST[AWE] indicates whether the watchdog has been served in time. In case the application failed to serve the application watchdog, interrupt flag TTIR[AW] is set. For software development, the application watchdog may be disabled by programming TTOCF[AWL] to 0x00 (refer to *Section 52.4.7.10.1: Application Watchdog*).

52.4.10.2 Timing of interface signals

The timing events which cause a pulse at output M_TTCAN trigger time mark interrupt pulse and M_TTCAN register time mark interrupt pulse are generated in the CAN clock domain. There is a clock domain crossing delay to be considered before the same event is visible in the Host clock domain (TTIR[TTMI] resp. TTIR[RTMI] set). The signals can be connected for example to the timing input(s) of another M_TTCAN node, in order to automatically synchronize two M_TTCAN networks.

Output M_TTCAN start of cycle gets active whenever a reference message is completed (either transmitted or received). The output is controlled in the Host clock domain.

52.4.11 Message scheduling

TTOCF[TM**]** controls whether the M_TTCAN operates as a potential time master or as a time slave. If it is a potential time master, the three LSBs of the reference message's identifier TTRMC[RID] define the master priority, 0 giving the highest and 7 giving the lowest priority. There may not be two nodes in the network using the same master priority. TTRMC[RID] is used for recognition of reference messages. TTRMC[RMPS] is not relevant for time slaves.

The Initial Reference Trigger Offset TTOCF[IRTO] is a 7-bit-value that defines (in NTUs) how long a backup time master waits before it starts the transmission of a reference message when a reference message is expected but the bus remains idle. The recommended value for TTOCF[IRTO] is the master priority multiplied with a factor depending on the expected clock drift between the potential time masters in the network. The sequential order of the backup time masters, when one of them starts the reference message in case the current time master fails, should correspond to their master priority, even with maximum clock drift.

TTOCF[OM] decides whether the node operates in M_TTCAN Level 0, Level 1, or Level 2. In one network, all potential time masters have to operate on the same level. Time slaves

may operate on Level 1 in a Level 2 network, but not vice versa. The configuration of the M_TTCAN operation mode via TTOCF[OM] is the last step in the setup. With TTOCF[OM] = ‘00’ (event-driven CAN communication), the M_TTCAN operates according to ISO 11898-1, without time triggers.With TTOCF[OM] = ‘01’ (Level 1), the M_TTCAN operates according to ISO 11898-4, but without the possibility to synchronize the basic cycles to external events, the Next_is_Gap bit in the reference message is ignored. With TTOCF[OM] = ‘10’ (Level 2), the M_TTCAN operates according to ISO 11898-4, including the event-synchronized start of a basic cycle. With TTOCF[OM] = ‘11’ (Level 0), the M_TTCAN operates as event-driven CAN but maintains a calibrated global time base as in Level 2.

TTOCF[EECS] enables the external clock synchronization, allowing the application program of the current time master to update the TUR configuration during time-triggered operation, to adapt the clock speed and (in Level 0,2 only) the global clock phase to an external reference.

TTMLM[ENTT] in the TT Matrix Limits register specifies the number of expected Tx_Triggers in the system matrix. This is the sum of Tx_Triggers for exclusive, single arbitrating and merged arbitrating windows, excluding the Tx_Ref_Triggers. Note that this is usually not the number of Tx_Trigger memory elements; the number of basic cycles in the system matrix and the trigger’s repeat factors have to be taken into account. An inaccurate configuration of TTMLM[ENTT] will result in either a TxCount Underflow (TTIR[TXU] = ‘1’ and TTOST[EL] = ‘01’, severity 1) or in a Tx Count Overflow (TTIR[TXO] = ‘1’ and TTOST[EL] = ‘10’, severity 2).

Note: In case the first Reference Message seen by a node does not have Cycle_Count zero, this node may finish its first matrix cycle with its Tx count resulting in a Tx Count Underflow condition. As long as a node is in state Synchronizing its Tx_Triggers will not lead to transmissions.

TTMLM[CCM] specifies the number of the last basic cycle in the system matrix. The counting of basic cycles starts at 0. In a system matrix consisting of 8 basic cycles TTMLM[CCM] would be 7. TTMLM[CCM] is ignored by time slaves, a receiver of a reference message considers the received cycle count as the valid cycle count for the actual basic cycle.

TTMLM[TXEW] specifies the length of the Tx enable window in NTUs. The Tx enable window is that period of time at the beginning of a time window where a transmission may be started. If a transmission of a message cannot be started inside the Tx enable window because of for example, a slight overlap from the previous time window’s message, the transmission cannot be started in that time window at all. TTMLM[TXEW] has to be chosen with respect to the network’s synchronization quality and with respect to the relation between the length of the time windows and the length of the messages.

52.4.11.1 Trigger memory

The trigger memory is part of the external Message RAM to which the M_TTCAN is connected via its Generic Master Interface (refer to *Figure 754*). It stores up to 64 trigger elements. A trigger memory element consists of Time Mark TM, Cycle Code CC, Trigger Type TYPE, Filter Type FTYPE, Message Number MNR, Message Status Count MSC, Time Mark Event Internal TMIN, Time Mark Event External TMEX, and Asynchronous Serial Communication ASC (refer to *Section 52.4.6.6: Trigger Memory element*).

The time mark defines at which cycle time a trigger becomes active. The triggers in the trigger memory have to be sorted by their time marks. The trigger element with the lowest time mark is written to the first trigger memory word. Message number and cycle code are

ignored for triggers of type Tx_Ref_Trigger, Tx_Ref_Trigger_Gap, Watch_Trigger, Watch_Trigger_Gap, and End_of_List.

When the cycle time reaches the time mark of the actual trigger, the FSE switches to the next trigger and starts to read the following trigger from the trigger memory. In case of a transmit trigger, the Tx Handler starts to read the message from the Message RAM as soon as the FSE switches to its trigger. The RAM access speed defines the minimum time step between a transmit trigger and its preceding trigger, the Tx Handler has to be able to prepare the transmission before the transmit trigger's time mark is reached. The RAM access speed also limits the number of non-matching (with regard to their cycle code) triggers between two matching triggers, the next matching trigger must be read before its time mark is reached. If the reference message is n NTU long, a trigger with a time mark< n will never become active and will be treated as a configuration error.

Starting point of the cycle time is the sample point of the reference message's start of frame bit. The next reference message is requested when cycle time reaches the Tx_Ref_Trigger's time mark. The M_TTCAN reacts on the transmission request at the next sample point. A new Sync_Mark is captured at the start of frame bit, but the cycle time is incremented until the reference message is successfully transmitted (or received) and the Sync_Mark is taken as the new Ref_Mark. At that point in time, cycle time is restarted. As a consequence, cycle time can never (with the exception of initialization) be seen at a value< n, with n being the length of the reference message measured in NTU.

Length of a basic cycle: Tx_Ref_Trigger's time mark+1 NTU+1 CAN bit time.

The trigger list will be different for all nodes in the M_TTCAN network. Each node knows only the Tx_Triggers for its own transmit messages, the Rx_Triggers for those receive messages that are processed by this node, and the triggers concerning the reference messages.

52.4.11.1.1 Trigger types

Tx_Ref_Trigger (TYPE = '0000') and Tx_Ref_Trigger_Gap (TYPE = '0001') cause the transmission of a reference message by a time master. A configuration error (TTOST[EL] = '11', severity 3) is detected when a time slave encounters a Tx_Ref_Trigger(_Gap) in its trigger memory. Tx_Ref_Trigger_Gap is only used in external event-synchronized time-triggered operation mode. In that mode, Tx_Ref_Trigger is ignored when the M_TTCAN synchronization state is In_Gap (TTOST[SY]S = '10').

Tx_Trigger_Single (TYPE = '0010'), Tx_Trigger_Continuous (TYPE = '0011'), Tx_Trigger_Arbitration (TYPE = '0100'), and Tx_Trigger_Merged (TYPE = '0101') cause the start of a transmission. They define the start of a time window.

Tx_Trigger_Single starts a single transmission in an exclusive time window when the message buffer's Transmission Request Pending bit is set. After successful transmission the Transmission Request Pending bit is reset.

Tx_Trigger_Continuous starts a transmission in an exclusive time window when the message buffer's Transmission Request Pending bit is set. After successful transmission the Transmission Request Pending bit remains set, and the message buffer is transmitted again in the next matching time window.

Tx_Trigger_Arbitration starts an arbitrating time window, Tx_Trigger_Merged a merged arbitrating time window. The last Tx_Trigger of a merged arbitrating time window must be of type Tx_Trigger_Arbitration. A Configuration Error (TTOST[EL] = '11', severity 3) is detected when a trigger of type Tx_Trigger_Merged is followed by any other Tx_Trigger than one of type Tx_Trigger_Merged or Tx_Trigger_Arbitration. Several Tx_Triggers may be defined for

the same Tx message buffer. Depending on their cycle code, they may be ignored in some basic cycles. The cycle code has to be considered when the expected number of Tx_Triggers (TTMLM[ENTT]) is calculated.

Watch_Trigger (TYPE = ‘0110’) and Watch_Trigger_Gap (TYPE = ‘0111’) check for missing reference messages. They are used by both time masters and time slaves. Watch_Trigger_Gap is only used in external event-synchronized time-triggered operation mode. In that mode, a Watch_Trigger is ignored when the M_TTCAN synchronization state is In_Gap (TTOST[SYS] = ‘10’).

Rx_Trigger (TYPE = ‘1000’) is used to check for the reception of periodic messages in exclusive time windows. Rx_Triggers are not active until state In_Schedule or In_Gap is reached. The time mark of an Rx_Trigger shall be placed after the end of that message’s transmission, independent of time window boundaries. Depending on their cycle code, Rx_Triggers may be ignored in some basic cycles. At the time mark of the Rx_Trigger, it is checked whether the last received message before this time mark and after start of cycle or previous Rx_Trigger had matched the acceptance filter element referenced by MNR. Accepted messages are stored in one of the two receive FIFOs, according to the acceptance filtering, independent of the Rx_Trigger. Acceptance filter elements which are referenced by Rx_Triggers should be placed at the beginning of the filter list to ensure that the filtering is finished before the Rx_Trigger’s time mark is reached.

Time_Base_Trigger (TYPE = ‘1001’) are used to generate internal/external events depending on the configuration of ASC, TMIN, and TMEX.

End_of_List (TYPE = ‘1010…1111’) is an illegal trigger type, a configuration error (TTOST[EL] = ‘11’, severity 3) is detected when an End_of_List trigger is encountered in the trigger memory before the Watch_Trigger and Watch_Trigger_Gap.

52.4.11.1.2 Restrictions for the node’s trigger list

There may not be two triggers that are active at the same cycle time and cycle count, but triggers that are active in different basic cycles (different cycle code) may share the same time mark.

Rx_Triggers and Time_Base_Triggers may not be placed inside the Tx enable windows of Tx_Trigger_Single/Continuous/Arbitration, but they may be placed after Tx_Trigger_Merged.

Triggers that are placed after the Watch_Trigger (or the Watch_Trigger_Gap when TTOST[SYS] = ‘10’) will never become active. The watch triggers themselves will not become active when the reference messages are transmitted on time.

All unused trigger memory words (after the Watch_Trigger or after the Watch_Trigger_Gap when TTOST[SYS] = ‘10’) must be set to trigger type End_of_List.

A typical trigger list for a potential time master will begin with a number of Tx_Triggers and Rx_Triggers followed by the Tx_Ref_Trigger and the Watch_Trigger. For networks with external event- synchronized time-triggered communication, this is followed by the Tx_Ref_Trigger_Gap and the Watch_Trigger_Gap. The trigger list for a time slave will be the same but without the Tx_Ref_Trigger and the Tx_Ref_Trigger_Gap.

At the beginning of each basic cycle, that is at each reception or transmission of a reference message, the trigger list is processed starting with the first trigger memory element. The FSE looks for the first trigger with a cycle code that matches the current cycle count. The FSE waits until cycle time reaches the trigger’s time mark and activates the trigger.

Afterwards the FSE looks for the next trigger in the list with a cycle code that matches the current cycle count.

Special consideration is needed for the time around Tx_Ref_Trigger and Tx_Ref_Trigger_Gap. In a time master competing for master ship, the effective time mark of a Tx_Ref_Trigger may be decremented in order to be the first node to start a reference message. In backup time masters the effective time mark of a Tx_Ref_Trigger or Tx_Ref_Trigger_Gap is the sum of its configured time mark and the Reference Trigger Offset TTOCF[IRTO]. In case error level 2 is reached (TTOST[EL] = '10'), the effective time mark is the sum of its time mark and 0x127. No other trigger elements should be placed in this range otherwise it may happen, that the time marks appear out of order and are flagged as a configuration error. Trigger elements which are coming after Tx_Ref_Trigger may never become active as long as the reference messages come in time.

There are interdependencies between the following parameters:

- Host clock frequency
- Speed and waiting time for Trigger RAM accesses
- Length of the acceptance filter list
- Number of trigger elements
- Complexity of cycle code filtering in the trigger elements
- Offset between time marks of the trigger elements

52.4.11.1.3 Example for trigger handling

The example below shows how the trigger list is derived from a node's system matrix. Assumed node A is first time master and has knowledge of the section of the system matrix shown below.

Table 954. System Matrix Node A

Cycle Count	Time Mark1	Time Mark2	Time Mark3	Time Mark4	Time Mark5	Time Mark6	Time Mark7
0	Tx7					TxRef	Error
1	Rx3		Tx2, Tx4			TxRef	Error
2						TxRef	Error
3	Tx7		Rx5			TxRef	Error
4	Tx7			Rx6		TxRef	Error

The cycle count starts with 0 and runs until 0, 1, 3, 7, 15, 31, 63 (the number of basic cycles in the system matrix is 1, 2, 4, 8, 16, 32, 64). The maximum cycle count is configured by TTMLM.CCM. The Cycle Code CC is composed of repeat factor (= value of most significant '1') and the number of the first basic cycle in the system matrix (= bit field after most significant '1').

Example: with a cycle code of 0b0010011 (repeat factor: 16, first basic cycle: 3) and a maximum cycle count of TTMLM.CCM = '0x3F' matches occur at cycle counts 3, 19, 35, 51.

A trigger element consists of Time Mark TM, Cycle Code CC, Trigger Type TYPE, and Message Number MNR. For transmission MNR references the Tx Buffer number (0...31). For reception MNR references the number of the filter element (0 to 127) that matched

during acceptance filtering. Depending on the configuration of the Filter Type FTYPE, the 11-bit or 29-bit message ID filter list is referenced.

In addition a trigger element can be configured for Asynchronous Serial Communication ASC, generation of Time Mark Event Internal TMIN, and Time Mark Event External TMEX. The Message Status Count MSC holds the counter value (0...7) for scheduling errors for periodic messages in exclusive time windows at the point in time when the time mark of the trigger element became active.

Table 955. Trigger List Node A

Trigger	Time Mark TM[15:0]	Cycle Code CC [6:0]	Trigger Type TYPE [3:0]	Mess No. MNR [6:0]
0	Mark 1	0b0000100	Tx_Trigger_Single	7
1	Mark 1	0b1000000	Rx_Trigger	3
2	Mark 1	0b1000011	Tx_Trigger_Single	7
3	Mark 3	0b1000001	Tx_Trigger_Merged	2
4	Mark 3	0b1000011	Rx_Trigger	5
5	Mark 4	0b1000001	Tx_Trigger_Arbitration	4
6	Mark 4	0b1000100	Rx_Trigger	6
7	Mark 6	n.a.	Tx_Ref_Trigger	0 (Ref)
8	Mark 7	n.a.	Watch_Trigger	n.a.
9	n.a.	n.a.	End_of_List	n.a.

Tx_Trigger_Single, Tx_Trigger_Continuous, Tx_Trigger_Merged, Tx_Trigger_Arbitration, Rx_Trigger, and Time_Base_Trigger are only valid for the specified cycle code. For all other trigger types the cycle code is ignored.

The FSE starts the basic cycle with scanning the trigger list starting from zero until a trigger with time mark > cycle time and with its Cycle Code CC matching the actual cycle count is reached, or a trigger of type Tx_Ref_Trigger, Tx_Ref_Trigger_Gap, Watch_Trigger, or Watch_Trigger_Gap is encountered.

When the cycle time reached the Time Mark TM, the action defined by Trigger Type TYPE and Message Number MNR is started. There is an error in the configuration when End_of_List is reached.

At Mark6 the reference message (always TxRef) is transmitted. After transmission of the reference message the FSE returns to the beginning of the trigger list. When the Watch Trigger at Mark7 is reached, the node was not able to transmit the reference message; error treatment is started.

52.4.11.1.4 Detection of configuration errors

A configuration error is signaled via TTOST[EL] = ‘11’ (severity 3) when:

- The FSE comes to a trigger in the list with a cycle code that matches the current cycle count but with a time mark that is less than the cycle time.
- The previous active trigger was a Tx_Trigger_Merged and the FSE comes to a trigger in the list with a cycle code that matches the current cycle count but that is neither a

Tx_Trigger_Merged nor a Tx_Trigger_Arbitration nor a Time_Base_Trigger nor an Rx_Trigger.

- The FSE of a node with TTOCF[TM]=‘0’ (time slave) encounters a Tx_Ref_Trigger or a Tx_Ref_Trigger_Gap.
- Any time mark placed inside the Tx enable window (defined by TTMLM[TXEW]) of a Tx_Trigger with a matching cycle code.
- A time mark is placed near the time mark of a Tx_Ref_Trigger and the Reference Trigger Offset TTOST[RTO] causes a reversal of their sequential order measured in cycle time.

52.4.11.2 M_TTCAN schedule initialization

The synchronization to the M_TTCAN’s message schedule starts when CCCR[INIT] is reset. The M_TTCAN can operate strictly time-triggered (TTOCF[GEN] = ‘0’) or external event-synchronized time-triggered (TTOCF[GEN] = ‘1’). All nodes start with cycle time zero at the beginning of their trigger list with TTOST[SYS] = ‘00’ (out of synchronization), no transmission is enabled with the exception of the reference message. Nodes in external event-synchronized time-triggered operation mode will ignore Tx_Ref_Trigger and Watch_Trigger and will use instead Tx_Ref_Trigger_Gap and Watch_Trigger_Gap until the first reference message decides whether a Gap is active.

52.4.11.2.1 Time slaves

After configuration, a time slave will ignore its Watch_Trigger and Watch_Trigger_Gap when it did not receive any message before reaching the Watch_Triggers. When it reaches Init_Watch_Trigger, interrupt flag TTIR[IWT] is set, the FSE is frozen, and the cycle time will become invalid, but the node will still be able to take part in CAN bus communication (to give acknowledge or to send error flags). The first received reference message will restart the FSE and the cycle time.

Note: Init_Watch_Trigger is not part of the trigger list. It is implemented as an internal counter which counts up to 0xFFFF = maximum cycle time.

When a time slave has received any message but the reference message before reaching the Watch_Triggers, it will assume a fatal error (TTOST[EL] = ‘11’, severity 3), set interrupt flag TTIR[WT], switch off its CAN bus output, and enter the bus monitoring mode (CCCR[MON] set to ‘1’). In the bus monitoring mode it is still able to receive messages, but it cannot send any dominant bits and therefore cannot give acknowledge.

Note: To leave the fatal error state, the Host has to set CCCR[INIT] = ‘1’. After reset of CCCR[INIT], the node restarts M_TTCAN communication.

When no error is encountered during synchronization, the first reference message sets TTOST[SYS] = ‘01’ (Synchronizing), the second sets the M_TTCAN synchronization state (depending on its Next_is_Gap bit) to TTOST[SYS] = ‘11’ (In_Schedule) or TTOST[SYS] = ‘10’ (In_Gap), enabling all Tx_Triggers and Rx_Triggers.

52.4.11.2.2 Potential time masters

After configuration, a potential time master will start the transmission of a reference message when it reaches its Tx_Ref_Trigger (or its Tx_Ref_Trigger_Gap when in external event-synchronized time-triggered operation). It will ignore its Watch_Trigger and Watch_Trigger_Gap when it did not receive any message or transmit the reference message successfully before reaching the Watch_Triggers (assumed reason: all other

nodes still in reset or configuration, giving no acknowledge). When it reaches Init_Watch_Trigger, the attempted transmission is aborted, interrupt flag TTIR[IWT] is set, the FSE is frozen, and the cycle time will become invalid, but the node will still be able to take part in CAN bus communication (to give acknowledge or to send error flags). Resetting TTIR[IWT] will re-enable the transmission of reference messages until next time the Init_Watch_Trigger condition is met, or another CAN message is received. The FSE will not be restarted by the reception of a reference message.

When a potential time master reaches the Watch_Triggers after it has received any message but the reference message, it will assume a fatal error (TTOST[EL] = ‘11’, severity 3), set interrupt flag TTIR[WT], switch off its CAN bus output, and enter the bus monitoring mode (CCCR[MON] set to ‘1’). In bus monitoring mode, it is still able to receive messages, but it cannot send any dominant bits and therefore cannot give acknowledge.

When no error is detected during initialization, the first reference message sets TTOST[SYS] = ‘01’ (synchronizing), the second sets the M_TTCAN synchronization state (depending on its Next_is_Gap bit) to TTOST[SYS] = ‘11’ (In_Schedule) or TTOST[SYS] = ‘10’ (In_Gap), enabling all Tx_Triggers and Rx_Triggers.

A potential time master is current time master (TTOST[MS] = ‘11’) when it was the transmitter of the last reference message, else it is backup time master (TTOST[MS] = ‘10’).

When all potential time masters have finished configuration, the node with the highest time master priority in the network will become the current time master.

52.4.12 M_TTCAN gap control

All functions related to Gap control apply only when the M_TTCAN is operated in external event synchronized time-triggered mode (TTOCF[GEN] = ‘1’). In this operation mode the M_TTCAN message schedule may be interrupted by inserting Gaps between the basic cycles of the system matrix. All nodes connected to the CAN network have to be configured for external event- synchronized time-triggered operation.

During a Gap, all transmissions are stopped and the CAN bus remains idle. A Gap is finished when the next reference message starts a new basic cycle. A Gap starts at the end of a basic cycle that itself was started by a reference message with bit Next_is_Gap = ‘1’ , for example, Gaps are initiated by the current time master.

The current time master has two options to initiate a Gap. A Gap can be initiated under software control when the application program writes TTOCN[NIG] = ‘1’. The Next_is_Gap bit will be transmitted as ‘1’ with the next reference message. A Gap can also be initiated under hardware control when the application program enables the M_TTCAN event trigger input pin by writing TTOCN[GCS] = ‘1’. When a reference message is started and TTOCN[GCS] is set, a HIGH level at M_TTCAN event trigger pin will set Next_is_Gap = ‘1’.

As soon as that reference message is completed, the TTOST[WFE] bit will announce the Gap to the time master as well as to the time slaves. The current basic cycle will continue until its last time window. The time after the last time window is the Gap time.

For the actual time master and the potential time masters, TTOST[GSI] will be set when the last basic cycle has finished and the Gap time starts. In nodes that are time slaves, bit TTOST[GSI] will remain at ‘0’.

When a potential time master is in synchronization state In_Gap (TTOST[SYS] = ‘10’), it has four options to intentionally finish a Gap:

Under software control by writing TTOCN[FGP] = ‘1’.

Under hardware control (TTOCN[GCS] = ‘1’) an edge from HIGH to LOW at the M_TTCAN event-trigger input pin sets TTOCN[FGP] and restarts the schedule.

The third option is a time-triggered restart. When TTOCN[TMG] = ‘1’, the next register time mark interrupt (TTIR[RTMI] = ‘1’) will set TTOCN[FGP] and start the reference message.

Finally any potential time master will finish a Gap when it reaches its Tx_Ref_Trigger_Gap, assuming that the event to synchronize on did not occur in time.

Neither of these options can cause a basic cycle to be interrupted with a reference message.

Setting of TTOCN[FGP] after the Gap time has started will start the transmission of a reference message immediately and will thereby synchronize the message schedule. When TTOCN[FGP] is set before the Gap time has started (while the basic cycle is still in progress), the next reference message is started at the end of the basic cycle, at the Tx_Ref_Trigger – there will be no Gap time in the message schedule.

In strictly time-triggered operation, bit Next_is_Gap = ‘1’ in the reference message will be ignored, as well as the M_TTCAN event-trigger input pin and the bits TTOCN[NIG], TTOCN[FGP], and TTOCN[TMG].

52.4.13 Stop watch

The stop watch function enables capturing of M_TTCAN internal time values (local time, cycle time, or global time) triggered by an external event.

To enable the stop watch function, the application program first has to define local time, cycle time, or global time as stop watch source via TTOCN[SWS].When TTOCN[SWS] is ≠ ‘00’ and TT Interrupt Register flag TTIR[SWE] is ‘0’, the actual value of the time selected by TTTOCN[SWS] will be copied into TTCPT[SWV] on the next rising/falling edge (as configured via TTOCN[SWP]) on M_TTCAN stop watch trigger pin. This will set interrupt flag TTIR[SWE]. After the application program has read TTCPT[SWV], it may enable the next stop watch event by resetting TTIR[SWE] to ‘0’.

52.4.14 Local time, cycle time, global time, and external clock synchronization

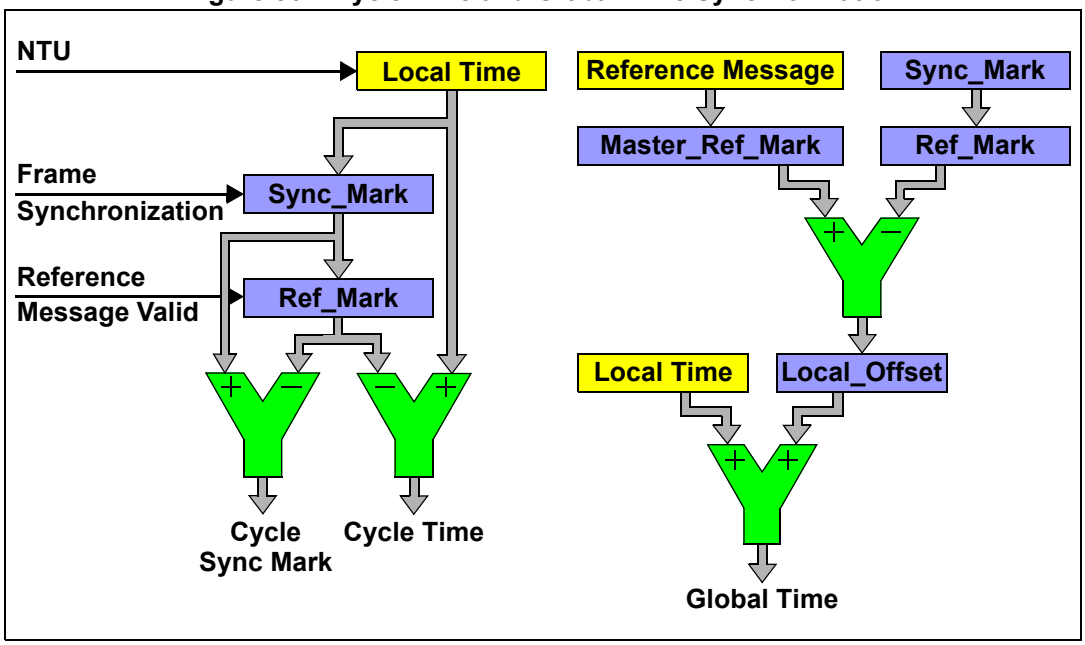
There are two possible levels in time-triggered CAN: Level 1 and Level 2. Level 1 only provides time-triggered operation using cycle time. Level 2 additionally provides increased synchronization quality, global time and external clock synchronization. In both levels, all timing features are based on a local time base - the local time.

The local time is a 16-bit cyclic counter, it is incremented once each NTU. Internally the NTU is represented by a 3-bit counter which can be regarded as a fractional part (three binary digits) of the local time. Generally, the 3-bit NTU counter is incremented 8 times each NTU. If the length of the NTU is shorter than 8 CAN clock periods (as may be configured in Level 1, or as a result of clock calibration in Level 2), the length of the NTU fraction is adapted, and the NTU counter is incremented only 4 times each NTU.

Figure 867 describes the synchronization of the cycle time and global time, performed in the same manner by all M_TTCAN nodes, including the time master. Any message received or transmitted invokes a capture of the local time taken at the message is frame synchronization event. This frame synchronization event occurs at the sample point of each Start of Frame (SoF) bit and causes the local time to be stored as Sync_Mark. Sync_Marks and Ref_Marks are captured including the 3-bit fractional part.

Whenever a valid reference message is transmitted or received, the internal Ref_Mark is updated from the Sync_Mark. The difference between Ref_Mark and Sync_Mark is the Cycle Sync Mark (Cycle Sync Mark = Sync_Mark - Ref_Mark) stored in register **TTCSM**. The most significant 16 bits of the difference between Ref_Mark and the actual value of the local time is the cycle time (Cycle Time = Local Time - Ref_Mark).

Figure 867. Cycle Time and Global Time Synchronization



The cycle time that can be read from TTCTC[CT] is the difference of the node's local time and Ref_Mark, both synchronized into the Host clock domain and truncated to 16 bit.

The global time exists for M_TTCAN Level 0 and Level 2 only, in Level 1 it is invalid. The node's view of the global time is the local image of the global time in (local) NTUs. After configuration, a potential time master will use its own local time as global time. The time master establishes its own local time as global time by transmitting its own Ref_Marks as Master_Ref_Marks in the reference message (bytes 3,4). The global time that can be read from TTLGT[GT] is the sum of the node's local time and its local offset, both synchronized into the Host clock domain and truncated to 16 bit. The fractional part is used for clock synchronization only.

A node that receives a reference message calculates its local offset to the global time by comparing its local Ref_Mark with the received Master_Ref_Mark. The node's view of the global time is local time + local offset. In a potential time master that has never received another time master's reference message, Local_Offset will be zero. When a node becomes the current time master after first having received other reference messages, Local_Offset will be frozen at its last value. In the time receiving nodes, Local_Offset may be subject to small adjustments, due to clock drift, when another node becomes time master, or when there is a global time discontinuity, signaled by Disc_Bit in the reference message. With the exception of global time discontinuity, the global time provided to the application program by register TTLGT is smoothed by a low-pass filtering to have a continuous monotonic value.

Figure 868. M_TTCAN Level 0 and Level 2 Drift Compensation

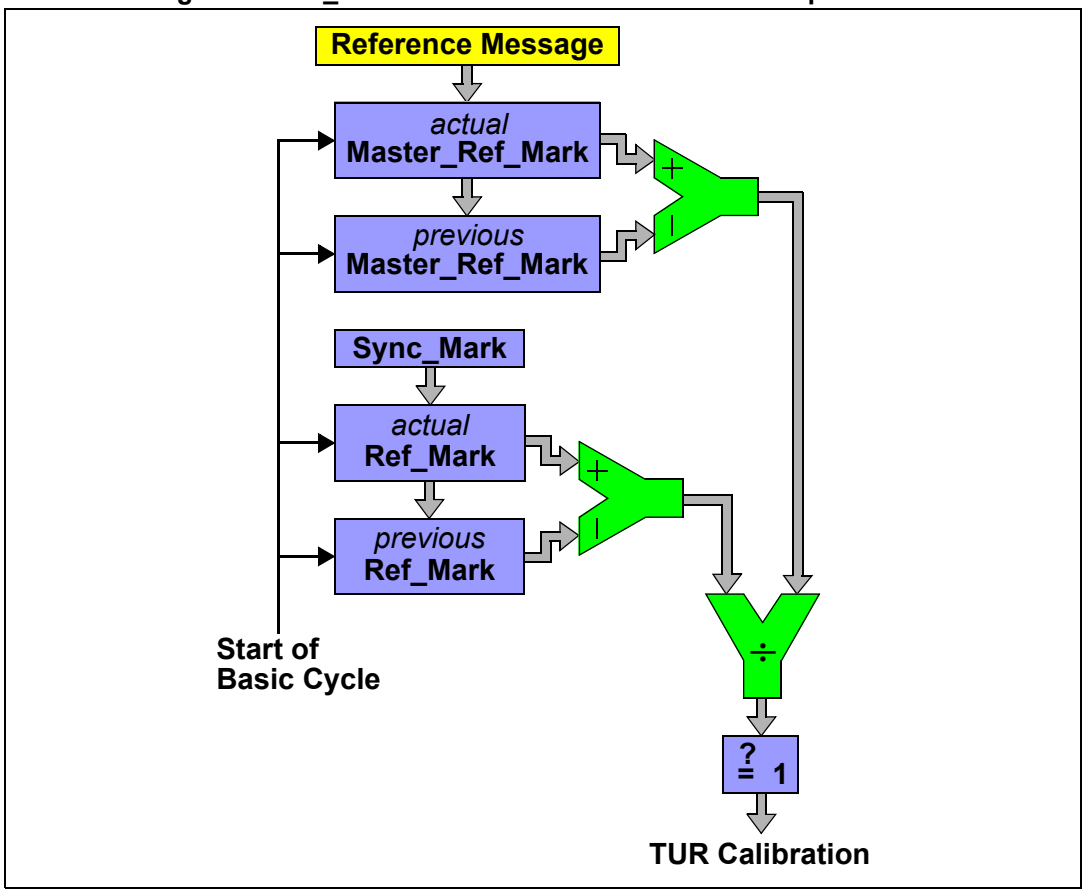


Figure 868 describes how in M_TTCAN Level 0,2 each time receiving node compensates the drift between its own local clock and the time master's clock by comparing the length of a basic cycle in local time and in global time. If there is a difference between the two values and the Disc_Bit in the reference message is not set, a new value for TURNA[NAV] is calculated. If the Synchronization Deviation SD = |NC - TURNA[NAV]| ≤ SDL (Synchronization Deviation Limit), the new value for TURNA[NAV] takes effect. Else the automatic drift compensation is suspended.

In M_TTCAN Level 0 and Level 2, TTOST[QCS] indicates whether the automatic drift compensation is active or suspended. In M_TTCAN Level 1, TTOST[QCS] is always '1'.

The current time master may synchronize its local clock speed and the global time phase to an external clock source. This is enabled by bit TTOCF[EECS].

The stop watch function (refer to *Section 52.4.13*) may be used to measure the difference in clock speed between the local clock and the external clock. The local clock speed is adjusted by first writing the newly calculated Numerator Configuration Low to TURCF[NCL] (TURCF[DC] cannot be updated during operation). The new value takes effect by writing TTOCN[ECS] to '1'.

The global time phase is adjusted by first writing the phase offset into the TT Global Time Preset register TTGTP. The new value takes effect by writing TTOCN[SGT] to '1'. The first reference message transmitted after the global time phase adjustment will have the Disc_Bit set to '1'.

TTOST[QGTP] shows whether the node's global time is in phase with the time master's global time. TTOST[QGTP] is permanently '0' in M_TTCAN Level 1 and when the Synchronization Deviation Limit is exceeded in M_TTCAN Level 0,2 (TTOST[QCS] = '0'). It is temporarily '0' while the global time is low-pass filtered to supply the application with a continuous monotonic value. There is no low-pass filtering when the last reference message contained a Disc_Bit = '1' or when TTOST[QCS]='0'.

52.4.15 M_TTCAN error level

The ISO 11898-4 specifies four levels of error severity:

- S0: No Error
- S1: Warning.
  Only notification of application, reaction application-specific.
- S2: Error.
  Notification of application. All transmissions in exclusive or arbitrating time windows are disabled (that is no data or remote frames may be started). Potential time masters still transmit reference messages with the Reference Trigger Offset TTOST[RTO] set to the maximum value of 127.
- S3: Severe Error.
  Notification of application. All CAN bus operations are stopped, that is transmission of dominant bits is not allowed, and CCCR[MON] is set. The S3 error condition remains active until the application updates the configuration (set CCCR[CCE]).

If several errors are detected at the same time, the highest severity prevails. When an error is detected, the application is notified by TTIR[ELC]. The error level is monitored by TTOST[EL].

The M_TTCAN signals the following error conditions as required by ISO 11898-4:

- Config_Error (S3):
  Sets Error Level TTOST[EL] to '11' when a merged arbitrating time window is not properly closed or when there is a Tx_Trigger with a time mark beyond the Tx_Ref_Trigger.
- Watch_Trigger_Reached (S3):
  Sets Error Level TTOST[EL] to '11' when a watch trigger was reached because the reference message is missing.
- Application_Watchdog (S3):
  Sets Error Level TTOST[EL] to '11' when the application failed to serve the application watchdog.The application watchdog is configured via TTOCF[AWL]. It is served by reading register TTOST.When the watchdog is not served in time, bit TTOST[AWE] and interrupt flag TTIR[AW] are set, all M_TTCAN communication is stopped, and the M_TTCAN is set into bus monitoring mode (CCCR[MON]set to '1').
- CAN_Bus_Off (S3):
  Entering CAN_Bus_Off state sets error level TTOST[EL] to '11'. CAN_Bus_Off state is signaled by PSR[BO] = '1' and CCCR[INIT] = '1'.
- Scheduling_Error_2 (S2):
  Sets Error Level TTOST[EL] to '10' if the MSC of one Tx_Trigger has reached 7. In addition interrupt flag TTIR[SE2] is set. The Error Level TTOST[EL] is reset to '00' at the beginning of a matrix cycle when no Tx_Trigger has an MSC of 7 in the preceding matrix cycle.
- Tx_Overflow (S2):
  Sets Error Level TTOST[EL] to '10' when the Tx count is equal or superior to the

expected number of Tx_T riggers TTMLM[ENTT] and a Tx_Trigger event occurs. In addition interrupt flag TTIR[TXO] is set. The Error Level TTOST[EL] is reset to '00' when the Tx count is no more than TTMLM[ENTT] at the start of a new matrix cycle.

- Scheduling_Error_1 (S1):
  Sets Error Level TTOST[EL] to '01' if within one matrix cycle the difference between the maximum MSC and the minimum MSC for all trigger memory elements (of exclusive time windows) is larger than 2, or if one of the MSCs of an exclusive Rx_Trigger has reached 7. In addition interrupt flag TTIR[SE1] is set. If within one matrix cycle none of these conditions is valid, the Error Level TTOST[EL] is reset to '00'.
- Tx_Underflow (S1):
  Sets Error Level TTOST[EL] to '01' when the Tx count is less than the expected number of Tx_Triggers TTMLM[ENTT] at the start of a new matrix cycle. In addition interrupt flag TTIR[TXU] is set. The Error Level TTOST[EL] is reset to '00' when the Tx count is at least TTMLM[ENTT] at the start of a new matrix cycle.

52.4.16 M_TTCAN message handling

52.4.16.1 Reference message

For potential time masters the identifier of the reference message is configured via TTRMC[RID]. No dedicated Tx Buffer is required for transmission of the reference message. When a reference message is transmitted, the first data byte (M_TTCAN Level 1) resp. the first four data bytes (M_TTCANM_TTCAN Level 0 and Level 2) will be provided by the FSE.

In case the Reference Message Payload Select TTRMC[RMPS] is set, the rest of the reference message's payload (Level 1: bytes 2-8, Level 0,2: bytes 5-6) is taken from Tx Buffer 0. In this case the data length DLC code from message buffer 0 is used.

Table 956. Number of Data Bytes transmitted with a Reference Messages

TTRMC.RMPS	TXBRP.TRP0	Level 0	Level 1	Level 2
0	0	4	1	4
0	1	4	1	4
1	0	4	1	4
1	1	4+MBO	1+MBO	4+MBO

To send additional payload with the reference message in Level 1 a DLC > 1 has to be configured, for Level 0, 2 a DLC > 4 is required. In addition the transmission request pending bit TXBRP[TRP0] of message buffer 0 must be set (refer to *Table 956*). In case bit TXBRP[TRP0] is not set when a reference message is started, the reference message is transmitted with the data bytes supplied by the FSE only.

For acceptance filtering of reference messages the Reference Identifier TTRMC[RID] is used.

52.4.16.2 Message reception

Message reception is done via the two Rx FIFOs in the same way as for event-driven CAN communication (refer to *Section 52.3.10*).

The Message Status Count MSC is part of the corresponding trigger memory element and has to be initialized to zero during configuration. It is updated while the M_TTCAN is in

synchronization states In_Gap or In_Schedule. The update happens at the message's Rx_Trigger. At this point in time it is checked at which acceptance filter element the latest message received in this basic cycle had matched. The matching filter number is stored as the acceptance filter result. If this is the same the filter number as defined in this trigger memory element, the MSC is decremented by one. If the acceptance filter result is not the same filter number as defined for this filter element, or if the acceptance filter result is cleared, the MSC is incremented by one. At each Rx_Trigger and at each start of cycle, the last acceptance filter result is cleared.

The time mark of an Rx_Trigger should be set to a value where it is ensured that reception and acceptance filtering for the targeted message has completed. This has to take into consideration the RAM access time and the order of the filter list. It is recommended, that filters which are used for Rx_Triggers are placed at the beginning of the filter list. It is not recommended to use an Rx_Trigger for the reference message.

52.4.16.3 Message transmission

For time-triggered message transmission the M_TTCAN supplies 32 dedicated Tx buffers (refer to *Section 52.4.7.20: Dedicated Tx buffers*). A Tx FIFO or Tx queue is not available when the M_TTCAN is configured for time-triggered operation (TTOCF[OM] = '01' or '10').

Each Tx_Trigger in the trigger memory points to a particular Tx buffer containing a specific message. There may be more than one Tx_Trigger for a given Tx buffer if that Tx buffer contains a message that is to be transmitted more than once in a basic cycle or matrix cycle.

The application program has to update the data regularly and on time, synchronized to the cycle time. The Host CPU is responsible that no partially updated messages are transmitted. To assure this the Host has to proceed in the following way:

Tx_Trigger_Single / Tx_Trigger_Merged / Tx_Trigger_Arbitration

- Check whether the previous transmission has completed by reading TXBTO
- Update the Tx buffer's configuration or payload, or both
- Issue an Add Request to set the Tx Buffer Request Pending bit

Tx_Trigger_Continous

- Issue a Cancellation Request to reset the Tx Buffer Request Pending bit
- Check whether the cancellation has finished by reading TXBCF
- Update Tx buffer's configuration or payload, or both
- Issue an Add Request to set the Tx Buffer Request Pending bit

The message's MSC stored with the corresponding Tx_Trigger provides information on the success of the transmission.

The MSC is incremented by one when the transmission could not be started because the CAN bus was not idle within the corresponding transmit enable window or when the message was started and could not be completed successfully. The MSC is decremented by one when the message was transmitted successfully or when the message could have been started within its transmit enable window but was not started because transmission was disabled (M_TTCAN in Error Level S2 or Host has disabled this particular message).

The Tx buffers may be managed dynamically, that is several messages with different identifiers may share the same Tx buffer element. In this case the Host has to assure that no transmission request is pending for the Tx buffer element to be reconfigured by checking TXBRP.

If a Tx buffer with pending transmission request should be updated, the Host first has to issue a cancellation request and check whether the cancellation has completed by reading TXBCF before it starts updating.

The Tx Handler will transfer a message from the Message RAM to its intermediate output buffer at the trigger element which becomes active immediately before the Tx_Trigger element which defines the beginning of the transmit window. During and after the transfer time the transmit message may not be updated and its TXBRP bit may not be changed. To control this transfer time, an additional trigger element may be placed before the Tx_Trigger. This may be example of a Time_Base_Trigger which need not cause any other action. The difference in time marks between the Tx_Trigger and the preceding trigger has to be large enough to guarantee that the Tx Handler can read four words from the Message RAM even at high RAM access load from other modules.

52.4.16.3.1 Transmission in exclusive time windows

A transmission is started time-triggered when the cycle time reaches the time mark of a Tx_Trigger_Single or Tx_Trigger_Continuous. There is no arbitration on the bus with messages from other nodes. The MSC is updated according the result of the transmission attempt. After successful transmission started by a Tx_Trigger_Single the respective Tx Buffer Request Pending bit is reset. After successful transmission started by a Tx_Trigger_Continuous the respective Tx Buffer Request Pending remains set. When the transmission was not successful due to disturbances, it will be repeated next time (one of) its Tx_Trigger(s) become(s) active.

52.4.16.3.2 Transmission in arbitrating time windows

A transmission is started time-triggered when the cycle time reaches the time mark of a Tx_Trigger_Arbitration. Several nodes may start to transmit at the same time. In this case the message has to arbitrate with the messages from other nodes. The MSC is not updated. When the transmission was not successful (lost arbitration or disturbance), it will be repeated next time (one of) its Tx_Trigger(s) become(s) active.

52.4.16.3.3 Transmission in merged arbitrating time windows

The purpose of a merged arbitrating time window is to enable multiple nodes to send a limited number of frames which are transmitted in immediate sequence, the order given by CAN arbitration. It is not intended for burst transmission by a single node. Since the node does not have exclusive access within this time window, it may happen that not all requested transmissions are successful.

Messages which have lost arbitration or were disturbed by an error, may be re-transmitted inside the same merged arbitrating time window. The re-transmission will not be started if the corresponding Transmission Request Pending flag was reset by a successful Tx cancellation.

In single transmit windows, the Tx Handler transmits the message indicated by the message number of the trigger element. In merged arbitrating time windows, it can handle up to three message numbers from the trigger list. Their transmissions will be attempted in the sequence defined by the trigger list. If the time mark of a fourth message is reached before the first is transmitted (or canceled by the Host), the four the request will be ignored.

The transmission inside a merged arbitrating time window is not time-triggered. The transmission of a message may start before its time mark, or after the time mark if the bus was not idle.

The messages transmitted by a specific node inside a merged arbitrating time window will be started in the order of their Tx_Triggers, so a message with low CAN priority may prevent the successful transmission of a following message with higher priority, if their is compelling bus traffic. This has to be considered for the configuration of the trigger list. Time_Base_Triggers may be placed between consecutive Tx_Triggers to define the time until the data of the corresponding Tx Buffer needs to be updated.

52.4.17 M_TTCAN interrupt and error handling

The TT Interrupt Register TTIR consists off our segments. Each interrupt can be enabled separately by the corresponding bit in the TT Interrupt Enable register TTIE. The flags remain set until the Host clears them. A flag is cleared by writing a '1' to the corresponding bit position.

The first segment consists of flags CER, AW, WT, and IWT. Each flag indicates a fatal error condition where the CAN communication is stopped. With the exception of IWT, these error conditions require a re-configuration of the M_TTCAN module before the communication can be restarted.

The second segment consists of flags ELC, SE1, SE2, TXO, TXU, and GTE. Each flag indicates an error condition where the CAN communication is disturbed. If they are caused by a transient failure, for example by disturbances on the CAN bus, they will be handled by the M_TTCAN protocol's failure handling and do not require intervention by the application program.

The third segment consists of flags GTD, GTW, SWE, TTMI, and RTMI. The first two flags are controlled by global time events (Level 0,2 only) that require a reaction by the application program. With a Stop Watch Event triggered by a rising edge on stop watch trigger pin internal time values are captured. The Trigger Time Mark Interrupt notifies the application that a specific Time_Base_Trigger is reached. The Register Time Mark Interrupt signals that the time referenced by TTOCN[TMC] (Cycle, Local, or Global) equals time mark TTTMK[TM]. It can also be used to finish a Gap.

The fourth segment consists of flags SOG, CSM, SMC, and SBC. These flags provide a means to synchronize the application program to the communication schedule.

52.4.18 Level 0

M_TTCAN Level 0 is not part of ISO11898-4. This operation mode makes the hardware, that in M_TTCAN Level 2 maintains the calibrated global time base, also available for event-driven CAN according to ISO11898-1.

Level 0 operation is configured via TTOCF[OM] = '11'. In this mode the M_TTCAN operates in event driven CAN communication, there is no fixed schedule, the configuration of TTOCF[GEN] is ignored. External event-synchronized operation is not available in Level 0. A synchronized time base is maintained by transmission of reference messages.

In Level 0 the trigger memory is not active and therefore needs not to be configured. The time mark interrupt flag (TTIR[TTMI]) is set when the cycle time has reached TTOCF[IRTO] $\times$ 0x200, it reminds the Host to set a transmission request for message buffer 0. The Watch_Trigger interrupt flag (TTIR[WT]) is set when the cycle time has reached 0xFF00. These values were chosen to have enough margin for a stable clock calibration. There are no further TT-error-checks.

Register time mark interrupts (TTIR[RTMI]) are also possible.

The reference message is configured as for Level 2 operation. Received reference messages are recognized by the identifier configured in register TTRMC. For the transmission of reference messages only message buffer 0 may be used. The node transmits reference messages any time the Host sets a transmission request for message buffer 0, there is no reference trigger offset.

Level 0 operation is configured via:

- TTRMC
- TTOCF except EVTP, AWL, GEN
- TTMLM except ENTT, TXEW
- TURCF

Level 0 operation is controlled via:

- TTOCN except NIG, TMG, FGP, GCS, TTMIE
- TTGTP
- TTTMK
- TTIR excluding bits CER, AW, IWT SE2, SE1, TXO, TXU, SOG (no function)
- TTIR the following bits have changed function
  - TTMI not defined by trigger memory: activated at cycle time TTOCF[IRTO] $\times$ 0x200
  - WT not defined by trigger memory: activated at cycle time 0xFF00

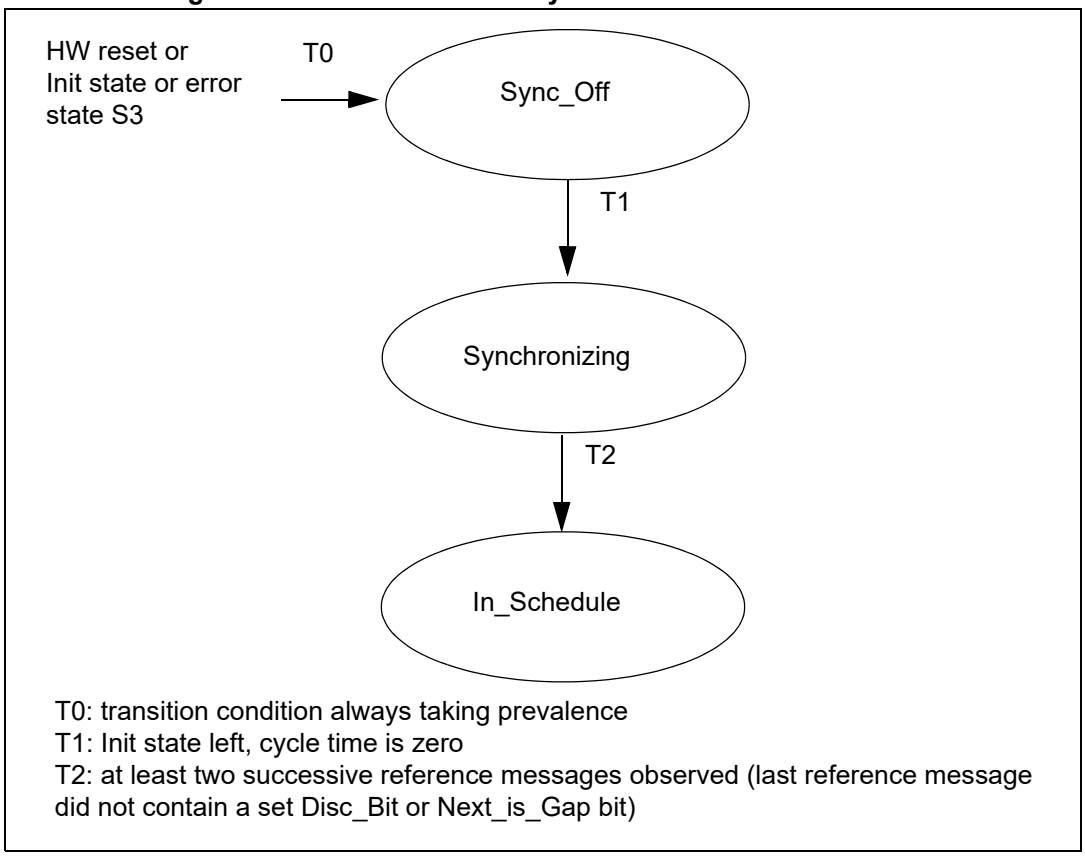
Level 0 operation is signaled via:

- TTOST excluding bits AWE, WFE, GSI, GFI, RTO (no function)

52.4.18.1 Synchronizing

Figure 869 describes the states and state transitions in M_TTCAN Level 0 operation. Level 0 has no In_Gap state.

Figure 869. Level 0 schedule synchronization state machine



52.4.18.2 Handling of error levels

During Level 0 operation only the following error conditions may occur:

- Watch_Trigger_Reached (S3), reached cycle time 0xFF00
- CAN_Bus_Off (S3)

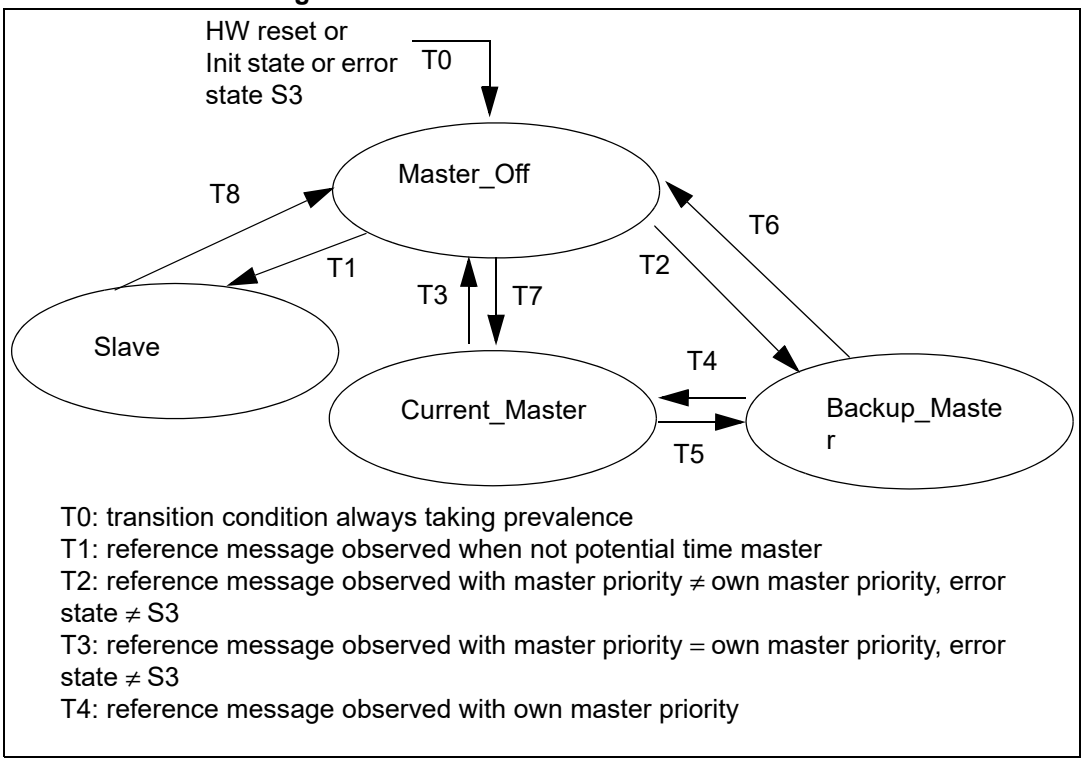
Since no S1 and S2 error are possible, the error level can only switch between S0 (No Error) and S3 (Severe Error). In M_TTCAN Level 0 an S3 error is handled differently. When error level S3 is reached, both TTOST[SYS] and TTOST[MS] are reset, and interrupt flags TTIR[GTE] and TTIR[GTD] are set.

When error level S3 (TTOST[EL] = ‘11’) is entered, bus monitoring mode is, contrary to M_TTCAN Level 1 and Level 2, not entered. S3 error level is left automatically after transmission (time master) or reception (time slave) of the next reference message.

52.4.18.3 Master slave relation

Figure 870 below describes the master slave relation in M_TTCAN Level 0. In case of an S3 error the M_TTCAN returns to state Master_Off.

Figure 870. Level 0 master to slave relation



52.4.19 Synchronization to external time schedule

This feature can be used to synchronize the phase of the M_TTCAN's schedule to an external schedule (such as that of a second M_TTCAN network or FlexRay network). It is applicable only when the M_TTCAN is current time master (TTOST[MS] = '11').

External synchronization is controlled by event trigger input pin. If bit TTOCN[ESCN] is set, a rising edge at the event trigger pin the M_TTCAN compares its actual cycle time with the target phase value configured by TTGTP[CTP].

Before setting TTOCN[ESCN] the Host has to adapt the phases of the two time schedules such as by using the M_TTCAN gap control (refer to *Section 52.4.12*). When the Host sets TTOCN[ESCN], TTOST[SPL] is set.

If the difference between the cycle time and the target phase value TTGTP[CTP] at the rising edge at M_TTCAN event trigger pin is greater than 9 NTU, the phase lock bit TTOST[SPL] is reset, and interrupt flag TTIR[CSM] is set. TTOST[SPL] is also reset (and TTIR[CSM] is set), when another node becomes time master.

If both TTOST[SPL] and TTOCN[ESCN] are set, and if the difference between the cycle time and the target phase value TTGTP[CTP] at the rising edge at event trigger pin is less or equal 9 NTU, the phase lock bit TTOST[SPL] remains set, and the measured difference is used as reference trigger offset value to adjust the phase at the next transmitted reference message.

Note: The rising edge detection at event trigger pin is enabled with the start of each basic cycle. The first rising edge triggers the compare of the actual cycle time with TTGTP[CTP]. All further edges until the beginning of the next basic cycle are ignored.

52.5 CAN RAM arbiter

This block acts as a dynamic arbiter between the various CAN nodes in one CAN subsystem and the CPU for granting access to the shared CAN memory. The dynamic arbitration ensures:

- 50% bandwidth for CPU, 50% shared by the CAN nodes when simultaneous access requested from all masters and CPU,
- Bandwidth dynamic upgrading to requesting masters if other masters do not request access.

Note: From the arbiter prospective the CPU bandwidth may be limited to 50%. The CPU may be waiting stated due to this bandwidth limitation when accessing the shared memory.

52.5.1 Features

- Active low asynchronous reset
- 1–CPU + up to 5–CAN arbiter
- Combined dynamic arbitration scheme: Bandwidth between CPU and M_CAN interfaces can be redistributed. If active, the CPU interface is granted a 50% portion of the available bandwidth
- Bandwidth dynamic upgrading to requesting masters if other masters do not request access
- The SRAM access by a single master needs two cycles (address and data) for each read/write command. The arbiter has a pseudo address pre-fetching mechanism. This mechanism acts when multiple masters are accessing the SRAM and it overlaps the address cycle of one master with the data cycle of another master. In case of multiple masters accessing the SRAM, the pseudo address pre-fetching scheme saves multiple clock cycles
- The peripheral GSI module enable acts as a request from each master. Each master will be made to wait at least one clock cycle before the grant is given. The arbiter ensures that the CPU does not wait for more than 1 clock cycle, hence ensuring a 50% guaranteed bandwidth
- Time slot is defined as one peripheral clock cycle

52.5.2 Functional overview using examples

The dynamic arbitration scheme is explained using the following examples.

- Example 1 (active CPU IF, 3 out of 3 CAN nodes are active):
  - CPU obtains every 2nd slot
  - CAN 1/2/3 obtains every 6th time slot
- Example 2 (inactive CPU IF, 4 out of 4 CAN nodes are active):
  - CPU obtains no time slots
  - CAN 1/2/3/4 obtains every 4th time slot
- Example 3 (active CPU IF, 2 out of 4 CAN node are active):
  - CPU obtains every 2nd time slot
  - Active CAN 1/2 obtains every 4th time slot
- Example 4 (inactive CPU IF, 2 out of 4 CAN node are active):
  - CPU obtains no time slots
  - Active CAN 1/2 obtains every 2nd time slot

52.6 SRAM interface and memory organization

The CAN subsystem will interface with an external RAM using this interface. The RAM sizes required for the CAN subsystem are detailed in *Section 52.11: Shared memory map*.

52.6.1 ECC controller

The RAM embeds ECC logic and code to provide Single Error Correction / Double Error Detection (SECCDED). The module does not guarantee any proper functionality if more than 2 bits are in errors. It supports only 32-bit write access and 8-/16-/32-bit read access. The ECC error is reported to each CAN module as well as available at the CAN subsystem top, so that the same can be forwarded to the Error Management Module.

The module uses Hamming code for single error correction and double error detection logic. It involves transmitting data with multiple ecc_check bits and decoding the associated ecc_check bits when receiving data to detect errors. The SECDED Hamming code is not able to detect three bit errors. Rather, in the presence of a three bit error, a conventional SECDED code returns an error code that is indistinguishable from an error code resulting from a single bit error. Hence, this module does not guarantee any proper functionality when more than 2 bits are in error.

The ecc_check bits are parallel parity bits generated from XORing certain bits in the original data word. If bit error(s) are introduced in the codeword, several ecc_check bits show parity errors after decoding the retrieved codeword. The combination of these ecc_check bit errors displays the nature of the error. In addition, the position of any single bit error is identified from the ecc_check bits.

The ECC error address is reported at the top of the subsystem using a 16 bit ECC error address signal. This address is valid only when the ECC bit error output is valid.

52.6.1.1 Features

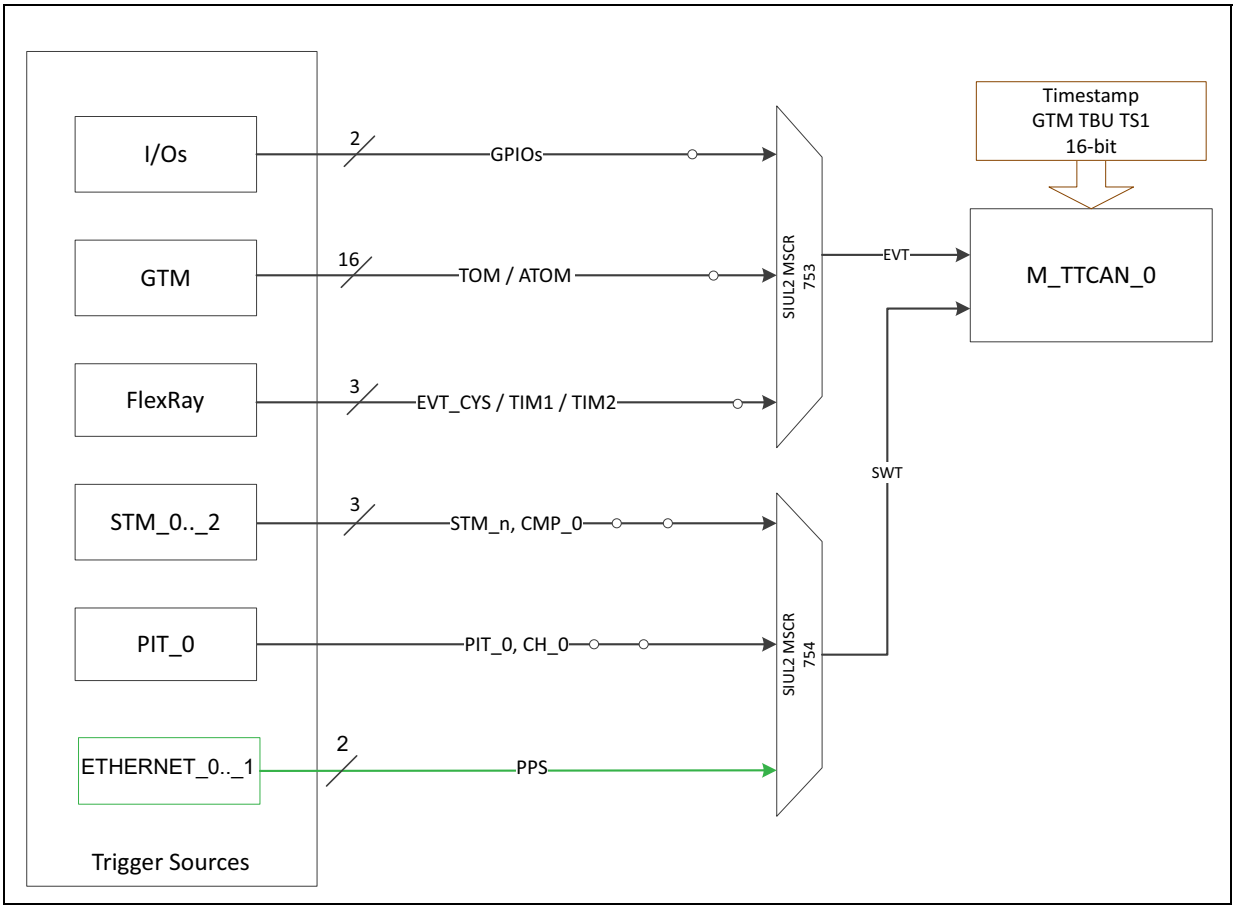
The ECC module supports the following:

- ecc_error = ‘00’ No error is detected in the read data from memory
- ecc_error = ‘01’ This indicates single bit error occurred within the 39-bit codeword. In addition, the error is corrected, and the forwarded data to the master is error free.
- ecc_error = ‘10’ This indicates that a two bit error has occurred within the codeword. In this case, no error correction is possible.
- ecc_error = ‘11’ This indicates that errors beyond the detection capability have occurred within the codeword and no error correction is possible. This is an invalid error type.

52.7 Time Triggered CAN matrix and basic cycles management

Figure 871 shows hardware control for M_TTCAN cycles management.

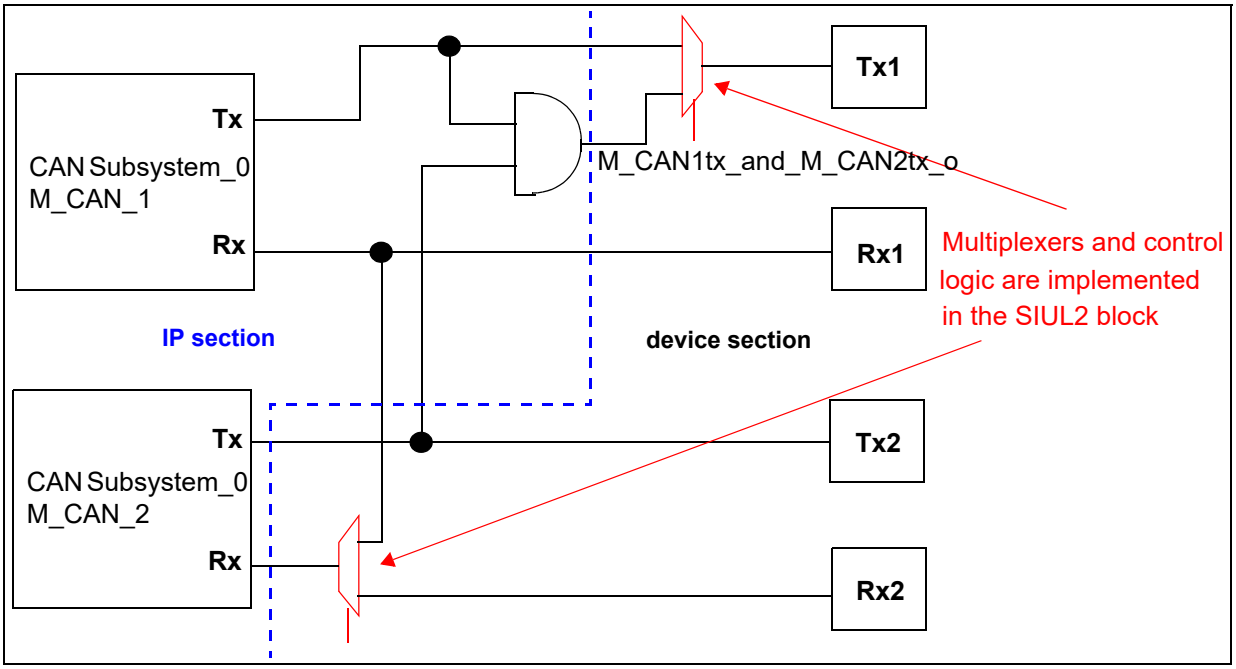
Figure 871. Hardware control for M_TTCAN cycles management



52.8 CAN nodes 1 and 2 I/Os sharing

Each CAN node has its Tx/Rx signals connected to separated GPIO pairs. It is required for M_CAN_1 and M_CAN_2 to share one common GPIO pair. This feature is described in *Figure 872*.

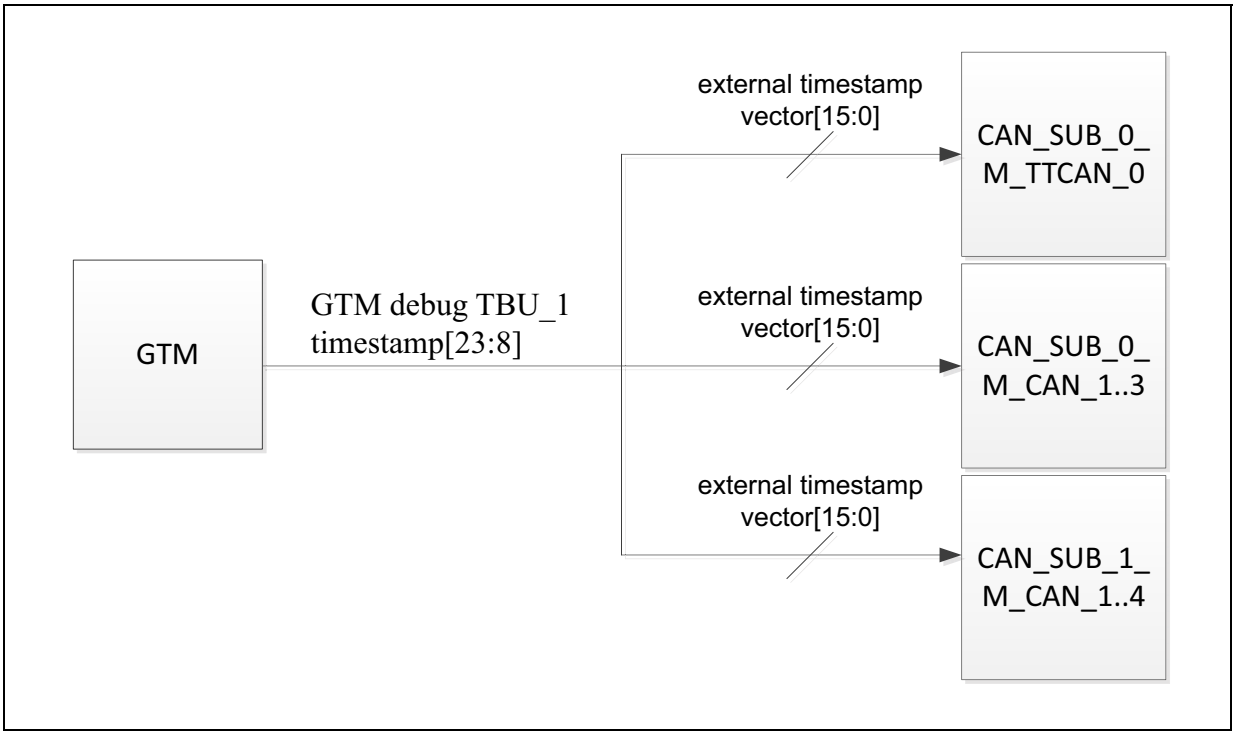
Figure 872. CAN Subsystem_0 M_CAN_1 and M_CAN_2 connections to GP I/Os



52.9 External Time Stamp

The external 16-bit time stamp bus of all M_CAN and M_TTCAN nodes is connected to the upper 16-bit of the 24-bit GTM time base unit TBU_1 as shown in *Figure 873*.

Figure 873. CAN external time base



52.10 External signal description

The CAN subsystem has following external pins.

- M_CAN_x_RX: M_CAN x receive input
- M_CAN_x_TX: M_CAN x transmit output
- M_TTCAN_x_RX: M_TTCAN x receive input
- M_TTCAN_x_TX: M_TTCAN x transmit output
- M_CAN_x_RXFD: M_CAN_x Receive Fast Data control signal
- M_CAN_x_TXFD: M_CAN_x Transmit Fast Data control signal

52.11 Shared memory map

Table 957 shows the shared memory map vs CAN subsystem transmit, receive and filters elements.

Table 957. Shared memory map versus CAN transmit, receive and filters elements

Start Address Offset (byte)	End Address Offset (byte)	CAN block/sub block		Comment
CAN Subsystem_0 - PBRIDGE_2				
0x0000	0x3DFF	CAN_SUB_0_M_TTCAN_0 CAN_SUB_0_M_CAN_1 CAN_SUB_0_M_CAN_2 CAN_SUB_0_M_CAN_3	Standard Filters	This available physical memory space could be configured for Filters/ FIFO/Buffer. This memory space can be flexible, assigned to any of the implemented M_CAN modules.
			Extended Filters	
			Rx FIFO 0	
			Rx FIFO 1	
			Rx Buffer	
			Tx Event FIFO	
			Tx Buffers	
			M_TTCAN triggers	
CAN Subsystem_1 - PBRIDGE_1				
0x0000	0x3DFF	CAN_SUB_1_M_CAN_1 CAN_SUB_1_M_CAN_2 CAN_SUB_1_M_CAN_3 CAN_SUB_1_M_CAN_4	Standard Filters	This available physical memory space could be configured for Filters/ FIFO/Buffer. This memory space can be flexible, assigned to any of the implemented M_CAN modules.
			Extended Filters	
			Rx FIFO 0	
			Rx FIFO 2	
			Rx Buffer	
			Tx Event FIFO	
			Tx Buffers	
			Reserved	

Note: There is no write access protection from a CAN module to the message memory space.

The start address for every memory section is to be configured by the user in the relevant CAN modules registers. However there is no hardware check in the RAM interface for the start address consistency.

However the number of filters, the Tx/Rx buffers and the triggers used, it is mandatory for the user to configure the start address registers, specified in Table 958, in order to fit the memory mapping specified in Table 957.

The CPU uses the peripheral interface to access the memory locations. It must be noted that there is no transfer error reported to the CPU in case of access to reserved memory locations.

Table 958. Detailed mapping in CAN sub-blocks

CAN sub-block	Element name	Comment
Standard Filters	Standard Message ID filter 0 ... Standard Message ID filter 127	address configured in SIDFC [FLSSA]
Extended Filters	Extended Message ID filter 0 ... Extended Message ID filter 63	address configured in XIDFC [FLESA]
Rx FIFO0	Rx FIFO0 element 0 ... Rx FIFO0 element 63	address configured in RXF0C[F0SA]
Rx FIFO1	Rx FIFO1 element 0 ... Rx FIFO1 element 63	address configured in RXF1C[F1SA]
Rx Buffer	Rx Buffer element 0 ... Rx Buffer element 63	address configured in RXBC[RBSA]
Tx Event FIFO	Tx event FIFO element 0 ... Tx event FIFO element 31	address configured in TXEFC[EFSA]
Tx Buffers	Tx Buffer element 0 ... Tx Buffer element 31	address configured in TXBC[TBSA]
M_TTCAN triggers (on M_TTCAN0 only)	Trigger element 0 --- Trigger element 63	address configured in TMC[TMSA]

53 Serial Interprocessor Interface (SIPI)

53.1 Introduction

The Serial Interprocessor Interface (SIPI) is an application layer protocol that runs on top of the LFAST module. It is used by the local device to access the shared memory of a remote device. SIPI defines point-to-point full duplex communication between two devices, where LFAST works as a physical medium of communication between both the devices.

53.1.1 Scalability

The SIPI protocol is designed to provide a sophisticated, high bandwidth, multi-master, multi-channel memory interface between two devices using few interconnecting signals.

Main scalable features:

- Number of concurrent channels:
  - Full Implementation = 4
  - Minimum Implementation = 1
- Full implementation has a node both as Initiator and Target of commands. Minimum Implementation either Initiator or Target.
- Full Implementation includes a block transfer feature, but this feature is optional.

The rest of this section describes a full implementation of SIPI, which includes:

- Advanced High-Performance Bus (AHB-Lite) master interface
- Direct Memory Access (DMA) interface
- LFAST Tx/Rx (transmit/receive) interfaces along with Peripheral Bus Interface (IPS)

53.2 Overview

An instance of SIPI can act as initiator, or target or both. SIPI can access the shared memory directly through its AHB master interface or through its DMA interface. DMA interface is used when the node acts as an initiator while the AHB Master interface will be used when the node acts as target. SIPI has four channels, with one channel (Channel 2) having data streaming capability. Payload width for channel 0, 1 and 3 is 32 bits, and for channel 2 data widths can be 32 bits, or 256 bits when streaming. Any of these channels can be used for DMA access or bus interface access depending on the setting of the SIPI_CCR*n*. CRC encoder calculates the CRC on the command frame. Then the SIPI initiator appends the CRC to the end of the frame before transmission.

Figure 874. Interprocessor communication diagram

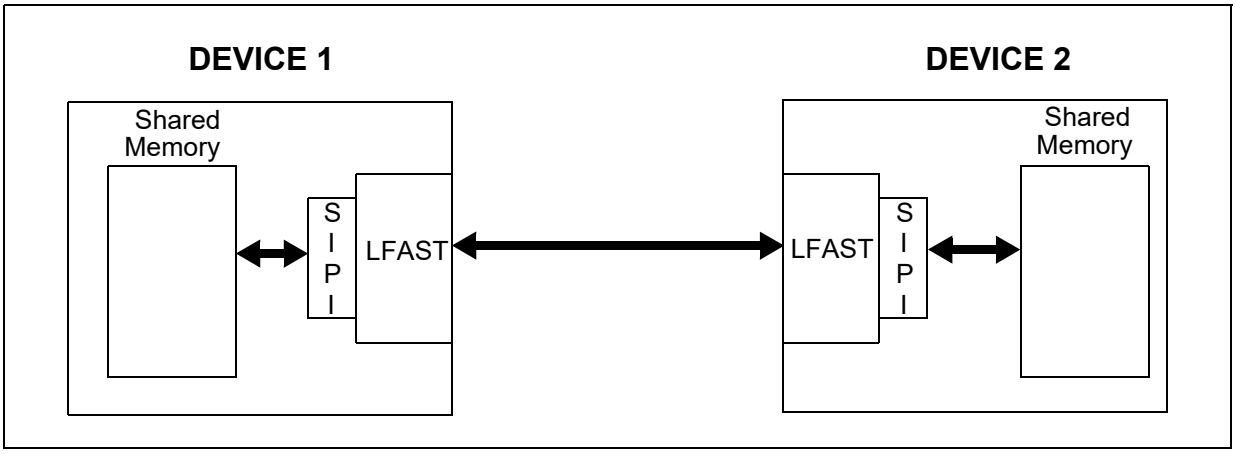
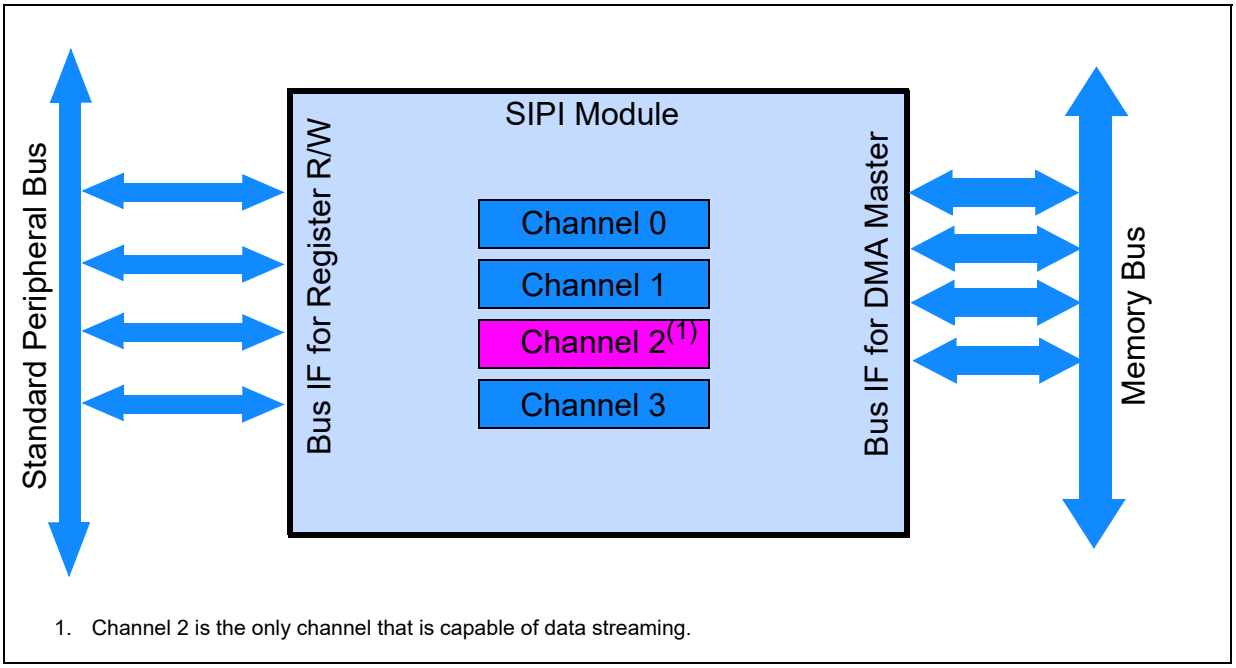


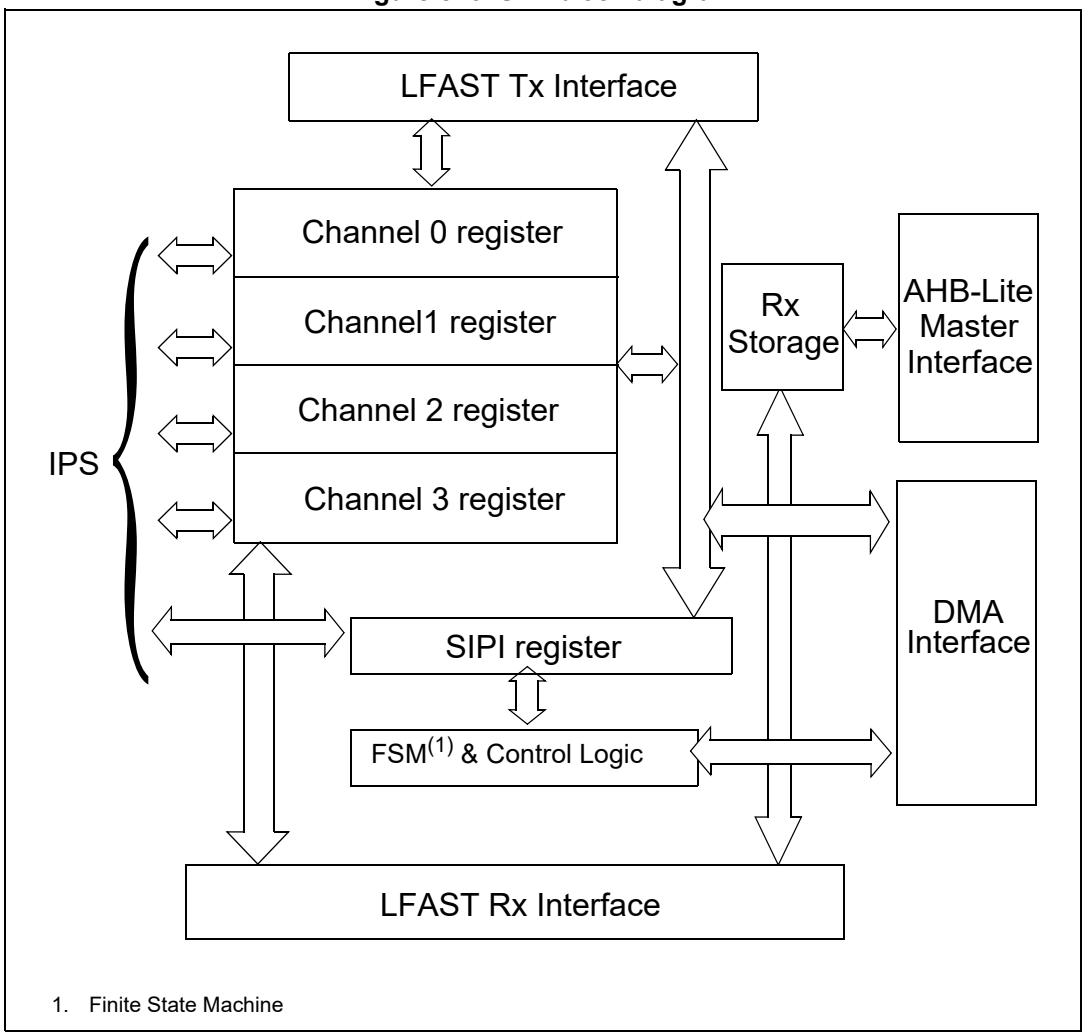
Figure 875. SIPI module



1. Channel 2 is the only channel that is capable of data streaming.

53.3 SIPI block diagram

Figure 876. SIPI block diagram



1. Finite State Machine

53.4 Feature description

This section describes the features of the SIPI module.

53.4.1 Main features

- Point-to-point communication between two devices
- Full duplex communication
- Four channels, including one channel with data streaming capability

- Configurable DMA access for each channel
- CRC protection mechanism
- Timeout protection mechanism
- Fixed priority channel selection
- Data size up to 256 bits on streaming channel
- Common tag pool for assigning sequential transfer IDs to every new transfer
- AHB master interface that is used by target node to access shared memory
- Up to two outstanding requests are supported at initiator

53.4.2 Standard features

- IPS bus interface
- SPP DMA2x bus interface
- AHB master interface
- LFAST Tx/Rx interfaces
- Cyclic Redundancy Check error detection (CRC16)

53.5 SIPI operation from reset

When the SIPI module exits reset, it is operational when SIPI_MCR[MOEN] = 1 and target mode SIPI_MCR[TEN]=1, without the need to configure more control registers.

53.6 Functional description

53.6.1 External signals

The SIPI has no chip external signals.

53.6.2 Frame format

All frames have the same general format:

- 16-bit header
- Address, address and data, or nothing
- 16-bit CRC

There are 2 main groups of command; read and write. Within those 2 groups are three read/write formats:

- 32-bit
- 16-bit
- 8-bit

Each command generates one of three different responses:

- Read data
- Write acknowledge
- Error

There is one additional command that requests the module ID from the Target node. The ID is a unique 32-bit ID that is specific to a particular device. It is normally the same as the JTAG ID.

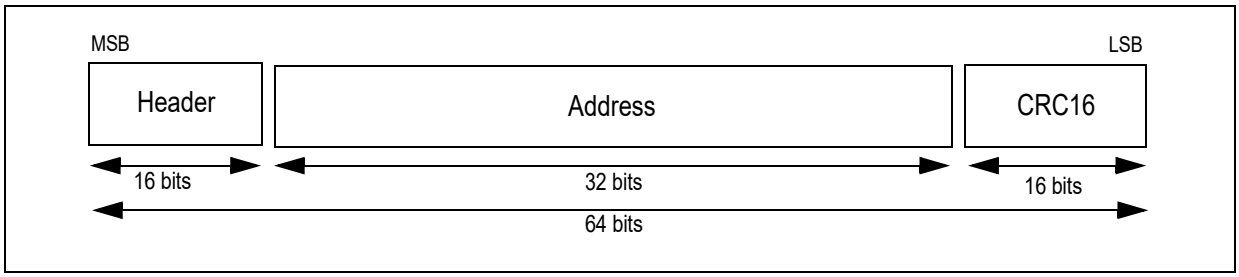
The sections below illustrate the four different frame formats that are used to implement all the commands and responses.

The number of frames depends on the data buffers (associated with every channel), the frame data is stored in data buffers at the initiator. Address and data are both transmitted in the same order in which they are stored.

Note: SIPI module architecture is always big endian but the storage in host memory depends on endianness architecture of host memory.

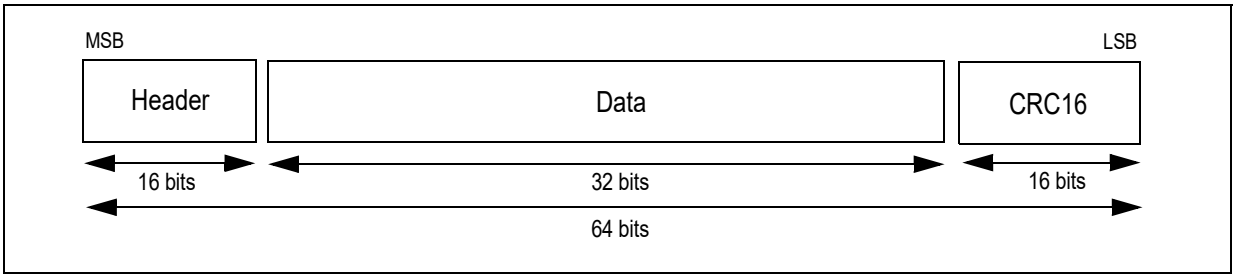
53.6.2.1 Register read request

Figure 877. SIPI register read request



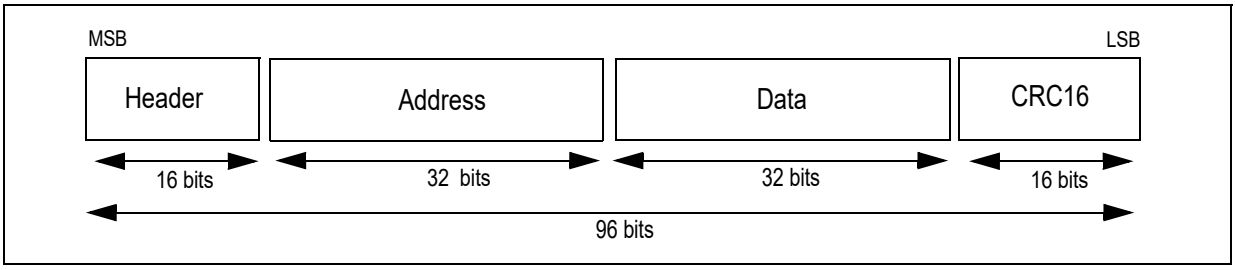
53.6.2.2 Register read response, ID request response

Figure 878. SIPI read response



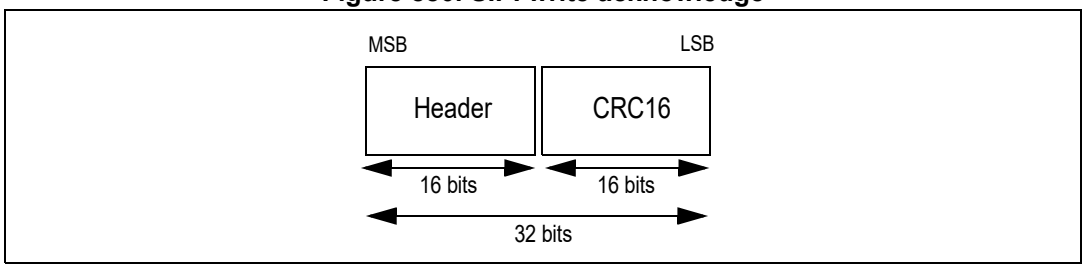
53.6.2.3 Register write request

Figure 879. SIPI register write request



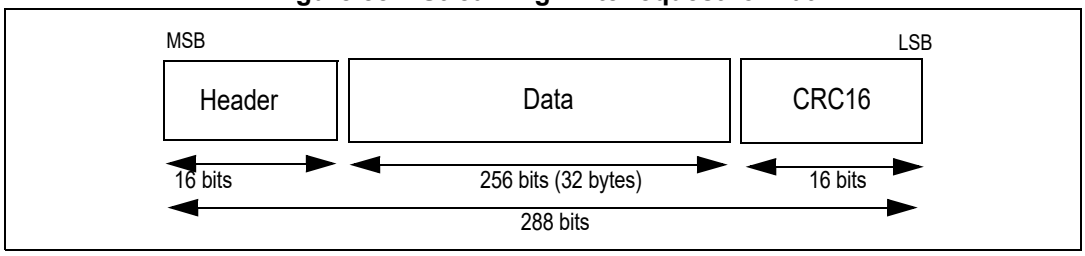
53.6.2.4 Trigger transfer, ID transfer, write acknowledge and streaming write acknowledge

Figure 880. SIPI write acknowledge



53.6.2.5 Streaming write request

Figure 881. Streaming write request format



Note: Direct write operations (IPS) are used to set the streaming address (refer to Figure 889: Streaming transfer with data in Section 53.7.3.2.1: Streaming write transfer with address).

Streaming data write is performed using the format shown in Figure 881.

53.6.2.6 Header field

Header field contains 16 bits of configuration information. MSB will be transmitted first.

53.6.2.6.1 SIPI header coding

Figure 882, *Table 959*, *Figure 883*, and *Table 960* show how the SIPI header bits are coded.

Figure 882. SIPI header coding

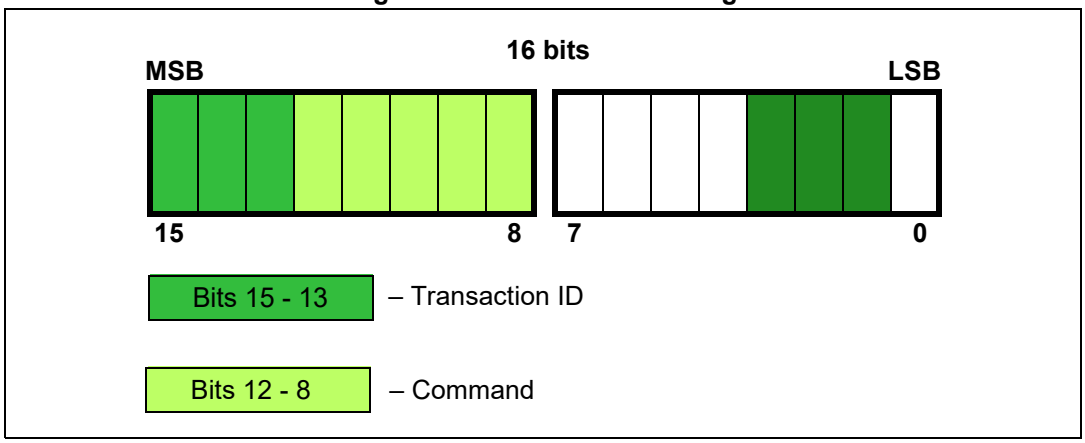


Table 959 shows the command coding for the bits[12:8] of the header.

Table 959. SIPI header command coding

b[12:8] (hex)	Command	Payload Size
0x00	Read 8 bits	64
0x01	Read 16 bits	64
0x02	Read 32 Bits	64
0x03	Reserved	—
0x04	Write 8 bits with ACK	96
0x05	Write 16 bits with ACK	96
0x06	Write 32 bits with ACK	96
0x07	Reserved	—
0x08	ACK – OK	32
0x09	ACK – Fault	32
0x0A	Read Answer – OK	64
0x0B	Reserved	—
0x0C	Trigger comm with ACK	32
0x0D	Reserved	—
…	…	…
0x11	Reserved	—
0x12	ID Register Read Request	32
0x13	Reserved	—
…	…	…
0x16	Reserved	—
0x17	Stream Data with ACK (32 bytes)	288
0x18	Reserved	—

Table 959. SIPI header command coding (continued)

b[12:8] (hex)	Command	Payload Size
…	…	...
0x1F	Reserved	—

Figure 883 shows the content of bits[7:0] in the SIPI header.

Figure 883. SIPI header channel field

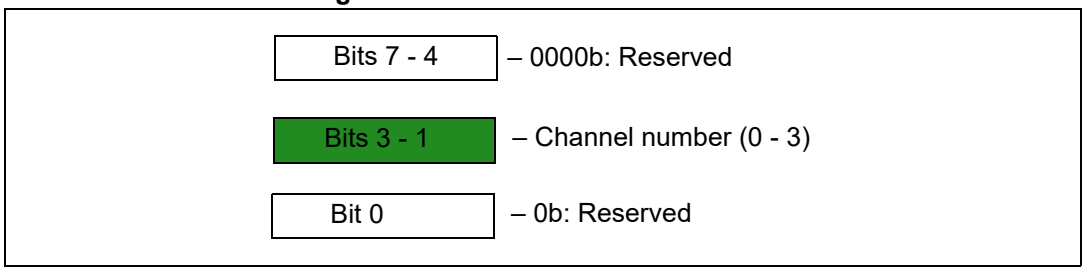


Table 960 shows the channel number coding.

Table 960. SIPI Header Channel Number Coding

SIPI channel name	Channel number coding in header(s)		Comment
	SIPI channel coding select bit		
	Code table I (1)	Code table II (0)	
0 (Channel A)	000b	100b	In use
1 (Channel B)	001b	101b	In use
2 (Channel C)	010b	110b	In use
3 (Channel D)	011b	111b	In use
4 (Channel E)	100b	000b	Reserved
5 (Channel F)	101b	001b	Reserved
6 (Channel G)	110b	010b	Reserved
7 (Channel H)	111b	011b	Reserved

53.6.2.7 Address field

The address field is 32 bits wide with the MSB transmitted first.

53.6.2.8 Payload field

Table 961 shows the payload sizes of LFAST frames.

Table 961. Payload size of LFAST channel frame

LFAST code for payload size	SIPI Code	LFAST payload size (bits)(1)	LFAST payload size (bytes)
000b	—	8	1
001b	—	32	4
010b	010b	64	8
011b	011b	96	12
100b	100b	128	16
101b	101b	256	32
111b	111b	288	36

1. Only 32, 64, 96 and 288 bits sizes are used for SIPI.

Table 962 shows the coding of LFAST channel for a given SIPI channel code.

Table 962. Converted coding of LFAST channel code for SIPI headers

LFAST channel code	SIPI channel code	Data channel(1)
0100b	**100b**	**Data channel A**
0101b	**101b**	**Data channel B**
0110b	**110b**	**Data channel C**
0111b	**111b**	**Data channel D**
1000b	**000b**	**Data channel E (not used by SIPI)**
1001b	**001b**	**Data channel F (not used by SIPI)**
1010b	**010b**	**Data channel G (not used by SIPI)**
1011b	**011b**	**Data channel H (not used by SIPI)**

1. SIPI channel 0 sends all commands on LFAST channel A, commands received on LFAST channel A, are processed and the response sent back on LFAST channel A.
SIPI channel 1 sends all commands on LFAST channel B, commands received on LFAST channel B, are processed and the response sent back on LFAST channel B.
SIPI channel 2 sends all commands on LFAST channel C, commands received on LFAST channel C, are processed and the response sent back on LFAST channel C.
SIPI channel 3 sends all commands on LFAST channel D, commands received on LFAST channel D, are processed and the response sent back on LFAST channel D.

53.6.2.9 CRC field

CRC field is 16 bits wide with calculation always enabled using CRC-16-CCITT syndrome (x16 + x12 + x5 + 1). The data stream is transmitted with the MSB sent first.

53.6.2.9.1 CRC examples

Example 1 – 32-bit write on channel 1 with ID1

- Header = 0x260A
- Address = 0x1122_3344
- Data = 0xCCDD_EEFF
- CRC = 0xBF7D

Example 2 – 32-bit read on channel 2 with ID2

- Header = 0x420C
- Address = 0x89AB_CDEF
- CRC = 0x6B80

Example 3 – Event command on channel 3 with ID3

- Header = 0x6C0E
- CRC = 0xB286

53.7 Transfer types

This section describes the available transfer types of the SIPI module. The SIPI frame is inserted inside the payload of the LFAST frame as shown in the figures of the following examples.

53.7.1 Read transfer

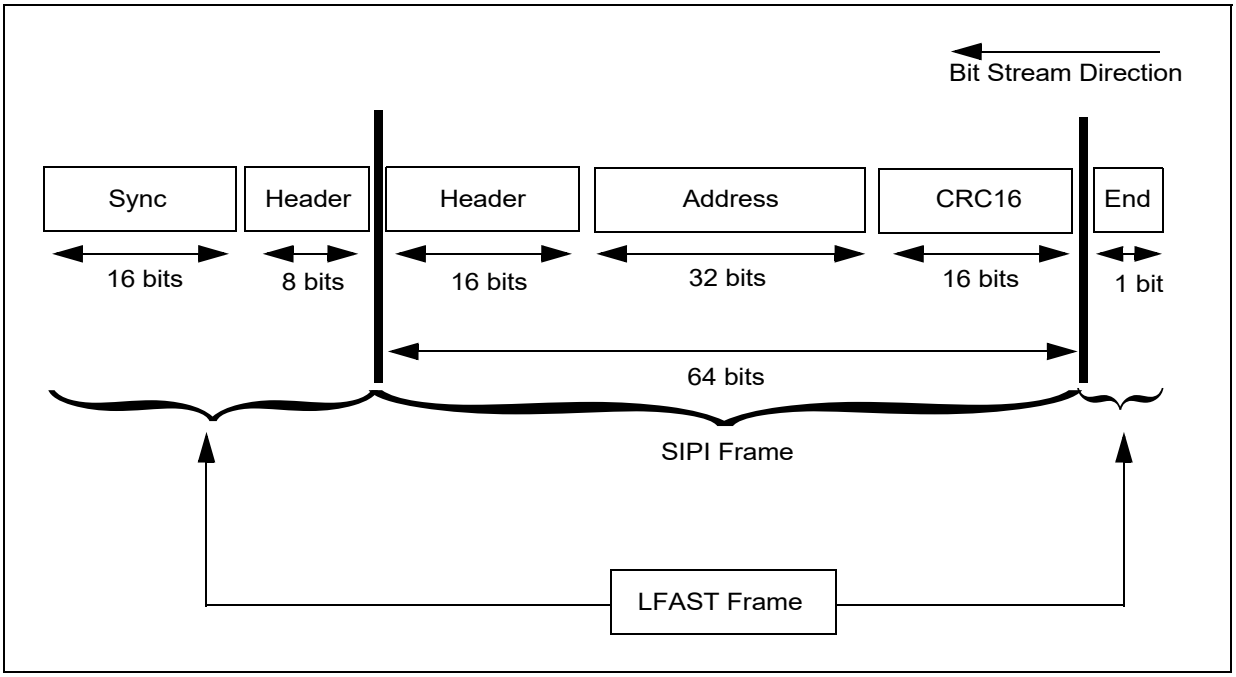
A Read transfer can be of two types:

- Read request transfer (at initiator node)
- Read response transfer (at target node)

53.7.1.1 Register read request transfer

If there is a read request transfer, the initiator node will send header, address and CRC bits as shown in *Figure 884*.

Figure 884. Read request transfer (showing LFAST frame encapsulation)



53.7.2 Register read answer transfer

In response to a read request by the initiator node, the target node will send header, payload and CRC16. Data transfer could be in 8-bit, 16-bit or 32-bit modes (refer to *Figure 885*). In the case of 8-bit or 16-bit modes, copies of the SIPI data is sent in the payload (refer to *Figure 886*).

Figure 885. Read answer transfer (LFAST frame encapsulation is not shown)

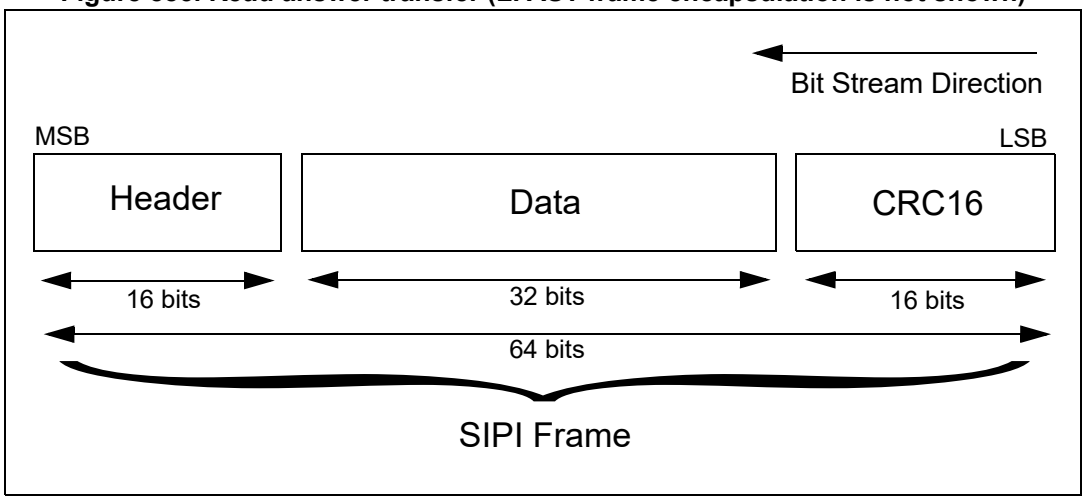
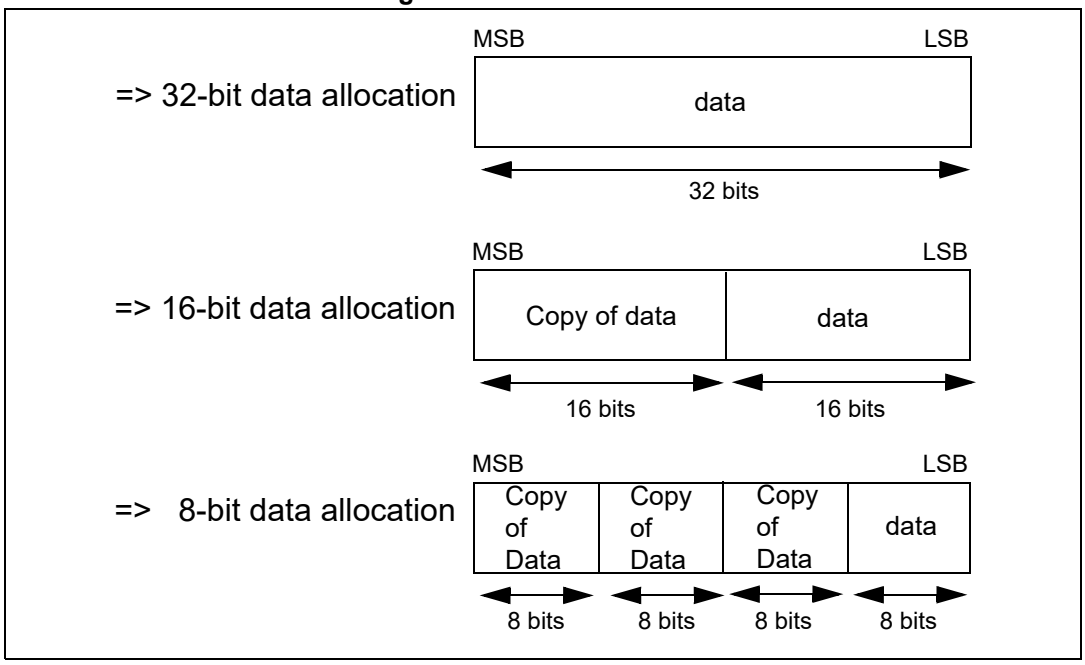


Figure 886. Data allocation



Note: For an 8-bit transfer, address bits [31:2] are used as address and bits [1:0] are used as byte enables (big endian):

Bits[1:0]:

- *00: Byte 3 enabled (MSB)*
- *01: Byte 2 enabled*
- *10: Byte 1 enabled*
- *11: Byte 0 enabled (LSB)*

For a 16-bit transfer, address bits [31:1] are used as address and bit [0] is used as half word enable (big endian).

Bit[0]:

- *0: Half-word 1 enabled (MSB)*
- *1: Half-word 0 enabled (LSB)*

Note: Byte Ordering depends from address written in CAR and transfer width:
Example:

- *CDR content: 0x12345678*
- *CAR content: see table columns*

Table 963. Target Memory content dependent from CAR content

	Target Memory Byte Address	CAR = 0x02040	CAR = 0x02041	CAR = 0x02042	CAR = 0x02043
32-bit transfer	0x02040	12	12	12	12
	0x02041	34	34	34	34
	0x02042	56	56	56	56
	0x02043	78	78	78	78

Table 963. Target Memory content dependent from CAR content (continued)

	Target Memory Byte Address	CAR = 0x02040	CAR = 0x02041	CAR = 0x02042	CAR = 0x02043
16-bit transfer	0x02040	12	56	xx	xx
	0x02041	34	78	xx	xx
	0x02042	xx	xx	12	56
	0x02043	xx	xx	34	78
8-bit transfer	0x02040	12	xx	xx	xx
	0x02041	xx	34	xx	xx
	0x02042	xx	xx	56	xx
	0x02043	xx	xx	xx	78

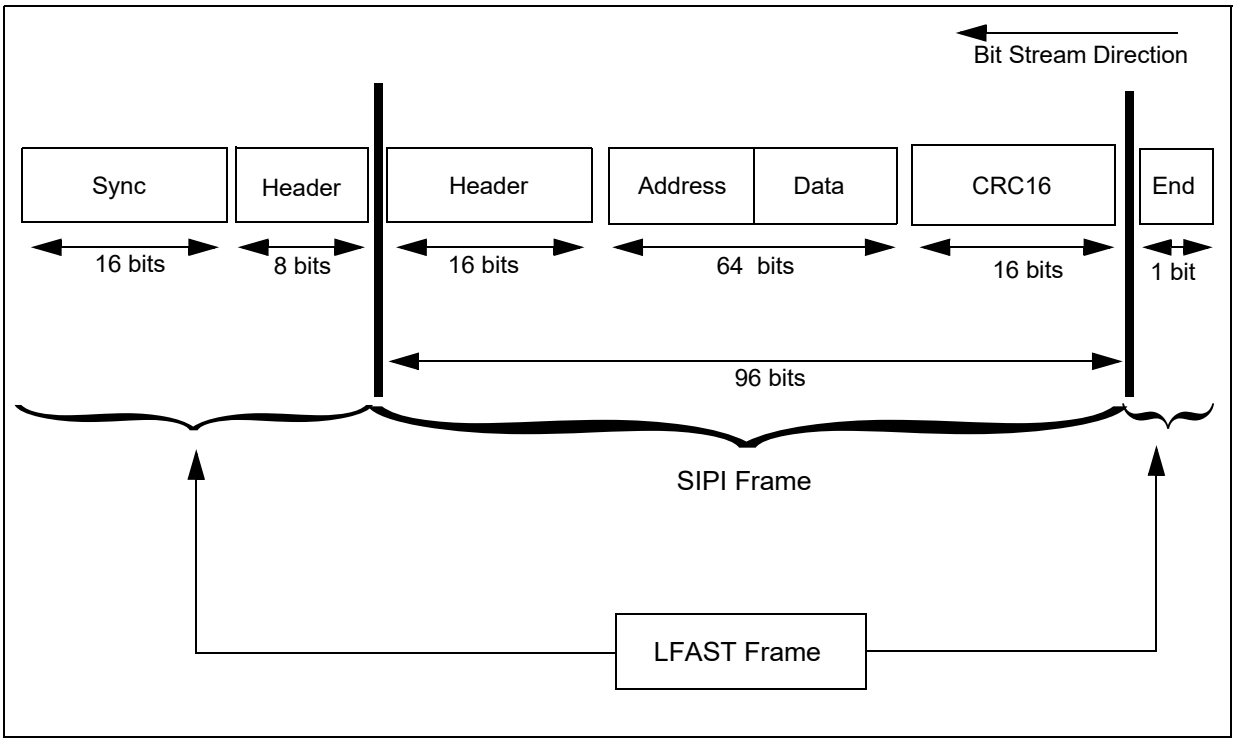
53.7.3 Register Write transfer

Register Write transfer can be of two types:

- Normal Write transfer – channels 0, 1, 2 and 3
- Streaming Data transfer – channel 2 only

A Register Write transfer can be done through Normal Write transfer - channels 0, 1, 2 and 3 (refer to *Section 53.7.3.1: Normal write transfer*) as shown in *Figure 887*.

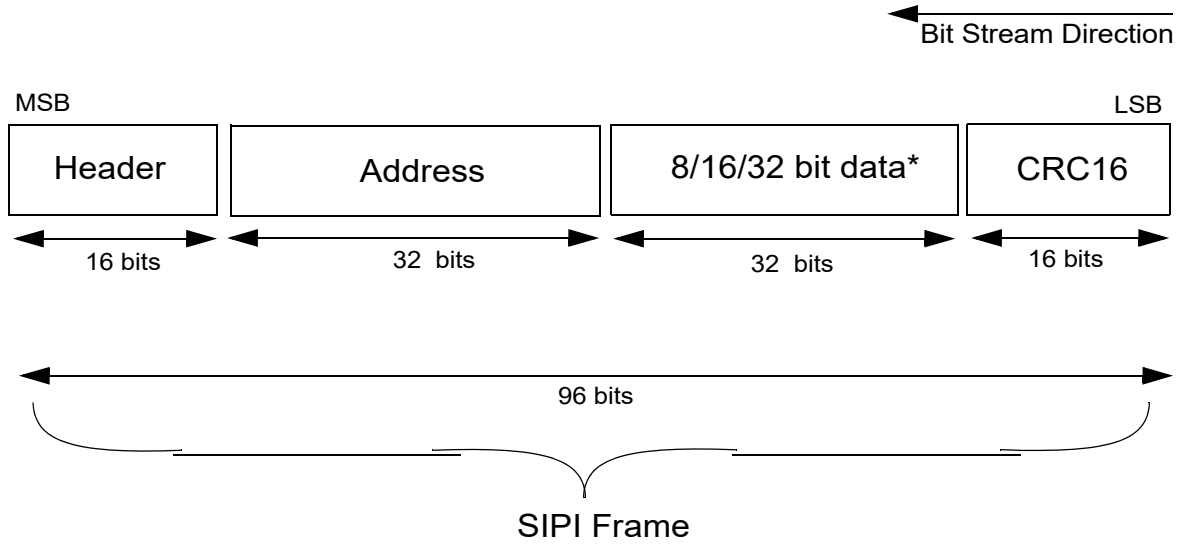
Figure 887. Register write transfer (showing LFAST frame encapsulation)



53.7.3.1 Normal write transfer

A normal write transfer contains header, address, data (32-bit) and 16-bit CRC as shown in *Figure 888*.

Figure 888. Write transfer (LFAST frame encapsulation is not shown)



Note: In the case of an 8-bit transfer, address bits [31:2] are used as address and bits [1:0] are used as a byte enable:

Bits[1:0]:

- *00 – byte 3 enabled (MSB)*
- *01 – byte 2 enabled*
- *10 – byte 1 enabled*
- *11 – byte 0 enabled (LSB)*

In the case of a 16-bit transfer, address bits [31:1] are used as address and bit [0] is used as a half-word enable:

Bit[0]:

- *0 – half-word 1 enabled (MSB)*
- *1 – half-word 0 enabled (LSB)*

53.7.3.2 Streaming write transfer

Streaming write transfers have 256 bits of payload.

53.7.3.2.1 Streaming write transfer with address

The related registers need to be set on target side using direct register write (32-bit).

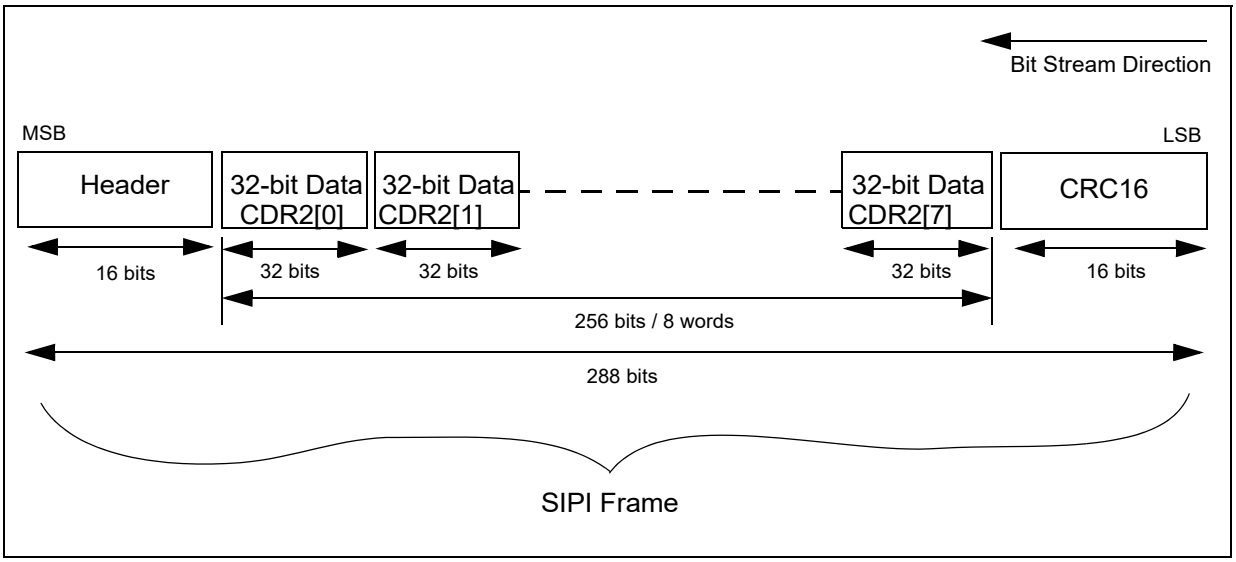
Setting the address is performed using a direct register write. SIPI_MAXCR and SIPI_ARR are written before the SIPI_ACR. This is to ensure that the unwanted behavior of the SIPI

trying to access non-shared memory is avoided. This is only true for the very first streaming command, subsequent streaming command(s) may not have the need to write SIPI_MAXCR and SIPI_ARR (refer to *Section 53.15.1.10: SIPI Max Count Register (SIPI_MAXCR)*, *Section 53.15.1.12: SIPI Address Count Register (SIPI_ACR)*, and *Section 53.15.1.11: SIPI Address Reload Register (SIPI_ARR)*.

53.7.3.2.2 Streaming transfer with data

Figure 889 shows the packet structure of streaming transfers that contain data.

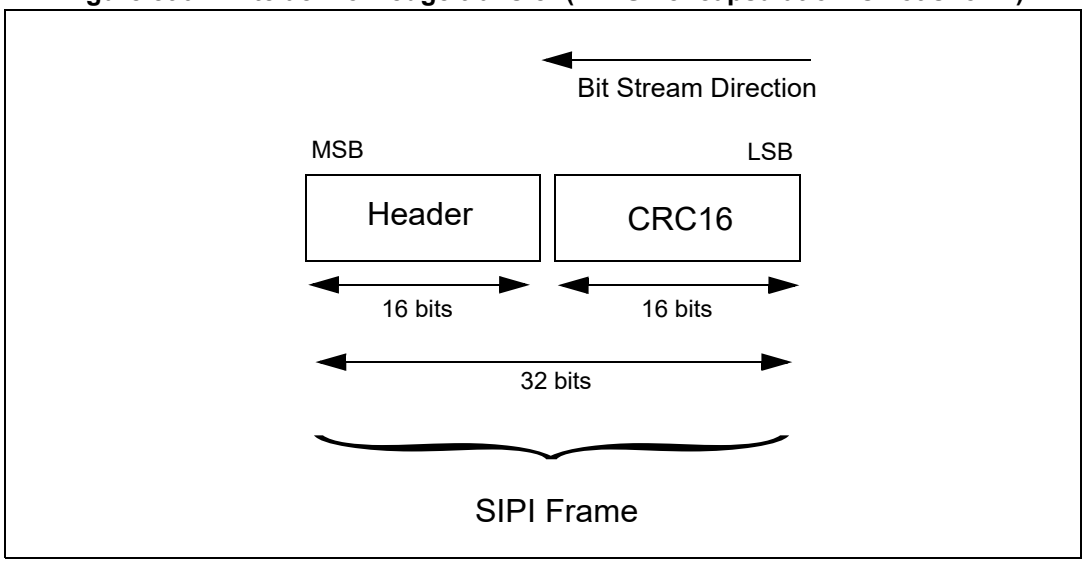
Figure 889. Streaming transfer with data



53.7.4 Write acknowledge transfer

A write acknowledge transfer contains only header and CRC bits (refer to *Figure 890*). The CRC bits are calculated on the header field.

Figure 890. Write acknowledge transfer (LFAST encapsulation is not shown)

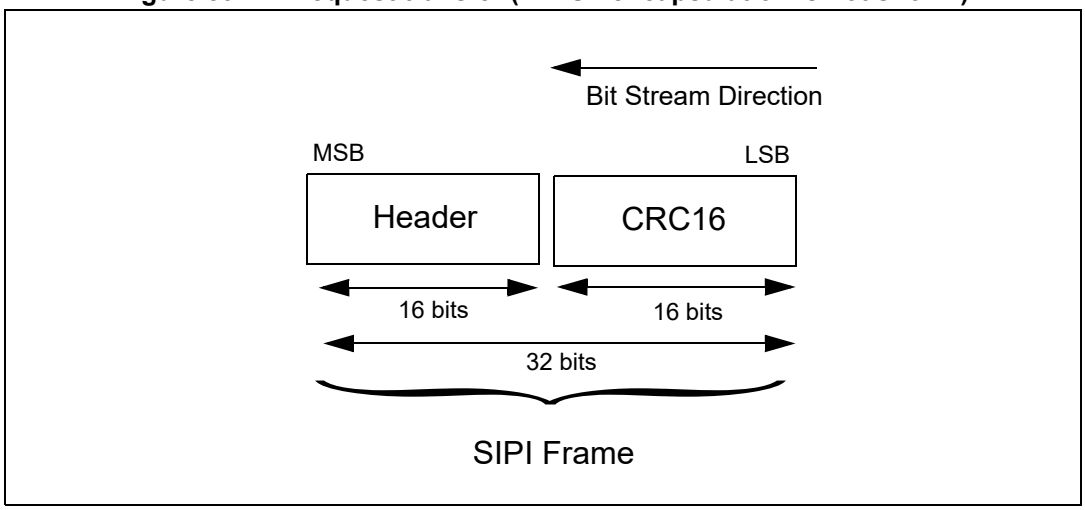


53.7.5 ID request response

53.7.5.1 ID request transfer

An ID request is transmitted by the initiator node as shown in *Figure 891*.

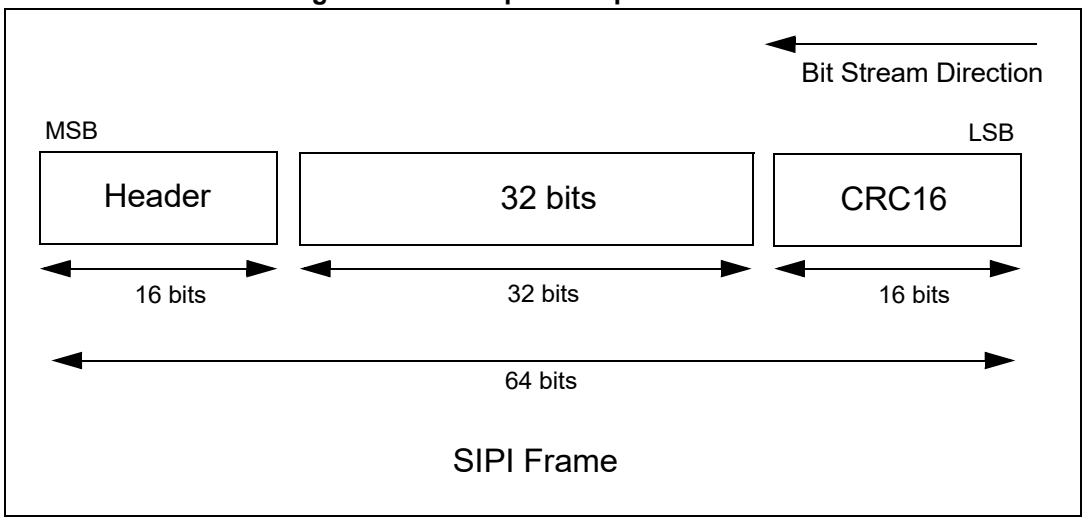
Figure 891. ID request transfer (LFAST encapsulation is not shown)



53.7.5.2 ID request response transfer

An ID request response is transmitted by the target node as shown in *Figure 892*.

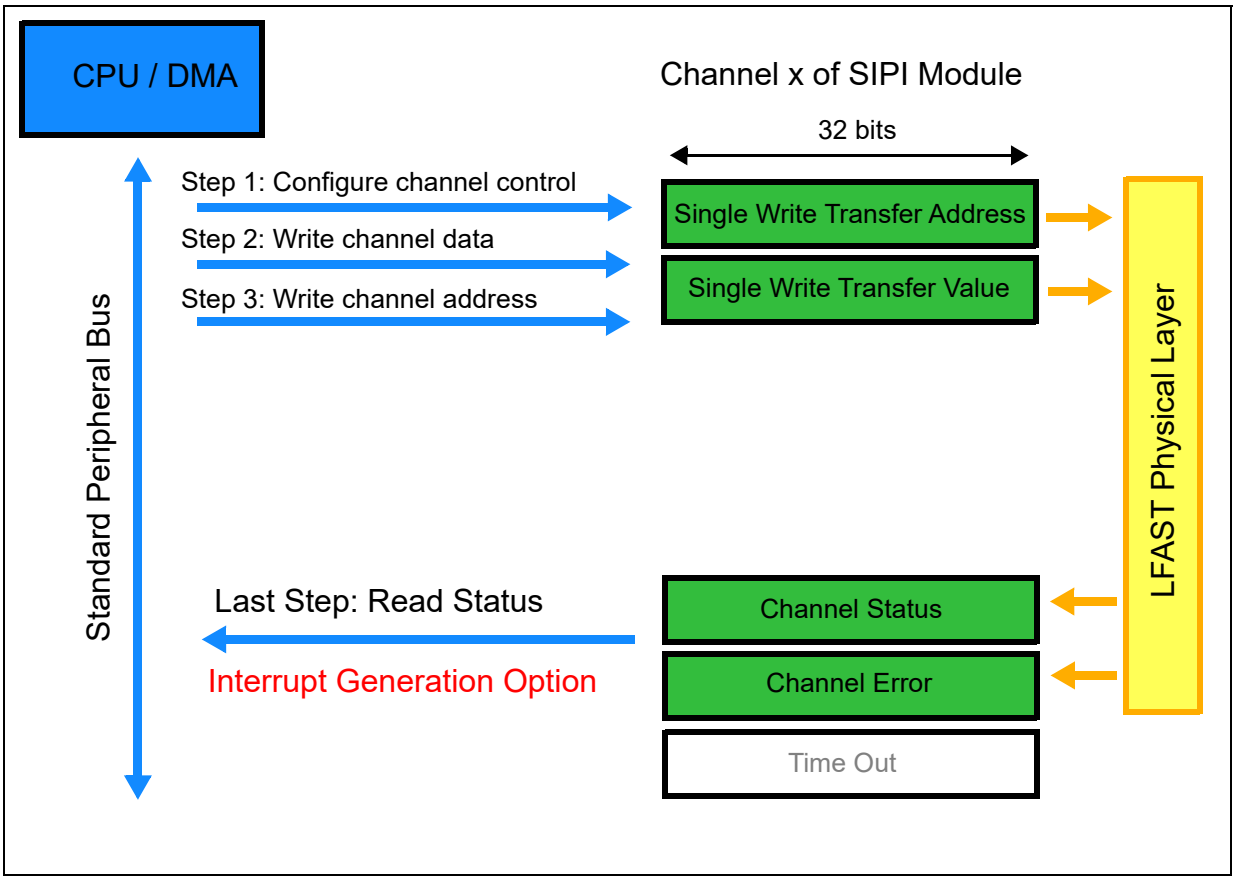
Figure 892. ID request response transfer



Note: The ID request response contains the value of the CHIP ID in place of data.

53.8 Transfer API and flow charts

Figure 893. SIPI single register write API



For a single write transfer request generation (refer to *Figure 893*):

1. Configure SIPI_CCR*n* at the initiator node.
2. Write data in SIPI_CDR*n* at the initiator node.
3. Configure SIPI_CAR*n* at the initiator node.
   As soon as the channel address register (SIPI_CAR*n*) is written and if SIPI_CCR*n*[CHEN] = 1, the initiator SIPI will calculate the CRC on header, address and data field and will start transmitting data to LFAST.
4. Software polls the status register bits (CSR*n*) to determine when the request completes, SIPI_CSR*n*[ACKR] = 1. An interrupt will be generated if the corresponding SIPI_CIR*n*[ACKIE] = 1.

On a single write transfer request reception (refer to *Figure 894*):

1. Target node will place the address, data and control information on its AHB master interface.
2. When the process in completed, the target node will generate an acknowledge frame and send it back to the LFAST.

Figure 894. SIPI single register write API – Flow chart

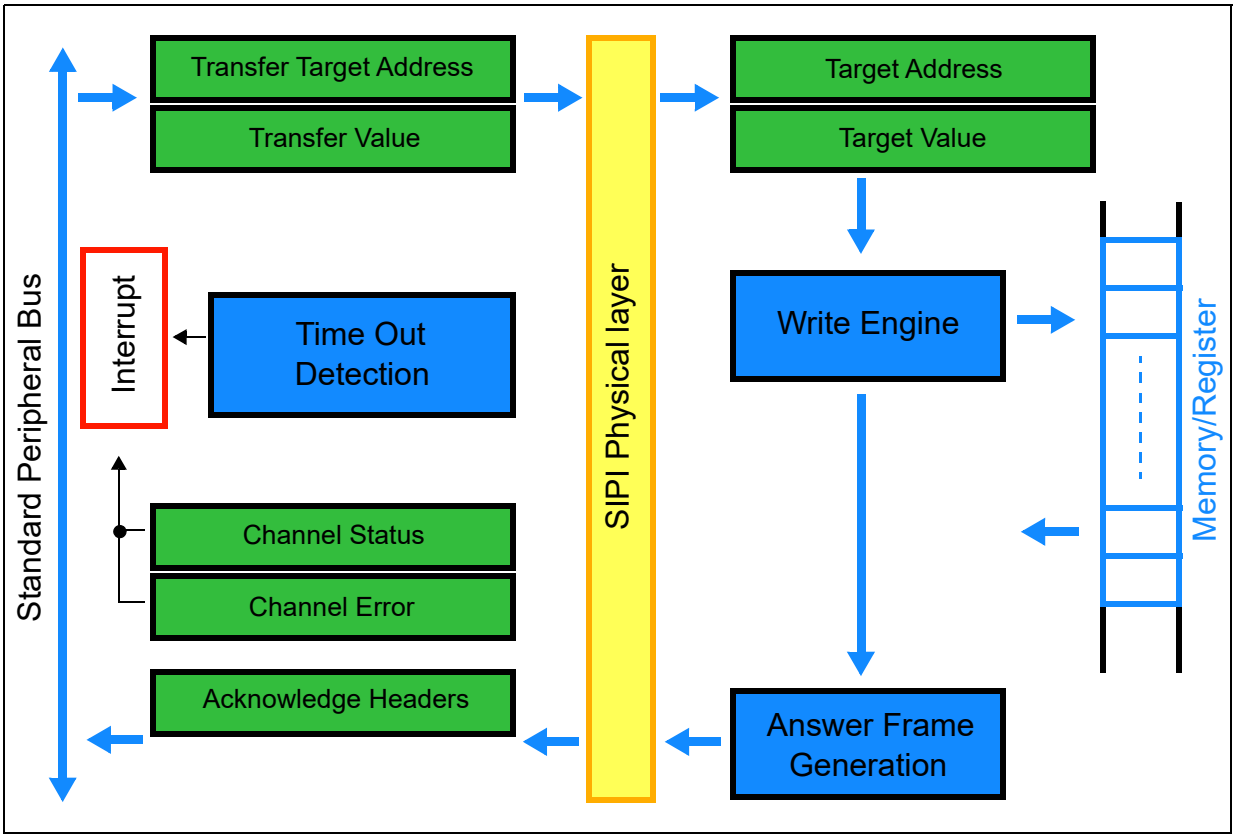
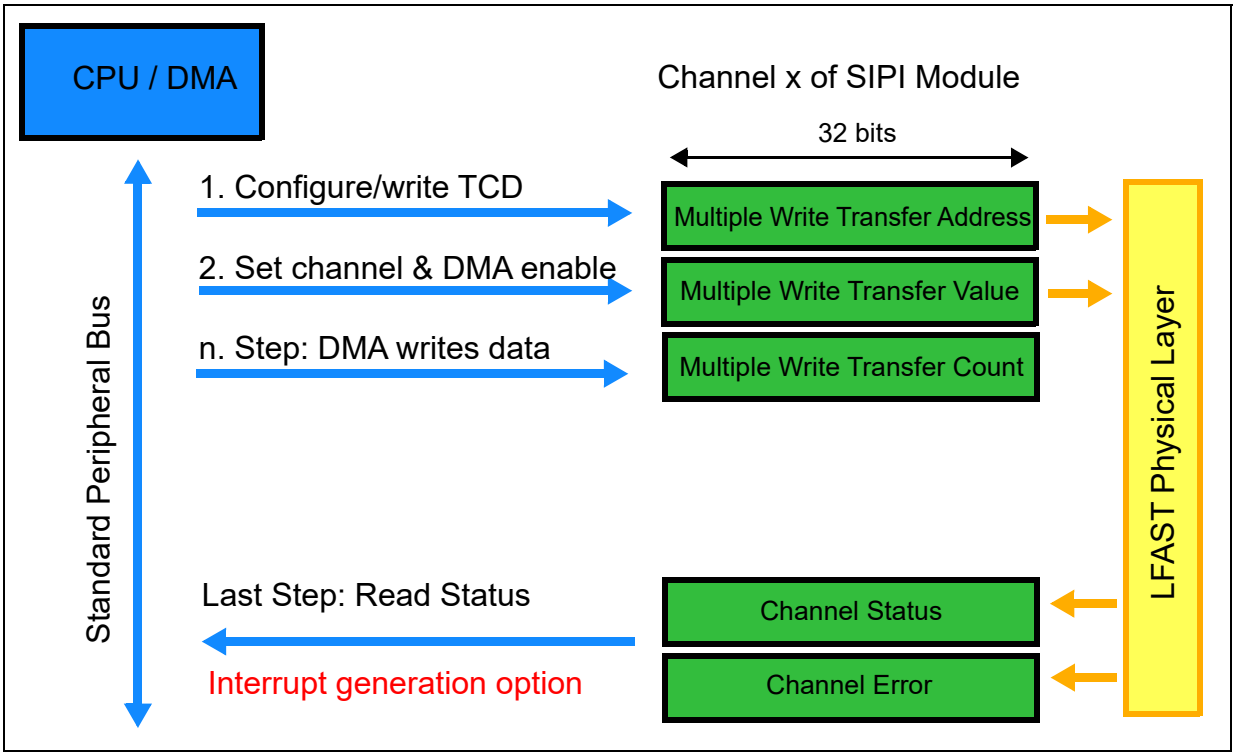


Figure 895. SIPI multiple register write API



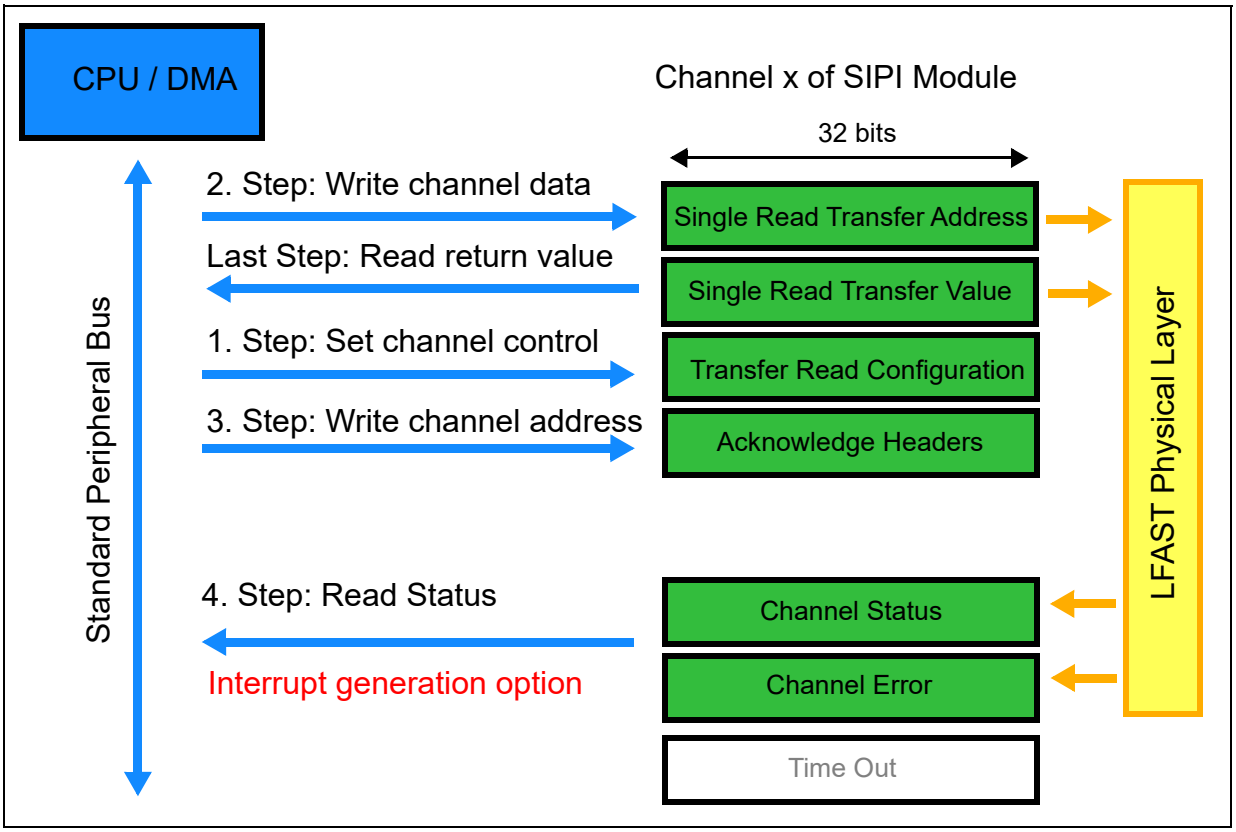
For multiple write transfer request (including streaming) generation (refer to *Figure 895*):

1. Software will configure the Transfer Control Descriptor (TCD) of the DMA.
2. Software will write SIPI_CCR*n*[CHEN] = 1 and SIPI_CCR*n*[DEN] = 1.
3. SIPI will generate request (DMA Request) and send it to the DMA controller.
4. SIPI will start copying data into the SIPI_CDR*n* through its DMA interface, depending on the transfer count and data registers size.
5. When the copying process is completed, initiator SIPI will calculate CRC on header, address and data field and start transmitting data to LFAST after SIPI_CARx (SIPI_CAR2 is valid for streaming mode only) is written.
6. Software should poll the SIPI_CSR*n* status register bits to determine if the request has completed. If SIPI_CSR*n*[ACKR] = 1, an interrupt will be generated (if the corresponding SIPI_CIR*n*[ACKIE] = 1).
7. If SIPI_CCR*n*[DEN] = 1, the SIPI request will generate a request and send it to the DMA controller. If not, the request will not be sent and the state machine will enter idle mode.
8. Steps 4 to 7 will be repeated.

On multiple write transfer request reception (refer to *Figure 895*):

1. Target node will place the address, data and control information on its AHB Master Interface.
2. When the process is completed, target node will generate an acknowledge frame and send it back to the LFAST.

Figure 896. SIPI single register read API



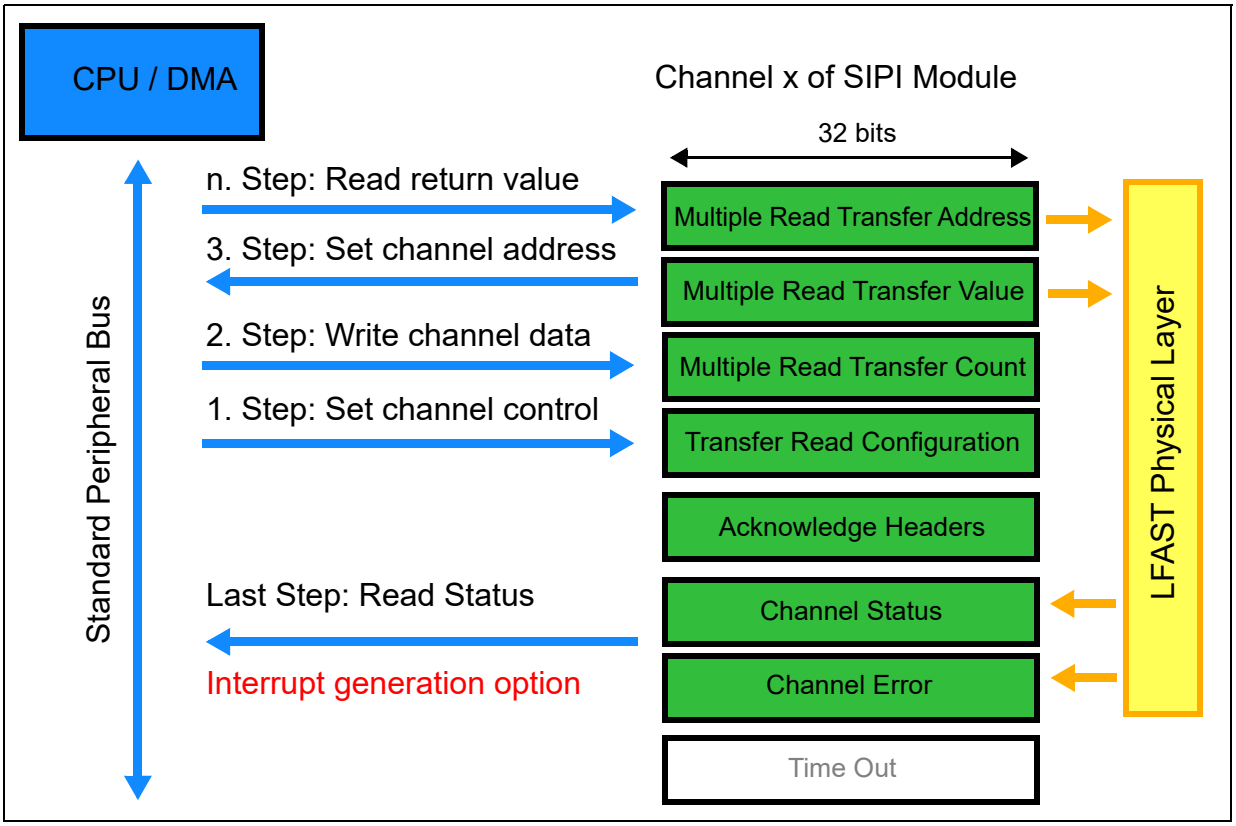
For single read transfer request generation (refer to *Figure 896*):

1. Software/DMA will configure SIPI_CCR*n,* with the last step by writing SIPI_CAR*n*.
2. As soon as SIPI_CAR is written, the initiator SIPI will calculate CRC of the header and address fields and start transmitting data to the LFAST.
3. Software should poll SIPI_CSR*n* to determine when the request has completed. If SIPI_CSR*n*[RAR] = 1, and interrupt will be generated (if SIPI_CIR*n*[RAIE] = 1). If SIPI_CSR*n*[RAR] does not set then SIPI_ERR[TOE*n*] = 1, which indicates a timeout has occurred.
4. If SIPI_CSR*n*[RAR] = 1, then software can read the data register, or can take other necessary action if SIPI_CSR*n*[RAR] = 0.

On single read transfer request reception (refer to *Figure 896*):

1. Target node will place the address and control information on its AHB Master Interface.
2. When the process in completed, target node will send read response back to LFAST.

Figure 897. SIPI multiple register read API



For multiple read transfer request generation (refer to *Figure 897*):

1. Software/DMA will configure SIPI_CCR*n,* with the last step by writing SIPI_CAR*n*.
2. As soon as channel address register is written, initiator SIPI will calculate CRC on the header and address fields, then start transmitting data to the LFAST.
3. Software should poll SIPI_CSR*n* to determine when the request has completed. If SIPI_CSR*n*[RAR] = 1 and SIPI_CIR*n*[RAIE] = 1, an interrupt will be generated. If SIPI_CSR*n*[RAR] does not set, then SIPI_ERR[TOE*n*] = 1, which indicates a timeout.
4. Steps 2 to 3 will be repeated for multiple requests.

On multiple read transfer request reception (refer to *Figure 897*):

1. Target node then will start reading data through its AHB interface.
2. When the transfer is completed/all data registers are full, it will calculate CRC and send the response frame back to LFAST.

Figure 898, *Figure 899*, *Figure 900* and *Figure 901* show the SIPI data streaming of data.

Figure 898. SIPI data stream model

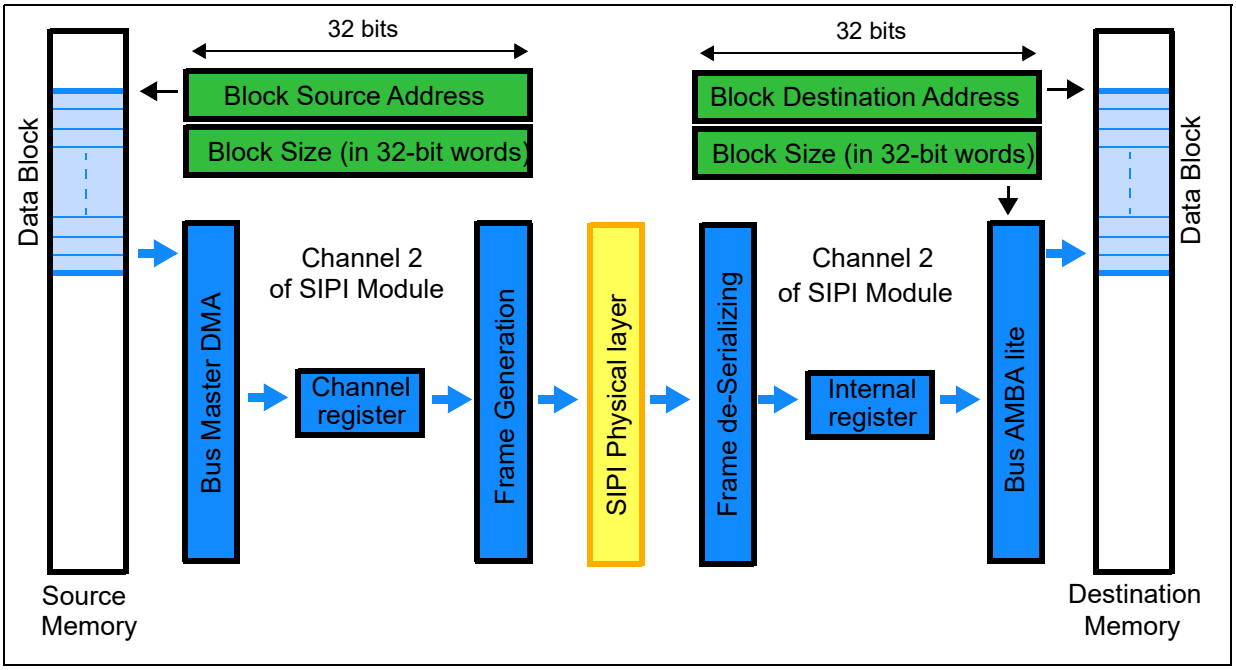


Figure 899. SIPI data stream model - Initiator

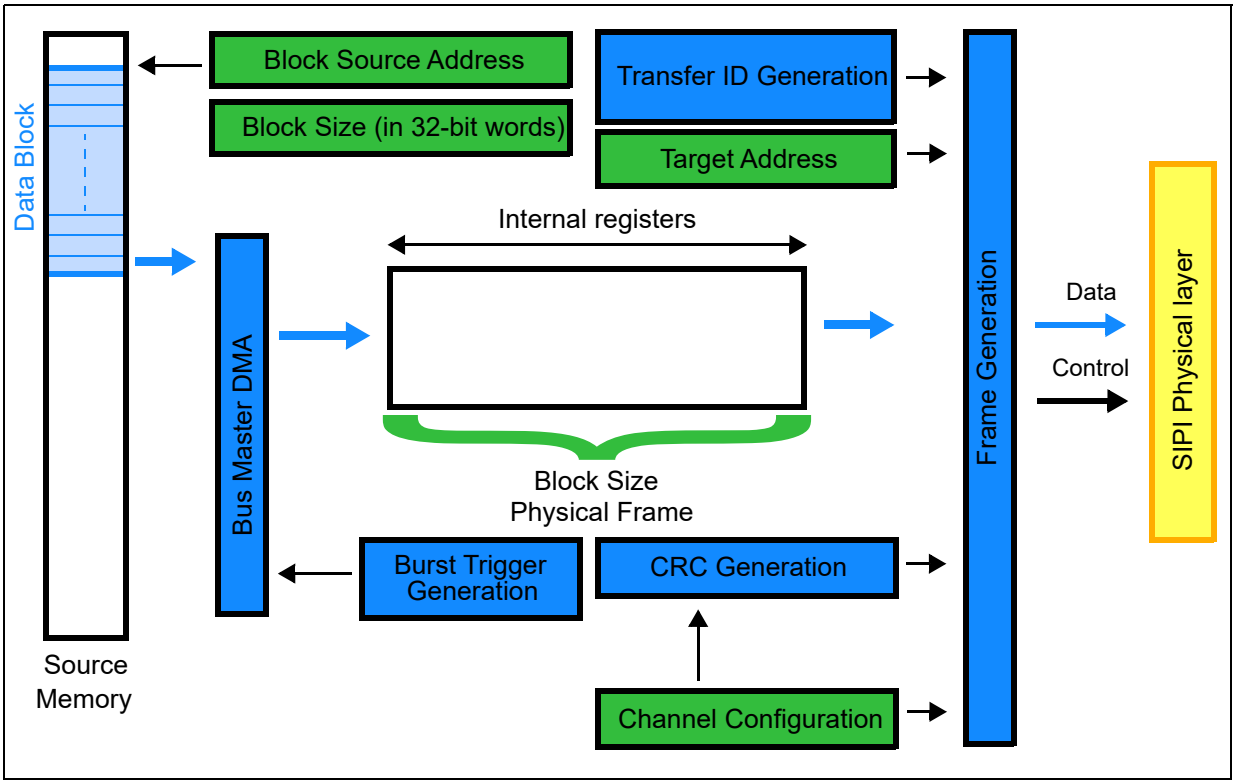


Figure 900. SIPI data stream model - Target

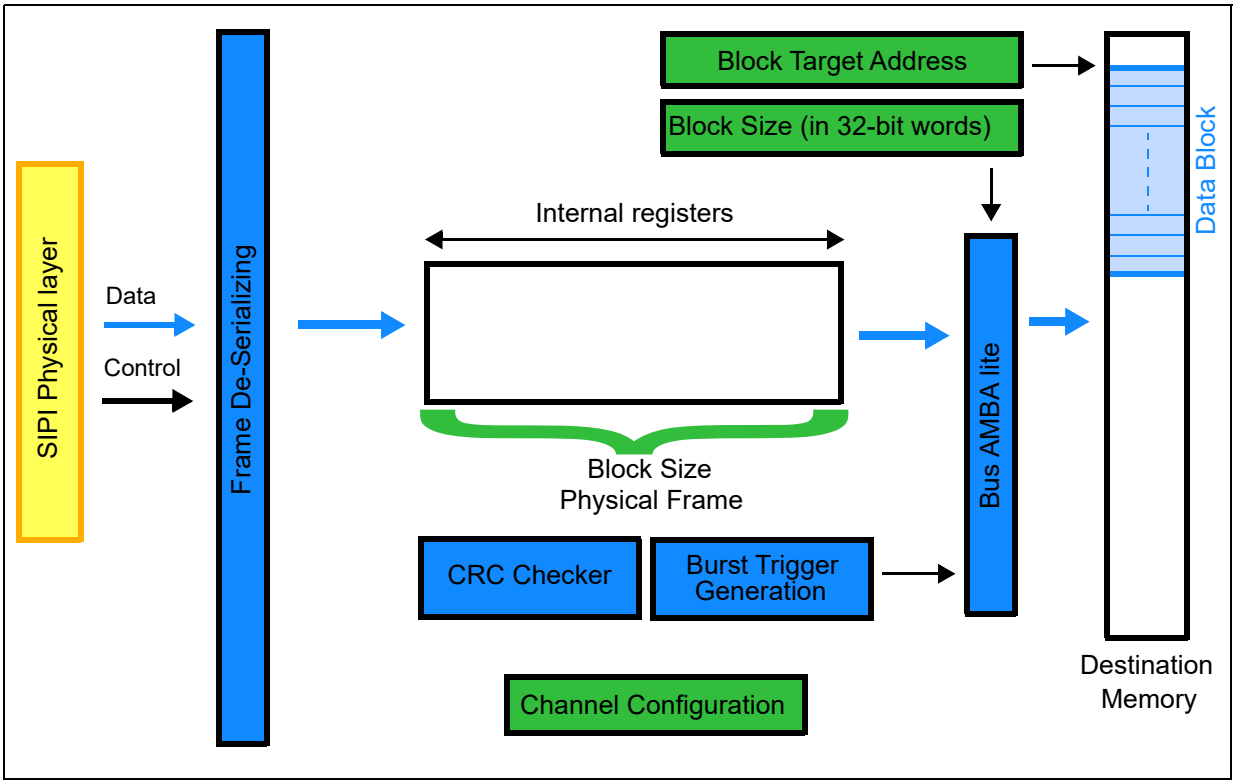
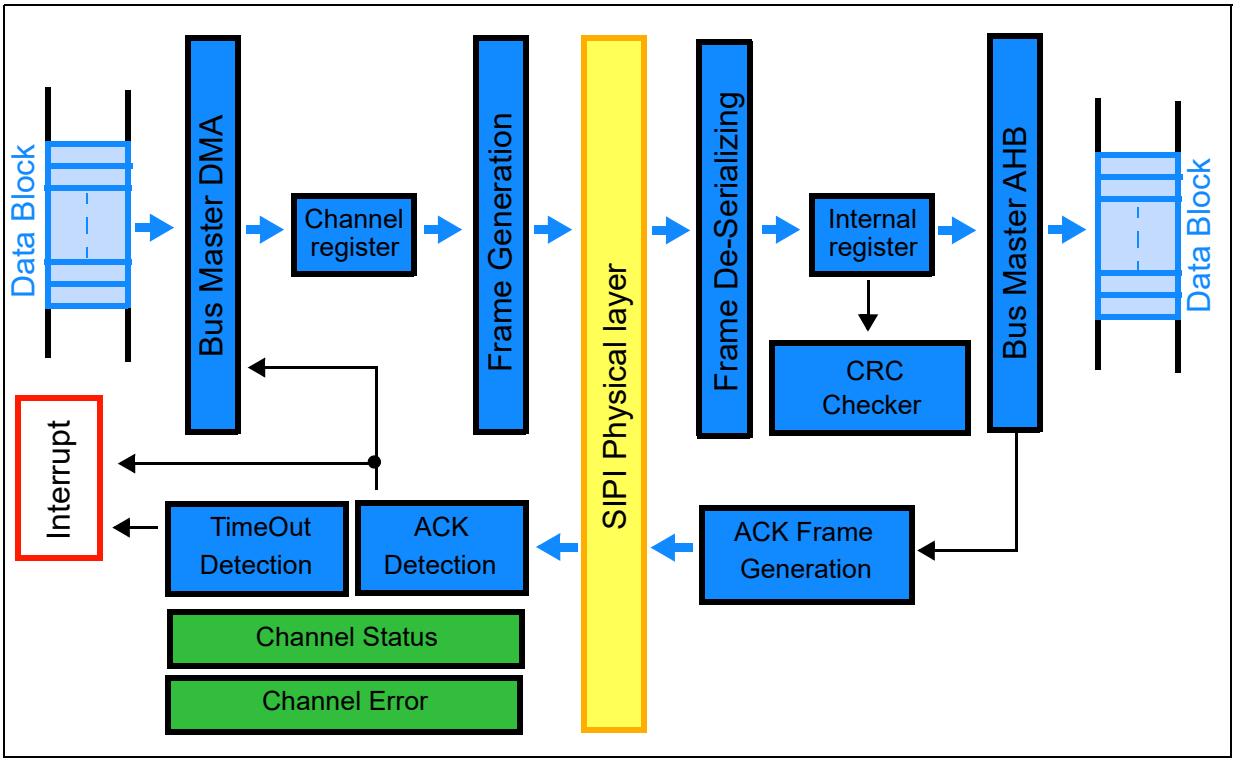


Figure 901. SIPI data stream with acknowledge trigger



53.9 DMA handling in transfer initiator mode

DMA handling of transfer initiator: the initiator of the SIPI transfer (SIPI master) uses the external DMA module on system level to exchange the data between System RAM and SIPI registers via the Peripheral Bridge Interface.

DMA handling of transfer target: the target of the SIPI transfer (SIPI slave) uses its built-in DMA capabilities to write received data into System RAM or to read requested data from System RAM via its data master port on the device interconnect.

1. Software configures TCD (DMA data).
2. DMA transfers from RAM to SIPI registers.
3. SIPI starts transferring the data through LFAST Tx ports.
4. If an acknowledge or fault response is received, SIPI will repeat the process. If any error response is received, SIPI will move to an idle state. An interrupt will be flagged by SIPI, and software will take control of the SIPI.
5. If SIPI has received a read response, and error flags are not set on the LFAST Rx port, SIPI will assert a request. In case of an error, move to Step 8.
6. As soon as the request is asserted, DMA transfer starts between the SIPI registers and RAM.
7. If DMA transfer minor loop/major loop is complete, the request is negated.
8. If an error occurs:
   a) The SIPI generates an interrupt.
   b) Software takes control of the SIPI.
9. Steps 5 to 8 are repeated until the transfer is finished.

53.10 Modes of operation

SIPI has three operating modes:

- Initialization
- Normal
- Module Disable

After hardware reset the SIPI is in Module Disable mode, which helps reduce power consumption (refer to *Section 53.10.3*).

53.10.1 Initialization mode (INIT)

To enter Initialization mode software writes SIPI_MCR[INIT] = 1 (SIPI_MCR[MOEN] = 1 must be written before attempting to write SIPI_MCR[INIT]). To exit Initialization mode software writes SIPI_MCR[INIT] = 0 (refer to *Section 53.15.1.8: SIPI Module Configuration Register (SIPI_MCR)*). SIPI_MCR[PRSCLR], SIPI_MCR[AID], SIPI_CCRn[DEN], SIPI_CCRn[CHEN], SIPI_CTORn[TOR], SIPI_CCRn[TC] fields can only be written in INIT mode.A channel already enabled will not be disabled on entry to INIT mode unless the SIPI_CCRn[CHEN] field is written a value 0.

Note: It is recommended that software checks the SIPI state (SIPI_MCR[MOEN]) before setting SIPI_MCR[INIT].

53.10.2 Normal Mode (NM)

Once software has completed initialization of SIPI (SIPI_MCR[INIT] = 1), it can enter Normal mode by writing SIPI_MCR[INIT] = 0.

Note: SIPI must be enabled before attempting to write to SIPI_MCR[INIT] (for example, SIPI_MCR[MOEN] = 1).

53.10.3 Module Disable (MD)

Module Disable mode in the SIPI is used to help reduce power consumption. By default, SIPI is in disable mode, SIPI_MCR[MOEN] = 0, and is exited by writing SIPI_MCR[MOEN] = 1 (refer to *Section 53.15.1.8: SIPI Module Configuration Register (SIPI_MCR)*). Future SIPI Tx transfers are disabled in Module Disable mode. To disable Rx functionality, the Target Enable (TEN) bit field has to be deasserted, then the internal Rx state machine resets to initial state.

53.11 Errors

This section describes the potential errors that can occur during SIPI operation.

53.11.1 Timeout error

This error is generated at the initiator node when the acknowledge/response is not received within the configured time in the corresponding SIPI_CTOR*n*[TOR] field setting (refer to *Section 53.15.1.4: SIPI Channel Timeout Register (SIPI_CTORn)*). SIPI_ERR[TOE*n*] = 1 indicates that a channel timeout error has occurred (refer to *Section 53.15.1.13: SIPI Channel Error Register (SIPI_ERR)*).

Note: Upon detecting timeout, the application can check the DEN bit in order to ascertain if the missing acknowledge comes from the last packet or an intermediate one:

- *DEN = 0: ack missing for last packet*
- *DEN = 1: ack missing for intermediate packet.*

Then, in order to recover functionality, the application can:

- *write the soft reset bit SIPI_MCR[SR] for the last streaming frame case; in this case, DEN bit access is gained back immediately after writing SR*
- *for non-last streaming frame, clear the timeout to automatically resume data transfer.*

53.11.2 CRC error

A CRC error is generated at the target nodes when the CRC received with the frame does not match with the calculated CRC. When a CRC error is detected the SIPI_SR[GCRCE] is set (refer to *Section 53.15.1.9: SIPI Status Register (SIPI_SR)*) and an interrupt will be generated if SIPI_MCR[CRCIE] = 1 (refer to *Section 53.15.1.8: SIPI Module Configuration Register (SIPI_MCR)*).

Note: In the case of a CRC error, the target node will not send an acknowledge to the initiator node. Then an interrupt will be generated on the target side if the corresponding interrupt enable bit is set. The initiator node will detect a time-out and take necessary action.

53.11.3 Maximum count reached error

This error is always generated at the target node only. The maximum count reached error is generated when the value of the SIPI_ACR equals SIPI_MAXCR (refer to *Section 53.15.1.12: SIPI Address Count Register (SIPI_ACR)* and *Section 53.15.1.10: SIPI Max Count Register (SIPI_MAXCR)* for more details). The SIPI_SR[MCR] will be set (refer to *Section 53.15.1.9: SIPI Status Register (SIPI_SR)*), and an interrupt will be generated if SIPI_MCR[MCRIE] = 1 (refer to *Section 53.15.1.8: SIPI Module Configuration Register (SIPI_MCR)*).

53.11.4 Transaction ID error

This error is always generated at the initiator node only. It is generated when header bits 15:13 do not match SIPI_CSR*n*[TID] (for transaction ID bits, refer to *Section 53.15.1.2: SIPI Channel Status Register (SIPI_CSRn)*). The SIPI_CSR*n*[TIDE] bit will be set when a TID error is detected, and an interrupt will be generated if SIPI_CIR*n*[TIDIE] = 1 (refer to *Section 53.15.1.3: SIPI Channel Interrupt Register (SIPI_CIRn)*).

53.11.5 Acknowledge error

When an incorrect Acknowledge is received the SIPI_ERR[ACKR*n*] will be set (refer to *Section 53.15.1.13: SIPI Channel Error Register (SIPI_ERR)*), at the initiator node only, and an interrupt will be generated if SIPI_CIR*n*[WAIE] = 1 (refer to *Section 53.15.1.3: SIPI Channel Interrupt Register (SIPI_CIRn)*).

53.12 CRC calculation

Example 1

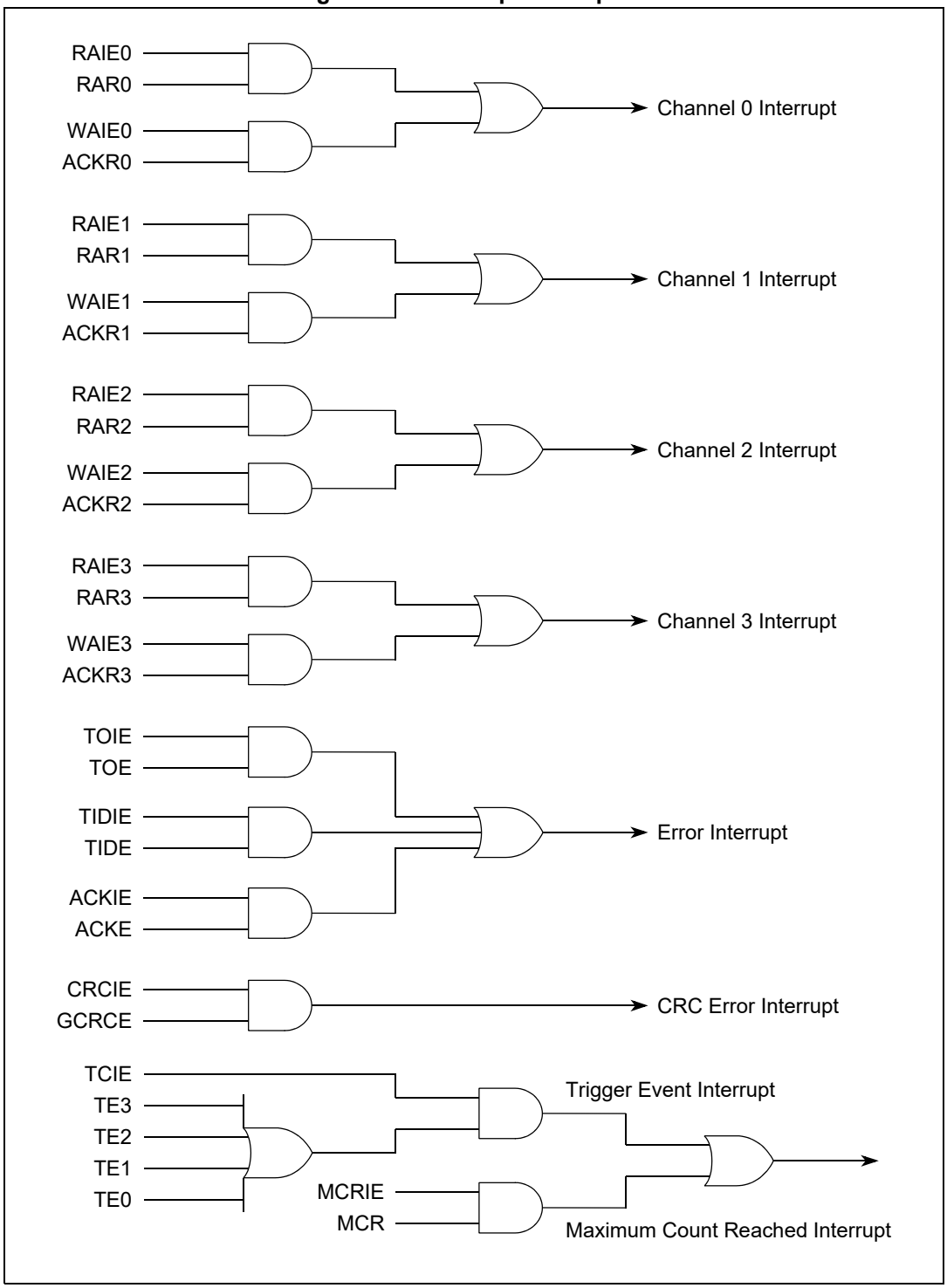
If header is 0xAABB, address is 0x1122_3344 and data is 0xCCDD_EEFF. Then CRC calculation will take place as follows:

1. The CRC seed is initialized by 0xFFFF_FFFF.
2. All the data will be mirrored before sending to CRC engine (for example, MSB will be sent as LSB).
3. So the header will be sent as 0xDD55_0000, byte_enable = 0xC.
4. Address will be sent as 0x22CC_4488, byte_enable = 0xF.
5. Data will be sent as 0xFF77_BB33, byte enable = 0xF.

53.13 Interrupt logic

A description of the interrupt logic can be found in *Figure 902*.

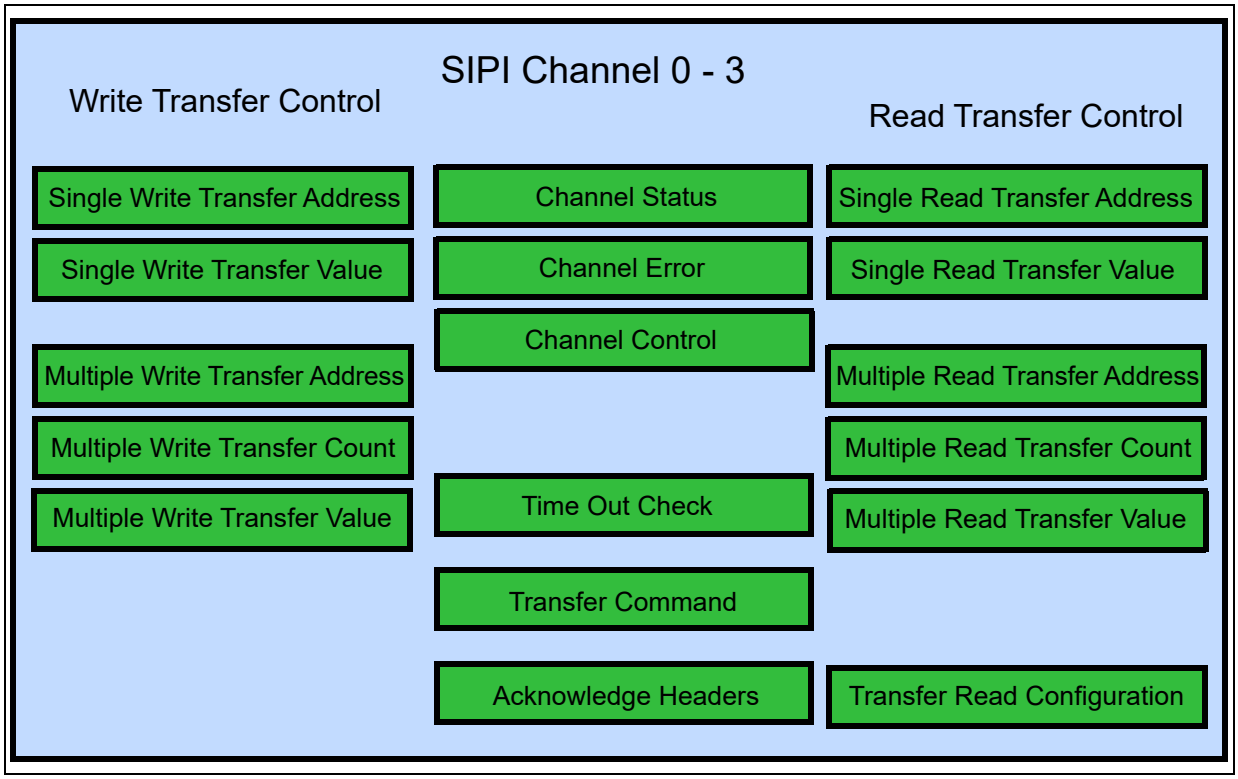
Figure 902. Interrupt description



53.14 SIPI control and status overview

The diagram below shows the relationship between transfers and the SIPI control and status registers.

Figure 903. SIPI control & status overview – Register transfer



53.15 Memory map and register description

Table 964. SIPI memory map

Offset from SIPI base (hex)	Register	Section/Page
0x0000	SIPI Channel Control Register 0 (SIPI_CCR0)	*Section 53.15.1.1*
0x0004	SIPI Channel Status Register 0 (SIPI_CSR0)	*Section 53.15.1.2*
0x0008–0x000B	Reserved	
0x000C	SIPI Channel Interrupt Register 0 (SIPI_CIR0)	*Section 53.15.1.3*
0x0010	SIPI Channel Timeout Register 0 (SIPI_CTOR0)	*Section 53.15.1.4*
0x0014	SIPI Channel CRC Register 0 (SIPI_CCRC0)	*Section 53.15.1.5*
0x0018	SIPI Channel Address Register 0 (SIPI_CAR0)	*Section 53.15.1.6*
0x001C	SIPI Channel Data Register 0 (SIPI_CDR0)	*Section 53.15.1.7*
0x0020	SIPI Channel Control Register 1 (SIPI_CCR1)	*Section 53.15.1.1*
0x0024	SIPI Channel Status Register 1 (SIPI_CSR1)	*Section 53.15.1.2*
0x0028–0x002B	Reserved	
0x002C	SIPI Channel Interrupt Register 1 (SIPI_CIR1)	*Section 53.15.1.3*
0x0030	SIPI Channel Timeout Register 1 (SIPI_CTOR1)	*Section 53.15.1.4*

Table 964. SIPI memory map (continued)

Offset from SIPI base (hex)	Register	Section/Page
0x0034	SIPI Channel CRC Register 1 (SIPI_CCRC1)	*Section 53.15.1.5*
0x0038	SIPI Channel Address Register 1 (SIPI_CAR1)	*Section 53.15.1.6*
0x003C	SIPI Channel Data Register 1 (SIPI_CDR1)	*Section 53.15.1.7*
0x0040	SIPI Channel Control Register 2 (SIPI_CCR2)	*Section 53.15.1.1*
0x0044	SIPI Channel Status Register 2 (SIPI_CSR2)	*Section 53.15.1.2*
0x0048–0x004B	Reserved	
0x004C	SIPI Channel Interrupt Register 2 (SIPI_CIR2)	*Section 53.15.1.3*
0x0050	SIPI Channel Timeout Register 2 (SIPI_CTOR2)	*Section 53.15.1.4*
0x0054	SIPI Channel CRC Register 2 (SIPI_CCRC2)	*Section 53.15.1.5*
0x0058	SIPI Channel Address Register 2 (SIPI_CAR2)	*Section 53.15.1.6*
0x005C	SIPI Channel Data Register 2[0] (SIPI_CDR2[0])	*Section 53.15.1.7*
0x0060	SIPI Channel Data Register 2[1] (SIPI_CDR2[1])	*Section 53.15.1.7*
0x0064	SIPI Channel Data Register 2[2] (SIPI_CDR2[2])	*Section 53.15.1.7*
0x0068	SIPI Channel Data Register 2[3] (SIPI_CDR2[3])	*Section 53.15.1.7*
0x006C	SIPI Channel Data Register 2[4] (SIPI_CDR2[4])	*Section 53.15.1.7*
0x0070	SIPI Channel Data Register 2[5] (SIPI_CDR2[5])	*Section 53.15.1.7*
0x0074	SIPI Channel Data Register 2[6] (SIPI_CDR2[6])	*Section 53.15.1.7*
0x0078	SIPI Channel Data Register 2[7] (SIPI_CDR2[7])	*Section 53.15.1.7*
0x007C	SIPI Channel Control Register 3 (SIPI_CCR3)	*Section 53.15.1.1*
0x0080	SIPI Channel Status Register 3 (SIPI_CSR3)	*Section 53.15.1.2*
0x0084–0x0087	Reserved	
0x0088	SIPI Channel Interrupt Register 3 (SIPI_CIR3)	*Section 53.15.1.3*
0x008C	SIPI Channel Timeout Register 3 (SIPI_CTOR3)	*Section 53.15.1.4*
0x0090	SIPI Channel CRC Register 3 (SIPI_CCRC3)	*Section 53.15.1.5*
0x0094	SIPI Channel Address Register 3 (SIPI_CAR3)	*Section 53.15.1.6*
0x0098	SIPI Channel Data Register 3 (SIPI_CDR3)	*Section 53.15.1.7*
0x009C	SIPI Module Configuration Register (SIPI_MCR)	*Section 53.15.1.8*
0x00A0	SIPI Status Register (SIPI_SR)	*Section 53.15.1.9*
0x00A4	SIPI Max Count Register (SIPI_MAXCR)	*Section 53.15.1.10*
0x00A8	SIPI Address Reload Register (SIPI_ARR)	*Section 53.15.1.11*
0x00AC	SIPI Address Count Register (SIPI_ACR)	*Section 53.15.1.12*
0x00B0	SIPI Error Register (SIPI_ERR)	*Section 53.15.1.13*

53.15.1 Register descriptions

53.15.1.1 SIPI Channel Control Register (SIPI_CCR*n*)

Note: Back to back transactions are not supported on a single channel (for example, if the channel busy bit, SIPI_CSRn[CB] = 1, is configured for 1 channel, it cannot be triggered for another transfer until the data buffers are empty and the channel is free, SIPI_CSRn[CB] = 0).

PRIORITY SCHEDULING: The channel whose SIPI_CARn is written first will have access first, irrespective of its priority at the start of the transfer. As more transfers are requested, priority will be resolved as per the priority scheme (for example, channel 0 – channel 1 – channel 2 – channel 3). Priority scheduling occurs when the LFAST is unable to receive data, and no channel is pending transmission. The reason for the scheduling is that a common CRC module is shared by all channels.

Note: This register can only be written during Initialization mode (SIPI_MCR[INIT] = 1), except for command bits (for example, TC, WL, ST, IDT, RRT and WRT). Also, the command bits can be written only when the corresponding Channel busy bit is not asserted (SIPI_CSRn[CB] = 1, Section 53.15.1.2: SIPI Channel Status Register (SIPI_CSRn)). Software will need to poll the respective channel busy bits before attempting to change the command type.

Only one channel at a time can transmit. Therefore, only one channel can remain busy. Maximum time allowed to transmit a full command is nine system clock cycles (streaming command is 9 words long). The Command type of a channel can only be changed when it is no longer busy. Also, to send a command, we need to write to the corresponding SIPI_CARn (Section 53.15.1.6: SIPI Channel Address Register (SIPI_CARn).

Offset: 0x0000 (SIPI_CCR0), 0x0020 (SIPI_CCR1), 0x0040 (SIPI_CCR2), 0x007C (SIPI_CCR3)

Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	TC
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	WL		CHEN	ST	IDT(1)	RRT(1)	WRT(1)	DEN
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. The order of priority for writing to IDT, WRT and RRT must be in this order: 1) IDT, 2) RRT, 3) WRT.

Figure 904. SIPI Channel Control Register (SIPI_CCR*n*)

Table 965. SIPI_CCR*n* field descriptions

Field	Description
15 TC	Trigger Command A Trigger Command requires that only this bit to be set by software, the trigger does not depend on settings in the SIPI_CAR, this bit can not be set if SIPI_CSRn[CB] = 1. SW must read back to check the value and SW has to set it back to 0. 0 Trigger command not sent 1 Trigger command sent
24:25 WL	Word Length Transfer For Streaming write WL bits should be written 10. 00 8-bit 01 16-bit 10 32-bit 11 not used
26 CHEN	Channel Enable If all the channel enables are pending simultaneously, channel 0 will be given the highest priority for transmission. For channels, the lowest channel number is given highest priority. 0 Channel is disabled. 1 Channel is enabled.
27 ST	Streaming Transfer This bit is hard-coded to 0 for channels 0, 1 and 3. It can only be written for channel 2 (SIPI_CCR2). This bit can only be written when SIPI_CCR*n*[WRT] = 1. 0 Streaming transfer is disabled. 1 Streaming transfer is enabled.
28 IDT	ID Read Request Transfer This request returns the value of the CHIP ID. 0 ID read request not sent. 1 ID read request sent.
29 RRT	Read Request Transfer 0 Read request will not be sent 1 Read request transfer by the initiator. This bit cannot be written if SIPI_CCR*n*[IDT] = 1.
30 WRT	Write Request Transfer 0 No write request will be sent. 1 Write request transfer by the initiator. This bit cannot be written if SIPI_CCR*n*[IDT] = 1 or SIPI_CCR*n*[RRT] = 1.
31 DEN	DMA Enable When enabling DMA mode, this bit should be set by software. It will then be cleared by hardware after one major loop has completed (for streaming mode and multiple data transfer it is advised to assert DEN). 0 Channel will be used for bus interface access. 1 Channel will be used for DMA request to the host.

53.15.1.2 SIPI Channel Status Register (SIPI_CSR*n*)

The SIPI_CSR*n* contains the status bits for the current transfer.

Offset: 0x0004 (SIPI_CSR0)
0x0024 (SIPI_CSR1)
0x0044 (SIPI_CSR2)
0x0080 (SIPI_CSR3)

Access: User read/write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	RAR	TID			ACKR	CB	0	0
W									w1c				w1c			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 905. Channel Status Register (CSR*n*)

Table 966. CSR*n* field descriptions

Field	Description
24 RAR	Read Answer Reception 0 Read answer not received 1 Read answer received
25:27 TID	Transaction ID of transmitted frame Transaction ID is a random number associated with every Tx frame to match the received response/acknowledge with the command.
28 ACKR	Acknowledge Received 0 Acknowledge not received 1 Acknowledge received
29 CB	Channel Busy Indicates channel status. This bit is read only. 0 Channel free 1 Channel busy

53.15.1.3 SIPI Channel Interrupt Register (SIPI_CIR*n*)

The SIPI_CIR*n* contains the interrupt enable bits for the corresponding channel (*n*).

Offset: 0x000C (SIPI_CIR0)
0x002C (SIPI_CIR1)
0x004C (SIPI_CIR2)
0x0088 (SIPI_CIR3)

Access: User read/write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	WAIE	RAIE	TCIE	TOIE	TIDIE	ACKIE
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 906. SIPI Channel Interrupt Register (SIPI_CIR*n*)

Table 967. SIPI_CIR*n* field descriptions

Field	Description
26 WAIE	Write Acknowledge Interrupt Enable 0 Interrupt is disabled 1 Interrupt is enabled
27 RAIE	Read Answer Interrupt Enable 0 Interrupt is disabled 1 Interrupt is enabled
28 TCIE	Trigger Command Interrupt Enable 0 Interrupt is disabled 1 Interrupt is enabled
29 TOIE	Timeout Error Interrupt Enabled 0 Interrupt is disabled 1 Interrupt is enabled
30 TIDIE	Transaction ID Error Interrupt Enable 0 Interrupt is disabled 1 Interrupt is enabled
31 ACKIE	Acknowledge Error Interrupt Enable 0 Interrupt is disabled 1 Interrupt is enabled

53.15.1.4 SIPI Channel Timeout Register (SIPI_CTOR*n*)

The SIPI_CTOR*n* contains the timeout value for Tx requests.

Offset: 0x0010 (SIPI_CTOR0), 0x0030 (SIPI_CTOR1), 0x0050 (SIPI_CTOR2), 0x008C (SIPI_CTOR3)

Access: User read/write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	TOR(1)							
W																
Reset	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1

1. This field can only be written during Initialization mode (SIPI_MCR[INIT] = 1).

Figure 907. SIPI Channel Timeout Register (SIPI_CTOR*n*)

Table 968. SIPI_CTOR*n* field descriptions

Field	Description
24:31 TOR	Timeout value for transmitted requests. Timeout counter runs on the prescaled peripheral clock. Prescaler is defined by SIPI_MCR[PRSCLR]. These bits can only be written by software in initialization mode, SIPI_MCR[INIT] = 1.

53.15.1.5 SIPI Channel CRC Register (SIPI_CCRC*n*)

SIPI_CCRC*n* is read-only register that returns the CRC calculated value on the header, address, and data bits.

Offset: 0x0014 (SIPI_CCRC0), 0x0034 (SIPI_CCRC1), 0x0054 (SIPI_CCRC2), 0x0090 (SIPI_CCRC3)

Access: User read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	CRCI															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	CRCT															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 908. SIPI Channel CRC Register (SIPI_CCRC*n*)

Table 969. CCRC*n* field descriptions

Field	Description
0:15 CRCI	Reflects received CRC value at initiator
16:31 CRCT	Reflects received CRC value at target

53.15.1.6 SIPI Channel Address Register (SIPI_CAR*n*)

The SIPI_CAR*n* contains the data transmission target address. The SIPI_CAR*n* is a 32-bit register and can be written anytime by software or DMA.

Offset: 0x0018 (SIPI_CAR0)
0x0038 (SIPI_CAR1)
0x0058 (SIPI_CAR2)
0x0094 (SIPI_CAR3)

Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	CAR(1)															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	CAR(1)															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. This field holds the start address except for streaming command. For Streaming, address is got from SIPI_ACR.

Figure 909. SIPI Channel address Register (SIPI_CAR*n*)

Table 970. SIPI_CAR*n* field descriptions

Field	Description
0:31 CAR	These bits contain the address of the target node. Single and multiple write request: these bits contain the address of the target node. For streaming mode, this register must be written to start the transmission but contents are not used.

53.15.1.7 SIPI Channel Data Register (SIPI_CDR*n*)

The SIPI_CDR*n* contains the data to be transmitted, and can be written anytime by software or DMA. The data can be written in 8, 16 or 32-bit formats.

Offset: 0x001C (SIPI_CDR0)
0x003C (SIPI_CDR1)
0x0098 (SIPI_CDR3)

Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	CDR															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	CDR															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 910. SIPI Channel Data Register (SIPI_CDR*n*)

Table 971. SIPI_CDR*n* field descriptions

Field	Description
0:31 CDR	Data register bits Contains the data that will be transmitted, or received.

Table 972. Channel data Register address offsets

Register	Address Offset (hex)
SIPI_CDR0	0x001C
SIPI_CDR1	0x003C
SIPI_CDR2*i*	0x005C + *i**(0x4) for *i* = 0 to 7 (refer to *Section 53.15.1.7.1*)
SIPI_CDR3	0x0098

53.15.1.7.1 SIPI Channel Data Register (SIPI_CDR2*i*)

The SIPI_CDR2*i* contains the data to be transmitted, and can be written anytime by software or DMA. The data can be written in 8, 16 or 32-bit formats.

Offset: 0x005C + *i**0x4 (*i* = 0 to 7)

Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	CDR															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	CDR															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 911. SIPI Channel Data Register (SIPI_CDR2*i*)

Table 973. SIPI_CDR*2i* field descriptions

Field	Description
0:31 CDR	Data register bits Contains the data that will be transmitted, or received.

53.15.1.8 SIPI Module Configuration Register (SIPI_MCR)

The SIPI_MCR is a 32-bit global configuration register of the SIPI.

Offset: 0x009C Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	FRZ	0	HALT	0	0	PRSCLR(1)										
W																
Reset	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	AID(1)		0	0	0	CRCIE(2)	MCRIE(2)	0	0	0	0	CHNSB(3)	TEN(2)	INIT(2)	MOEN	SR
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. Can be written in initialization mode only (SIPI_MCR[INIT] = 1).
2. Can only be written when SIPI_MCR[MOEN] = 1.
3. Can be written only once after reset.

Figure 912. SIPI Module Configuration Register (SIPI_MCR)

Table 974. SIPI_MCR field descriptions

Field	Description
0 FRZ	Freeze Enable The FRZ bit specifies the SIPI behavior when Debug mode is requested at the MCU level. When FRZ is asserted, the SIPI is enabled to enter Freeze mode. Negation of this bit field causes SIPI to exit Freeze mode. 0 Not enabled to enter Freeze mode 1 Enabled to enter Freeze mode
2 HALT	Halt Mode Enable Assertion of this bit puts SIPI into Freeze mode. No Rx or Tx is performed in the SIPI until this bit is cleared. If this bit is enabled in during Tx or Rx communications, the current activity will finish, then the SIPI will enter Freeze mode. 0 No Freeze mode request. 1 Enters Freeze mode if FRZ bit is asserted.

Table 974. SIPI_MCR field descriptions (continued)

Field	Description
5:15 PRSCLR	These bits should be programmed by software in initialization mode, SIPI_MCR[INIT] = 1, writes during other times are ignored. The timeout counter runs on the prescaled system clock. Default value is 64 (0x040). The allowed programmable values are: 0x040: 64 0x080: 128 0x100: 256 0x200: 512 0x400: 1024
16:17 AID	Address Increment/Decrement (for streaming mode) These bits define the type of address change at the target node for the next word write, in streaming mode. This field can be written only in INIT mode that is SIPI_MCR[INIT] = 1. This field dictates how it's memory is written when it receives a streaming command from outside world. 00 No change. address stays same 01 Address increments by 4 10 Address decrements by 4. 11 Reserved
21 CRCIE	CRC Error Interrupt Enable 0 Interrupt is disabled 1 Interrupt is enabled
22 MCRIE	Max Count Reached Interrupt Enable 0 Interrupt is disabled. 1 Interrupt is enabled.
27 CHNSB	Channel coding select bit 0, 1 Channel number select as per *Table 960: SIPI Header Channel Number Coding*.
28 TEN	Target Enable Setting this bit enables the target functionality at SIPI. This bit can be read or written anytime. This bit is automatically negated by hardware when target detects an error in “streaming without ACK” mode. This bit has to be enabled for transmission operations as well.
29 INIT	Setting this bit puts the module in initialization mode. This bit should be cleared by software. Most register bits are configured using this bit. The SIPI_MCR[MOEN] bit needs to be set first, and then the INIT bit can be set and both bits cannot be enabled together. 0 Normal Mode 1 Initialization Mode
30 MOEN	Module Enable This bit should be set or cleared by software. When this bit is negated, future SIPI Tx transfers are disabled.
31 SR	Soft Reset Setting this bit clears all status and error registers and FSMs are moved to idle state. This bit is automatically cleared by hardware once the reset operation is complete. Statical configuration, data and address registers remain unchanged.

53.15.1.9 SIPI Status Register (SIPI_SR)

The SIPI_SR is the global status register of SIPI.

Offset: 0x00A0 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	FRZACK	LPMACK	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	GCRCE	MCR	0	TE				STATE			
W						w1c	w1c		w1c							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 913. SIPI Status Register (SIPI_SR)

Table 975. SIPI_SR field descriptions

Field	Description
0 FRZACK	Freeze Mode Acknowledge This read-only bit indicates that SIPI is in Freeze mode. The Freeze mode request cannot be granted until current transmission or reception processes have finished. The software can poll the FRZACK bit to find when the SIPI has actually entered Freeze mode. If Freeze mode is requested while SIPI is in any of the low power modes, then the FRZACK bit will only be set when the low power mode is exited. 0 SIPI not in Freeze mode 1 SIPI in Freeze mode
1 LPMACK	Low Power Mode Acknowledge This read-only bit indicates that SIPI is in Disable Mode. Disable mode can not be entered until all current transmission or reception processes have finished. The CPU can poll the LPMACK bit to know when SIPI has actually entered low power mode. 0 SIPI not in any low power modes 1 SIPI is in Disable mode
21 GCRCE	Global CRC Error Bit 0 No CRC error 1 CRC error occurred
22 MCR	Maximum Count Reached This bit will be set whenever SIPI_ACR[ADCNT] = SIPI_MAXCR[MXCNT]. An interrupt will be generated when this bit is set only if SIPI_MCR[MCRIE] = 1. This bit should be cleared by software.

Table 975. SIPI_SR field descriptions (continued)

Field	Description
24:27 TE	Trigger Event on Respective Channels This field enables interrupts for target nodes. TE3 = 1 - Channel 3 trigger command received TE2 = 1 - Channel 2 trigger command received TE1 = 1 - Channel 1 trigger command received TE0 = 1 - Channel 0 trigger command received
28:31 STATE	These bits reflect the transmit state machine status. They can be polled for determination of state machine status. These bits reflect what is being transmitted that is STATE = HEADER_AND_ADDRESS_FIELD means that HEADER and ADDRESS were transmitted. Boundary of transmission is word and with the command formats, half address sent in one clock and other half in the next. 0000 IDLE 0001 HEADER_AND_ADDRESS_FIELD 0010 HEADER_AND_DATA_FIELD 0011 HEADER_AND_CRC_FIELD 0100 ADDRESS_AND_CRC_FIELD 0101 ADDRESS_AND_DATA_FIELD 0110 DATA_AND_CRC_FIELD 0111 DATA_FIELD

53.15.1.10 SIPI Max Count Register (SIPI_MAXCR)

The SIPI_MAXCR contains the maximum address count value at target node. It is programmed via direct write request through the initiator. This register can be read or written by software anytime.

Offset: 0x00A4 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	MXCNT															
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	MXCNT														0	0
W																
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0

Figure 914. SIPI Max Count Register (SIPI_MAXCR)

Table 976. SIPI_MAXCR field descriptions

Field	Description
0:29 MXCNT	This field contains the maximum address count value at the target node. It should be programmed via direct write request through the initiator.

53.15.1.11 SIPI Address Reload Register (SIPI_ARR)

The SIPI_ARR contains the reload value for the address counter at the target node. It should be configured by direct write request from the initiator. This register can be read or written anytime.

Offset: 0x00A8 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	ADRLD															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	ADRLD														0	0
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 915. SIPI Address Reload Register (SIPI_ARR)

Table 977. SIPI_ARR field descriptions

Field	Description
0:29 ADRLD	Contains the reload value for the address counter at the target node. It should be configured by direct write request from the initiator.

53.15.1.12 SIPI Address Count Register (SIPI_ACR)

This register reflects the address for streaming command. It should be configured by direct write from initiator. This register can be read/written by software anytime.

Offset: 0x00AC Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	ADCNT															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	ADCNT														0	0
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 916. SIPI Address Count Register (SIPI_ACR)

Table 978. SIPI_ACR field descriptions

Field	Description
0:29 ADCNT	Reflects the value of address at target node. It should be configured by direct write from the initiator. This register can be read/written by software anytime. At the end of the streaming operation, SIPI_ACR will auto update to the next address to be written.

Note: SIPI_ARR, SIPI_ACR and SIPI_MAXCR will be configured by direct write operation by the initiator.
When a streaming write command is received, the target will start the write operation from the address present in SIPI_ACR.
After each 32-bit write has completed, SIPI_ACR[ADCNT] is compared against SIPI_MAXCR[MXCNT]. If they are equal, SIPI_ACR[ADCNT] is loaded with the value stored in SIPI_ARR[ADRLD]. If they are not equal, the SIPI_ACR[ADCNT] will increment by 4, decrement by 4, or remain the same (depending on AID field). In either case, the SIPI_ACR[ADCNT] will contain the address at which the next streaming write will occur.

53.15.1.13 SIPI Channel Error Register (SIPI_ERR)

This register contains the error bits for the last transaction(s) for all the channels. Communication errors are generated only for out of range address accesses, areas that are read only and where accesses do not lead to generation of transfer errors. Also, writing to read only registers will not generate communication errors.

Offset: 0x00B0 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	TOE3	TIDE3	ACKE3	0	0	0	0	0	TOE2	TIDE2	ACKE2
W						w1c	w1c	w1c						w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	TOE1	TIDE1	ACKE1	0	0	0	0	0	TOE0	TIDE0	ACKE0
W						w1c	w1c	w1c						w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 917. SIPI Channel Error Register (SIPI_ERR)

Table 979. SIPI_ERR field descriptions

Field	Description
5 TOE3	Timeout Error for Channel 3 0 Timeout error has not occurred. 1 Timeout error has occurred.
6 TIDE3	Transaction ID Error for Channel 3 0 Received transaction ID matched with the stored ID. 1 Received transaction ID did not match with the stored ID.
7 ACKE3	Acknowledge Error for Channel 3 0 Acknowledge received is correct. 1 Acknowledge received is not correct.
13 TOE2	Timeout Error for Channel 2 0 Timeout error has not occurred. 1 Timeout error has occurred.

Table 979. SIPI_ERR field descriptions (continued)

Field	Description
14 TIDE2	Transaction ID Error for Channel 2 0 Received transaction ID matched with the stored ID. 1 Received transaction ID didn't match with the stored ID.
15 ACKE2	Acknowledge Error for Channel 2 0 Acknowledge received is correct. 1 Acknowledge received is not correct.
21 TOE1	Timeout Error for Channel 1 0 Timeout error has not occurred. 1 Timeout error has occurred.
22 TIDE1	Transaction ID Error for Channel 1 0 Received transaction ID matched with the stored ID. 1 Received transaction ID didn't match with the stored ID.
23 ACKE1	Acknowledge Error for Channel 1 0 Acknowledge received is correct. 1 Acknowledge received is not correct.
29 TOE0	Timeout Error for Channel 0 0 Timeout error has not occurred. 1 Timeout error has occurred.
30 TIDE0	Transaction ID Error for Channel 0 0 Received transaction ID matched with the stored ID. 1 Received transaction ID didn't match with the stored ID.
31 ACKE0	Acknowledge Error for Channel 0 0 Acknowledge received is correct. 1 Acknowledge received is not correct.

54 Zipwire

54.1 Overview

The SIPI and LFAST modules work together as a single unit called Zipwire. The LFAST portion of the two modules allows for high speed inter-device communications. The SIPI allows memory to be shared between devices which have SIPI and LFAST communication modules.

The LFAST can operate in either slave or master mode configurations. The node running in master mode controls the serial link, but SIPI, in both master and slave modes, can act as both initiator and target for SIPI commands simultaneously. Please see the SIPI and LFAST chapters for detailed information on module configuration and functionality.

54.2 Introduction

Zipwire is a group of modules that allow one MCU to have a fast, low pin count, serial communication link directly into the memory mapped peripherals or memories of another MCU (or both) or smart ASIC. Zipwire is implemented in hardware so there is no CPU load for the initiator or target node. Zipwire supports 8-bit, 16-bit or 32-bit reads and writes to any 32-bit address at the target node.

Zipwire architecture is fully pipelined and supports multiple outstanding commands to allow maximum use of the serial link bandwidth.

The serial link runs at 320 Mbaud using LVDS physical layer. One LVDS pair for Tx and another pair for Rx, with a separate 20 MHz reference clock for a total of five pins.Zipwire also supports a streaming mode for the transfer of large blocks of data.

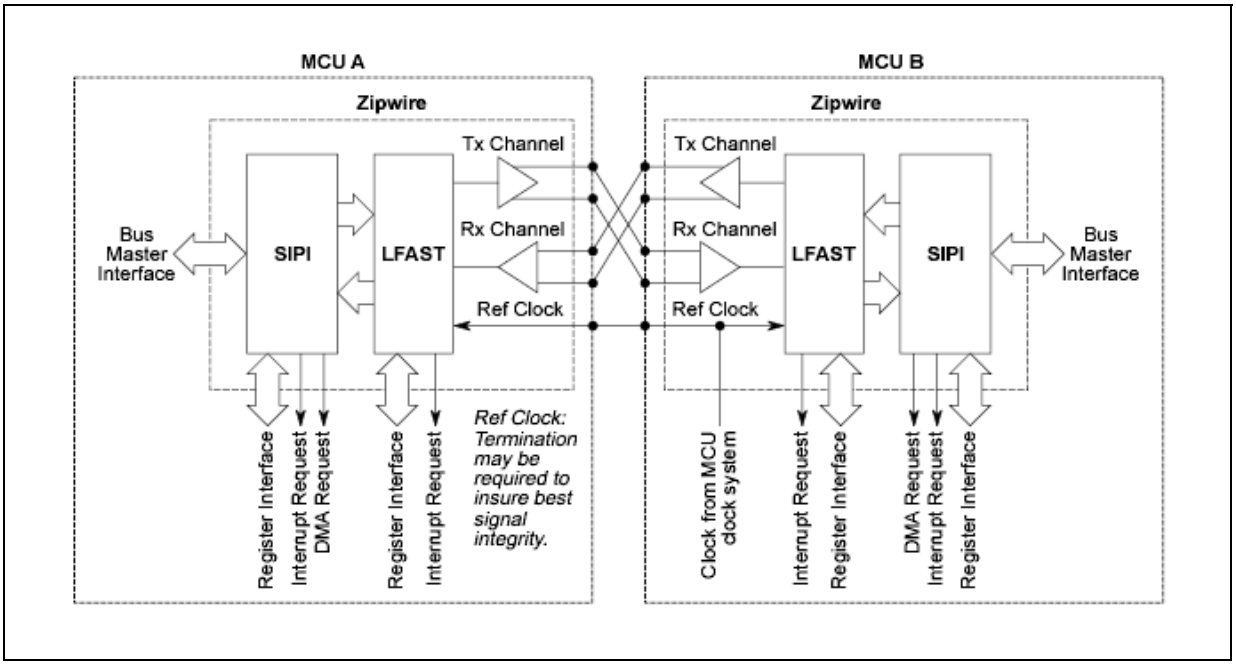
In streaming mode Zipwire has a high transfer rate. For single random access, 32-bit read, from any 32-bit address location, the latency is approximately 1 µs.

54.3 Zipwire block diagram

Figure 918 shows two MCUs with Zipwire connected using a five wire interface. Both MCUs support Target and Initiator modes of Zipwire. The SIPI Bus Master Interface is used for all Target Mode transactions.The SIPI Register Interface is used for all Initiator Transactions and initial setup. The LFAST Register Interface is used for Initial Setup.

The diagram shows the LFAST Reference clock coming from the MCU B clock system. The Ref_Clock can come from either MCU. It is user configurable, but it must be the same clock that drives both LFAST modules.

Figure 918. Zipwire connection diagram



54.4 Architecture

Zipwire is composed of several modules and system resources. The physical layer is LVDS with 1.2 V common mode voltage and 200 mV swing.The LVDS pads are shared with other I/O and GPIO.

The transport layer is a protocol called LFAST. LFAST is an asynchronous protocol, using non-return to zero encoding. The LFAST protocol is composed of the following:

- A fixed 16-bit sync frame to allow the receiver to detect the optimal point to sample the incoming data.
- Followed by an 8-bit LFAST header, that defines the channel number and the size of the LFAST payload.
- Finally the payload which can be between 8 and 288 bits.

The application layer protocol is called Serial Inter Processor Interface (SIPI). SIPI runs on top of LFAST and is fully encapsulated within the LFAST payload. SIPI uses four fixed sizes of LFAST payload in Zipwire:

- 32-bit
- 64-bit
- 96-bit
- 288-bit

The SIPI protocol implements a suite of commands initiated by one MCU, for reading and writing any 32-bit address location in another connected MCU. The SIPI protocol allows either or both MCUs to initiate commands. The protocol also supports interrupt requests from one MCU to the other.

The SIPI module implements both the initiator part of the protocol and the recipient part of the protocol. The module implements four SIPI Initiator channels that can run independently

of each other and can run simultaneously. The SIPI module is a bus master on the low speed XBAR. As the target MCU for a command from the initiating MCU, SIPI can perform read and write accesses to any address location within the target MCU. The software running on the local MCU, should configure the system MPU to allow/deny memory accesses to MCU memory and resources as required.

The initiator part of the module is connected to the DMA controller and is able to generate DMA requests as a command is completed. This allows a series of read or write commands to be queued in the Inititator MCU and executed by DMA and SIPI without CPU intervention at either Initiator or Target MCU.

54.5 LFAST clocking

The LFAST implementation includes a dedicated PLL for multiplying a low speed clock source up to 320 MHz for the transmission and reception of data. The high speed asynchronous nature of LFAST, requires that both the local node and remote node, operate from the same, stable, low jitter clock sources. This common low speed clock is referred to as the Reference Clock.

The MCU includes clock multiplexing to allow the source of the LFAST reference clock to come from one of three sources via a programmable clock divider:

- XOSC
- Output of PLL0 (PLL0:PHI)
- External pin

The most expedient source for the reference clock is the 20 MHz crystal oscillator (XOSC) from one MCU. This 20 MHz clock can be used as the LFAST Reference clock for the local LFAST. This same 20 MHz clock must be exported from the local MCU to the remote MCU, to be used as the LFAST reference at the remote MCU.

The 20 MHz clock may be used in the following ways:

- Shared between the MCUs using a dedicated reference clock pin on each MCU
- Divided by 2 and shared as a 10 MHz reference
- Shared as the fundamental clock for the MCU system clock and the LFAST reference clock

The LFAST PLL for generating the 320 MHz clock, is shared between LFAST for SIPI.

54.6 Zipwire interconnections

LFAST has the following connections:

- Tx and Rx connections to the pads via the SIUL and the MSCR registers
- Peripheral Bridge interface (PBRIDGE): allows software to read and write configuration registers
- Tx Data Port/Rx Data Port: directly connected to the SIPI module. Allows received data to be efficiently transferred to SIPI and transmit data to be transferred from SIPI.
- Interrupt Request connections: Allows the module to flag to the CPU when it requires servicing.

SIPI has the following connections:

- Peripheral Bridge interface (PBRIDGE): allows software to read and write configuration registers and SIPI Interface Registers.
- DMA connections: allows SIPI command sequences to be queues and initiated without CPU intervention.
- Crossbar master port: allows SIPI to execute requested commands to read and write MCU address space.
- Tx Data Port/Rx Data Port: directly connected to the LFAST module. Allows received data to be efficiently transferred from LFAST and transmit data to be transferred to LFAST.
- Interrupt Request connections: allows the module to flag to the CPU when it requires servicing.

55 LVDS Fast Asynchronous Serial Transmission – Interprocessor Communications

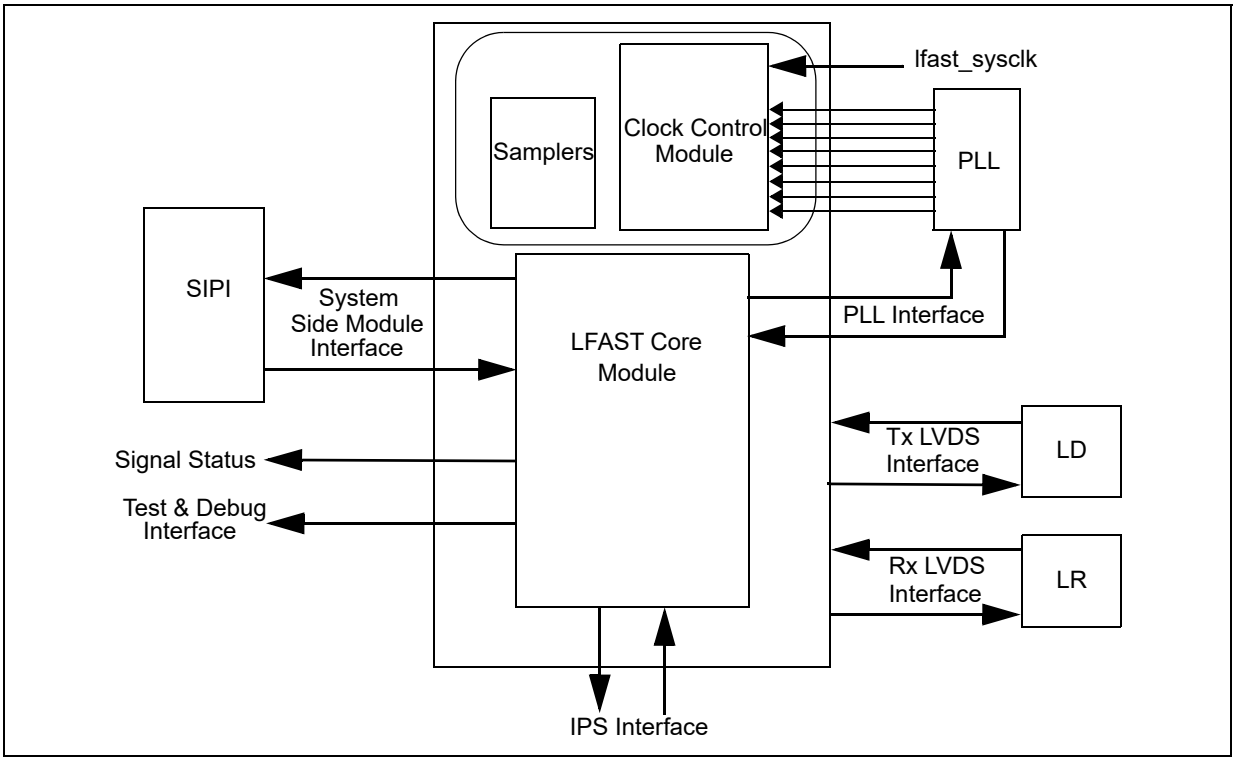
55.1 Introduction

This chapter describes the specifications of the LVDS Fast Asynchronous Serial Transmission (LFAST) module. LFAST is used in dual mode (software configurable master/slave operation) for interprocessor communications.

55.2 Block diagram

Figure 919 depicts LFAST interaction with other modules on the device.

Figure 919. LFAST block diagram



55.3 External signals

LFAST is a five pin interface with the following signals:

- lfast_sysclk
  - Reference clock of the LFAST master and slave
- txdatap/txdatan
  - Differential transmit (Tx) interface pair
- rxdatap/rxdatan
  - Differential receive (Rx) interface pair

LFAST interface is an asynchronous high speed LVDS interface.

55.3.1 LFAST operating data rates

The change of data rate is controlled by the LFAST master by issuing appropriate Interface Control Logical Channel (ICLC) packets to the LFAST slave. Henceforth, the data rate 6.5 Mbps/5 Mbps (lfast_sysclk ÷ 4 or lfast_sysclk ÷ 2) is referred to as low data rate, and the high data rate is in the Data Sheet.

55.4 LFAST frame structure

LFAST frames are transmitted and received across a serial interface using a packeted asynchronous communication format.

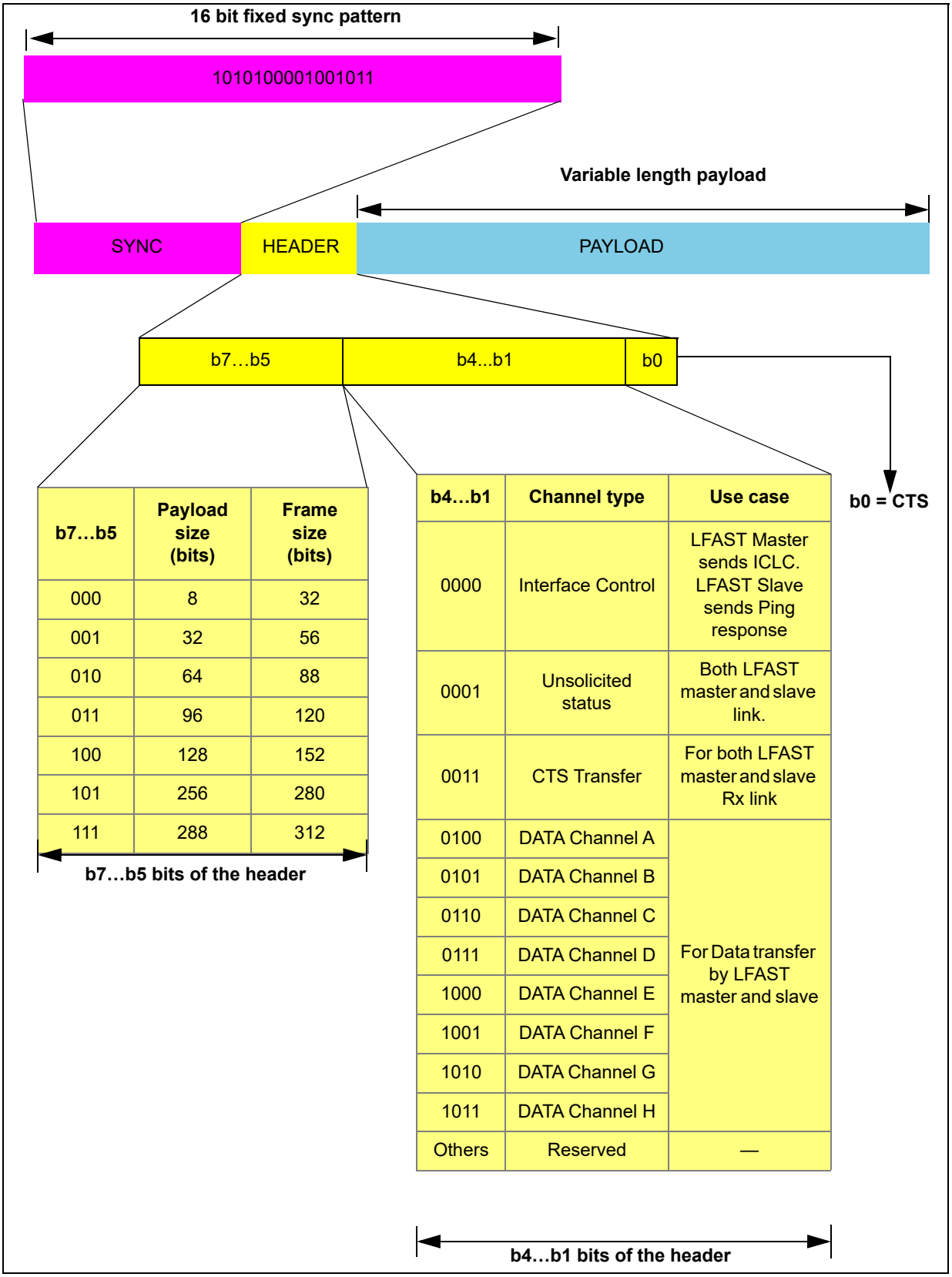
A LFAST frame is made up of three fields:

- Sync pattern
- Header
- Payload

Sync pattern and header fields are of fixed length. A sync pattern is used to synchronize incoming data stream in LFAST module.

Header field of the frame distinguishes various types of data and control transferred and also contains information about the length of the payload. The payload field is the actual data that is transferred across the channel. See *Figure 920* for details of the LFAST frame.

Figure 920. LFAST frame structure

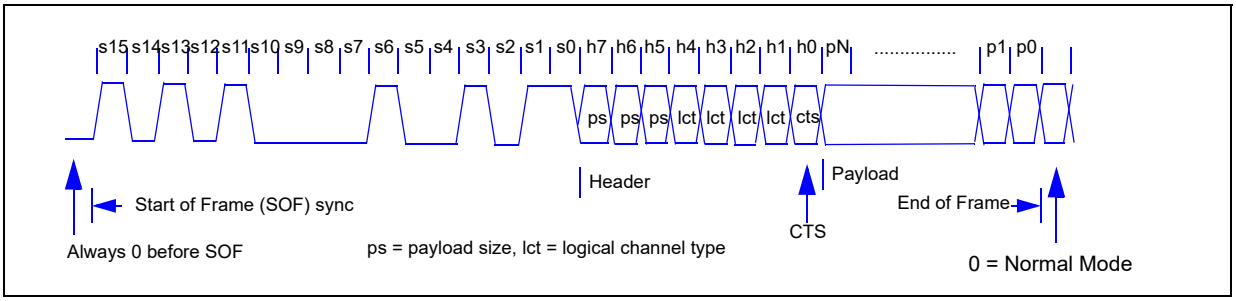


b7…b5	Payload size (bits)	Frame size (bits)
000	8	32
001	32	56
010	64	88
011	96	120
100	128	152
101	256	280
111	288	312

b4…b1	Channel type	Use case
0000	Interface Control	LFAST Master sends ICLC. LFAST Slave sends Ping response
0001	Unsolicited status	Both LFAST master and slave link.
0011	CTS Transfer	For both LFAST master and slave Rx link
0100	DATA Channel A	For Data transfer by LFAST master and slave
0101	DATA Channel B	
0110	DATA Channel C	
0111	DATA Channel D	
1000	DATA Channel E	
1001	DATA Channel F	
1010	DATA Channel G	
1011	DATA Channel H	
Others	Reserved	—

The same protocol is used on both transmit and receive interfaces for communications of data, control and status information. A synchronization pattern (16 bits) and header (8 bits) are present in every frame.

Figure 921. Serial frame structure



The frame structure is shown in *Figure 921* and consists of the following:

- **16-bit synchronization pattern:** Used for clock synchronization and pattern recognition. The frame synchronization code at the start of every frame is a unique reserved word that is used to identify whether the data received is the start of the frame and for synchronization (clock phase extraction) with the stream data. A synchronous sequence is used to give a high quality auto-correlation. The LFAST 16-bit synchronization sequence = 1010_1000_0100_1011b = A84Bh.
- **Header - 3 MSB (b7 - b5):** Defines the payload size according to *Table 980*.

Table 980. Header payload sizes

b7-b5 (bin)	Payload size (bits)	Frame size (bits)
000	8	32
001	32	56
010	64	88
011	96	120
100	128	152
101	256	280
110	—	—
111	288	312

- **Header - (b4 - b1):** Defines the logical channel types, which indicate the type of payload that the frame carries. How the payload field of a frame is used for any other logical channel type is system side module specific except in the case of the interface control logical channel type and clear to send (CTS) frame.
- **Header - (b0):** CTS on the both LFAST master and slave devices.
- **Payload: Content dependent upon frame type.**
- **Bit after frame:** This bit determines entry into Sleep mode (1 = Sleep mode, 0 = normal mode).

55.5 Features

- Supports dual mode (register configurable Master/Slave)
- Supports asynchronous data transfer rates up to the maximum data rated shown in the product Datasheet
- Transmits and receives data, CTS, ICLC and unsolicited frames
- Provision of five interrupts for Tx and Rx channels.
- Supports processor controlled transfer of ICLC frame with 8-bit payload size to implement the data rate changes and test modes
- Supports LFAST defined CTS controlled data transfer. No dependency between data transmission and reception unless CTS mode is enabled
- Supports flow control using sliding window protocol
- Provides configurable frame length for data frame with variable payload sizes of 32, 64, 96, 128, 256 or 288 bits
- Provides configurable frame length for unsolicited frame with variable payload sizes of 8, 32, 64, 96, 128, 256 or 288 bits
- Supports PLL configuration (such as feedback loop divider) through registers
- Supports LVDS configuration through registers
- Supports multiple loopback modes for checking the physical interface
- Supports automatic ping response generation in slave mode
- Supports for detection of unsupported channel number and unsupported payload size

55.6 Memory map and register definition

All registers are 32-bit wide.

55.6.1 Memory map

Table 981. LFAST memory map

Offset from LFAST_BASE (hex)	Register	Access	Reset Value (hex)	Location
0000	LFAST Mode Configuration Register (MCR)	R/W	0x0000_0010	*Section 55.6.2.1*
0004	LFAST Speed Control Register (SCR)	R/W	0x0001_0000	*Section 55.6.2.2*
0008	LFAST Correlator Control Register (COCR)	R/W	0x0000_000E	*Section 55.6.2.3*
000C	LFAST Test Mode Control Register (TMCR)	R/W	0x0000_0000	*Section 55.6.2.4*
0010	LFAST Auto Loopback Control Register (ALCR)	R/W	0x0000_0000	*Section 55.6.2.5*
0014	LFAST Rate Change Delay Control Register (RCDCR)	R/W	0x000F_0000	*Section 55.6.2.6*
0018	LFAST Wakeup Delay Control Register (SLCR)	R/W	0x1201_5F02	*Section 55.6.2.7*
001C	LFAST ICLC Control Register (ICR)	R/W	0x0000_0000	*Section 55.6.2.8*
0020	LFAST Ping Control Register (PICR)	R/W	0x0000_80CA	*Section 55.6.2.9*
0024–0028	Reserved			

Table 981. LFAST memory map (continued)

Offset from LFAST_BASE (hex)	Register	Access	Reset Value (hex)	Location
002C	LFAST Rx FIFO CTS Control Register (RFCR)	R/W	0x000F_0009	*Section 55.6.2.10*
0030	LFAST Tx Interrupt Enable Register (TIER)	R/W	0x0000_0000	*Section 55.6.2.11*
0034	LFAST Rx Interrupt Enable Register (RIER)	R/W	0x0000_0000	*Section 55.6.2.12*
0038	LFAST Rx ICLC Interrupt Enable Register (RIIER)	R/W	0x0000_0000	*Section 55.6.2.13*
003C	LFAST PLL Control Register (PLLCR)	R/W	0x0000_005C	*Section 55.6.2.14*
0040	LFAST LVDS Control Register (LCR)	R/W	0x0000_5024	*Section 55.6.2.15*
0044	LFAST Unsolicited Tx Control Register (UNSTCR)	R/W	0x0000_0000	*Section 55.6.2.16*
0048	LFAST Unsolicited Tx Data Registers 8 (UNSTDR8)	R/W	0x0000_0000	*Section 55.6.2.17*
004C	LFAST Unsolicited Tx Data Registers 7 (UNSTDR7)	R/W	0x0000_0000	*Section 55.6.2.17*
0050	LFAST Unsolicited Tx Data Registers 6 (UNSTDR6)	R/W	0x0000_0000	*Section 55.6.2.17*
0054	LFAST Unsolicited Tx Data Registers 5 (UNSTDR5)	R/W	0x0000_0000	*Section 55.6.2.17*
0058	LFAST Unsolicited Tx Data Registers 4 (UNSTDR4)	R/W	0x0000_0000	*Section 55.6.2.17*
005C	LFAST Unsolicited Tx Data Registers 3 (UNSTDR3)	R/W	0x0000_0000	*Section 55.6.2.17*
0060	LFAST Unsolicited Tx Data Registers 2 (UNSTDR2)	R/W	0x0000_0000	*Section 55.6.2.17*
0064	LFAST Unsolicited Tx Data Registers 1 (UNSTDR1)	R/W	0x0000_0000	*Section 55.6.2.17*
0068	LFAST Unsolicited Tx Data Registers 0 (UNSTDR0)	R/W	0x0000_0000	*Section 55.6.2.17*
006C–007C	Reserved			
0080	LFAST Global Status Register (GSR)	R	0x8000_0000	*Section 55.6.2.18*
0084	LFAST Ping Status Register (PISR)	R	0x0000_0000	*Section 55.6.2.19*
0088–0090	Reserved			
0094	LFAST Data Frame Status Register (DFSR)	R	0x0000_0000	*Section 55.6.2.20*
0098	LFAST Tx Interrupt Status Register (TISR)	R/W	0x0000_0000	*Section 55.6.2.21*
009C	LFAST Rx Interrupt Status Register (RISR)	R/W	0x0000_0000	*Section 55.6.2.22*
00A0	LFAST Rx ICLC Interrupt Status Register (RIISR)	R/W	0x0000_0000	*Section 55.6.2.23*
00A4	LFAST PLL and LVDS Status Register (PLLLSR)	R	0x0002_0003	*Section 55.6.2.24*
00A8	LFAST Unsolicited Rx Status Register (UNSRSR)	R/W	0x0000_0000	*Section 55.6.2.25*
00AC	LFAST Unsolicited Rx Data Register 8 (UNSRDR8)	R	0x0000_0000	*Section 55.6.2.26*
00B0	LFAST Unsolicited Rx Data Register 7 (UNSRDR7)	R	0x0000_0000	*Section 55.6.2.26*

Table 981. LFAST memory map (continued)

Offset from LFAST_BASE (hex)	Register	Access	Reset Value (hex)	Location
00B4	LFAST Unsolicited Rx Data Register 6 (UNSRDR6)	R	0x0000_0000	*Section 55.6.2.26*
00B8	LFAST Unsolicited Rx Data Register 5 (UNSRDR5)	R	0x0000_0000	*Section 55.6.2.26*
00BC	LFAST Unsolicited Rx Data Register 4 (UNSRDR4)	R	0x0000_0000	*Section 55.6.2.26*
00C0	LFAST Unsolicited Rx Data Register 3 (UNSRDR3)	R	0x0000_0000	*Section 55.6.2.26*
00C4	LFAST Unsolicited Rx Data Register 2 (UNSRDR2)	R	0x0000_0000	*Section 55.6.2.26*
00C8	LFAST Unsolicited Rx Data Register 1 (UNSRDR1)	R	0x0000_0000	*Section 55.6.2.26*
00CC	LFAST Unsolicited Rx Data Register 0 (UNSRDR0)	R	0x0000_0000	*Section 55.6.2.26*

55.6.2 Register descriptions

55.6.2.1 Mode Configuration Register (MCR)

Offset: 0x0000 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	MSEN(1)	0	0	0	0	0	0	IPGDBG	0	0	0	0	0	0	0	LSSEL
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	DRFEN	RXEN	TXEN	0	0	0	0	0	0	0	0	TXARBD	CTSEN(2)	0	DRFRST(3)	DATAEN
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0

1. Writable only once after the asynchronous reset.
2. Only writable when MCR[DRFEN] = 0.
3. Set by user software and cleared by system hardware.

Figure 922. Mode Configuration Register (MCR)

Table 982. MCR field descriptions

Field	Description
0 MSEN	LFAST Master or Slave mode Enable This bit selects either the LFAST master or slave functionality. This bit can be written only once after the asynchronous reset. 0 Enable the modules LFAST Slave functionality only. 1 Enable the modules LFAST Master functionality only.
7 IPGDBG	Control bit to enable support for IPG Debug mode This mode is indicated by assertion of IPG debug mode signal. 0 IPG debug mode enable signal will be ignored. 1 IPG debug mode enable signal will not be ignored.
15 LSSEL	Selects the fraction of sysclk in Low Speed Select mode Refer to *Section 55.10.2: Slow speed clock* for more detailed description. 0 Low Speed Mode in which where lfast_sysclk input is used to generate 4 phases of lfast_sysclk/2. 1 Low Speed Mode in which where lfast_sysclk input is used to generate 4 phases of lfast_sysclk/4.
16 DRFEN	LFAST Enable This bit enables/disables the reception and transfer of LFAST device. 0 LFAST is immediately disabled. All current/pending requests are terminated and the Tx and Rx data FIFOs are flushed. If this bit is cleared in the middle of a transmit/receive operation, then that operation is terminated immediately and nothing is transmitted/received further. All the programmable registers retain their values and status registers are cleared to their reset values. Registers read/write operations can be performed through the IPS Bus. 1 LFAST is Enabled.
17 RXEN	LFAST Receiver Enable This bit controls the reception of the frames and decoding on the LFAST device. This bit also disables the Rx LVDS LR. 0 Receiver Interface is disabled. If this bit is cleared during a data transfer, the current frame is received and then the Rx block is disabled. After the Rx block is disabled, all new frames from LFAST peer device are ignored. System Side Module Rx interface is not disabled by this bit. 1 Receiver Interface is Enabled.
18 TXEN	LFAST Transmitter Enable This bit controls the transmission of frames from the LFAST device and disables the Tx LVDS LD. This bit can also be set and cleared by LFAST slave hardware on reception of an ICLC command frame. 0 LFAST transmitter Interface is disabled. No new request is accepted but ongoing request is served. 1 Transmitter is Enabled.
27 TXARBD	Tx Arbiter Disable This bit enables/disables the Tx block arbiter. Current frame transfer is completed, but new frame requests are ignored. 0 Enable Tx arbiter and framer. When enabled it takes all the frame request and services based on priority. 1 Disable Tx arbiter and framer. All frame requests are ignored.

Table 982. MCR field descriptions (continued)

Field	Description
28 CTSEN	CTS Enable This bit defines the Push-Pull mode of the LFAST devices receiver. This bit is used to enable/disable CTS mode of the Tx block. 0 CTS mode is disabled. Indicates that the device is in Push mode. The CTS bit of frames transmitted is 1. The CTS bit does not represent the status of Rx FIFO. 1 CTS mode is enabled. The CTS bit of all transmit frames is set when the Rx FIFO empty space is on or above higher threshold, and cleared when the Rx FIFO empty space is on or below lower threshold.
30 DRFRST	LFAST Soft Reset This bit is automatically cleared after reset. 0 No soft reset. 1 Soft reset to LFAST is asserted. When set it causes a reset of the LFAST module; all the registers will be reset to their default values and all the FIFOs will be flushed.
31 DATAEN	DATA Frame Enable This bit enables/disables the transmission and reception of data frames between the LFAST master and slave devices. 0 Data frame transmission and reception is disabled. Tx data frame requests are ignored by the transmitter. Frame with LCT of data frame is ignored by the receiver. 1 Data frame transmission and reception is enabled. Tx data frame requests are serviced by the transmitter. Frame with LCT of data frame is received and placed into the Rx data FIFO.

55.6.2.2 Speed Control Register (SCR)

The SCR is used to configure the Rx and Tx data rate of the LFAST.

Offset: 0x0004 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	DRMD(1)
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	RDR	0	0	0	0	0	0	0	TDR
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. Only writable when MCR[DRFEN] = 0.

Figure 923. Speed Control Register (SCR)

Table 983. SCR field descriptions

Field	Description
15 DRMD	Data Rate Controller mode Defines the mode set for LFAST slave device by the software or the LFAST master. 0 The software controls the Data Rate controller mode. In LFAST Slave the ICLC frames for rate change have no effect on the Data rate. 1 In LFAST Slave the reception of ICLC frame for rate change sets appropriate speed mode. In LFAST Master the SCR[DRMD] should be 0.
23 RDR	Receiver Data Rate This bit defines the receiver data rate. For LFAST Master: – The software should program this bit only after transmission of an ICLC frame changing the speed mode of the slaves Tx interface. For LFAST Slave: – When SCR[DRMD] = 1, the hardware programs this bit on reception of ICLC frame for changing speed mode of Slaves Rx interface. – The software can program this bit when SCR[DRMD] = 0. – This bit is cleared on MCR[DRFEN] negation. 0 Data rate of Rx block is low speed. 1 Data rate of Rx block is high speed.
31 TDR	Transmit Data Rate This bit defines the transmitter data rate. For LFAST Master: – The software should program this bit only after transmission of an ICLC frame changing the speed mode of the slaves Rx interface. For LFAST Slave: – When SCR[DRMD] = 1, the hardware writes this bit on receipt of an ICLC frame changing the speed mode of the slaves Tx interface. – The software can program this bit when SCR[DRMD] = 0. – This bit is cleared onMCR[DRFEN] = 0. 0 Data rate of Tx block is low speed. 1 Data rate of Tx block is high speed.

55.6.2.3 Correlator Control Register (COCR)

The COCR is used to select the sampler data path and the number of bits of correlation to be used.

Offset: 0x0008 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	SMPSEL(1)								0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	CORRTH(1)			PHSSEL(1)
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	0

1. Only writable only when MCR[RXEN] = 0.

Figure 924. Correlator Control Register (COCR)

Table 984. COCR field descriptions

Field	Description
0:7 SMPSEL	Sampler Data Path Selector (overrides the correlator selection) Defines the sampler data path to be activated at all the times. All the bits should be 0 (00h) for Sampler Data Path to be selected by the correlator. In Low Speed mode only Sampler Data Paths 0-3 are valid. 0x00 Sampler Data Path selected by correlator 0x01 Sampler Data Path 0 selected 0x02 Sampler Data Path 1 selected 0x04 Sampler Data Path 2 selected 0x08 Sampler Data Path 3 selected 0x10 Sampler Data Path 4 selected 0x20 Sampler Data Path 5 selected 0x40 Sampler Data Path 6 selected others Sampler Data Path 7 selected
28:30 CORRTH	Correlator threshold level. Defines the correlation threshold level. 000 9 Bits of correlation 001 10 Bits of correlation …. 110 15 Bits of correlation 111 16 Bits of correlation
31 PHSSEL	Polyphase 8 or 4 phase selection Defines the number of phases for the polyphase generator used. This bit is ignored in low speed mode since only 4 Phases are used in low speed mode. In High Speed mode phase 0, 2, 4 and 6 are used when 4 phase mode is selected. 0 8 phases 1 4 phases

55.6.2.4 Test Mode Control Register (TMCR)

The TMCR enables and configures the LFAST clock test and loopback modes

Offset: 0x000C Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	CLKTST	LPON	0	0	0	0	0	LPMOD(1)		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	LPFRMTH(1)															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. Writable only when TMCR[LPON] = 0.

Figure 925. Test Mode Control Register (TMCR)

Table 985. TMCR field descriptions

Field	Description
6 CLKTST	Clock Test mode The software can define when the clock test mode is enabled. This bit can also be set and cleared by LFAST slave hardware on reception of an ICLC command. This field can only be written when MCR[DRFEN] = 1. 1 Clock Test mode enabled 0 Clock Test mode disabled
7 LPON	Loopback mode Logic Enable The software can define when the loopback logic is enabled. This bit can also be written by LFAST slave hardware on reception of an ICLC command. This field can only be written when MCR[DRFEN] = 1. 1 Loopback mode is enabled. 0 Loopback mode is disabled.
13:15 LPMOD	Loopback mode Defines the type of loopback mode enabled. 000 Rx loopback 001 Rx LVDS loopback 010 Tx loopback without automatic frame generation 011 Tx loopback with automatic frame generation 100 Tx LVDS loopback (external) with automatic frame generation Others - Reserved (should not be written by the software)
16:31 LPFRMTH	Loopback check mode valid pass frames threshold value. Defines the number of frames to verify before setting GCR[LPFPDV] when running in Automatic Loopback Frame mode. The loopback frame is considered pass when the payload is CBh, header is 13h and sync is valid. This mode is valid only when TMCR[LPMOD] = 011b or TMCR[LPMOD] = 100b. 0x0000: Reserved (Not to be used) 0x0001: Check 1 frame have correct sync, header and payload. … 0xFFFE: Check 65534 frames have correct sync, header and payload. 0xFFFF: Check 65535 frames have correct sync, header and payload.

55.6.2.5 Auto Loopback Control Register (ALCR)

Offset: 0x0010 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	LPCNTEN(1)
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	LPFMCNT(1)															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. Writable only when TMCR[LPON] = 0.

Figure 926. Auto Loopback Control Register (ALCR)

Table 986. ALCR field descriptions

Field	Description
15 LPCNTEN	Auto Loopback Frame Transmission Count Enable Enables fixed number of auto predefined loopback frame transmission. 0 Infinite predefined loopback frame transmission enabled 1 Fixed count of predefined loopback frame transmission enabled
16:31 LPFMCNT	Auto Loopback Frame Transmission Count Defines the number of predefined auto loopback frames to be sent. The predefined loopback frame has payload CBh, header 13h and valid sync. This mode is valid if TMCR[LPMOD] = 011b or TMCR[LPMOD] = 100b. 0x0000: Reserved (Not to be used) 0x0001: Send 1 frame with correct sync, header and payload. ... 0xFFFE: Send 65534 frames with correct sync, header and payload. 0xFFFF: Send 65535 frames with correct sync, header and payload.

55.6.2.6 Rate Change Delay Control Register (RCDCR)

Offset: 0x0014 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	DRCNT(1)			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1

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

1. Only writable when MCR[DRFEN] = 1.

Figure 927. Rate Change Delay Control Register (RCDCR)

Table 987. RCDCR field description

Field	Description
12:15 DRCNT	Data Rate Controller Counter Value Defines the number of cycles of Phase 0 clock needed by the Tx interface Data rate change controller to switch from one speed mode to another. The arbitrator ignores all requests during this period. 0x0: 0 cycles 0x1: 1 cycles ... 0xE: 14 cycles 0xF: 15 cycles

55.6.2.7 Wakeup Delay Control Register (SLCR)

Offset: 0x0018 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	HSCNT(1)								0	0	0	0	LSCNT(1)			
W																
Reset	0	0	0	1	0	0	1	0	0	0	0	0	0	0	0	1

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	HWKCNT(1)								0	0	0	0	LWKCNT(1)			
W																
Reset	0	1	0	1	1	1	1	1	0	0	0	0	0	0	1	0

1. Only writable when MCR[DRFEN] = 0.

Figure 928. Wakeup Delay Control Register (SLCR)

Table 988. SLCR field descriptions

Field	Description
0:7 HSCNT	High Speed Sleep mode Exit Time Defines 1/4 of the number of the high speed clock cycle wait after the negation of the LVDS LD sleep signal, and before the start of new frame. This wait is the summation of settling time of the Line Driver (LD) after negation of its sleep signal, and the wakeup time of the LR of the peer LFAST. 0x00: cycle 0x01: 1 cycle … 0x12: 18 cycles (200 ns + 8 cycles of High speed clock) … 0xFE: 254 cycles 0xFF: 255 cycles
12:15 LSCNT	Low Speed Sleep mode Exit Time Defines 1/4 of the number of Low speed clock cycle wait after the negation of LVDS LD sleep signal, and before the start of new frame. This wait is the summation of settling time of LD after negation of LVDS LD sleep signal, and the wakeup time of the LR of the peer LFAST. 0x0: 0 cycle 0x1: 1 cycle (200ns + 1 cycle of Low speed clock) … 0xE: 14 cycles 0xF: 15 cycles
16:23 HWKCNT	Wake Up time for the LD Defines the 1/4 of the number of High speed clock cycles used during the wakeup of the LD from shutdown mode. This is time required by the LD to move from moving from Shutdown to Normal State in High speed mode. 0x00: 0 cycles 0x01: 1 cycle … 0x5F: 95 cycles (1.18 us) … 0xFF: 255 cycles Maximum
28:31 LWKCNT	Wake Up time for the LD Defines the 1/4 of the number Low speed clock used during the wakeup of the LD from shutdown mode. This is time required by the LD to move from Shutdown to Normal State in Low speed mode. 0x0: 0 cycle 0x1: 1 cycle 0x2: 2 cycles (1.18 us) … 0xE: 14 cycles 0xF: 15 cycles

55.6.2.8 ICLC Control Register (ICR)

Offset: 0x001C Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	ICLCSEQ	SNDICLC(1)
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	ICLCPLD(2)							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. Set by user software and cleared by system hardware.
2. Writable only when ICR[SNDICLC] = 0.

Figure 929. ICLC Control Register (ICR)

Table 989. ICR field descriptions

Field	Description
14 ICLCSEQ	ICLC enabled This bit should be set whenever the software is performing a series of ICLC frame transfers. This bit ensures only ICLC frame transmission request is serviced, other pending frame requests are ignored. 1 The software is performing ICLC frame transfers. Only the ICLC frames will be scheduled during this period. All the other frames will be scheduled only after ICR[ICLCSEQ] = 0. 0 Single ICLC frame transfer
15 SNDICLC	ICLC frame request This bit is set to initiate the transfer of ICLC frame by LFAST master. This bit should be set (ICR[SNDICLC] = 1) after writing the required ICLC payload to be transmitted in the ICLCPLD field of the register. This bit is self clearing, which will be cleared when ICLC frame transfer is complete. 1 Valid ICLC frame for transfer. 0 No Valid ICLC frame for transfer.
24:31 ICLCPLD	ICLC Payload This field is used to program the payload of the ICLC frame to be transmitted. New ICLC payload should be set when ICR[SNDICLC] = 0. Refer to *Table 1014* for supported ICLC payloads.

55.6.2.9 Ping Control Register (PICR)

Offset: 0x0020 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	PNGREQ(1)
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	PNGAUTO(2)	0	0	0	0	0	0	0	PNGPYLD(2)							
W																
Reset	1	0	0	0	0	0	0	0	1	1	0	0	1	0	1	0

1. Set by user software and cleared by system hardware.
2. Only writable when MCR[DRFEN] = 0.

Figure 930. Ping Control Register (PICR)

Table 990. PICR field descriptions

Field	Description
15 PNGREQ	Ping Response Frame Request This bit is set to initiate the transfer of Ping response frame. Cleared after transmission of Ping response frame. This field can only be written when MCR[DRFEN] = 1. 1 Ping response frame transmission request is queued. 0 No pending Ping response frame transmission request.
16 PNGAUTO	Ping Response Enable Defines when ping response should be automatically sent on reception of Ping ICLC frame from LFAST master 0 Ping response should not be automatically sent. 1 Ping response should be automatically sent.
24:31 PNGPYLD	LFAST Slave: Defines the LFAST slaves ping reply frame payload content. LFAST Master: Defines the expected payload of ping response frame. Used for comparison with ping frame received.

55.6.2.10 Rx FIFO CTS Control Register (RFCR)

Offset: 0x002C Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	RCTSMX(1)					
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	RCTSMN(1)					
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1

1. Only writable when MCR[DRFEN] = 0.

Figure 931. Rx FIFO CTS Control Register (RFCR)

Table 991. RFCR field descriptions

Field	Description
10:15 RCTSMX	Rx FIFO Maximum Threshold Defines the condition for CTS bit of frames to be negated.
26:31 RCTSMN	Rx FIFO Minimum Threshold Defines the condition for CTS bit of frames to be set.

55.6.2.11 Tx Interrupt Enable Register (TIER)

Offset: 0x0030 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	TXIIE	TXOVIE
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	TXPNGIE	0	TXUNSIE	TXICLCIE	TXDTIE
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 932. Tx Interrupt Enable Register (TIER)

Table 992. TIER field descriptions

Field	Description
14 TXIIE	Tx Data Interface Not Enabled (Mask) Enables or disables the interrupt. Tx Data Interface not enabled and a frame is ready to be transmitted. 0 Interrupt is disabled. 1 Interrupt is enabled.
15 TXOVIE	Transmit Data FIFO Overflow Interrupt Enable 0 Interrupt is disabled. 1 Interrupt is enabled.
27 TXPNGIE	Ping Response Frame Transmitted Interrupt Enable 0 Interrupt is disabled. 1 Interrupt is enabled.
29 TXUNSIE	Unsolicited Frame transmitted Interrupt Enable 0 Interrupt is disabled. 1 Interrupt is enabled.
30 TXICLCIE	ICLC Frame transmitted Interrupt Enable 0 Interrupt is disabled. 1 Interrupt is enabled.
31 TXDTIE	Data Frame transmitted Interrupt Enable 0 Interrupt is disabled. 1 Interrupt is enabled.

55.6.2.12 Rx Interrupt Enable Register (RIER)

Offset: 0x0034 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	RXUOIE	RXMNIE	RXMXIE	RXUFIE	RXOFIE	RXSZIE	RXICIE	RXLCEIE
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	RXCTSIE	RXDIE	RXUNSIE	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 933. Rx Interrupt Enable Register (RIER)

Table 993. RIER field descriptions

Field	Description
8 RXUOIE	Unsolicited frame register overflow Indicates existing unsolicited frame has not been read and a new unsolicited frame has arrived. 0 Interrupt is disabled. 1 Interrupt is enabled.
9 RXMNIE	Rx Data FIFO Min Threshold reached 0 Interrupt is disabled. 1 Interrupt is enabled.
10 RXMXIE	Rx Data FIFO Max Threshold reached 0 Interrupt is disabled. 1 Interrupt is enabled.
11 RXUFIE	Rx Data FIFO Underflow 0 Interrupt is disabled. 1 Interrupt is enabled.
12 RXOFIE	Rx Data FIFO Overflow 0 Interrupt is disabled. 1 Interrupt is enabled.
13 RXSZIE	Frame with unsupported frame size received Valid frame sizes are defined in *Table 1018*. 0 Interrupt is disabled. 1 Interrupt is enabled.
14 RXICIE	Invalid ICLC code Received 0 Interrupt is disabled. 1 Interrupt is enabled.
15 RXLCEIE	Invalid Logical Channel Type 0 Interrupt is disabled. 1 Interrupt is enabled.
28 RXCTSIE	Frame with CTS bit Low Received 0 Interrupt is disabled. 1 Interrupt is enabled.
29 RXDIE	Data frame received 0 Interrupt is disabled. 1 Interrupt is enabled.
30 RXUNSIE	Unsolicited Frame received 0 Interrupt is disabled. 1 Interrupt is enabled.

55.6.2.13 Rx ICLC Interrupt Enable Register (RIIER)

Offset: 0x0038 Access: User read/write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	ICPFIE	ICPSIE	ICPRIE	ICTOIE	ICLPIE	ICCTIE	ICTDIE	ICTEIE	ICRFIE	ICRSIE	ICTFIE	ICTSIE	ICPOFIE	ICPONIE
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 934. Rx ICLC Interrupt Enable Register (RIIER)

Table 994. RIIER field descriptions

Field	Description
18 ICPFIE	Ping Frame Response Payload Mismatch 0 Interrupt is disabled. 1 Interrupt is enabled.
19 ICPSIE	Ping Frame Response successful 0 Interrupt is disabled. 1 Interrupt is enabled.
20 ICPRIE	ICLC frame for Ping Frame Request received 0 Interrupt is disabled. 1 Interrupt is enabled.
21 ICTOIE	ICLC frame for Test mode off received 0 Interrupt is disabled. 1 Interrupt is enabled.
22 ICLPIE	ICLC frame for Loopback On received 0 Interrupt is disabled. 1 Interrupt is enabled.
23 ICCTIE	ICLC frame for Clk Test mode on received 0 Interrupt is disabled. 1 Interrupt is enabled.
24 ICTDIE	ICLC frame for LFAST Slaves Tx Interface Disable received 0 Interrupt is disabled. 1 Interrupt is enabled.
25 ICTEIE	ICLC frame for LFAST Slaves Tx Interface Enable received 0 Interrupt is disabled. 1 Interrupt is enabled.
26 ICRFIE	ICLC frame for LFAST Slaves Rx Interface fast mode switch received 0 Interrupt is disabled. 1 Interrupt is enabled.

Table 994. RIIER field descriptions (continued)

Field	Description
27 ICRSIE	ICLC frame for LFAST Slaves Rx Interface slow mode switch received 0 Interrupt is disabled. 1 Interrupt is enabled.
28 ICTFIE	ICLC frame for LFAST Slaves Tx Interface fast mode switch received 0 Interrupt is disabled. 1 Interrupt is enabled.
29 ICTSIE	ICLC frame for LFAST Slaves Tx Interface slow mode switch received 0 Interrupt is disabled. 1 Interrupt is enabled.
30 ICPOFIE	ICLC frame for PLL OFF received 0 Interrupt is disabled. 1 Interrupt is enabled.
31 ICPONIE	ICLC frame for PLL ON received 0 Interrupt is disabled. 1 Interrupt is enabled.

55.6.2.14 PLL Control Register (PLLCR)

This register exists only when LFAST controls an external PLL.

Offset: 0x003C Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	IPTMOD			0	0	0	0	0	0	0	0	0	0	0	SWPOFF(1)	SWPON(1)
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	REFINV	LPCFG		0	0	PLCKCW		FDIVEN	FBDIV(2)						PREDIV(2)	
W																
Reset	0	0	0	0	0	0	0	0	0	1	0	1	1	1	0	0

1. Set by user software and cleared by system hardware.
2. Only writable when MCR[DRFEN] = 0.

Figure 935. PLL Control Register (PLLCR)

Table 995. PLLCR field descriptions

Field	Description
0:2 IPTMOD	Test mode programmability 000 Functional mode 001 Closed Loop 1 010 Force Vctrl 011 Charge Pump Up 100 Charge Pump Up Internal Test 101 Charge Pump Idle 110 Charge Pump Down 111 Closed Loop 2
14 SWPOFF	Software signal to turn OFF the PLL 0 No effect 1 PLL will be turned OFF.
15 SWPON	Software signal to turn ON the PLL 0 No effect 1 PLL will be turned ON
16 REFINV	Inverts reference clock edge to PFD If System PLL PFD using same reference clock, enabling this feature will minimize noise injection crosstalk via the substrate. 0 Not inverted 1 Invert
17:18 LPCFG	PLL Loop Optimization. Adjusts charge pump bias current. Higher current increases PLL bandwidth, decreasing jitter while degrading stability. 00 1 x IBASE 01 0.5 x IBASE 10 1.5 x IBASE 11 2 x IBASE
21:22 PLCKCW	PLL Lock Ready Count Width Defines the number of cycles PLL waits for before asserting lock_ready flag High. 00 1040 cycles 01 520 cycles 10 320 cycles 11 200 cycles
23 FDIVEN	Enable fraction division mode in feedback divider Enables the division of vco clock output by a factor of (FBDIV + 0.5). 0 Fraction division mode not enabled 1 Fraction division mode enabled

Table 995. PLLCR field descriptions (continued)

Field	Description
24:29 FBDIV	Feedback Division factor for VCO output clock 0x00: No clock output 0x01: Reserved … 0x0A: Reserved 0x0B: Divide by 11 (11.5, if FDIVEN = 1) … 0x1F: Divide by 31 (31.5, if FDIVEN = 1) 0x20: Divide by 32 (32.5, if FDIVEN = 1) 0x21: Reserved … 0x3F: Reserved
30:31 PREDIV	Division factor for PLL Reference Clock input 00 Direct clock passed 01 Divide by 2 10 Divide by 3 11 Divide by 4

55.6.2.15 LVDS Control Register (LCR)

Offset: 0x0040 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	SWWKLD(1)	SWSLPLD(1)	SWWKLR(1)	SWSLPLR(1)	SWOFFLD(1)	SWONLD(1)	SWOFFLR(1)	SWONLR(1)
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	LVRXOFF	LVTXOE	TXCMUX	LVRFEN	LVLPEN	0	0	0	0	0	LVRXOP_TR	Reserved	LVRXOP_BR	LVTXOP	LVCKSS	LVCKP
W																
Reset	0	1	0	1	0	0	0	0	0	0	1	0	0	1	0	0

1. Set by user software and cleared by system hardware.

Figure 936. LVDS Control Register (LCR)

Table 996. LCR field description

Field	Description
8 SWWKLD	Software signal to take LVDS LD out of Sleep mode This bit can be changed only when DRFEN is high. 0 No effect 1 LVDS LD will be taken out of sleep (provided no other source is trying to put it in sleep).
9 SWSLPLD	Software signal to put LVDS LD into Sleep mode This bit can be changed only when DRFEN is high. 0 No effect 1 LVDS LD will be put in sleep.
10 SWWKLR	Software signal to take LVDS LR out of Sleep mode This bit can be changed only when DRFEN is high. 0 No effect 1 LVDS LR will be taken out of sleep (provided no other source is trying to put it in sleep).
11 SWSLPLR	Software signal to put LVDS LR into Sleep mode This bit can be changed only when DRFEN is high. 0 No effect 1 LVDS LR will be put in sleep (provided no other source is trying to wake it up).
12 SWOFFLD	Software signal to turn OFF the LVDS LD This bit can be changed only when DRFEN is high. 0 No effect 1 LVDS LD will be turned OFF (provided no other source is trying to turn the LD ON).
13 SWONLD	Software signal to turn ON the LVDS LD This bit can be changed only when DRFEN is high. 0 No effect 1 LVDS LD will be turned ON.
14 SWOFFLR	Software signal to turn OFF the LVDS LR This bit can be changed only when DRFEN is high. 0 No effect 1 LVDS LR will be turned OFF (provided no other source is trying to turn the LR ON).
15 SWONLR	Software signal to turn ON the LVDS LR This bit can be changed only when DRFEN is high. 0 No effect 1 LVDS LR will be turned ON.
16 LVRXOFF	Indicates the value driven onto LVDS LR output when in shutdown mode.
17 LVTXOE	LVDS LD output buffer enable 0 LVDS LD output buffer enable is disabled. 1 LVDS LD output buffer enabled

Table 996. LCR field description (continued)

Field	Description
18 TXCMUX	Tx and Clock Mux The bit can be used to bring out PLL Phase 0 clock on Tx LVDS pad. 0 No effect 1 PLL Phase 0 clock will be brought out to Tx LVDS pad.
19 LVRFEN	LVDS pad reference enable 0 LVDS reference pad disabled 1 LVDS reference pad enabled
20 LVLPEN	Tx LVDS internal loopback enable The internal loopback is not intended for board level functionality, so this feature should not be implemented. 0 Tx LVDS normal mode enabled 1 Tx LVDS internal loopback mode enabled
26 LVRXOP_TR	Used to enable or disable the on-chip receiver termination resistor in LFAST mode (applies to LVDS pad use for LFAST only). 0 Disable on-chip LFAST receiver termination 1 Enable on-chip LFAST receiver termination
28 LVRXOP_BR	Used to set the bias current for the receiver in LFAST mode. It is recommended to always write 1 to this bit when using the LFAST interface. Writing 0 will allow for a small power savings during lower baud rates (applies to LVDS pad use for LFAST only). 0 Use for LFAST receiver baud rates less than maximum baud rate 1 Required for LFAST receiver maximum baud rate
29 LVTXOP	Control signal for LFAST and Micro-Second Bus selection 0 Micro-Second Bus selection 1 LFAST Bus selection
30 LVCKSS	LVDS Clock Sync Select This bit is used to adjust the LVDS data sampling to the duty cycle of the clock. It is recommended that a value of zero is used in all cases. A value of one is reserved for specific devices/revisions with different clock timing. 0 Normal clock used to sample the LVDS data 1 Adjusted clock used to sample the LVDS data
31 LVCKP	LVDS clock select This bit is used for selecting use of direct pll clock or inverted pll clock in LVDS. 0 Direct pll clock to be used inside the LVDS 1 Inverted pll clock to be used inside the LVDS

55.6.2.16 Unsolicited Tx Control Register (UNSTCR)

Offset: 0x0044 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	USNDRQ(1)
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	UNSHDR(2)						
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. Set by user software and cleared by system hardware.
2. Only writable when UNSTCR[USNDRQ] = 0.

Figure 937. Unsolicited Tx Control Register (UNSTCR)

Table 997. UNSTCR field descriptions

Field	Description
15 USNDRQ	Tx Unsolicited send request. This field can only be written when MCR[DRFEN] = 1. 0 No valid Unsolicited frame exists 1 Valid Unsolicited frame exists for transmission
25:31 UNSHDR	Tx Unsolicited message header

55.6.2.17 LFAST Unsolicited Tx Data Registers (UNSTDR*n*)

Offset: 0x0048 (UNSTDR8)
0x004C (UNSTDR7)
0x0050 (UNSTDR6)
0x0054 (UNSTDR5)
0x0058 (UNSTDR4)
0x005C(UNSTDR3)
0x0060 (UNSTDR2)
0x0064 (UNSTDR1)
0x0068 (UNSTDR0)

Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	UNTXD(1)															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	UNTXD(1)															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. Only writable when UNSTCR[USNDRQ] = 0.

Figure 938. Unsolicited Tx Data Registers (UNSTDR8–UNSTDR0)

Table 998. UNSTDR8–UNSTDR0 field descriptions

Field	Description
0:31 UNTXD	Unsolicited Transmit Data 8–0 This represents 9 registers for Unsolicited transmit data. The first bit to transmitted as part of the payload will be from UNTXD8[31], second bit from UNTXD8[30], and so on. So the last bit transmitted will be from UNTXD0[0] in case of 288 bit payload.

55.6.2.18 Global Status Register (GSR)

Offset: 0x0080 Access: User read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	DUALMD	0	0	0	0	0	0	0	0	0	0	0	0	LRMD	LDSM	DRSM
W																
Reset	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	LPTXDN	LPFPDV	LPCPDV	LPCHDV	LPCSDV
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 939. Global Status Register (GSR)

Table 999. GSR field descriptions

Field	Description
0 DUALMD	Indicates the LFAST module is in Dual mode 0 LFAST Module Slave only mode 1 LFAST Module in dual mode
13 LRMD	Indicates if the Rx Controller is idle/active and that the Rx clocks are enabled. In functional mode this will always be active. 0 Rx Controller is in Idle state. 1 Rx Controller is active.
14 LDSM	Transmit Interface Data Rate Status Indicates the current speed rate of the Tx controller. 0 Data rate of LOW speed mode 1 Data rate of HIGH speed mode
15 DRSM	Receive Interface Data Rate Status Indicates the current speed rate of the Rx controller. 0 Data rate of LOW speed mode 1 Data rate of HIGH speed mode
27 LPTXDN	Auto loopback frame transmission count reached The Count of frame is defined by *Auto Loopback Control Register (ALCR)* LPFMCNT. 0 Auto loopback frame transmission count not reached 1 Auto loopback frame transmission count reached
28 LPFPDV	Loopback frame pass threshold reached 0 Pass frame threshold not reached 1 Pass frame threshold achieved

Table 999. GSR field descriptions (continued)

Field	Description
29 LPCPDV	Valid payload received during loopback check mode Indicates whether the Loop back frame received payload is 0xCB. 0 Payload received is not 0xCB. 1 Payload received is 0xCB.
30 LPCHDV	Valid header received during loopback check mode Indicates whether the Loop back frame received header is 0x13. 0 Header received is not 0x13. 1 Header received is 0x13.
31 LPCSDV	Valid synchronization received 0 Valid Synchronization pattern not detected 1 Valid Synchronization pattern detected

55.6.2.19 Ping Status Register (PISR)

Offset: 0x0084 Access: User read-only

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	RXPNGD							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 940. Ping Status Register (PISR)

Table 1000. PISR field descriptions

Field	Description
24:31 RXPNGD	Ping Data Register In LFAST Master mode the Ping response ICLC frame received is stored into this register.

55.6.2.20 Data frame Status Register (DFSR)

Offset: 0x0094 Access: User read only

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	RXDCNT						0	0	0	0	0	RXFCNT		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R			TXDCNT						0	0	0	0	0	TXFCNT		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 941. Data frame Status Register (DFSR)

Table 1001. DFSR field descriptions

Field	Description
2:7 RXDCNT	Unread Rx Frame Data Count Indicates the number of unread data stored in the Rx Data FIFO.
13:15 RXFCNT	Unread Rx Frame Count Indicates the number of unread data frames stored in the Rx Data FIFO.
18:23 TXDCNT	Unread Tx Frame Data Count Indicates the number of unread data stored in the Tx Data FIFO.
29:31 TXFCNT	Unread Tx Frame Count Count of pending Data Frames programmed by System Side Module.

55.6.2.21 Tx Interrupt Status Register (TISR)

Offset: 0x0098 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	TXIEF	TXOVF
W															w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	TXPNGF	0	TXUNSF	TXICLCF	TXDTF
W												w1c		w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 942. Tx Interrupt Status Register (TISR)

Table 1002. TISR field descriptions

Field	Description
14 TXIEF	Tx Data Interface not enabled Tx Data Interface not enabled and a frame is ready to be transmitted. 0 Interrupt event has not occurred. 1 Interrupt event has occurred.
15 TXOVF	Transmit Data FIFO Overflow Interrupt 0 Interrupt event has not occurred. 1 Interrupt event has occurred.
27 TXPNGF	Ping response frame transmitted interrupt 0 Interrupt event has not occurred. 1 Interrupt event has occurred.
29 TXUNSF	Unsolicited Frame transmitted Interrupt 0 Interrupt event has not occurred. 1 Interrupt event has occurred.
30 TXICLCF	ICLC Frame transmitted Interrupt 0 Interrupt event has not occurred. 1 Interrupt event has occurred.
31 TXDTF	Data Frame transmitted Interrupt 0 Interrupt event has not occurred. 1 Interrupt event has occurred.

55.6.2.22 Rx Interrupt Status Register (RISR)

Offset: 0x009C Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	RXUOF	RXMNF	RXMXF	RXUFF	RXOFF	RXSZF	RXICF	RXLCEF
W									w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	RXCTSF	RXDF	RXUNSF	0
W													w1c	w1c	w1c	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 943. Rx Interrupt Status Register (RISR)

Table 1003. RISR field descriptions

Field	Description
8 RXUOF	Unsolicited frame register overflow Indicates existing unsolicited frame has not been read and a new unsolicited frame has arrived. 0 Interrupt event has not occurred. 1 Interrupt event has occurred.
9 RXMNF	Rx Data FIFO Min Threshold reached 0 Interrupt event has not occurred. 1 Interrupt event has occurred.
10 RXMXF	Rx Data FIFO Max Threshold reached 0 Interrupt event has not occurred. 1 Interrupt event has occurred.
11 RXUFF	Rx Data FIFO Underflow 0 Interrupt event has not occurred. 1 Interrupt event has occurred.
12 RXOFF	Rx Data FIFO Overflow 0 Interrupt event has not occurred. 1 Interrupt event has occurred.
13 RXSZF	Frame with unsupported frame size received 0 Interrupt event has not occurred. 1 Interrupt event has occurred: On reception of frame with payload size for a frame other than mentioned in *Table 1018*.
14 RXICF	Invalid ICLC code Received 0 Interrupt event has not occurred. 1 Interrupt event has occurred.
15 RXLCEF	Invalid Logical Channel Type 0 Interrupt event has not occurred. 1 Interrupt event has occurred: On reception of frame other than mentioned in *Table 1018*.
28 RXCTSF	Frame with CTS bit Low Received 0 Interrupt event has not occurred. 1 Interrupt event has occurred.
29 RXDF	Data frame received 0 Interrupt event has not occurred. 1 Interrupt event has occurred.
30 RXUNSF	Unsolicited Frame received 0 Interrupt event has not occurred. 1 Interrupt event has occurred.

55.6.2.23 Rx ICLC Interrupt Status Register (RIISR)

Offset: 0x00A0 Access: User read/write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	ICPFF	ICPSF	ICPRF	ICTOF	ICLPF	ICCTF	ICTDF	ICTEF	ICRFF	ICRSF	ICTFF	ICTSF	ICPOFF	ICPONF
W			w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 944. Rx ICLC Interrupt Status Register (RIISR)

Table 1004. RIISR field descriptions

Field	Description
18 ICPFF	Ping Frame Response Payload mismatch 0 Interrupt event has not occurred. 1 Interrupt event has occurred.
19 ICPSF	Ping Frame Response successful 0 Interrupt event has not occurred. 1 Interrupt event has occurred.
20 ICPRF	ICLC Ping Frame Request received 0 Interrupt event has not occurred. 1 Interrupt event has occurred.
21 ICTOF	ICLC frame for Test mode off received 0 Interrupt event has not occurred. 1 Interrupt event has occurred.
22 ICLPF	ICLC frame for Loopback On received 0 Interrupt event has not occurred. 1 Interrupt event has occurred.
23 ICCTF	ICLC frame for Clk Test mode received 0 Interrupt event has not occurred. 1 Interrupt event has occurred.
24 ICTDF	ICLC frame for LFAST Slaves Tx Interface Disable received 0 Interrupt event has not occurred. 1 Interrupt event has occurred.
25 ICTEF	ICLC frame for LFAST Slaves Tx Interface Enable received 0 Interrupt event has not occurred. 1 Interrupt event has occurred.
26 ICRFF	ICLC frame for LFAST Slaves Rx Interface fast mode switch received 0 Interrupt event has not occurred. 1 Interrupt event has occurred.

Table 1004. RIISR field descriptions (continued)

Field	Description
27 ICRSF	ICLC frame for LFAST Slaves Rx Interface slow mode switch received 0 Interrupt event has not occurred. 1 Interrupt event has occurred.
28 ICTFF	ICLC frame for LFAST Slaves Tx Interface fast mode switch received 0 Interrupt event has not occurred. 1 Interrupt event has occurred.
29 ICTSF	ICLC frame for LFAST Slaves Tx Interface slow mode switch received 0 Interrupt event has not occurred. 1 Interrupt event has occurred.
30 ICPOFF	ICLC frame for PLL OFF received 0 Interrupt event has not occurred. 1 Interrupt event has occurred.
31 ICPONF	ICLC frame for PLL ON received 0 Interrupt event has not occurred. 1 Interrupt event has occurred.

55.6.2.24 PLL and LVDS Status Register (PLLLSR)

Offset: 0x00A4 Access: User read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	PLLDIS	PLDCR
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	LRSLPS	LDSLPS	LDPDS	LRPDS
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1

Figure 945. PLL and LVDS Status Register (PLLLSR)

Table 1005. PLLLSR field descriptions

Field	Description
14 PLLDIS	PLL disable Status When asserted, PLL is put in the power down state. 0 PLL disable signal is negated. 1 PLL disable signal is asserted.
15 PLDCR	PLL Lock Delay Counter Ready When asserted this bit indicates that the PLL is locked after N number of reference/PLLCR[PREDIV] cycles. 0 PLL Lock delay counter is not decremented to 0. 1 PLL Lock delay counter is decremented to 0.
28 LRSLPS	This bit indicates the real time status of the LR sleep signal. 0 LR sleep signal is negated. 1 LR sleep signal is asserted.
29 LDSLPS	This bit indicates the real time status of LD sleep signal. 0 LD power sleep signal is negated. 1 LD power sleep signal is asserted.
30 LDPDS	This bit indicates the real time status of LD power down signal. When asserted, LD is put in the power down state. 0 LD power down signal is negated. 1 LD power down signal is asserted.
31 LRPDS	This bit indicates the real time status of LR power down signal. When asserted, LR is put in the power down state. 0 LR power down signal is negated. 1 LR power down signal is asserted.

55.6.2.25 Unsolicited Rx Status Register (UNSRSR)

Offset: 0x00A8 Access: read/write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	URXDV	0	0	0	0	0	URPCNT		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 946. Unsolicited Rx Status Register (UNSRSR)

Table 1006. UNSRSR field descriptions

Field	Description
23 URXDV	Unsolicited data valid Indicates a valid frame exists in the Unsolicited Data registers.
29:31 URPCNT	Rx Unsolicited payload Indicates the number of bytes of valid Rx unsolicited data payload present in the *Unsolicited Rx Data Registers (UNSRDRn)*. The bitfield coding is same as b7-b5 in *Table 980*.

55.6.2.26 Unsolicited Rx Data Registers (UNSRDR*n*)

Offset: 0x00AC (UNSRDR8), 0x00B0 (UNSRDR7), 0x00B4 (UNSRDR6), 0x00B8 (UNSRDR5), 0x00BC (UNSRDR4), 0x00C0 (UNSRDR3), 0x00C4 (UNSRDR2), 0x00C8 (UNSRDR1), 0x00CC (UNSRDR0) — Access: User Read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	UNRXD															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	UNRXD															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 947. Unsolicited Rx Data Registers (UNSRDR8–UNSRDR0)

Table 1007. UNSRDR8–UNSRDR0 field descriptions

Field	Description
0:31 UNRXD	Unsolicited Receive Data This represents 9 registers for Unsolicited received data.It is read only register. The first bit received as part of the payload will be stored at UNRXD8[31], second bit at UNRXD8[30], and so on. So the last bit will be stored at UNRXD0[0] in case of 288-bit payload.

55.6.3 Register safety classification requirements

This module is classified as NoSaMo (Non-Safety relevant Module) for safety requirements (Refer to the “Functional safety” chapter for register classification details. All registers of the LFAST module are classified as non-safety relevant.

The MCR[IPGDBG] fulfills the safety requirements for monitoring of the debug freeze signal.

55.7 Functional description

55.7.1 Startup procedure

The LFAST requires a sequence of operations before it can be used for communication. The procedure to enable LFAST for proper operation is listed below.

55.7.1.1 LFAST master interface startup procedure

1. After reset the SLCR and RCDCR are programmed according to the LVDS parameters (refer to the default reset values of the related registers).
2. The PLLCR is programmed with configuration parameters for the PLL.
3. The LCR is programmed with configuration parameters of the LVDS.
4. Write MCR[MSEN] = 1. Then select LFAST modes by configuring MCR[CTSEN] and MCR[DATAEN].
5. Write MCR[DRFEN] = 1 to enable the LFAST.
6. Write MCR[RXEN] = 1 and MCR[TXEN] = 1 to negate the LD powerdown, LR disable and LR powerdown signals.
7. Write ICR[ICLCPLD] = 31h to enable the Slaves Tx Interface.
8. Write ICR[SNDICLC] = 1 and ICR[ICLCSEQ] = 1.
9. Write MCR[TXARBD] = 0.
10. The ICLC transmission is confirmed by verifying one of the following:
    – ICR[SNDICLC] = 0.
    – TISR[TXICLCF] = 1.
11. Write ICR[ICLCPLD] = 0x00 to check the LFAST slaves status.
12. Write ICR[SNDICLC] = 1.
13. The LFAST slave status is confirmed by occurrence of one of the following:
    – LFAST slave is enabled if RIISR[ICPSF] = 1. Proceed to step 14.
    – If RIISR[ICPSF] = 1: The LFAST slave is configured to mode matching to High speed mode of the LFAST master mode.
    – If RIISR[ICPSF] = 0 or RIISR[ICPFF] = 1: The LFAST slave is not providing response matching with High speed mode of the LFAST master. The LFAST slave is most likely still in SLOW speed mode.
    – LFAST slave is disabled or wrongly configured if RIISR[ICPSF] = 0 or RIISR[ICPFF] = 1. The LFAST master must wait and restart from Step 7.

Speed mode change:

14. Write LCR[SWPON] = 1 to enable the LFAST masters PLL.
15. Write ICLC start PLL frame, ICR[ICLCPLD] = 02h.
16. Write ICR[SNDICLC] = 1.
17. The ICLC transmission is confirmed by occurrence of one of the following:
    – ICR[SNDICLC] = 0.
    – TISR[TXICLCF] = 1.
18. To change LFAST masters Tx interface speed both of the following operations are performed:
    (**Note:** The slaves Rx interface speed should be changed first.)
    – Write ICR[ICLCPLD] = 0x10 for Tx data fast frame.
    – Write ICR[SNDICLC] = 1.
19. Both of the following operations are performed to change the LFAST masters Rx interface speed:
    (**Note:** The slaves Tx interface speed mode should be changed first)
    – Write ICR[ICLCPLD] = 0x80 to select Rx data fast frame.
    – Write ICR[SNDICLC] = 1.
20. The ICLC transmission is confirmed by the occurrence of one of the following:
    – ICR[SNDICLC] = 0
    – TISR[TXICLCF] = 1.
21. Write SCR[TDR] = 1 to change the LFAST masters Tx interface speed.
22. Write SCR[RDR] = 1 to change the LFAST masters Rx interface speed.
23. Write ICR[ICLCSEQ] = 0 to enable speed change for next ICLC frame.
24. Write ICR[ICLCPLD] = 0x00 to confirm the change in speed of the LFAST slave. This frame should be written after waiting for the expected delay in the start of the PLL and speed mode change delay at the LFAST slave.
25. Write ICR[SNDICLC] = 1
26. Write MCR[TXARBD] = 1, after some delay to ensure the step 24 ICLC frame is send but no other frame is sent to arbitration.
27. The LFAST slaves speed mode is confirmed by the occurrence of the following:
    – If RIISR[ICPSF] = 1: The LFAST slave is configured to mode matching to High speed mode of the LFAST master mode.
    – If RIISR[ICPSF] = 0 or RIISR[ICPFF] = 1: The LFAST slave is not providing response matching with High speed mode of the LFAST master. The LFAST slave is most likely still in SLOW speed mode.
28. If RIISR[ICPSF] = 1, then all frame arbitration is enabled by the following operation: Write MCR[TXARBD] = 0.
29. If RIISR[ICPFF] = 1 modifying the LFAST slaves speed needs to be repeated by:
    – Write SCR[TDR] = 0 to return the Tx interface to Low Speed mode.
    – Write MCR[TXARBD] = 0, to enable frame transmission.
    – The operations from step 14 need to be executed again.

55.7.1.2 LFAST slave interface startup procedure

1. After reset the SLCR and RCDCR are programmed according to the LVDS parameters in the device datasheet.
2. The PLLCR is programmed with the configuration parameters for the PLL.
3. The LCR is programmed with the configuration parameters for the LVDS.
4. Write MCR[MSEN] = 0. Then select LFAST modes by configuring MCR[CTSEN] and MCR[DATAEN].
5. Write MCR[DRFEN] = 1 to enable the LFAST.
6. The LR is enabled by writing MCR[RXEN] = 1. This negates the LR disable signal and LR's power down signal.
7. The LD enable signal needs to be asserted. This is done when one of the following is true:
   – When an ICLC frame with payload 0x31 is received the hardware will write RIISR[ICTEF] = 1 and MCR[TXEN] = 1.
   – MCRMCR[TXEN] = 1
8. After a write to MCR[TXARBD] = 0, a ping frame will be sent on one of the following conditions:
   – When PICR[PNGAUTO] = 1 and an ICLC frame with payload 0x00 frame is received.the hardware writes [ICRPF] = 1
   – PICR[PNGREQ] = 1.

Speed mode change:

9. The PLL will start when one of the following conditions is met:
   – When an ICLC frame with payload 0x02 is received. The hardware writes RIISR[IPONF] = 1
   – Write PLLCR[SWPON] = 1.
10. The speed of the Tx interface is changed on one of the following conditions:
   – When SCR[DRMD] = 1 and an ICLC frame with payload 0x80 is received. The hardware writes RIISR[ICTFF] = 1 and SCR[TDR] = 1.
   – SCR[TDR] = 1 when SCR[DRMD] = 0
11. The speed of the Rx interface is changed on one of the following conditions:
   – When SCR[DRMD] = 1 and an ICLC frame with payload 0x10 is received. The hardware writes RIISR[ICRFF] = 1 and SCR[RDR] = 1.
   – SCR[RDR] = 1, when SCR[DRMD] = 0
12. A Ping Frame is sent on one of the following conditions:
   – When PICR[PNGAUTO] = 1 and an ICLC frame with payload 0x00 is received.The hardware writes RIISR[ICRPF] = 1
   – Write PICR[PNGREQ] = 1.

55.7.2 Line Receiver (LR)

55.7.2.1 Introduction

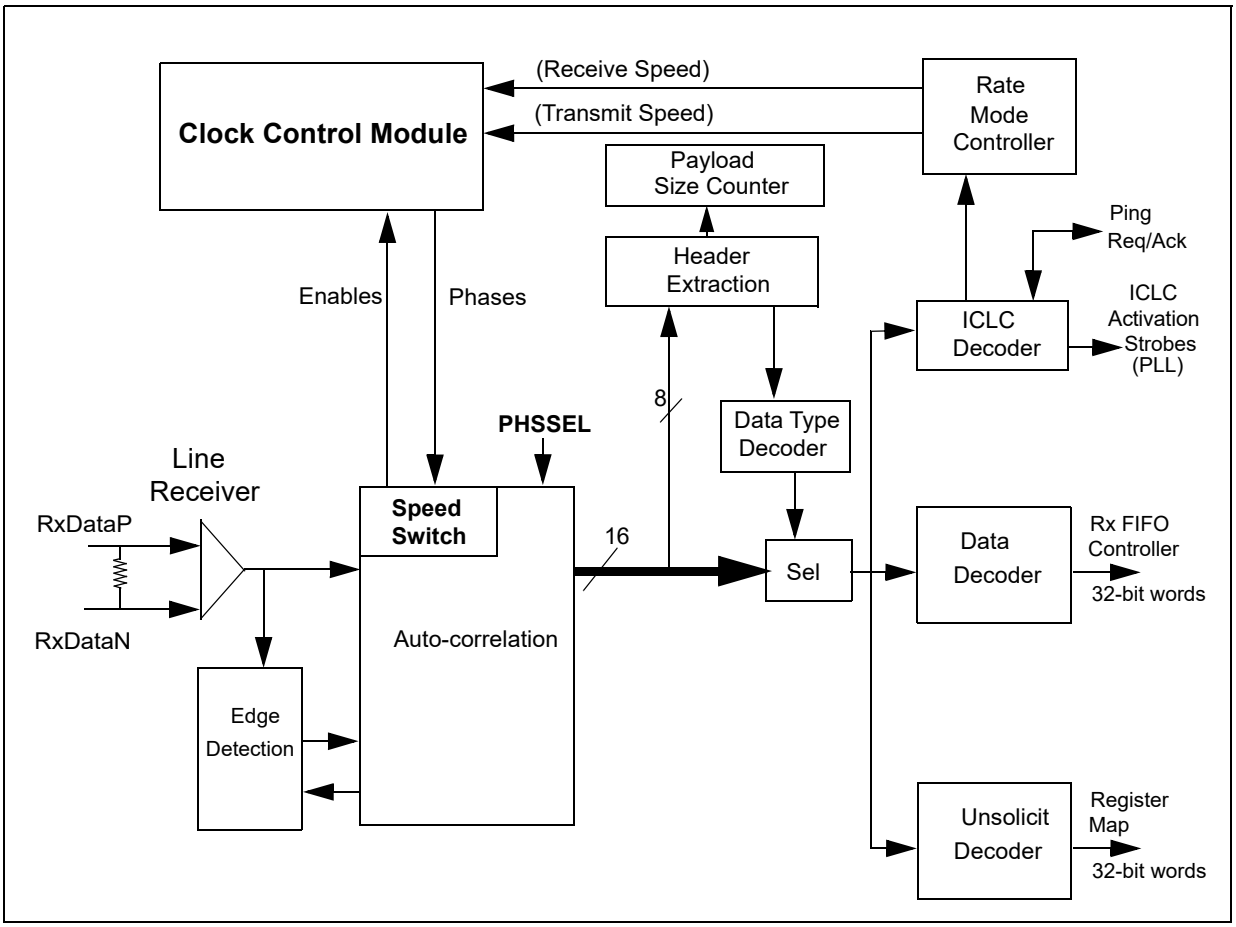
The LR detects the voltage swing on the differential pair and converts it to a CMOS logic level that feeds the adaptive auto-correlation block. The received data is sampled using the best of the 8 (default) or 4 (alternative setting) possible sampling edges from the high speed

clock or 4 phases using the low speed clock. The sampling edge is chosen by checking which of the 8/4 Correlators provide the maximum correlation. If more than one sampling edge provides the maximum then the state machine will choose the sampling edge based on a defined selection algorithm.

After the correct sampling edge is chosen, the remaining unused sampling edges are turned off. Once the header is received the length of the frame (payload size) and logical channel definition can be obtained. The logical channel definition will determine the data type of the payload and therefore will determine the destination for the payload. Decoding of the ICLC commands from the payload is required in order to extract the interface control, rate mode and PLL commands.

Figure 948 shows the block diagram of Uplink controller.

Figure 948. Top level receive controller



55.7.2.2 Edge-detection and auto-correlation

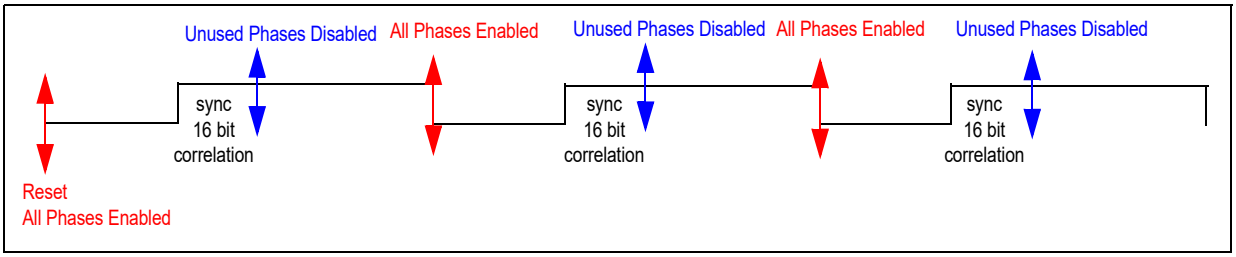
The edge-detection and auto-correlation are described together, as the mode setting of the auto-correlation block impacts largely on whether the edge-detection circuitry is used or not. The edge-detection will be performed first if required before auto-correlation.

55.7.2.2.1 Auto-correlation modes

55.7.2.2.1.1 Hunt correlation mode

On reset the Receive Controller will always come up in Hunt Correlation mode. In this mode all the Correlators are enabled and the Receive Controller is always hunting for the synchronization pattern. The phase enables (either 8 or 4) to the external clock control module are always high. There is no edge detection for the first bit of the synchronization pattern. Hunt Correlation mode is considered the safest mode as the Receive Controller is always checking for the synchronization pattern, but it is the mode that consumes the most power. In Hunt Correlation mode the correlation is performed over all 16 bits of the synchronization pattern.

Figure 949. Hunt correlation mode



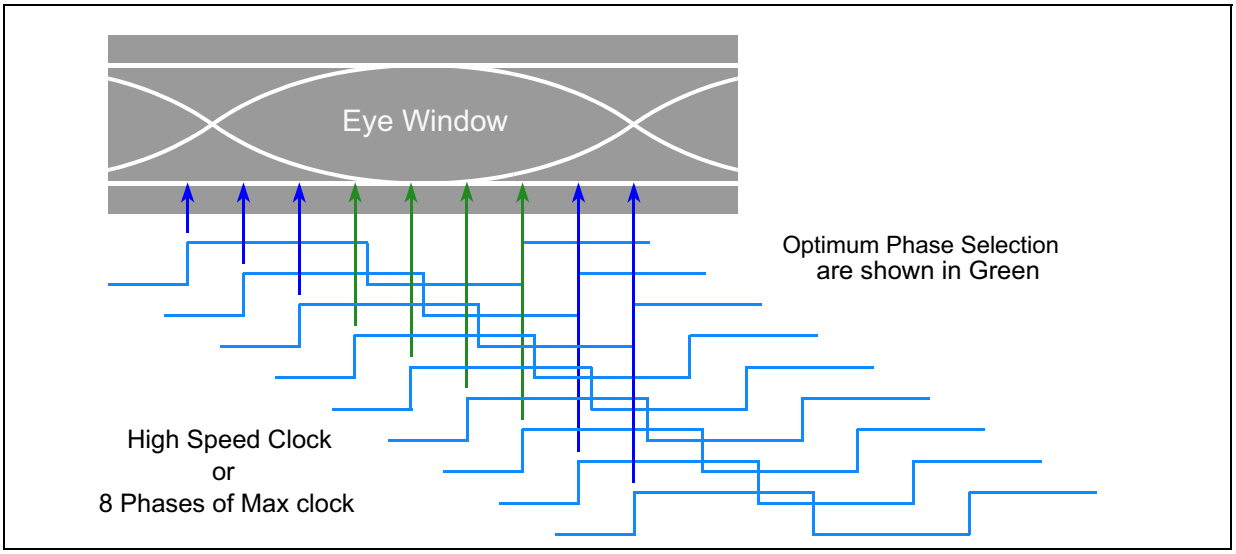
55.7.2.2.2 Edge detection

The edge detector is disabled in Hunt Correlation mode.

55.7.2.2.3 Auto-correlation

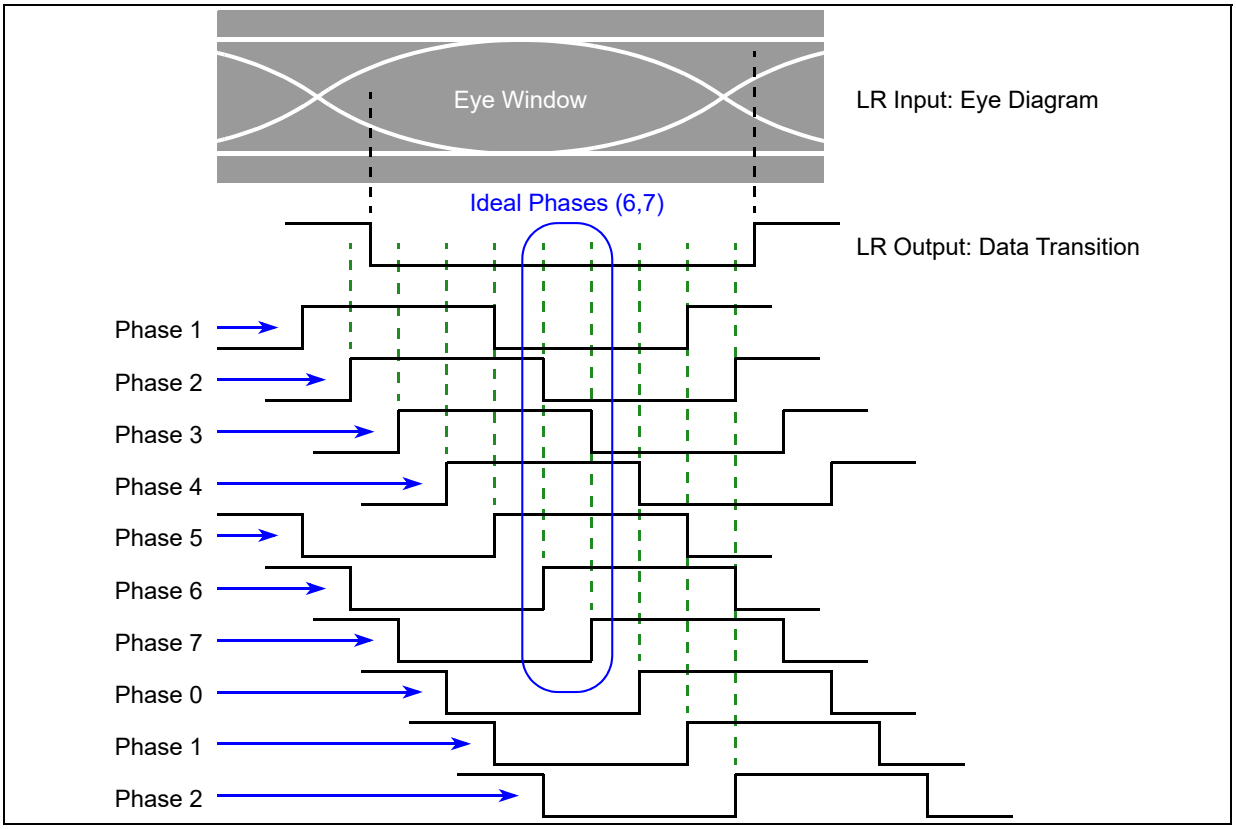
The data transmission between the 2 devices LD and LR is asynchronous in nature. Hence the Receive Controller does not have the knowledge about the correct clock phase to be used for extracting the data. The task of the auto-correlation (or synchronization) scheme is to estimate the best clock phase (8 or 4) for extraction of data as shown in *Figure 950*.

Figure 950. High speed 8 phase selection



The objective of the auto-correlation is to select the clock phase that occurs closest to the center of the eye diagram window. For 8 Phase clock alignment the worst case selection is when the clock edge is just after the LR output transition. The 8 clock phases will sample the correct value but the middle clock phases are the ideal selection. If one of the middle phases are selected then the minimum distance to the LR transition is 3 clock phases as shown in *Figure 951*.

Figure 951. High speed 8 phase clock alignment example



Each of the 8 high speed phases are 45 degrees separated. For 4 phases, whether high or low speed, the phases are 90 degrees separated but can have different phases enabled as shown in *Figure 952* and *Figure 953*.

Figure 952. High speed 4 phase clock alignment example

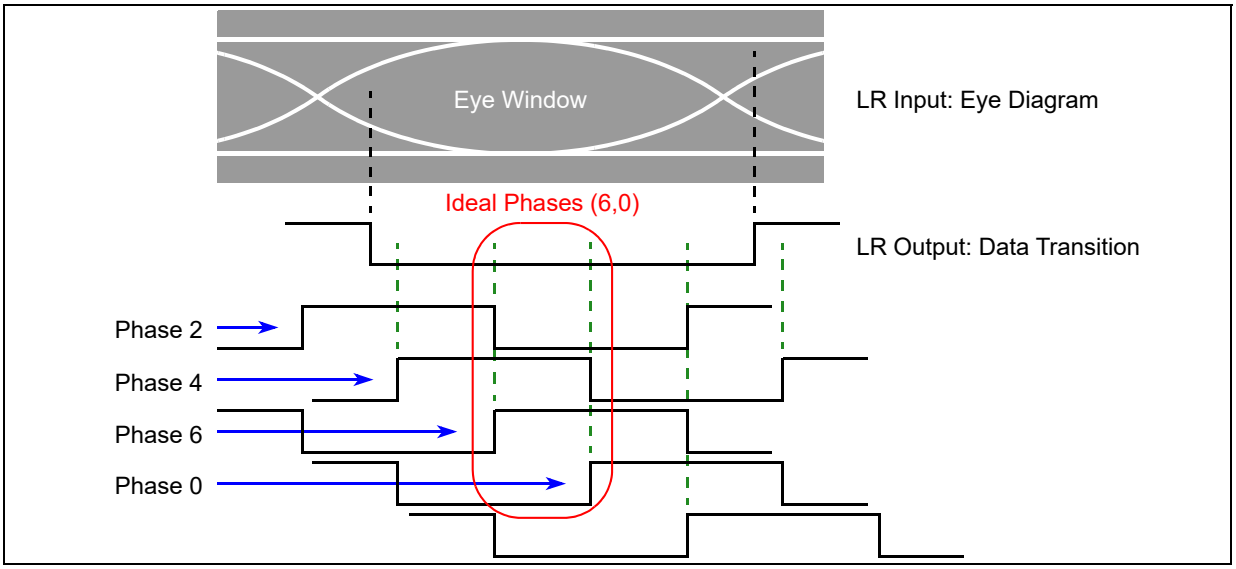
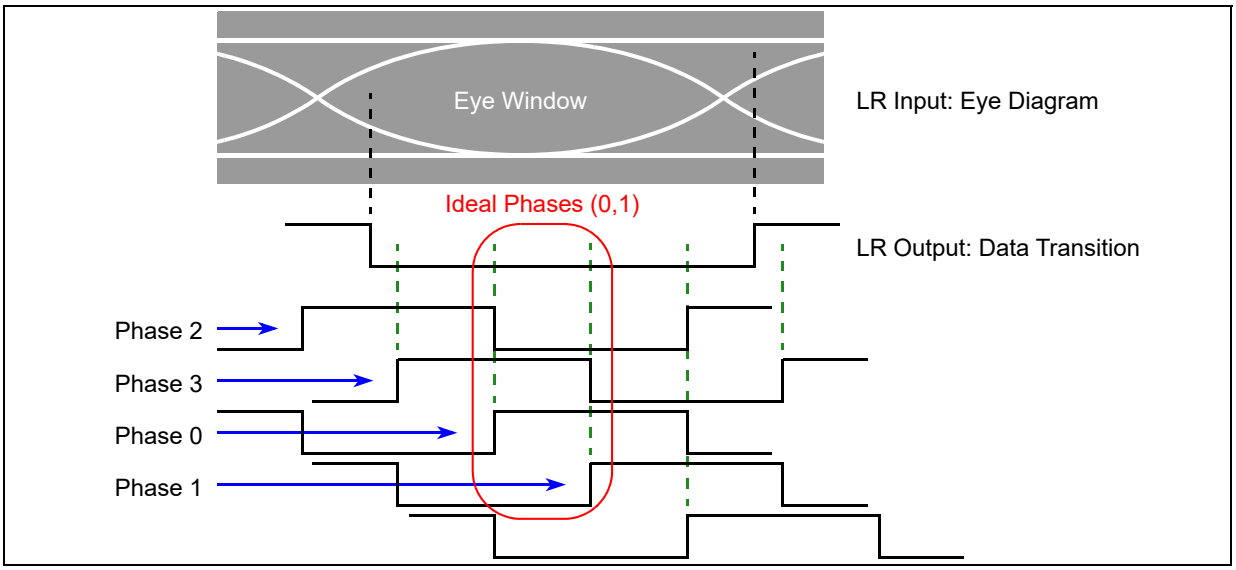


Figure 953. Low speed 4 phase clock alignment example



The auto-correlation and selection of the correct clock phase starts after the edge detection finds a 0 to 1 transition. The high speed 8 or 4 phases from the PLL and the low speed 4 phases (generated inside the Clocking Module) are muxed inside the clocking module. The LFAST interface block will determine which phases are to be enabled and disabled. The only time the interface does not choose the phases is when the Clocking Module overrides the phase enables using COCR[SMPSEL].

The input data path can be sampled by different samplers depending on the configuration:

- High Speed 8 Phases: Samplers 0,1,2,3,4,5,6,7
- High Speed 4 Phases: Samplers 0,2,4,6
- Low Speed 4 Phases: Samplers 0,1,2,3

Each sampler block samples the data path with a different clock phase from the Clocking Module, and re-samples it to an intermediate phase as shown in *Table 1008*, *Table 1009*,

and *Table 1010* before re-sampling it to Phase 0. The intermediate phase is used to make static timing between the initial phase and the final phase 0. The final phase, Phase 0 is chosen so all the clock trees after the sampler are clocked by the same clock.

Table 1008. High speed 8 phase selection - sampling procedure

Samplers	Initial Sample	Intermediate Sample	Final Sample
0	Phase 0	Phase 0	Phase 0
1	Phase 1	Phase 0	Phase 0
2	Phase 2	Phase 0	Phase 0
3	Phase 3	Phase 0	Phase 0
4	Phase 4	Phase 2	Phase 0
5	Phase 5	Phase 2	Phase 0
6	Phase 6	Phase 4	Phase 0
7	Phase 7	Phase 4	Phase 0

Table 1009. High speed 4 phase selection - sampling procedure

Samplers	Initial Sample Phase	Intermediate Sample Phase	Final Sample Phase
0	Phase 0	Phase 0	Phase 0
1	Disabled	Disabled	Disabled
2	Phase 2	Phase 0	Phase 0
3	Disabled	Disabled	Disabled
4	Phase 4	Phase 2	Phase 0
5	Disabled	Disabled	Disabled
6	Phase 6	Phase 4	Phase 0
7	Disabled	Disabled	Disabled

For High speed 4 Phase selection, See *Table 1009*, Samplers 1, 3, 5, 7 are disabled. In the Phase Select algorithm Phase 1 is mapped to Phase0, Phase 3 is mapped to Phase 2, Phase 5 is mapped to Phase 4, Phase 7 is mapped to Phase 6 so it simplifies the algorithm and allows the algorithm to be the same independent of 4 or 8 Phases in high speed.

Table 1010. Low speed 4 phase selection - sampling procedure

Samplers	Initial Sample Phase	Intermediate Sample Phase	Final Sample Phase
0	Phase 0	Phase 0	Phase 0
1	Phase 1	Phase 0	Phase 0
2	Phase 2	Phase 0	Phase 0
3	Phase 3	Phase 0	Phase 0

Table 1010. Low speed 4 phase selection - sampling procedure (continued)

Samplers	Initial Sample Phase	Intermediate Sample Phase	Final Sample Phase
4	Disabled	Disabled	Disabled
5	Disabled	Disabled	Disabled
6	Disabled	Disabled	Disabled
7	Disabled	Disabled	Disabled

For low speed 4 Phase selection, See *Table 1010*, Samplers 4, 5, 6, 7 are disabled.The first four Samplers (0,1,2,3) of the low 4 phase speed selection algorithm are the same as the four samplers required for the high 8 phase speed. Therefore the same architecture can be used for both high speed and low speed with the other four data paths disabled for low speed.

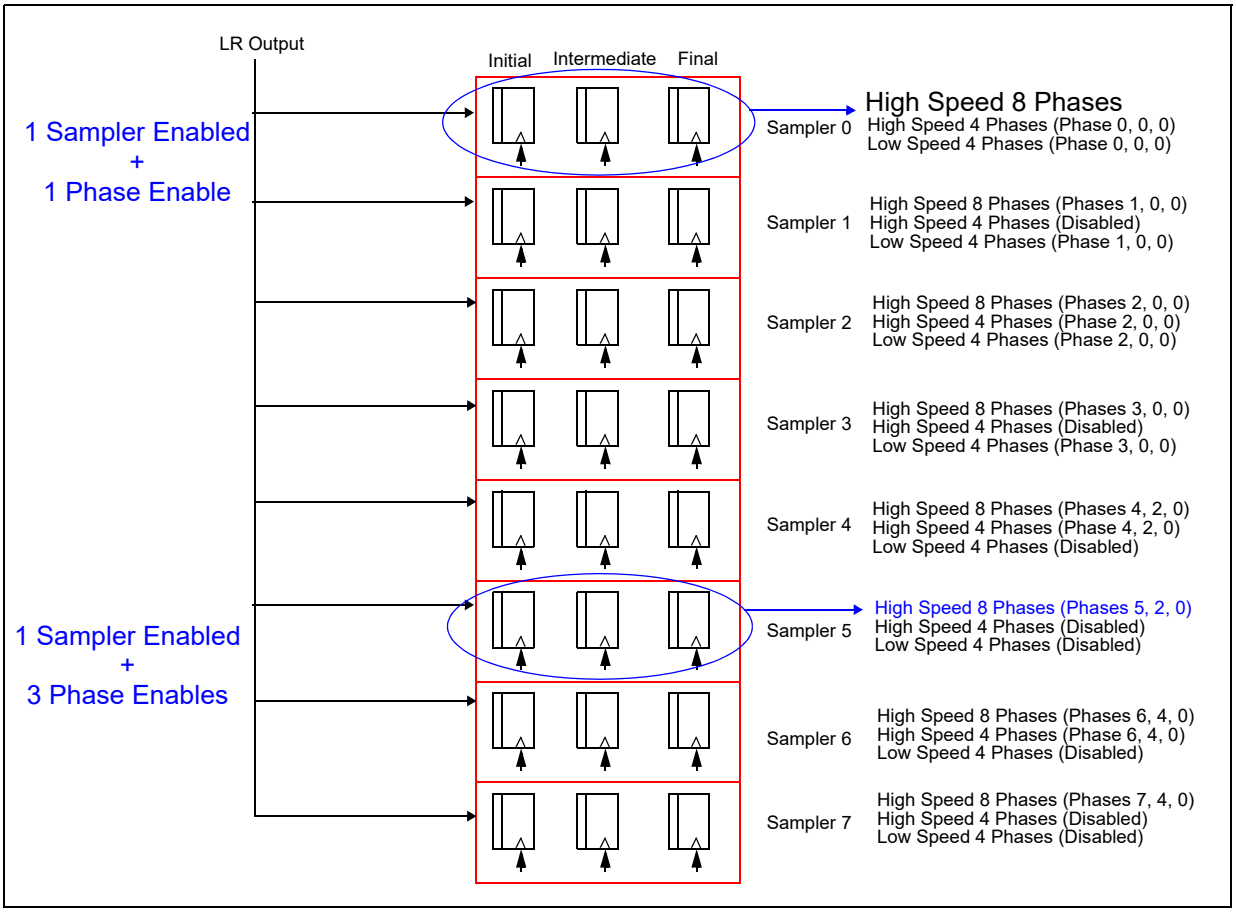
55.7.2.2.4 Sampler block and phase enable and disable

There are 8 data sampler paths in total inside the auto-correlation block, each data sampler has 3 sampling registers with the possibility of each register being clocked by a different phase (in example, Sampler 5 for high speed can have Phases 5, 2 and 0). Therefore after the correct data sampler has been selected for either high or low speed, the block can turn off all the sampler paths except for one, therefore 3 out of 24 registers will remain enabled while the others are disabled, as shown in *Figure 954*.

After correlation and the correct data sampler has been selected all the other data samplers whose initial phase does not match the select phase are disabled. The selected sampler will then keep the required phases needed enabled, this can be max 3 phases (for instance Sampler 5 has Phases 5, 2, 0) or min 1 phase (in example, Sampler 0 has all Phase 0 content).

The end result is that the Rx Controller can disable the required number of unused sampler registers and disable any unnecessary phases from the Clocking Module by de-asserting the respective phase enables.

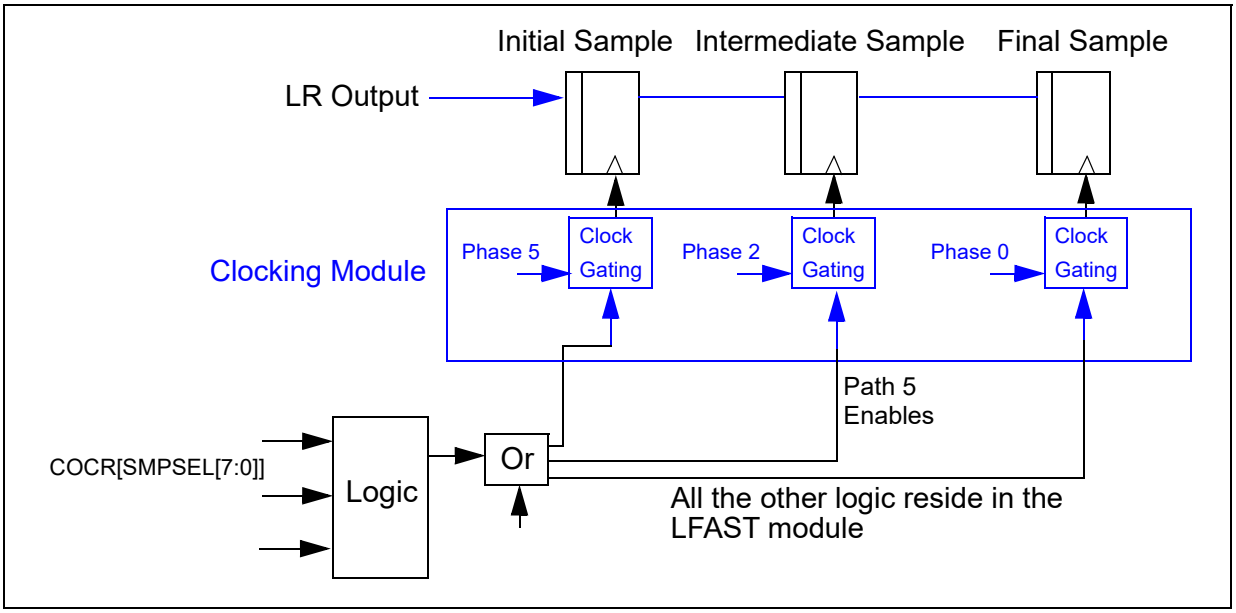
Figure 954. Samplers, each sampler has 3 registers



In order to achieve the sampler and phase enable requirements each Sampler block will require the logic as shown in *Figure 955*.

The Clock Gating Element resides inside the Clocking Module block. The interface will provide the enables to the Clocking Module and the Clocking Module will in return provide the individual clocks to the sampling registers.

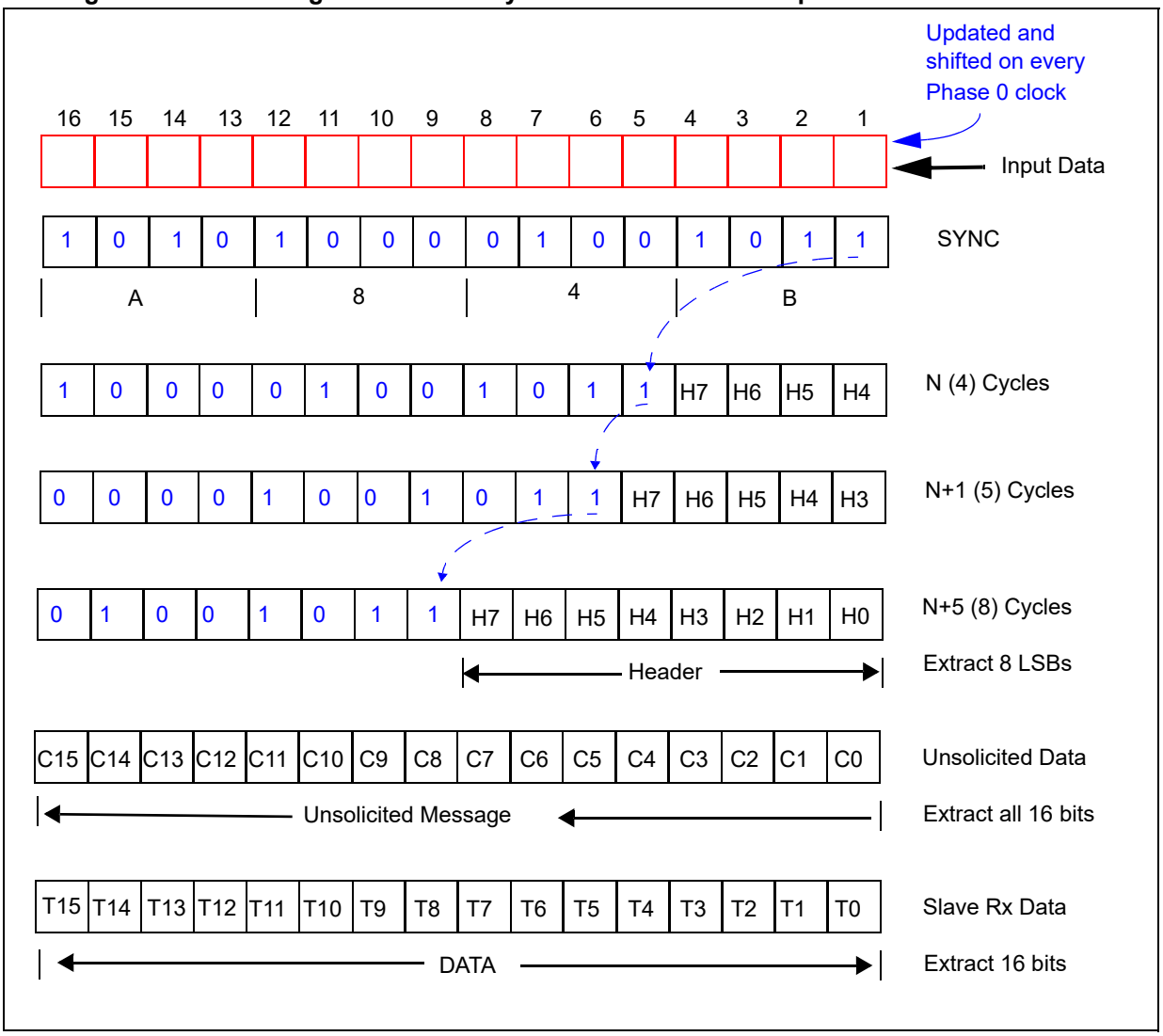
Figure 955. Sampler 5 Logic



55.7.2.3 Header and Payload extraction

Once the Phases/Samplers are chosen after Synchronization and Correlation the next part of the flow is the header extraction. The Receive Controller will output a 16-bit word, bit shifted every Phase 0 clock as shown in *Figure 956*.

Figure 956. Extracting Header and Payload from the 16-bit output from Auto-correlation



55.7.2.3.1 Line Receiver states

The LD has the following states:

- Shutdown
  The LR enters in the shutdown state when the power down signal is high. In this state the LR regulator is supplying a Vdd level, however the LR is not enabled. In this state the LR holds its off value. Once power down signal is negated, a predefined settling time is required before the LR may be used for communication. During the settling time the peer LFAST device should not start its transfer.
- Sleep
  The LR enters in the Sleep state when the sleep signal is high. In the sleep state, the LR is enabled, but held in a power-saving state. Sleep mode may be used during inter-frame gaps that are long compared to the frame durations but not long enough to allow the interface(s) or high-speed clock generators to be powered down completely. In this

mode the receiver current consumption is reduced. Hysteresis in the LR ensures that it continues to present a 1 to the LFAST module.

- Normal
  In the Normal state, the LR is primed for reception. Receiver converts differential signals to a single ended logic signal.

Table 1011. Line Receiver state

LR State	LFAST Master Triggers (at least one of the following)	LFAST Slave Triggers (at least one of the following)
Shutdown	– Programing of LCR[SWOFFLR] or LCR{SWONLR] – Programing MCR[RXEN] and MCR[TXEN]	– LFAST interface enable assertion/negation – Programing of LCR[SWOFFLR] or LCR{SWONLR] – Programing MCR[RXEN] and MCR[TXEN]
Sleep	– Programing of LCR[SWSLPLR] or LCR[SWWKLR]	– Programing of LCR[SWSLPLR] or LCR[SWWKLR]
Normal	Absence of above mentioned conditions	Absence of above mentioned conditions

55.7.3 Transmit controller

55.7.3.1 Introduction

The Transmit Controller uses Rx information, status information and error control data and codes them into the appropriate frame structure for transmission to the LD. This includes the synchronization and header, in preparation for the LD, which transmits the frame to peer LFAST IC at either low or high speed. The Transmit Controller creates a frame structure that is converted to a serial format for the LD.

An arbitration block will determine which message has higher priority data, such as unsolicited, ICLC, CTS, ping. The data frames are extracted from the FIFO controller, the unsolicited message from the register block, the CTS messages from the Receive Controller and the ping response from the register block.

The Tx interface has two speed modes:

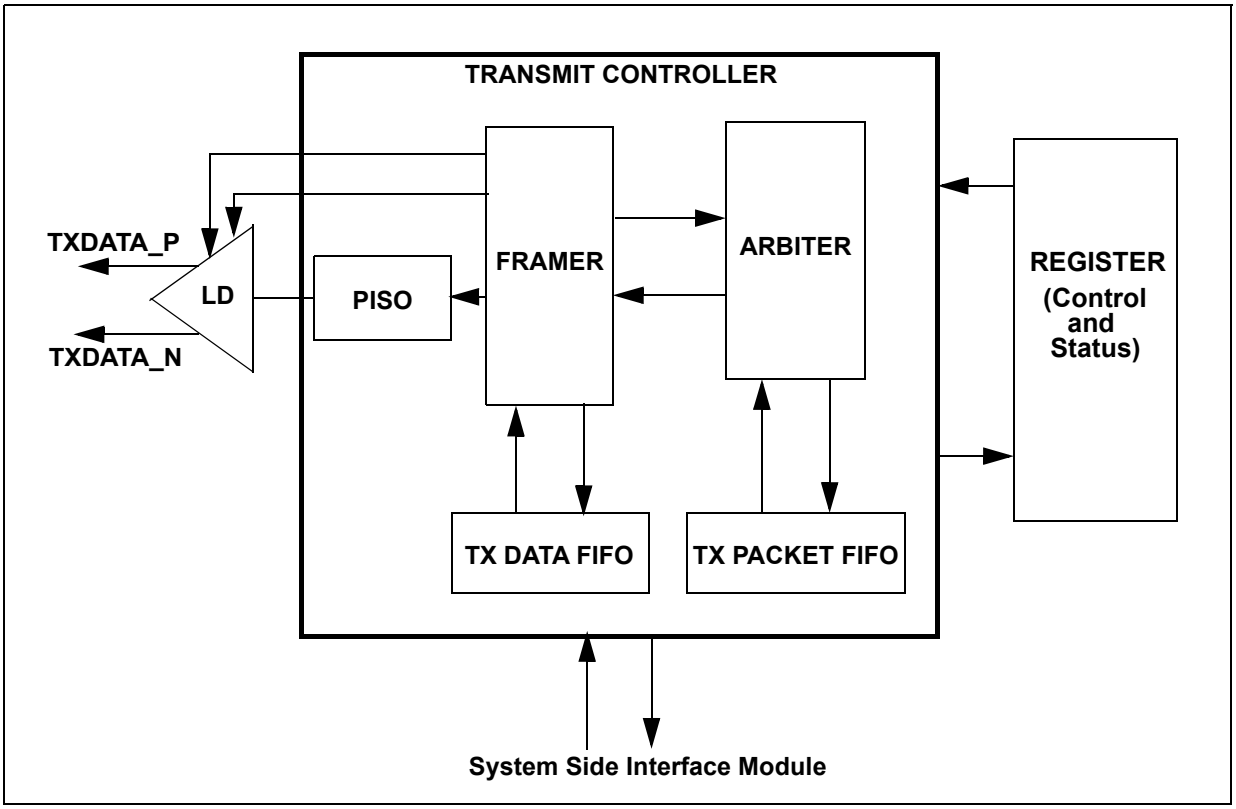
- Low Speed
- Fast Speed

The Transmit Controller consists of the following blocks as shown in *Figure 957*.

- Arbitrator
- Framer
- Request Clock Control

The arbitrator will grant access to the framer from the request that has the highest priority. The framer block will extract the payload data from the granted request source and build the frame (adding synchronization or header if required). The frame is fed into a PISO (Parallel In Serial Out) and eventually sent to the LD.

Figure 957. Transmit controller connections



55.7.3.2 Arbitration

The arbitration block prioritizes between requests from multiple sources like:

- Software programmable registers
- System side interface FIFO
- Rx interface controller

The level of priority for each request is shown in *Table 1012*.

Table 1012. Priority levels for the Transmit controller

Request	LFAST master priority	LFAST slave priority	Triggering conditions (at least one of the listed)
LFAST interface enable	1	1	– LFAST interface enable is asserted – LFAST interface enable is negated
Tx Interface disabled	2	2	– MCR[DRFEN] = 0 – MCR[TXEN] = 0 – MCR[DRFRST] = 1 – MCR[TXARBD] = 1 – LFAST Slave: ICLC frame with payload for “Disable Rx interface” received
Tx Interface speed mode change	3	3	– SCR[TDR] is modified – LFAST Slave: ICLC frame with payload for changing Rx interface speed received

Table 1012. Priority levels for the Transmit controller (continued)

Request	LFAST master priority	LFAST slave priority	Triggering conditions (at least one of the listed)
Loopback frame in Loopback mode	4	4	– TMCR[LPON] = 1 – LFAST Slave: ICLC frame with payload for loopback mode enable received **Note:** Valid only for following TMCR[LPMOD] settings: LPMOD[2:0] = 011b LPMOD[2:0] = 100b
ICLC frame request	5	–	– LFAST Master: ICR[SNDICLC] = 1 – LFAST Master: ICR[ICLCSEQ] = 1
Ping response request	–	5	– LFAST Slave: ICLC frame with payload for ping request received and PICR[PNGAUTO] = 1 – LFAST Slave: PICR[PNGREQ] = 1
Unsolicited frame request	6	6	– Last Frame with CTS = 1 received from the peer LFAST device – UNSTCR[USNDRQ] = 1
Data frame from Tx FIFO	7	7	– Tx FIFO contains one or more frames – Last Frame with CTS = 1 received from the peer LFAST device – MCR[DATAEN] = 1
CTS Frame	8	8	– MCR[CTSEN] = 1 – Rx FIFO reaches High/Low threshold, defined by TISR and no frame pending
Clock mode test	9	9	– TMCR[CLKTST] = 0 – LFAST Slave: ICLC frame with payload for clock test mode enable received

Once the arbitrator has granted access to a request, the Framer will commence building the frame. Any new requests for frame or data rate change will not be granted access until the framer has finished transmitting the current frame. If a data rate change request for the Tx interface is received while the Transmit controller is in the middle of sending a frame then the rate change request to the Clocking Module will be delayed. This allows for the frame to be completed before the change in speed mode. This prevents any speed mode change during the transmission of a frame. Once the speed mode request is sent to the Clocking Module by the Tx Interface Controller, it will then not allow any new requests to be processed for a specific time period defined by the bit field RCDCR[DRCNT].

55.7.4 CTS mode support

LFAST module supports flow control method defined in the LFAST3G standard. The CTS (Clear to send) is a protocol defined method to ensure no loss of frame, due to Rx FIFO unavailability. The optimal use of bandwidth of LFAST, without losing frames, is ensured by proper programing of the Rx FIFO Lower threshold (RFCR[CTSMN]) and Higher threshold (RFCR[CTSMX]). The CTS is supported by both the LFAST Master and Slave.

CTS Transmission:

The LFAST device sends CTS information with each frame to the peer LFAST device.

If the CTS mode is enabled by write MCR[CTSEN] = 1 then:

- Sends CTS bit as 0 in LFAST frame (bit[0] of Header field) whenever the Rx FIFO reaches the higher threshold defined by the bitfield TISR[CTSMX]. This indicates that the LFAST device does not want the peer device to send data. The CTS bit of all frame are sent 0 until the Rx FIFO pointer reaches the lower threshold, as described below.
- Sends CTS bit as 1 in LFAST frame (bit[0] of Header field) whenever the Rx FIFO reaches the Lower threshold defined by the bitfield TISR[CTSMN]. This indicates that the LFAST device is ready to receive data from the peer device. The CTS bit of all frame are sent 1 until the Rx FIFO pointer reaches the Higher threshold, as described above.
- The CTS bit can be sent through any type of frame (data, unsolicited or ICLC). When there are no frames available in LFAST to send to the peer device, a CTS frame is sent. In this frame, bit[4-1] of Header field are sent as 0011b and 8-bit payload of 0.

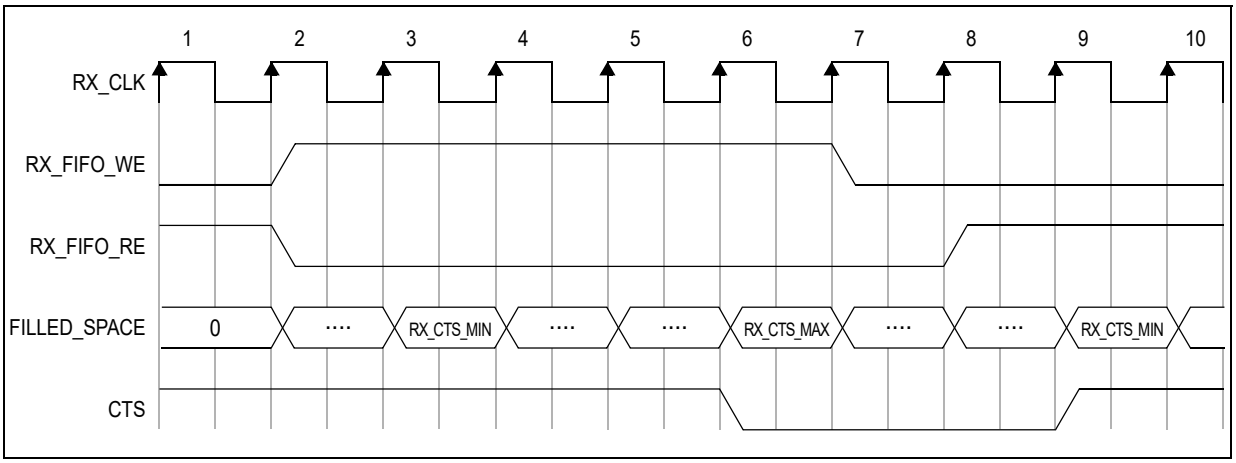
If the CTS mode is not enabled (for example, bit MCR[CTSEN] = 0) then:

- All the frames sent will have CTS bit as 1 in LFAST frame (bit[0] of Header field).

CTS Reception:

The reception of a LFAST frame with CTS bit 0 indicates that the peer device is not ready to receive data frames. The transmission of all pending data and unsolicited frames are postponed until a frame with CTS bit 1 is received.

Figure 958. CTS generation



55.7.5 Frames supported

Table 1013 provides the frames supported and their permissible payload sizes

Table 1013. Frames supported by LFAST interfaces

Frame Type	LCT Code	Supported by 1. LFAST Master Tx 2. LFAST Slave Rx	Supported by 1. LFAST Slave Tx 2. LFAST Master Rx	Payload Size (bits)
Data Frame	0100b – 1011b	YES	YES	32, 64, 96, 128, 256, 288
Unsolicited Frame	0001b	YES	YES	8, 32, 64, 96, 128, 256 and 288
ICLC Frame	0000b	YES	NO(1)	8
CTS Frame	0011b	YES	YES	8
Reserved	All others	—	—	—

1. Except the ICLC PING response frame.

If logical channel type (LCT) codes other than the above mentioned are received then the interrupt flag RISR[RXLCEF] = 1. If the payload size received does not match any value in *Table 1013* then the flag RISR[RXSZF] = 1. In both these error conditions the received frame is ignored. The software may have to take the necessary steps to recover from such an error.

55.7.6 Frame flow

The following sections describe Data Frame Flow, ICLC Flow and Rx Unsolicited Data Flow.

55.7.6.1 Data flow

LFAST supports the transfer of data between master and slave LFAST devices.

55.7.6.1.1 Data transmit

System Side Module is the initiator of all the Tx data frame to LFAST peer device. Data frame transmit is triggered whenever the Tx Data FIFO has at least one valid frame. The MCR[DATAEN], MCR[DRFEN] and MCR[TXEN] bits, should be set to enable the transfer of Tx data.

If MCR[DATAEN] = 0, then the valid frames in the in Tx data FIFO will be ignored for transmission. The Payload Size and channel type of each frame is specified by the System Side Module interface during transfer of the frame to the LFAST interface. The frame header is stored in the Tx packet FIFO and the payload in the Tx data FIFO.Whenever the Tx FIFO has at least one frame then a data transmit request is made to the Tx arbiter of the LFAST. Tx block arbitrates the data transmit request and schedules it depending on the priority of all the pending transmit requests. When data request is scheduled by Tx block, the required data is fetched from Tx data FIFO. The number of frames present in the Tx FIFO for transmission is indicated by the bitfield DFSR[TXFCNT].

55.7.6.1.1.1 Programing model for Tx data transmit

1. Program MCR[DATAEN] = 1, MCR[TXEN] = 1 and MCR[DRFEN] = 1 to enable the Tx path of LFAST device.
2. Select the desired clock rate at which data should be transmitted, using ICLC transfers and appropriate programing of SCR[TDR] bits.
3. Frame present in TX FIFO (Data and Packet FIFO) are sent.
4. TISR[TXDTF] = 1 after each frame transfer.
5. CTS is set depending on the push/pull mode setting defined by MCR[CTSEN].

55.7.6.1.2 Data receive

LFAST master and slave supports reception of data frame. The received frame is determined to be of data frame type by decoding channel type field of the header present in the received frame. The MCR[DATAEN], MCR[RXEN] and MCR[DRFEN] bits should be set to enable the data frame reception.When MCR[DATAEN] = 0 the received data frames will be ignored and will not be placed in the Rx data FIFO.

The Rx data frames received by Rx block are stored in the Rx FIFO. Whenever the frame is received in the Rx FIFO the System Side Module is indicated by assertion of LFAST Rx FIFO ready signal. The frame size and the Channel type is passed to the LFAST. The frame boundaries are indicated by start of frame and end of frame signals. The number of unread frames in the Rx Data FIFO are indicated by DFSR[RXFCNT]. When Rx DATA FIFO is full and cannot accommodate current frame completely, then the remaining data of the Rx frame is discarded. In this case, UNSRSR[RXOF] = 1 and an interrupt is generated if the RIER[RXOFIE] = 1.

55.7.6.2 Unsolicited flow

55.7.6.2.1 Unsolicited frame transmit flow

The software is the initiator for unsolicited frames to the LFAST peer device. The unsolicited frame header and payload is programmed into the Unsolicited Data and Control registers. Once the Payload is programmed UNSTCR[USNDRQ] is set, generating a request for unsolicited frame transfer to the Tx arbiter.

55.7.6.2.1.1 Programing model for unsolicited frame transmit

1. Program MCR[TXEN] = 1 and MCR[DRFEN] = 1 in the *Mode Configuration Register (MCR)* to enable the Tx path.
2. Select the desired clock rate at which data should be transmitted, using ICLC transfers and appropriate programing of SCR[TDR].
3. Read the UNSTCR[USNDRQ].
   - If UNSTCR[USNDRQ] = 1 then wait for either of the following:
     - UNSTCR[USNDRQ] = 0
     - TISR[TXUNSF] = 1.
   - If UNSTCR[USNDRQ] = 0 then:
     - Program the unsolicited payload in UNSTDR0–UNSTDR8, and can be written up to a payload of frame of a maximum of 288 bits.
     - Program the unsolicited frame header in UNSTCR[UNSHDR].
     - Program UNSTCR[USNDRQ] = 1.

4. UNSTCR[USNDRQ] is cleared by the Tx Block after the frame transfer.
5. TISR[TXUNSF] = 1 when the frame is transmitted.
6. CTS is set depending on the Push-Pull mode setting defined by MCR[CTSEN].

55.7.6.2.2 Unsolicited frame receive flow

Whenever an Unsolicited frame is received from the peer device, the following steps are performed:

If UNSRSR[URXDV] = 0 then:

1. The payload size field of the header is first saved into the UNSRSR[URPCNT].
2. The payload is stored in the UNSTDR0–UNSTDR8.
3. UNSRSR[URXDV] = 1 and the RISR[RXUNSF] = 1 indicating the successful reception of the frame.

If UNSRSR[URXDV] = 1 then:

1. The current unsolicited frame is ignored.
2. RISR[RXUOF] = 1.

Typical steps by the processor after RISR[RXUNSF] = 1 is as follows:

1. The processor reads UNSRSR to get the payload size of the frame.
2. It then reads the complete frame by reading UNSRDR8–UNSRDR0 for the payload size as received in the frame.
3. Then it clears the interrupt (write RISR[RXUNSF] = 1) and also the UNSRSR[URXDV] bit.

55.7.6.3 ICLC flow

The ICLC (Interface Control Logical Channel) is defined as a separate logical channel type, which is mainly meant for implementing the data rate change in the LFAST interface and initiating the test modes.

55.7.6.3.1 ICLC data transmit flow

Whenever the processor intends to transfer an ICLC frame to the LFAST slave then the following steps are to be followed.

1. Write the ICLC frame payload in the ICR[ICLCPLD] (see *Table 1014* for payloads as defined by the standard).
2. Program ICR[SNDICLC] = 1 to initiate the ICLC frame transfer. This bit clears itself when the ICLC frame has been transmitted. The processor needs to poll this bit to make sure it is cleared before setting this bit again (even when ICR[ICLCSEQ] = 1).
3. TISR[TXICLCF] = 1 after the transfer of the ICLC frame.
4. CTS is set depending on the Push-Pull mode setting defined by MCR[CTSEN].

To determine whether the ICLC frame has been sent or not, either the ICR[SNDICLC] bit can be polled, or wait for TISR[TXICLCF] = 1.

Table 1014. Supported ICLC payloads

ICLC code (hex)	ICLC function
00	Ping Request from LFAST Master to LFAST Slave
01	Reserved
02	Start PLL In preparation for High Speed mode
04	Stop PLL Fallback to low speed mode
08	Select TxData Slow (LFAST Slave Rx Interface)
10	Select TxData Fast (LFAST Slave Rx Interface)
20	Select RxData Slow (LFAST Slave Tx Interface)
40	Not Supported
80	Select RxData Fast (LFAST Slave Tx Interface)
31	Enable RxData Interface (LFAST Slave Tx Interface)
32	Disable RxData Interface (LFAST Slave Tx Interface)
34	Clock Test mode Send 101010… continuously using the currently configured clock rate (slow, fast)
FF	Turn payload loopback on
38	Turn Test mode (Loopback and Clock Test) off

Programing model for ICLC frame transmit:

1. Set the ICR[ICLCSEQ] bit (optional).
2. Write the ICLC frame payload in the ICR[ICLCPLD], corresponding to the desired Tx/Rx data rate change, as described in the *Table 1014*.
3. Write ICR[SNDICLC] = 1 to initiate the ICLC frame transfer.
4. LFAST master will schedule the ICLC transfer. Data present in ICR[ICLCPLD] is transmitted to the LFAST slave at the old Tx data rate. LFAST slave will configure its Tx/Rx interface data rate accordingly.
5. Reset the ICR[ICLCSEQ] to allow the scheduling of other types of frames, which will be sent at the new data rate.

When ICR[ICLCSEQ] = 0, though the ICLC frame transfer still has highest arbitration priority, other frames may be scheduled if no valid ICLC frame request exists.

Note: ICLC frame transmit will not trigger the internal data rate change of LFAST Master Tx/Rx interface. LFAST master Tx/Rx interface data rate will only be changed, when corresponding SCR[TDR] and SCR[RDR] bits are appropriately configured.

Processor requires to take care that no data frames are to be transmitted while data rate change is happening or ICLC is transmitting frames by setting ICR[ICLCSEQ] bit in Table 1014.

55.7.6.3.2 ICLC data receive flow

When the bits 4 to 1 of a receive frame are 0000b it indicates that the payload is an ICLC. ICLC payloads are always 8 bits.

The reception of an ICLC frame is indicated by an interrupt to the system. The ICLC status register RISR indicates the type of the ICLC frame received. The Rx block will decode and generate the appropriate signals. An invalid ICLC code besides the ones listed in *Table 1014* will cause RISR[RXICF] = 1.

55.7.6.3.2.1 Ping request ICLC

The LFAST slaves ICLC decoder will decode the ping request and set RIISR[ICPRF] = 1. The software can also write to PICR[PNGREQ] to indicate to the Tx block that a ping response frame needs to be transmitted. The PICR[PNGAUTO] indicates whether Tx block can respond automatically to the request by the LFAST master ICLC Ping Request frame.The Tx block will arbitrate the ping response frame request and send a ping frame with ping data defined by bitfield PICR[PNGPLYD]. Once the ping response frame has been sent the Tx block will set TISR[TXPNGF] and clear PICR[PNGREQ], if set.

55.7.6.3.2.2 Start PLL ICLC

The Start PLL ICLC command will write bit RIISR[ICPONF] = 1.

55.7.6.3.2.3 Stop PLL ICLC

The Stop PLL ICLC command will write RIISR[ICPOFF] = 1.

55.7.6.3.2.4 LFAST Slaves RxData Interface Slow/Fast ICLC

The LFAST Slaves Rx Interface has two speed modes supported: slow (Low speed) and fast (High speed). The ICLC command will write RIISR[ICRSF] = 1 (Low speed) or RIISR[ICRFF] = 1 (High speed).

55.7.6.3.2.5 LFAST Slaves TxData Interface Slow/Fast ICLC

The LFAST slaves Tx Interface Controller has two speed modes: slow (Low speed), and fast (High speed). The ICLC command will write RIISR[ICTSF] = 1 (Low speed) or RIISR[ICTFF] = 1 (High speed).

55.7.6.3.2.6 Enable/Disable LFAST Slaves TxData Interface ICLC

The ICLC commands, Enable RxData Interface and Disable RxData Interface are decoded and the appropriate enable/disable signal sent to the Tx block Controller. These ICLC command writes RIISR[ICTEF] = 1 and RIISR[ICTDF] = 1. The ICLC Tx enable command will write MCR[TXEN] = 1.

No frames can be sent on the LFAST Slave Tx interface until the either LFAST master has enabled it via an ICLC frame or software programs MCR[TXEN] = 1.

55.7.6.3.2.7 Clock Test mode ICLC

This ICLC command is decoded, and then is used to write TMCR[CLKTST] = 1. This indicates to the Tx interface to output an alternating pattern of 1 and 0 at the currently configured clock rate. The Rx interface generates RIISR[ICCTF] = 1 on reception of this ICLC command.

The exit from this mode happens on reception of Test mode off ICLC command.

55.7.6.3.2.8 Loopback payload on ICLC

This ICLC command is decoded and TMCR[LPON] = 1 to indicate to the Tx Block that the received payload is to be sent back. The exit from this mode happens on reception of Test mode off ICLC command. The Rx interface causes RIISR[ICLPF] = 1 on reception of this ICLC command.

55.7.6.3.2.9 Test mode off ICLC

This ICLC command is decoded and TMCR[LPON] and TMCR[CLKTST] are cleared to indicate to the Tx block to exit from loopback mode or clock test mode.

Another option is for the payload loopback option to remain enabled until negated on the toggling of LFAST interface enable .

55.7.6.3.2.10 Ping response ICLC

The LFAST master considers any ICLC frame payload as ping response data. The LFAST masters Rx block will store the received ping data in PISR[RXPNGD] and match it with PICR[PNGPYLD]. RIISR[ICPSF] = 1 if the Ping received matches with the one stored in PICR[PNGPYLD], or RIISR[ICPFF] = 1.

55.7.6.4 CTS flow

55.7.6.4.1 CTS Tx flow

The CTS frame, if enabled by the software, indicates the status of the Receive Data FIFO. CTS frame is triggered whenever Rx Data FIFO reaches either the High/Low threshold and no data, ICLC or unsolicited frame is ready for transmission.

Programing model for CTS frame transmit:

- Program RFCR[CTSMX] and RFCR[CTSMN].
- Enable the CTS frame Transmission by programming MCR[CTSEN] = 1.
- Program MCR[TXEN] = 1 and MCR[DRFEN] = 1 to enable the Tx path.
- The CTS bit sent with a valid frame (for example, data, ICLC and unsolicited) is 1 until the Rx Data FIFO reaches High Threshold.
- When the Rx Data FIFO reaches higher threshold and no valid frame (data, ICLC and unsolicited) is pending for transmission, then the CTS frame is triggered, with CTS bit of header = 0 and the status bit RISR[RXMXF] = 1.
- When the Rx Data FIFO reaches Low Threshold, due to system side reads and no valid frame (data, ICLC and unsolicited) is pending for transmission, then the CTS frame is triggered, with CTS bit of header = 1, and status bit RISR[RXMNF] = 1.

55.7.6.4.2 CTS Rx flow

Whenever a CTS frame header or any other frame with valid header is received with CTS bit = 0, then RISR[RXCTSF] = 1. The Tx Interface arbitration for unsolicited frame and data frame transmit request is turned off on reception of frame with CTS bit 0. The arbitration for unsolicited frame and data frame is enabled on reception of frame with CTS bit 1. Frames with LCT type CTS are not stored in the LFAST.

55.7.7 Test and debug support

The test and debug interface helps to debug the LFAST module. These signals can be brought out for ease of validation.

55.7.7.1 Loopback test mode

The loopback function allows to verify the correct operation of the physical interface and the basic checks for the LFAST module without a peer device.

There are certain prerequisites before entering the Loopback mode:

- LD and LR should be turned on.
- Tx and Rx mode should be enabled.
- Interrupts needed for software should be enabled.
- Both Tx and Rx interface should be in same speed mode.
- For Automatic Test mode TMCR[LPFRMTH] should be programmed.

The LFAST module supports four loopback modes, defined by TMCR[LPMOD].

Table 1015. Loopback modes

LPMOD[2:0]	Mode selected
000	Rx Loopback (default)
001	Rx LVDS Loopback
010	Tx Loopback without Automatic frame generation
011	Tx Loopback with Automatic frame generation
100	Tx LVDS Loopback (external) with Automatic frame generation

The TMCR[LPON] bit controls the state of the loopback function, and the default setting is 0 (off). This can be written by the software, or in the case of the LFAST slave, the TMCR[LPON] bit can be modified by the Rx interface controller on decoding of a valid ICLC loopback on or Test mode off commands. The LPMOD should not be changed when the LPON bit is set.

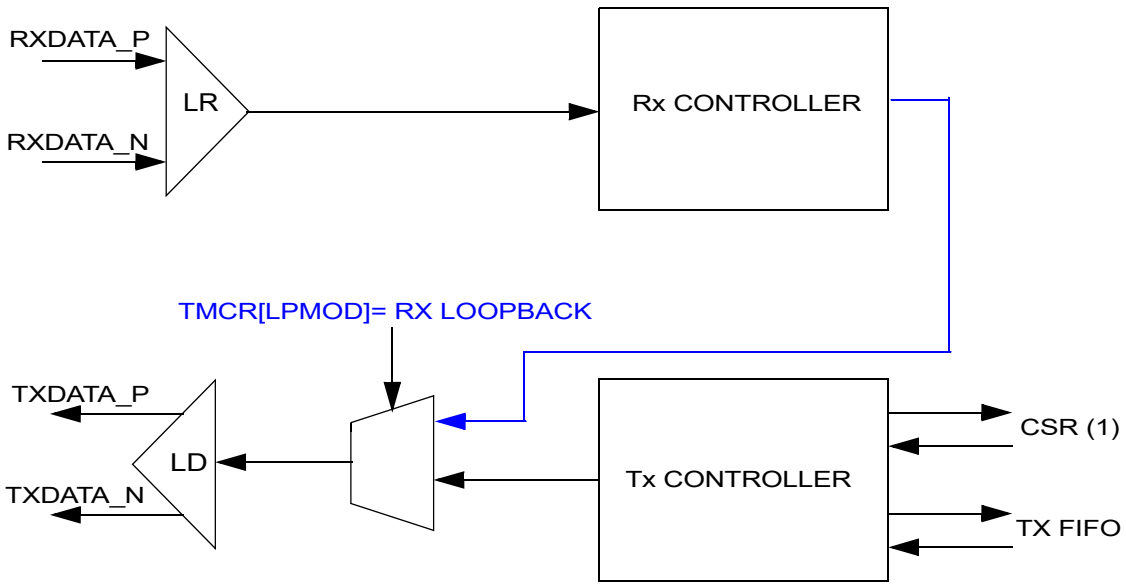
In all loopback modes, except Tx loopback without automatic frame generation, the decoding of frame and error flags is stopped. Only decoding of following is done:

- ICLC Turn Test mode Off frame.
- CTS bit of all frames.
- Automatic Loopback frame decoding (only in Automatic frame generation mode).

55.7.7.1.1 Rx Loopback mode

For Rx loopback mode the output of the LR is passed to the LD with manipulation via the Rx and Tx Interface controllers. CTS bit (Bit 0) of the header is guaranteed to be asserted when the incoming data from the other device is looped back. In Rx Loopback mode the Rx Interface controller operates in normal mode, decoding the frames (header, payloads). This allows the LFAST Master to control the Loopback mode on and off by sending ICLC commands.

Figure 959. Rx Loopback mode



1. CSR stands for Control and Status Registers (used to configure and indicate the status of the Tx/Rx controller).

Entry to Rx Loopback mode:

1. Software programs TMCR[LPMOD] = 000b.
2. Loopback can be turned on by either of the following methods:
   - Software programs TMCR[LPON] = 1.
   - For Slave Only: - Reception of ICLC Frame with Payload 0xFF (Loopback mode on), from LFAST Master

Exit from Rx Loopback mode can be done by any of the following methods:

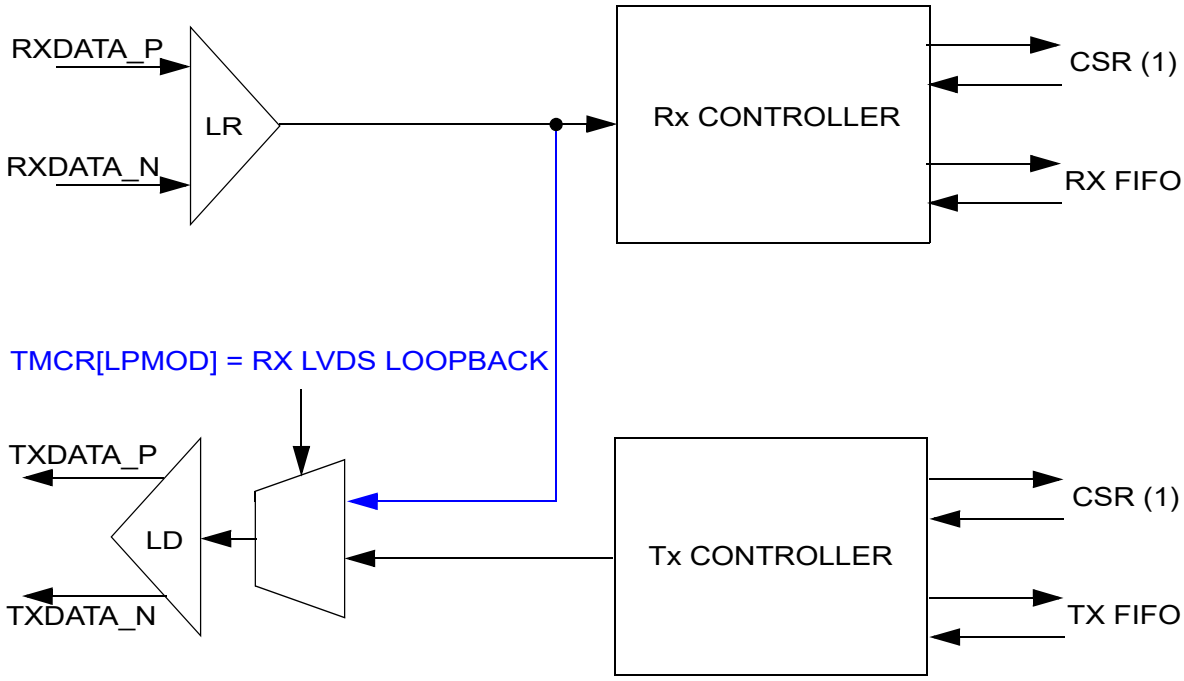
- Software programs TMCR[LPON] = 0
- For LFAST Slave: - Transmission of ICLC Frame with payload 0x38 (Test mode off), from LFAST Master

The peer LFAST device is required to maintain at least 2-bit IFG between two Loopback frames in this mode.

55.7.7.1.2 Rx LVDS Loopback mode

This loopback mode is provided to verify and characterize the LVDS pads. In this loopback mode the data received by LFAST on Rx LVDS input is loopback to Tx LVDS output, bypassing LFAST. For LVDS loopback the output of the LR is passed to the LD via “Rx LVDS LoopBack Mux”. Bit 0 of the header cannot be guaranteed to be asserted when the incoming data from the other device is looped back. In this loopback, the Rx Interface Controller operates in normal mode, decoding the frames (header, payloads) but the Tx Interface Controller is ignored.

Figure 960. Rx LVDS LoopBack mode



1. CSR stands for Control and Status Registers (used to configure and indicate the status of the Tx/Rx controller).

Entry to Rx LVDS Loopback mode:

1. Software programs TMCR[LPMOD] = 001b.
2. Loopback can be turned ON by either of the following methods:
   - The software writes TMCR[LPON] = 1.
   - For Slave Only: - Reception of ICLC Frame with Payload = 0xFF (Loopback mode ON), from LFAST Master.

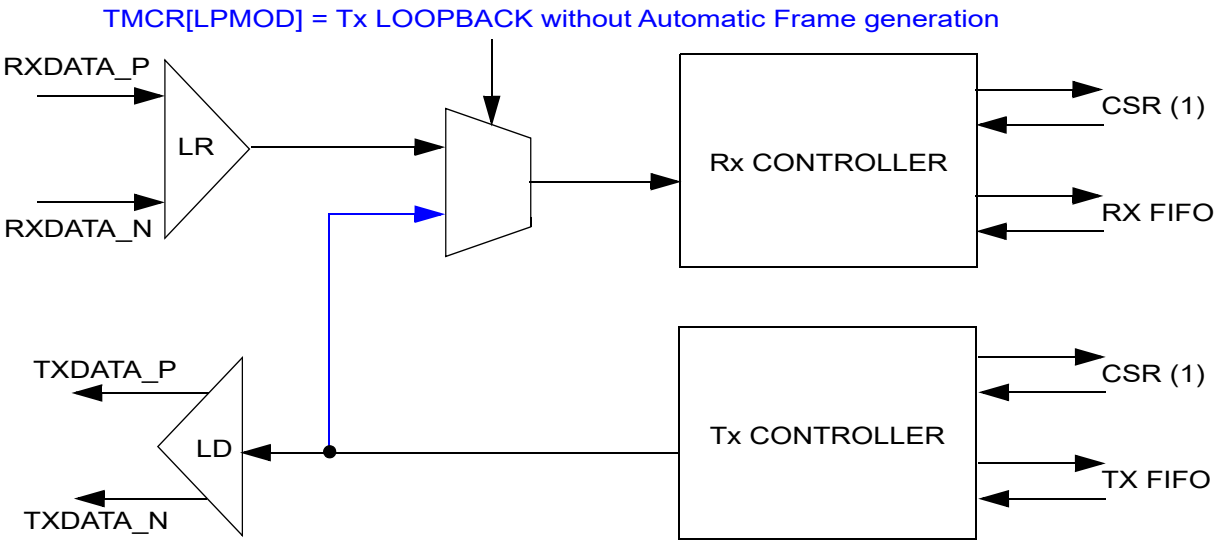
Exit from Rx LVDS Loopback mode can be done by any of the following methods:

- Software programs TMCR[LPON] = 0.
- For LFAST Slave:
  - Transmission of ICLC frame with payload 0x38 (Test mode off), from LFAST Master.

55.7.7.1.3 Tx loopback mode without automatic frame generation

This loopback mode is provided to verify the LFAST functionality, if LVDS pads are not functional. In this loopback mode the data transmitted by LFAST on Tx LVDS output is loopbacked internally on Rx LVDS input, bypassing LVDS pads.

Figure 961. Tx LoopBack mode without automatic frame generation



1. CSR stands for Control and Status Registers (used to configure and indicate the status of the Tx/Rx controller).

Entry to Tx Loopback without Automatic frame generation mode:

1. Software programs TMCR[LPMOD] = 010b.
2. Loopback can be turned on by either of the following methods:
   - Software programs TMCR[LPON] = 1.
   - For Slave Only: Reception of ICLC frame with payload 0xFF (Loopback mode on), from LFAST Master

Exit from Tx Loopback without Automatic frame generation mode can be done by any of the following methods:

- Software programs TMCR[LPON] = 0.
- For LFAST Slave: Reception of ICLC frame with payload 0x38 (Test mode off), from Tx interface (using unsolicited frame).

All frames and error flags are decoded in this mode.

55.7.7.1.4 Automatic frame generation

The LFAST module supports two Loopback modes where predefined frames are generated automatically by the Tx Interface and Rx Interface indicates its successful reception by dedicated signals and status registers. These modes are defined for BIST like check for the LFAST, with minimal software intervention.

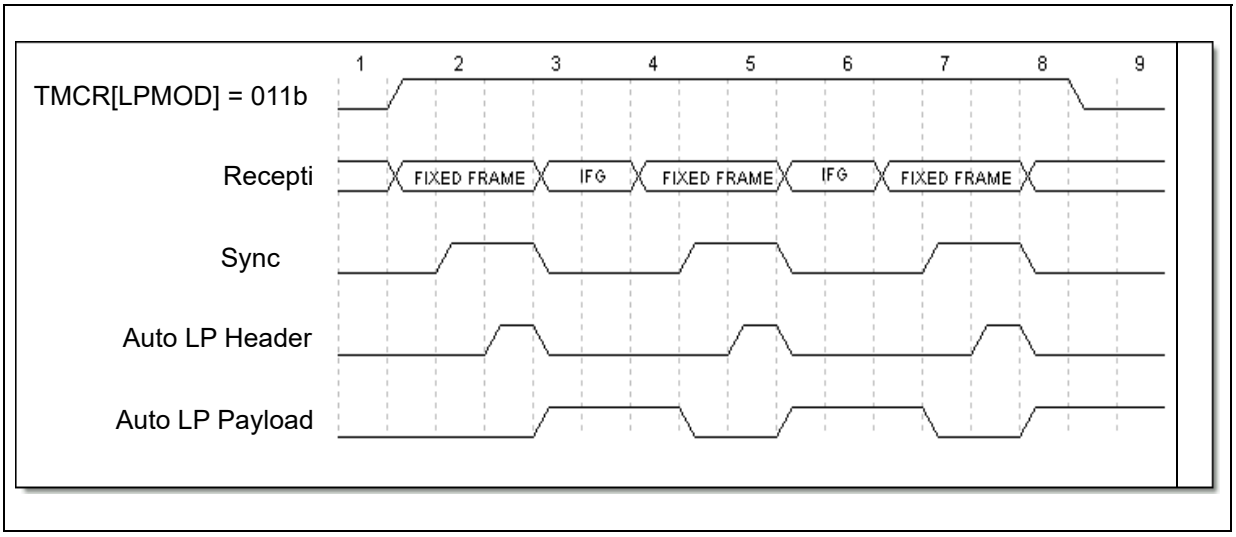
The Control register used for these modes are:

- ALCR[LPCNTEN] defines whether fixed number of auto loopback frames to be transmitted.
- ALCR[LPFMCNT] defines the number of loopback frames to be transmitted if ALCR[LPCNTEN] = 1.

The status signals and register used for these modes are:

- GSR[LPCSDV]: Valid Synchronization received.
- GSR[LPCHDV]: Frame with Header of 0x13 received.
- GSR[LPCPDV]: Frame with Payload of 0xCB received.
- GSR[LPTXDN]: Number of Auto Loopback frame transmitted with valid Synchronization, Header (13h) and Payload (CBh) is equal to ALCR[LPFMCNT].

Figure 962. Automatic Loopback test status signal timings



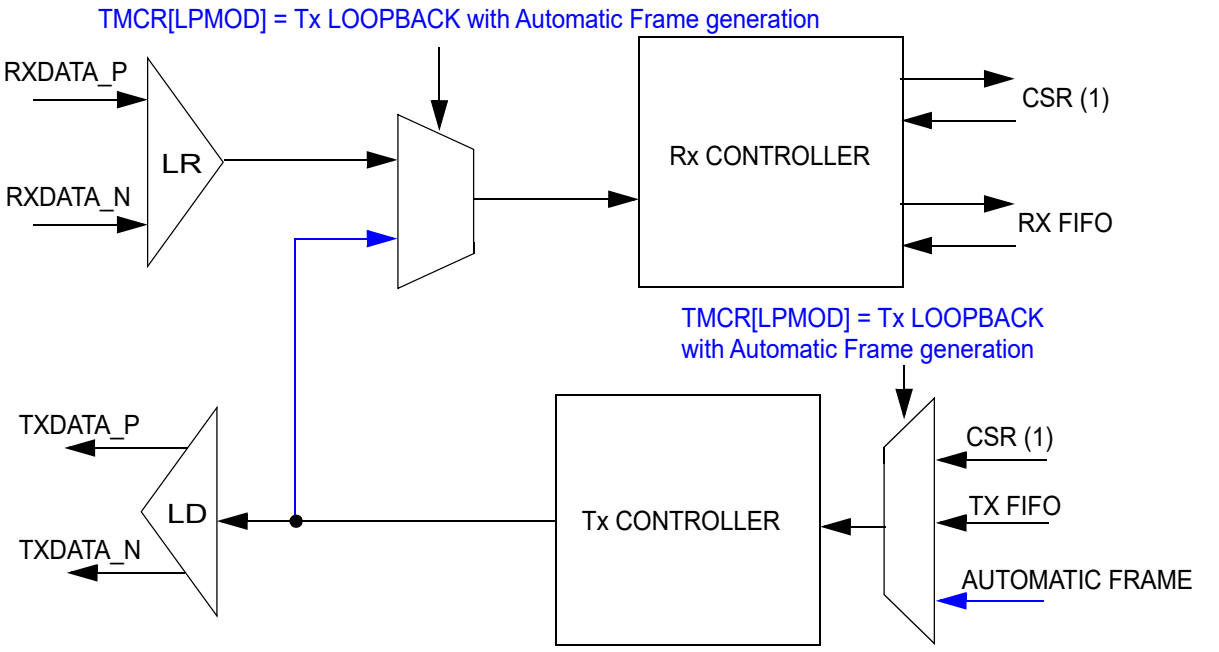
The two Loopback with automatic frame generation modes are:

- Tx Loopback with Automatic frame generation.
- Tx LVDS (external) with Automatic frame generation.

55.7.7.1.4.1 Tx LoopBack mode with automatic frame generation

When TMCR[LPMOD] = 011b (Tx Loopback mode with Automatic frame generation) the frames are generated by the Tx Interface. This mode helps to validate the Tx and Rx Path with minimal intervention from the software. The frames generated have fixed header of 0x13 and payload of 0xCB. The successful reception of this frame is indicated by the status signals and registers.

Figure 963. Tx LoopBack mode with automatic frame generation



1. CSR stands for Control and Status Registers (used to configure and indicate the status of the Tx/Rx controller).

Entry to Tx Loopback with automatic frame generation mode:

1. Software programs TMCR[LPMOD] = 011b.
2. Loopback can be turned on by either of the following methods:
   - Software programs TMCR[LPON] = 1.
   - For Slave Only: Reception of ICLC frame with payload 0xFF (Loopback mode on), from the LFAST Master.

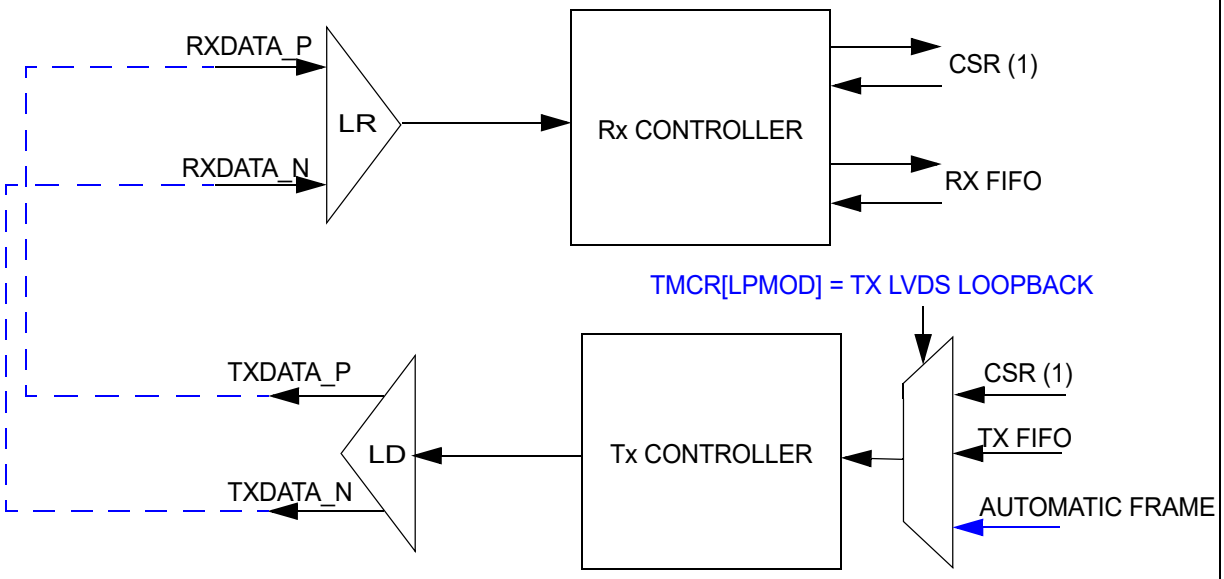
Exit from Tx Loopback with Automatic frame generation mode can be done by the following method:

- Software programs TMCR[LPON] = 0.

55.7.7.1.4.2 Tx LVDS loopback (external) mode with automatic frame generation

In this mode the LD/Tx Controller is used to test the LR/Rx Controller. The idea is to use a test frame (predefined) from the Tx Controller out through the LD, loopback completed on the DUT-board (LD connected to LR via a transmission line), back into the LR, synchronized and correlated in the Rx Controller and compared to what was sent in the Downlink controller.

Figure 964. Tx LVDS Loopback (external) mode with automatic frame generation



1. CSR stands for Control and Status Registers (used to configure and indicate the status of the Tx/Rx controller).

Entry to Tx LVDS loopback with automatic frame generation mode:

1. Software programs TMCR[LPMOD] = 100b.
2. Loopback can be turned on by either of the following methods:
   – Software programs TMCR[LPON] = 1.
   – For Slave Only: Reception of ICLC frame with payload 0xFF (Loopback mode on) from LFAST Master

Exit from Tx LVDS Loopback with automatic frame generation mode can be done by the following method:

- Software programs TMCR[LPON] = 0.

55.7.7.2 Clock test mode

The bit TMCR[CLKTST] enables or disables the Clock Test mode of the LFAST module. In this mode the LFAST sends fixed pattern out on the LD at the current configured RxData clock rate. It is a RWM bitfield and the default setting is 0 (off). This bit can be set under one of the following conditions:

- Software programs TMCR[CLKTST].
- For Slave Only: Reception of ICLC frame with payload 0x34 (Clock Test mode on) from LFAST Master.

The Tx Controller will send out a pattern of alternating 1 and 0 (pattern 101010...). This provides a divide by 2 test clock of the current Tx clock.

The Clock Test mode can be canceled by any of the following methods:

- Software programs TMCR[CLKTST] = 0.
- For LFAST Slave: Reception of ICLC frame with payload 0x38 (Test mode off) from LFAST Master.

In clock test mode the Tx Controller does not output any synchronization pattern or header, just a pattern of alternating 1's and 0's.

This mode does not affect the functionality of the Rx Interface.

55.7.8 Interrupts

The LFAST module generates five interrupts to the processor:

- Transmit complete interrupt
- Transmit exception interrupt
- Receive complete interrupt
- Receive exception interrupt
- ICLC receive interrupt

Each interrupt is asserted when any one of their status conditions is met.

55.7.8.1 Transmit complete interrupt

This interrupt indicates a transfer completion of a frame. This interrupt is asserted whenever any of the following bits are set in the TISR and also the corresponding mask bits are set in the TIER.

- TXDTF
- TXICLCF
- TXUNSF
- TXPNGF

Each of these bits can be individually maskable. So, those bits that are not required to generate an interrupt to the processor can be masked. Once the interrupt is asserted, it can be negated by clearing the corresponding flag bit in the TISR.

55.7.8.2 Transmit exception interrupt

This interrupt indicates occurrence of an exception condition in the Tx block. This interrupt is asserted whenever any of the following bits are set in the TISR along with the corresponding mask bits in the TIER.

- TXOVF
- TXIEF

Each of these bits can be individually masked. So, those bits that are not required to generate an interrupt to the processor can be masked. Once the interrupt is asserted, it can be negated by clearing the corresponding flag bit in the TISR.

Table 1016. Recommended transmit exception handling

Exception	Recommendation
TXOVF	– MCR[TXEN] = 0 – The system level interfaces transmit is reseted and enabled again – For Slave: After reception of ICLC Ping Response Request the MCR[TXEN] may be set – For Master: The MCR[TXEN] should be set and an ICLC frame Ping Response Request should be sent
TXIEF	– The MCR[TXEN] should be set

55.7.8.3 Receive complete interrupt

This interrupt indicates a reception of a frame. This interrupt is asserted whenever any of the following bits is set in the RISR along with the corresponding interrupt enable bit is also set in the RIER.

- RXCTSF
- RXDF
- RXUNSF

Each of these bits are individually maskable. So, the bits that are not required to generate an interrupt to the processor can be masked by clearing its bit in the RIER. Once the interrupt is asserted, it can be negated by clearing the corresponding flag bit in the RISR.

55.7.8.4 Receive exception interrupt

This interrupt indicates occurrence of an exception condition in Rx block. This interrupt is asserted whenever any of the following bits is set in the RISR and corresponding enable mask bit is also set in the RIER.

- RXLCEF
- RXICF
- RXSZF
- RXUOF
- RXUFF
- RXMXF
- RXMNF
- RXOFF

Each of these bits are individually maskable. So, those bits that are not required to generate an interrupt to the processor can be masked by clearing the appropriate bit in the RIER. For details on the cause of assertion of each bit refer to the RISR description. Once the interrupt is asserted, it can be negated by clearing the corresponding flag bit in the RISR.

Table 1017. Recommended receive exception handling mechanism

Exception	Recommendation
RXLCEF	The MCR[RXEN] may be set and cleared, as this exception can result in loss of subsequent packet
RXICF	No action required

Table 1017. Recommended receive exception handling mechanism (continued)

Exception	Recommendation
RXSZF	The MCR[RXEN] may be set and cleared, as this exception can result in loss of subsequent packet
RXOFF	– The MCR[RXEN] may be cleared – The system level interfaces receive may be reset and enabled, as current transfer may be corrupted
RXUFF	No action required
RXMXF	The system level interfaces receive should be enabled, if not enabled
RXMNF	No action required
RXUNSF	No action required

55.7.8.5 ICLC receive interrupt

This interrupt indicates reception of a valid ICLC frame. This interrupt is asserted whenever any of the following bits is set in the RIISR along with the corresponding enable bit mask bit is set in the RIIER.

- ICPONF
- ICPOFF
- ICTSF
- ICTFF
- ICRSF
- ICRFF
- ICTEF
- ICTDF
- ICCTF
- ICLPF
- ICTOF
- ICPRF
- ICPSF
- ICPFF

Each of these bits are individually maskable. So those bits that are not required to generate an interrupt to the processor can be masked. For details on the cause of assertion of each bit, refer RIISR. Once the interrupt is asserted, it can be negated by clearing the corresponding flag bit in RIISR.

55.8 Packet memory

The LFAST stores packet frames for transmission, and reads packet frames after reception. The transmitter has its own dedicated buffer and the receiver has it own dedicated buffer, and they are not shared between each other.

Table 1018. Frames supported by LFAST interfaces

Frame Type	Tx Buffer (in bits)	Rx Buffer (in bits)	Memory Type
Data Frame	38 × 32 max 6 packets	38 × 32 max 6 packets	FIFO
Unsolicited Frame	9 × 32 max 1 packet	9 × 32 max 1 packet	Registers UNSTD[8-0], UNSRDR[8-0]
ICLC Frame	1 × 8 max 1 packet(1)	1 × 8 max 1 packet(2)	Registers ICR, PISR
CTS Frame	N/A	N/A	N/A

1. Only for MCR[MSEN] = 1
2. Only for ping response data when MCR[MSEN] = 1

55.9 Resets

The various blocks in LFAST are reset as described in *Table 1019*.

Table 1019. Recommended receive exception handling mechanism

Reset	Blocks reset
Asynchronous Hardware reset	Polarity: Active Low – Clock control module (CCM) – LFAST Domain Logic (Rx and Tx block) – System Side Module Interface FIFOs – LFAST Register Space
DRFRST bit of *Mode Configuration Register (MCR)*	Polarity: Active High – Clock control module (CCM) – LFAST Domain Logic (Rx and Tx block) – System Side Module Interface FIFOs – LFAST Register Space
DRFEN bit of *Mode Configuration Register (MCR)*	Polarity: Active Low – Clock control module (CCM) – LFAST Domain Logic (Rx and Tx block) – System Side Module Interface FIFOs – LFAST Status Registers – SCR[RDR] and SCR[TDR] – TMCR[CLKTST] and TMCR[LPON] – ICR[ICLCSEQ] and ICR[SNDICLC] – PICR[PNGREQ] – UNSTCR[USNDRQ]

55.10 Clocks

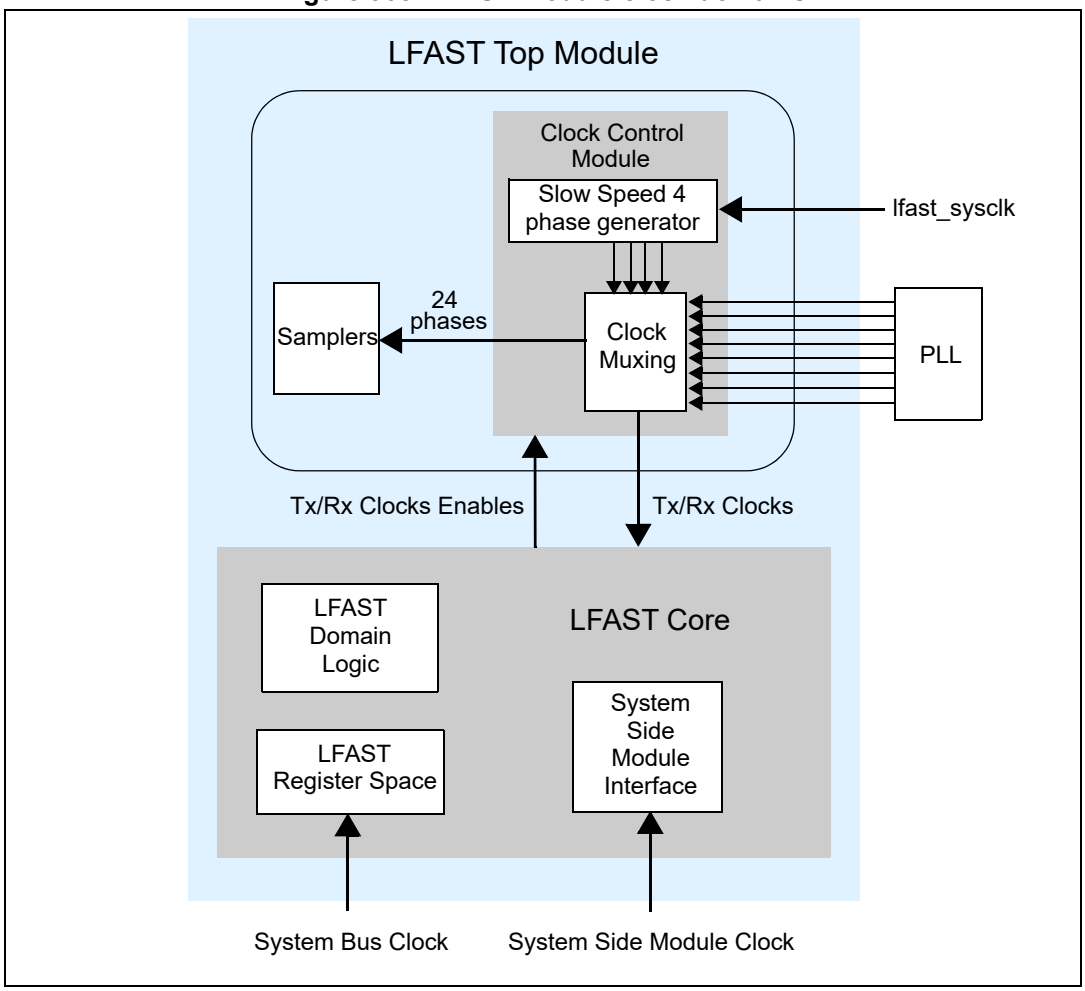
The LFAST mainly works on three clocks:

- System Bus Clock, used to program the registers.
- Protocol clock, which is either High Speed PLL clock or Low Speed clock depending on the speed mode, used for LFAST protocol operation.
- System Side Module clock, which is synchronous to the System Bus clock.

55.10.1 Clocking strategy

Figure 965 shows the clock domain in which each module functions. The PLL provides eight phases of the high speed clock to the Clock Muxing portion of the Clock Control Module. The Clock Module will then generate four phases of slow speed clock using both the edges of lfast_sysclk, muxes high speed and low speed clocks and provides a muxed clock to the Sampler Module.

Figure 965. LFAST module clock domains

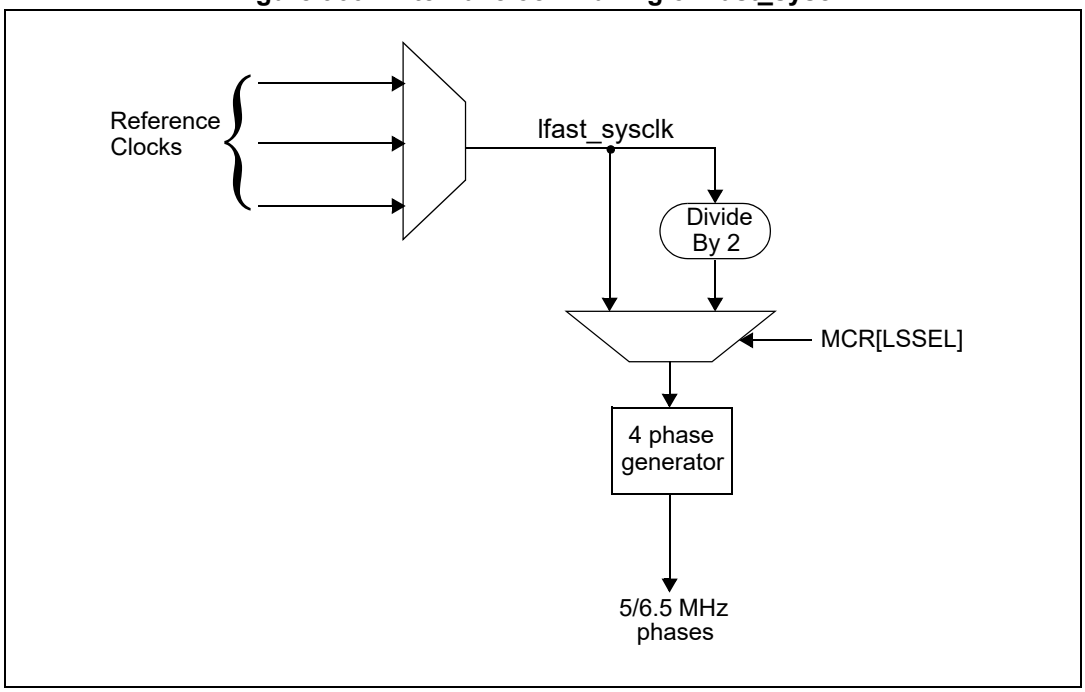


55.10.2 Slow speed clock

55.10.2.1 External muxing

The Clock Control Module will receive lfast_sysclk from the clocking subsystem as shown in *Figure 966*, *Figure 967* and *Figure 968*.

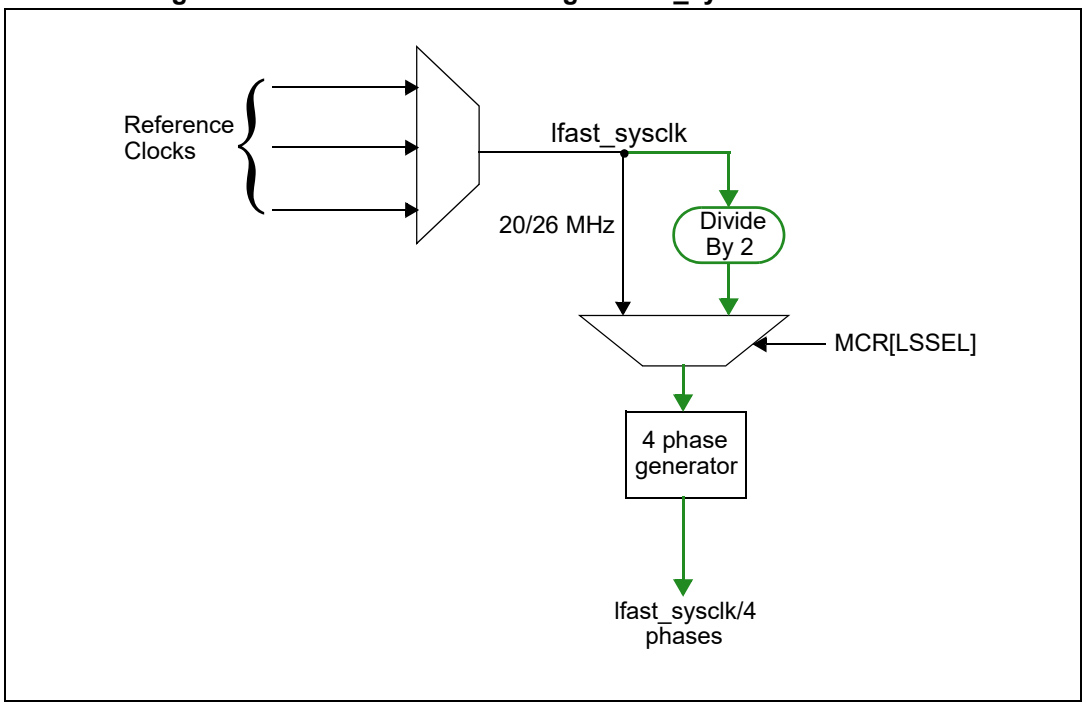
Figure 966. External clock muxing of lfast_sysclk



All the reference clocks will be muxed first to provide lfast_sysclk to the module and then either div/2 or direct muxed clock will be used to generate 4 slow speed phases in the Clock Control Module of LFAST as shown in *Figure 966*.

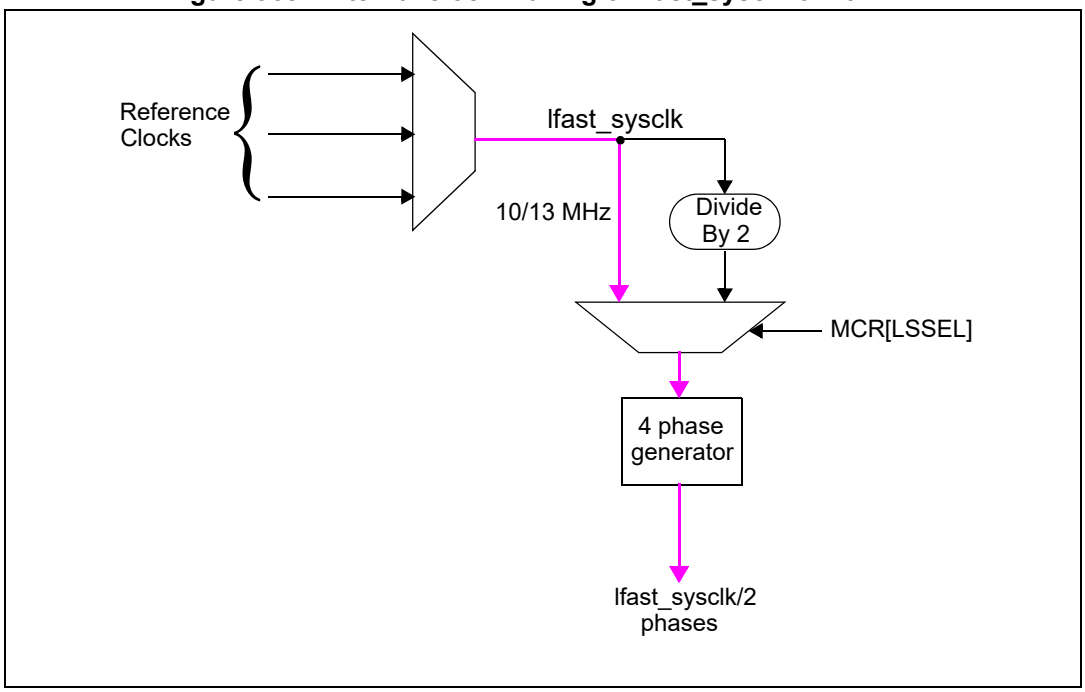
The lfast_sysclk could be either 20/26 MHz or 10/13 MHz. When lfast_sysclk is 20/26 MHz frequency, the clock control module will generate 4 phases of slow speed clock of frequency lfast_sysclk/4 when MCR[LSSEL] = 1. If lfast_sysclk is 20/26 MHz it needs to be first divided by 2 before using it for 4 phase generation in LFAST Clock Module, which generates 4 phases of half of the input frequency as shown in *Figure 967* (selected clock path shown in green).

Figure 967. External clock muxing of lfast_sysclk of 20/26 MHz



In this case, lfast_sysclk is 10/13 MHz frequency, the clock control module will generate 4 phases of slow speed clock of frequency lfast_sysclk/2 when MCR[LSSEL] = 0. If lfast_sysclk is 10/13 MHz then it does not need to be first divided by 2 before using it for 4 phase generation in LFAST Clock Module, which generates 4 phases of half of the input frequency as shown in *Figure 968* (selected clock path selected shown in magenta).

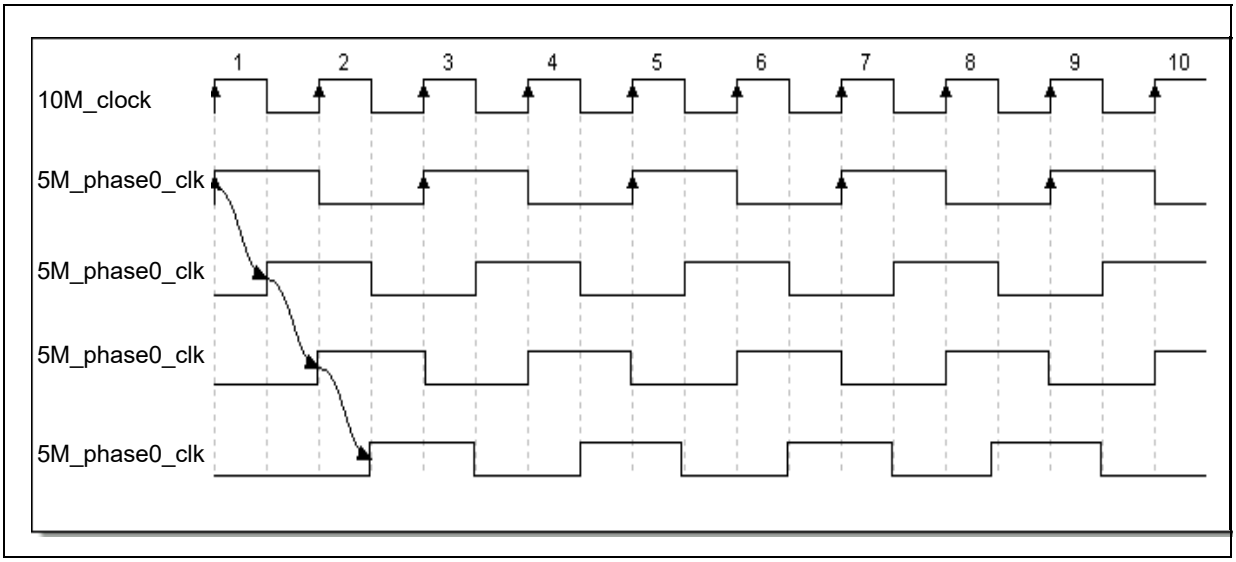
Figure 968. External clock muxing of lfast_sysclk of 10 MHz



55.10.2.2 Slow speed 4-phase generator

Clock control module will generate 4 phases of slow speed clock of frequency of 10/13 MHz. For instance, if lfast_sysclk is 10 MHz then 4 phases of 5 MHz will be generated. *Figure 969* Shows 4 phases getting generated using lfast_sysclk.

Figure 969. Slow speed 4 phases generation

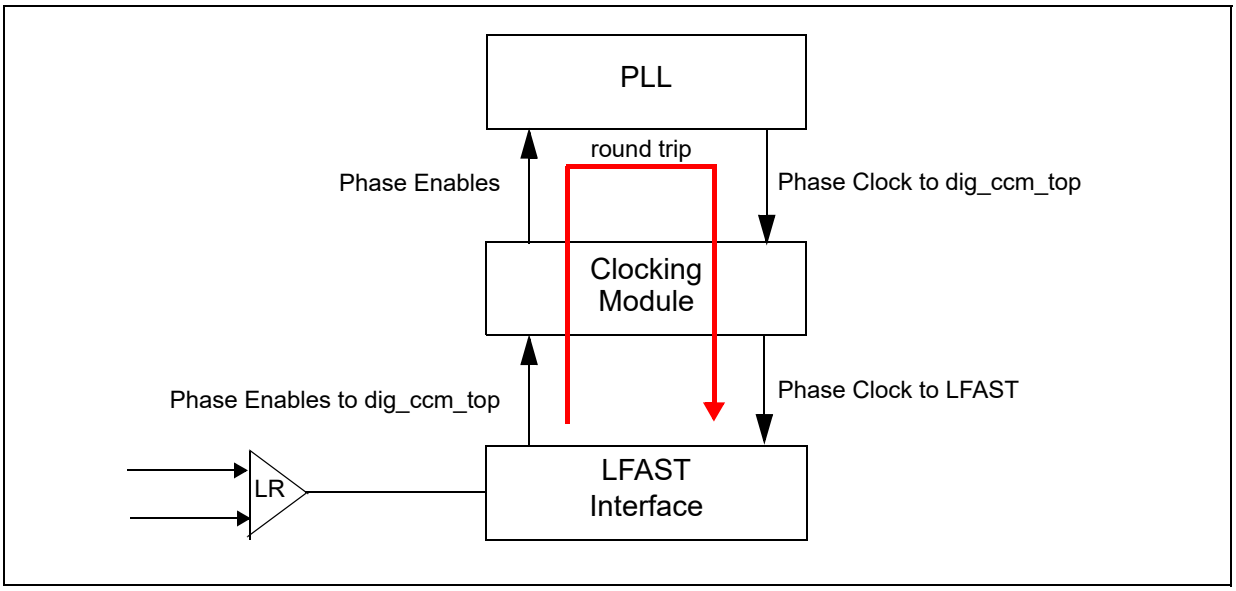


55.10.3 Rx controller clocks

55.10.3.1 Clocking module requirements for high speed phases

The high speed Phases are obtained from the PLL via the Clocking Module as shown in *Figure 970*.

Figure 970. Clock enables and clock paths



The LFAST block will know which of the phases will be required for the different modes of operations. Therefore, the LFAST block will provide enables and speed mode switches to the Clocking Module. The Clocking Module will use these signals to control the enables to the clock phases to reduce the power consumption.

The 8 phases of clocks signal are fed to the Clocking Module and muxed with the low speed phases. Depending on enables and data rate speed modes, the selected clocks are passed to the Sampler block and interface block.

COCR[PHSSEL] is connected to the Clocking Module, which selects whether 8 or 4 phases are selected for High Speed mode. It is only valid for high speed.

The routing of the 8 high speed phases to the interface is critical. Each clock phase will need to have the same routing track length from the PLL to the interface of Clocking Module. Each of the 8 high speed phases are separated by 45 degrees. This separation needs to be maintained from the phase generator to the interface.

55.10.3.2 Clock module requirements for low speed phases

55.10.3.2.1 Low speed

The low speed clock phases are generated in Clocking Module. These four phases are required to sample the data correctly at Low Speed mode. The LFAST block will provide the enable for the phases. The clock source for the low speed phases is SYSCLK. The Clocking Module block will need to generate the 4 phases of low speed clock 90 degrees apart.

Using the Low Speed mode on the interface allows the polyphase generation logic to be turned off and the requirement of high-speed-freq × 8 MHz clock from the PLL is not needed.

Table 1020. Rx clocks summary

Rx Speed mode	Source
Low Speed	lfast_sysclk
High Speed	PLL

55.10.4 Tx controller clocks

55.10.4.1 Clocking module requirements for speed phases

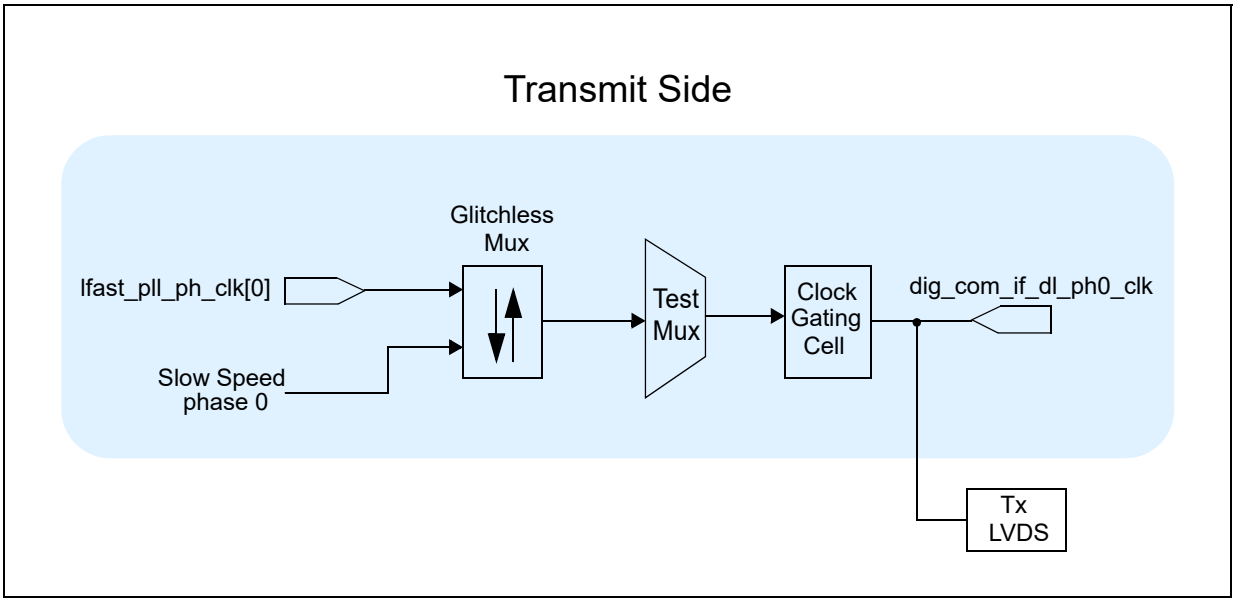
The low speed phase 0 clock is sourced from the lfast_sysclk and the high speed phase 0 clock is sourced from the PLL. See *Table 1021*.

Table 1021. Tx clocks summary

Tx Speed mode	Source
Low Speed	lfast_sysclk
High Speed	PLL

55.10.4.2 Transmit clock muxing

Figure 971. Transmit clocks muxing



55.10.4.3 Tx request clock control

The Tx phase 0 clock is enabled when all the resets are negated.

56 Ethernet

56.1 Overview

The Ethernet module enables a host to transmit and receive data over the Ethernet in compliance with the IEEE 802.3-2008. *Figure 972* shows a system-level block diagram of the Ethernet module.

As shown in *Figure 972*, the Ethernet module uses a 64-bit AMBA High-Performance Bus (AHB) on the application side. The AHB interface transfers the data to and from system memory through AHB master interface. One AHB master interface is connected to all DMA channels. The DMA arbiter helps in arbitration of all the paths (Transmit and Receive) in all channels.

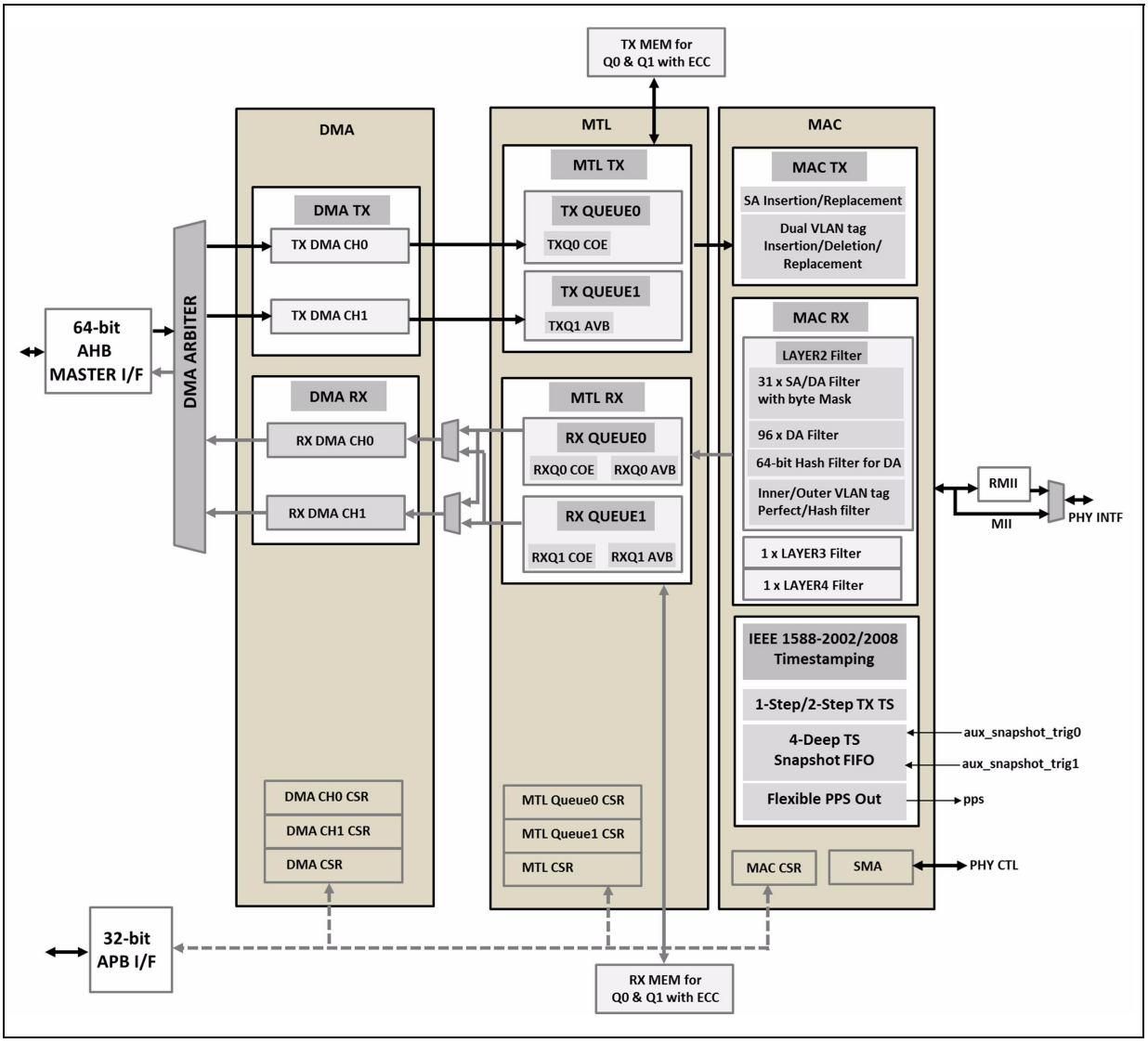
As shown in *Figure 972*, the Ethernet module uses a 32-bit APB interface to program the Control and Status registers (CSR). Each channel has a separate set of Control and Status registers (CSR) for managing the Transmit and Receive functions, descriptor handling, and interrupt handling.

The Transmit FIFO (Tx FIFO) buffers the data transferred from the application to the Ethernet module. Similarly, the Receive FIFO (Rx FIFO) stores the Ethernet frames received from the line until they can be transferred to the application. These are asynchronous FIFOs because they also transfer the data between the application clock and the MAC line clocks. Tx memory and Rx memory are one-ported RAM of 68-bit width for 64 data bus widths. The extra bits are used for storing the byte-enable information.

For multiple queues, all Tx queue share the Tx FIFO memory and all Rx queue share the Rx FIFO memory. The application can program the size of the FIFO memory allocated to each Tx or Rx queue. The Ethernet module provides a direct read or write access to the FIFO memories in Debug mode.

The Ethernet module can be programed to use either of MII or RMII PHY interfaces.

Figure 972. System Level block diagram



56.1.1 Ethernet module Features

56.1.1.1 Standard Compliance

The Ethernet module is compliant with the following standards:

- IEEE 802.3-2008 for Ethernet MAC, Media Independent Interface (MII)
- IEEE 1588-2008 for precision networked clock synchronization
- IEEE 802.1AS-2011 and 802.1-Qav-2009 for Audio Video (AV) traffic
- IEEE 802.3az-2010 for Energy Efficient Ethernet (EEE)
- AMBA 2.0 for AHB master and APB slave ports
- RMII specification version 1.2 from RMII consortium

56.1.1.2 MAC

The MAC supports the following features:

- MAC Tx and Rx features
- MAC Tx features
- MAC Rx features

56.1.1.2.1 MAC Tx and Rx features

Following is the list of MAC Tx and Rx features:

- Separate transmission, reception, and control interfaces to the application
- 10 and 100, Mbps data transfer rates with the following PHY interfaces:
  - IEEE 802.3-compliant MII interface to communicate with an external Fast Ethernet PHY
  - RMII interface to communicate with an external Fast Ethernet PHY
- Half-duplex operation:
  - CSMA/CD Protocol support
  - Flow control using backpressure support
  - Standard IEEE 802.3az-2010 for Energy Efficient Ethernet in MII PHYs.
- 64-bit data transfer interface on the application side
- Full-duplex flow control operations (IEEE 802.3x Pause packets and Priority flow control)
- Network statistics with RMON or MIB Counters (RFC2819/RFC2665)
- Support for Ethernet packet timestamping as described in IEEE 1588-2002 and IEEE 1588-2008 (64-bit timestamps given in the Tx or Rx status of PTP packet). Both one-step and two-step timestamping is supported in TX direction
- Flexibility to control the Pulse-Per-Second (PPS) output signal
- MDIO (Clause 22 and Clause 45) master interface for PHY device configuration and management

56.1.1.2.2 MAC Tx features

Following is the list of MAC Tx features:

- Preamble and start of packet data (SFD) insertion
- Separate 32-bit status for each packet transmitted from the application
- Automatic CRC and pad generation controllable on a per-packet basis
- Programmable packet length to support Standard or Jumbo Ethernet packets with up to 16 KB of size
- Programmable Inter Packet Gap (40–96 bit times in steps of 8)
- IEEE 802.3x Flow Control automatic transmission of zero-quanta Pause packet when flow control input transitions from assertion to de-assertion (in full-duplex mode)
- Source Address field insertion or replacement, and VLAN insertion, replacement, and deletion in transmitted packets with per-packet or static-global control
- Insertion, replacement, or deletion of up to two VLAN tags
- Option to transmit packets with reduced preamble size in full-duplex mode

56.1.1.2.3 MAC Rx features

Following is the list of MAC Rx features:

- Automatic Pad and CRC Stripping options
- Option to disable Automatic CRC checking
- Preamble and SFD deletion
- Separate 112-bit or 128-bit status
- Programmable watchdog timeout limit
- Flexible address filtering modes:
  - 128 MAC addresses for unicast 48-bit perfect filtering
    - 31 filters for destination address (DA) or source address (SA) comparison with masks for each byte
    - 97 filters for destination address (DA) that can be selected in blocks of 32 and 64 registers
  - 64 bit Hash filter for multicast and unicast (DA) addresses
  - Option to pass all multicast addressed packets
  - Promiscuous mode to pass all packets without any filtering for network monitoring
  - Pass all incoming packets (as per filter) with a status report
- Additional packet filtering:
  - VLAN tag-based: Perfect match and Hash-based filtering. Filtering based on either outer or inner VLAN tag is possible
  - Layer 3 and Layer 4-based: TCP or UDP over IPv4 or IPv6
- IEEE 802.1Q VLAN tag detection and option to delete the VLAN tags in received packets
- module to detect remote wake-up packets and AMD magic packets
- forwarding of received Pause packets to the application (in full-duplex mode)
- Receive module for Layer 3/Layer 4 checksum offload for received packets
- Stripping of up to two VLAN Tags and providing the tags in the status

56.1.1.3 MAC Transaction Layer (MTL)

The MTL block consists of the following FIFOs: Tx FIFO and Rx FIFO. The FIFO space is shared by multiple queues. The buffer size is configured for each queue in multiples of 256 bytes.

56.1.1.3.1 MTL Tx and Rx features

Following is the list of MTL Tx and Rx features:

- 64-bit Transaction Layer block (bridges the application and the MAC)
- Data transfers executed using simple FIFO protocol
- Synchronization for all clocks in the design (Transmit, Receive, and Application clocks)
- Optimization for packet-oriented transfers with packets delimiters
- Programmable threshold capability for each queue (default of 64 bytes)
- Debug and slave mode operation on Queue 0 (default queue)

56.1.1.3.2 MTL Tx features

Following is the list of MTL Tx features:

- Two single-port RAM-based (SPRAM) synchronous FIFO controllers
- 4 KB TX FIFO size
- Dual queues on the Transmit path with a common memory for both Tx queues
- Store-and-Forward mechanism or threshold mode (cut-through) for transmission to the MAC
- Programmable queue size. Each queue size can be programed in terms of 256 bytes
- Automatic retransmission of collision packets in half-duplex mode
- Discard packets on late collision, excessive collisions, excessive deferral, and under-run conditions with appropriate status
- Disabling of Data Memory RAM chip-select when inactive to reduce power consumption
- Checksum offload engine calculates and inserts IPv4 header checksum and TCP, UDP, or ICMP checksum
- Statistics by generating pulses for packets dropped (because of underflow) in the Tx FIFO
- Programmable interrupt options for different operational conditions
- Option to support dropping of Tx Status to improve the Transmit throughput
- Packet-level control for:
  - VLAN tag insertion or replacement
  - Ethernet source address insertion
  - Layer3/Layer4 Checksum insertion control
  - One-step timestamp
  - Timestamp control
  - CRC and pad control
- Following scheduling algorithms in configurations with multiple queues:
  - Weighted Round Robin (WRR)
  - Strict Priority (SP)
  - Credit-based Shaper (CBS) for AVB
- Statistics related to bandwidth consumption by each queue of up to 16 blocks over a 125 µs period

56.1.1.3.3 MTL Rx features

Following is the list of MTL Rx features:

- 8 KB Rx FIFO size
- Dual queues on the Receive path with a common memory for all Rx queues
- Programmable Rx queue threshold (default fixed at 64 bytes) in Threshold (or cut-through) mode
- Option to filter all error packets on reception and not forward them to the application in the store-and-forward mode
- Option to forward the undersized good packets
- Statistics by generating pulses for packets dropped (because of overflow) in the Rx FIFO
- Automatic generation of Pause packet control or backpressure signal to the MAC based on the Rx Queue fill level
- Arbitration among queues when multiple queues are present. The following arbitration schemes are supported:
  - Weighted Round Robin (WRR)
  - Weighted Strict priority (WSP)
  - Strict Priority (SP)

56.1.1.4 DMA block

The DMA block exchanges data between the MTL block and system memory. The well-defined descriptors structure acts as a software and hardware interface. The application can use a set of registers (DMA CSR) to control the DMA operations. The DMA block supports the following features:

- 64-bit data transfers
- 2-channel Transmit and 2-channel Receive engines
- Separate DMA channel in the Transmit path for each queue in MTL
- 2-DMA channels for queues in MTL Receive path
- Fully synchronous design operating on a single application clock except for CSR module
- Optimization for packet-oriented DMA transfers with packet delimiters
- Byte-aligned addressing for data buffer support
- Dual-buffer (ring) descriptor support
- Descriptor architecture to allow large blocks of data transfer with minimum CPU intervention (each descriptor can transfer up to 32 KB of data)
- Comprehensive status reporting for normal operation and transfers with errors
- Individual programmable burst length for Tx DMA and Rx DMA engines for optimal host bus utilization
- Programmable interrupt options for different operational conditions

- Per-packet Transmit or Receive Complete Interrupt control
- Round-robin or fixed-priority arbitration between the Receive and Transmit engines
- Start and Stop modes
- Separate ports for host CSR access and host data interface
- Routing of received packets to the RX DMA channels based on the DA or VLAN Priority in multi-channel DMA configurations
- Time-sensitive conditional packet transmission by comparing the Slot Time information provided in the descriptor (useful for AV applications)

56.1.1.5 AMBA AHB Master Interface

The AHB master interface supports the following features:

- Interfaces with the application through 64-bit AHB
- Option to select address-aligned bursts from AHB master port
- Split, Retry, and Error AHB responses
- AHB 1 K boundary burst splitting
- Software-selected type of AHB burst (fixed burst, indefinite burst, or mix of both)

The AHB master interface does not generate the following:

- Wrap burst
- Locked or Protected transfers

56.1.1.6 AMBA APB Slave Interface

The Ethernet module supports the 32-bit APB slave interface without any byte-enable to program the CSR registers.

56.1.1.7 Audio Video Support

The Ethernet module supports the following Audio Video (AV) features:

- Separate channels or queues for AV data transfer in 100 Mbps modes
- Up to 2-queues on the Receive paths for AV traffic and 1-queue on the Transmit path for AV traffic
- IEEE 802.1-Qav specified credit-based shaper (CBS) algorithm for Transmit channels
- Single Tx FIFO and Rx FIFO (MTL) for all selected queues (system-side interface [AHB] remains the same)

56.1.1.8 Monitoring, Testing, and Debugging Support

The Ethernet module supports the following features for monitoring, testing, and debugging:

- Internal loopback from Tx to Rx on the MII for debugging
- DMA states (Tx and Rx) as status bits
- Debug status register that gives status of FSMs in Transmit and Receive data paths and FIFO fill-levels
- Application Abort status bits
- MMC (RMON) module
- Current Tx or Rx Buffer pointer as status registers
- Current Tx or Rx Descriptor pointer as status registers
- Statistical counters to calculate the bandwidth served by each Transmit channel when AV support is enabled
- Tx or Rx Queues memory accessible through Slave port for debug

56.2 Memory map and register description

This section provides a detailed description of all registers accessible in this module.

Table 1022. Ethernet memory map

Address offset (hex)	Register	Location
0x0000	MAC Configuration register (MAC_CONFIGURATION)	*Section 56.2.1*
0x0004	MAC Extended Configuration register (MAC_EXT_CONFIGURATION)	*Section 56.2.2*
0x0008	MAC Packet Filter register (MAC_PACKET_FILTER)	*Section 56.2.3*
0x000C	Watchdog Timeout register (MAC_WATCHDOG_TIMEOUT)	*Section 56.2.4*
0x0010	Hash Table register 0 (MAC_HASH_TABLE_REG0)	*Section 56.2.5*
0x0014	Hash Table register 1 (MAC_HASH_TABLE_REG1)	*Section 56.2.6*
0x0018–0x004F	Reserved	
0x0050	VLAN Tag register (MAC_VLAN_TAG)	*Section 56.2.7*
0x0054–0x0057	Reserved	
0x0058	VLAN Hash Table register (MAC_VLAN_HASH_TABLE)	*Section 56.2.8*
0x005C–0x005F	Reserved	
0x0060	VLAN Tag Inclusion register (MAC_VLAN_INCL)	*Section 56.2.9*
0x0064	Inner VLAN Tag Inclusion register (MAC_INNER_VLAN_INCL)	*Section 56.2.10*
0x0068–0x006F	Reserved	
0x0070	MAC Q0 Flow Control register (MAC_Q0_TX_FLOW_CTRL)	*Section 56.2.11*
0x0074	MAC Q1 Flow Control register (MAC_Q1_TX_FLOW_CTRL)	*Section 56.2.12*
0x0078–0x008F	Reserved	
0x0090	MAC Receive Flow Control register (MAC_RX_FLOW_CTRL)	*Section 56.2.13*
0x0094–0x0097	Reserved	

Table 1022. Ethernet memory map (continued)

Address offset (hex)	Register	Location
0x0098	Transmit Queue Priority Mapping 0 register (MAC_TXQ_PRTY_MAP0)	*Section 56.2.14*
0x009C–0x009F	Reserved	
0x00A0	Receive Queue Control 0 register (MAC_RXQ_CTRL0)	*Section 56.2.15*
0x00A4	Receive Queue Control 1 register (MAC_RXQ_CTRL1)	*Section 56.2.16*
0x00A8	Receive Queue Control 2 register (MAC_RXQ_CTRL2)	*Section 56.2.17*
0x00AC–0x00AF	Reserved	
0x00B0	Interrupt Status register (MAC_INTERRUPT_STATUS)	*Section 56.2.18*
0x00B4	Interrupt Enable register (MAC_INTERRUPT_ENABLE)	*Section 56.2.19*
0x00B8	Receive Transmit Status register (MAC_RX_TX_STATUS)	*Section 56.2.20*
0x00BC–0x00BF	Reserved	
0x00C0	PMT Control and Status register (MAC_PMT_CONTROL_STATUS)	*Section 56.2.21*
0x00C4	Wake-up Packet Filter registers	*Section 56.2.22*
0x00C8–0x00CF	Reserved	
0x00D0	LPI Control and Status register (MAC_LPI_CONTROL_STATUS)	*Section 56.2.23*
0x00D4	LPI Timers Control register (MAC_LPI_TIMERS_CONTROL)	*Section 56.2.24*
0x00D8	LPI Entry Timer register (MAC_LPI_ENTRY_TIMER)	*Section 56.2.25*
0x00DC	1US Tic Counter register (MAC_1US_TIC_COUNTER)	*Section 56.2.26*
0x00E0–0x010F	Reserved	
0x0110	Version register (MAC_VERSION)	*Section 56.2.27*
0x0114	Debug register (MAC_DEBUG)	*Section 56.2.28*
0x0118–0x011B	Reserved	
0x011C	Hardware Feature0 register (MAC_HW_FEATURE0)	*Section 56.2.29*
0x0120	Hardware Feature1 register (MAC_HW_FEATURE1)	*Section 56.2.30*
0x0124	Hardware Feature2 register (MAC_HW_FEATURE2)	*Section 56.2.31*
0x0128–0x01FF	Reserved	
0x0200	MDIO Address register (MAC_MDIO_ADDRESS)	*Section 56.2.32*
0x0204	MDIO Data register (MAC_MDIO_DATA)	*Section 56.2.33*
0x0208–0x021F	Reserved	
0x0210	ARP Address register (MAC_ARP_ADDRESS)	*Section 56.2.34*
0x0214–0x02FF	Reserved	
0x0300	Address0 High register (MAC_ADDRESS0_HIGH)	*Section 56.2.35*
0x0304	Address0 Low register (MAC_ADDRESS0_LOW)	*Section 56.2.36*
(for *n* = 1 to 31)		
0x0300 + *n**0x8	MAC Address High register 1 to 31 (MAC_ADDRESS*n*_HIGH31_1)	*Section 56.2.37*

Table 1022. Ethernet memory map (continued)

Address offset (hex)	Register	Location
0x0304 + *n**0x8	MAC Address Low register 1 to 31 (MAC_ADDRESS*n*_LOW31_1)	*Section 56.2.38*
(for *n* = 32 to 63)		
0x0300 + *n**0x8	MAC Address High register 32 to 63 (MAC_ADDRESS*n*_HIGH63_32)	*Section 56.2.39*
0x0304 + *n**0x8	MAC Address Low register 32 to 63 (MAC_ADDRESS*n*_LOW63_32)	*Section 56.2.40*
(for *n* = 64 to 127)		
0x0300 + n*0x8	MAC Address High register 64 to 127 (MAC_ADDRESS*n*_HIGH127_64)	*Section 56.2.41*
0x0304 + *n**0x8	MAC Address Low register 64 to 127 (MAC_ADDRESS*n*_LOW127_64)	*Section 56.2.42*
0x0700	MMC Control register (MMC_CONTROL)	*Section 56.2.43*
0x0704	MMC Receive Interrupt register (MMC_RX_INTERRUPT)	*Section 56.2.44*
0x0708	MMC Tx Interrupt register (MMC_TX_INTERRUPT)	*Section 56.2.45*
0x070C	MMC Rx Interrupt mask register (MMC_RX_INTERRUPT_MASK)	*Section 56.2.46*
0x0710	MMC Tx Interrupt Mask register (MMC_TX_INTERRUPT_MASK)	*Section 56.2.47*
0x0714	Transmit Octet Count Good Bad register (TX_OCTET_COUNT_GOOD_BAD)	*Section 56.2.48*
0x0718	Transmit Packet Count Good Bad register (TX_PACKET_COUNT_GOOD_BAD)	*Section 56.2.49*
0x071C	Transmit Broadcast Packets Good register (TX_BROADCAST_PACKETS_GOOD)	*Section 56.2.50*
0x0720	Transmit Multicast Packets Good register (TX_MULTICAST_PACKETS_GOOD)	*Section 56.2.51*
0x0724	Transmit 64Octets Packets Good Bad register (TX_64OCTETS_PACKETS_GOOD_BAD)	*Section 56.2.52*
0x0728	Transmit 65to127Octets Packets Good Bad register (TX_65TO127OCTETS_PACKETS_GOOD_BAD)	*Section 56.2.53*
0x072C	Transmit 128To255Octets Packets Good Bad register (TX_128TO255OCTETS_PACKETS_GOOD_BAD)	*Section 56.2.54*
0x0730	Transmit 256To511Octets Packets Good Bad register (TX_256TO511OCTETS_PACKETS_GOOD_BAD)	*Section 56.2.55*
0x0734	Transmit 512To1023Octets Packets Good Bad register (TX_512TO1023OCTETS_PACKETS_GOOD_BAD)	*Section 56.2.56*
0x0738	Transmit 1024ToMaxOctets Packets Good Bad register (TX_1024TOMAXOCTETS_PACKETS_GOOD_BAD)	*Section 56.2.57*
0x073C	Transmit Unicast Packets Good Bad register (TX_UNICAST_PACKETS_GOOD_BAD)	*Section 56.2.58*
0x0740	Transmit Multicast Packets Good Bad register (TX_MULTICAST_PACKETS_GOOD_BAD)	*Section 56.2.59*
0x0744	Transmit Broadcast Packets Good Bad register (TX_BROADCAST_PACKETS_GOOD_BAD)	*Section 56.2.60*

Table 1022. Ethernet memory map (continued)

Address offset (hex)	Register	Location
0x0748	Transmit Underflow Error Packets register (TX_UNDERFLOW_ERROR_PACKETS)	*Section 56.2.61*
0x074C	Transmit Single Collision Good Packets register (TX_SINGLE_COLLISION_GOOD_PACKETS)	*Section 56.2.62*
0x0750	Transmit Multiple Collision Good Packets register (TX_MULTIPLE_COLLISION_GOOD_PACKETS)	*Section 56.2.63*
0x0754	Transmit Deferred Packets register (TX_DEFERRED_PACKETS)	*Section 56.2.64*
0x0758	Transmit Late Collision Packets register (TX_LATE_COLLISION_PACKETS)	*Section 56.2.65*
0x075C	Transmit Excessive Collision Packets register (TX_EXCESSIVE_COLLISION_PACKETS)	*Section 56.2.66*
0x0760	Transmit Carrier Error Packets register (TX_CARRIER_ERROR_PACKETS)	*Section 56.2.67*
0x0764	Transmit Octet Count Good register (TX_OCTET_COUNT_GOOD)	*Section 56.2.68*
0x0768	Transmit Packet Count Good register (TX_PACKET_COUNT_GOOD)	*Section 56.2.69*
0x076C	Transmit Excessive Deferral Error register (TX_EXCESSIVE_DEFERRAL_ERROR)	*Section 56.2.70*
0x0770	Transmit Pause Packets register (TX_PAUSE_PACKETS)	*Section 56.2.71*
0x0774	Transmit VLAN Packets Good register (TX_VLAN_PACKETS_GOOD)	*Section 56.2.72*
0x0778	Transmit OSize Packets Good register (TX_OSIZE_PACKETS_GOOD)	*Section 56.2.73*
0x077C–0x077F	Reserved	
0x0780	Receive Packets Count Good Bad register (RX_PACKETS_COUNT_GOOD_BAD)	*Section 56.2.74*
0x0784	Receive Octet Count Good Bad register (RX_OCTET_COUNT_GOOD_BAD)	*Section 56.2.75*
0x0788	Receive Octet Count Good register (RX_OCTET_COUNT_GOOD)	*Section 56.2.76*
0x078C	Receive Broadcast Packets Good register (RX_BROADCAST_PACKETS_GOOD)	*Section 56.2.77*
0x0790	Receive Multicast Packets Good register (RX_MULTICAST_PACKETS_GOOD)	*Section 56.2.78*
0x0794	Receive CRC Error Packets register (RX_CRC_ERROR_PACKETS)	*Section 56.2.79*
0x0798	Receive Alignment Error Packets register (RX_ALIGNMENT_ERROR_PACKETS)	*Section 56.2.80*
0x079C	Receive Runt Error Packets register (RX_RUNT_ERROR_PACKETS)	*Section 56.2.81*
0x07A0	Receive Jabber Error Packets register (RX_JABBER_ERROR_PACKETS)	*Section 56.2.82*
0x07A4	Receive Undersize Packets Good register (RX_UNDERSIZE_PACKETS_GOOD)	*Section 56.2.83*

Table 1022. Ethernet memory map (continued)

Address offset (hex)	Register	Location
0x07A8	Receive Oversize Packets Good register (RX_OVERSIZE_PACKETS_GOOD)	Section 56.2.84
0x07AC	Receive 64Octets Packets Good Bad register (RX_64OCTETS_PACKETS_GOOD_BAD)	Section 56.2.85
0x07B0	Receive 65To127Octets Packets Good Bad register (RX_65TO127OCTETS_PACKETS_GOOD_BAD)	Section 56.2.86
0x07B4	Receive 128To255Octets Packets Good Bad register (RX_128TO255OCTETS_PACKETS_GOOD_BAD)	Section 56.2.87
0x07B8	Receive 256To511Octets Packets Good Bad register (RX_256TO511OCTETS_PACKETS_GOOD_BAD)	Section 56.2.88
0x07BC	Receive 512To1023Octets Packets Good Bad register (RX_512TO1023OCTETS_PACKETS_GOOD_BAD)	Section 56.2.89
0x07C0	Receive 1024ToMax Octets Packets Good bad register (RX_1024TOMAXOCTETS_PACKETS_GOOD_BAD)	Section 56.2.90
0x07C4	Receive Unicast Packets Good register (RX_UNICAST_PACKETS_GOOD)	Section 56.2.91
0x07C8	Receive Length Error Packets register (RX_LENGTH_ERROR_PACKETS)	Section 56.2.92
0x07CC	Receive Out Of Range Type Packets register (RX_OUT_OF_RANGE_TYPE_PACKETS)	Section 56.2.93
0x07D0	Receive Pause Packets register (RX_PAUSE_PACKETS)	Section 56.2.94
0x07D4	Receive FIFO Overflow Packets register (RX_FIFO_OVERFLOW_PACKETS)	Section 56.2.95
0x07D8	Receive VLAN Packets Good Bad register (RX_VLAN_PACKETS_GOOD_BAD)	Section 56.2.96
0x07DC	Receive Watchdog Error Packets register (RX_WATCHDOG_ERROR_PACKETS)	Section 56.2.97
0x07E0	Receive Error Packets register (RX_RECEIVE_ERROR_PACKETS)	Section 56.2.98
0x07E4	Receive Control Packets Good register (RX_CONTROL_PACKETS_GOOD)	Section 56.2.99
0x07E8–0x07EB	Reserved	
0x07EC	Transmit LPI USEC Counter register (TX_LPI_USEC_CNTR)	Section 56.2.100
0x07F0	Transmit LPI Transaction Counter register (TX_LPI_TRAN_CNTR)	Section 56.2.101
0x07F4	Receive LPI USEC Counter register (RX_LPI_USEC_CNTR)	Section 56.2.102
0x07F8	Receive LPI Transaction Counter register (RX_LPI_TRAN_CNTR)	Section 56.2.103
0x07FC–0x07FF	Reserved	
0x0800	MMC IPC Receive Interrupt Mask register (MMC_IPC_RX_INTERRUPT_MASK)	Section 56.2.104
0x0804–0x0807	Reserved	

Table 1022. Ethernet memory map (continued)

Address offset (hex)	Register	Location
0x0808	MMC IPC Receive Interrupt register (MMC_IPC_RX_INTERRUPT)	*Section 56.2.105*
0x080C–0x080F	Reserved	
0x0810	Receive IPv4 Good Packets register (RXIPV4_GOOD_PACKETS)	*Section 56.2.106*
0x0814	Receive IPv4 Header Error Packets (RXIPV4_HEADER_ERROR_PACKETS)	*Section 56.2.107*
0x0818	Receive IPv4 No Payload Packets register (RXIPV4_NO_PAYLOAD_PACKETS)	*Section 56.2.108*
0x081C	Receive IPv4 Fragmented Packets register (RXIPV4_FRAGMENTED_PACKETS)	*Section 56.2.109*
0x0820	Receive IPv4 UDP Checksum Disabled Packets register (RXIPV4_UDP_CHECKSUM_DISABLED_PACKETS)	*Section 56.2.110*
0x0824	Receive IPv6 Good Packets register (RXIPV6_GOOD_PACKETS)	*Section 56.2.111*
0x0828	Receive IPv6 Header Error Packets register (RXIPV6_HEADER_ERROR_PACKETS)	*Section 56.2.112*
0x082C	Receive IPv6 Payload Packets register (RXIPV6_NO_PAYLOAD_PACKETS)	*Section 56.2.113*
0x0830	Receive UDP Good Packets register (RXUDP_GOOD_PACKETS)	*Section 56.2.114*
0x0834	Receive UDP Error Packets register (RXUDP_ERROR_PACKETS)	*Section 56.2.115*
0x0838	Receive TCP Good Packets register (RXTCP_GOOD_PACKETS)	*Section 56.2.116*
0x083C	Receive TCP Error Packets register (RXTCP_ERROR_PACKETS)	*Section 56.2.117*
0x0840	Receive ICMP Good Packets register (RXICMP_GOOD_PACKETS)	*Section 56.2.118*
0x0844	Receive ICMP Error Packets register (RXICMP_ERROR_PACKETS)	*Section 56.2.119*
0x0848–0x084F	Reserved	
0x0850	Receive IPv4 Good Octets register (RXIPV4_GOOD_OCTETS)	*Section 56.2.120*
0x0854	Receive IPv4 Header Error Octets register (RXIPV4_HEADER_ERROR_OCTETS)	*Section 56.2.121*
0x0858	Receive IPv4 No Payload Octets register (RXIPV4_NO_PAYLOAD_OCTETS)	*Section 56.2.122*
0x085C	Receive IPv4 Fragmented Octets register (RXIPV4_FRAGMENTED_OCTETS)	*Section 56.2.123*
0x0860	Receive IPv4 UDP Checksum Disable Octets register (RXIPV4_UDP_CHECKSUM_DISABLE_OCTETS)	*Section 56.2.124*
0x0864	Receive IPv6 Good Octets register (RXIPV6_GOOD_OCTETS)	*Section 56.2.125*
0x0868	Receive Header Error Octets register (RXIPV6_HEADER_ERROR_OCTETS)	*Section 56.2.126*
0x086C	Receive IPv6 No Payload Octets register (RXIPV6_NO_PAYLOAD_OCTETS)	*Section 56.2.127*
0x0870	Receive UDP Good Octets register (RXUDP_GOOD_OCTETS)	*Section 56.2.128*

Table 1022. Ethernet memory map (continued)

Address offset (hex)	Register	Location
0x0874	Receive UDP Error Octets register (RXUDP_ERROR_OCTETS)	*Section 56.2.129*
0x0878	Receive TCP Good Octets register (RXTCP_GOOD_OCTETS)	*Section 56.2.130*
0x087C	Receive TCP Error Octets register (RXTCP_ERROR_OCTETS)	*Section 56.2.131*
0x0880	Receive ICMP Good Octets register (RXICMP_GOOD_OCTETS)	*Section 56.2.132*
0x0884	Receive ICMP Error Octets register (RXICMP_ERROR_OCTETS)	*Section 56.2.133*
0x0888–0x088F	Reserved	
0x0900	Layer 3 and Layer 4 Control 0 register (MAC_L3_L4_CONTROL0)	*Section 56.2.134*
0x0904	Layer 4 Address 0 register (MAC_LAYER4_ADDRESS0)	*Section 56.2.135*
0x0908–0x090F	Reserved	
0x0910	Layer 3 Address 0 Register 0 (MAC_LAYER3_ADDRESS0_REG0)	*Section 56.2.136*
0x0914	Layer3 Address1 Register 0 (MAC_LAYER3_ADDRESS1_REG0)	*Section 56.2.137*
0x0918	Layer3 Address2 Register 0 (MAC_LAYER3_ADDRESS2_REG0)	*Section 56.2.138*
0x091C	Layer3 Address3 Register 0 (MAC_LAYER3_ADDRESS3_REG0)	*Section 56.2.139*
0x0920–0x0AFF	Reserved	
0x0B00	Timestamp Control register (MAC_TIMESTAMP_CONTROL)	*Section 56.2.140*
0x0B04	Sub Second Increment register (MAC_SUB_SECOND_INCREMENT)	*Section 56.2.141*
0x0B08	System Time Seconds register (MAC_SYSTEM_TIME_SECONDS)	*Section 56.2.142*
0x0B0C	System Time Nanoseconds register (MAC_SYSTEM_TIME_NANOSECONDS)	*Section 56.2.143*
0x0B10	System Time Seconds Update register (MAC_SYSTEM_TIME_SECONDS_UPDATE)	*Section 56.2.144*
0x0B14	System Time Nanoseconds Update register (MAC_SYSTEM_TIME_NANOSECONDS_UPDATE)	*Section 56.2.145*
0x0B18	Timestamp Addend register (MAC_TIMESTAMP_ADDEND)	*Section 56.2.146*
0x0B1C	System Time Higher Word Seconds register (MAC_SYSTEM_TIME_HIGHER_WORD_SECONDS)	*Section 56.2.147*
0x0B20	MAC Timestamp Status register (MAC_TIMESTAMP_STATUS)	*Section 56.2.148*
0x0B24–0x0B2F	Reserved	
0x0B30	MAC Transmit Timestamp Status Nanoseconds register (MAC_TX_TIMESTAMP_STATUS_NANOSECONDS)	*Section 56.2.149*
0x0B34	MAC Tx Transmit Timestamp Status Seconds register (MAC_TX_TIMESTAMP_STATUS_SECONDS)	*Section 56.2.150*
0x0B38–0x0B3F	Reserved	
0x0B40	MAC Auxiliary Control register (MAC_AUXILIARY_CONTROL)	*Section 56.2.151*
0x0B44–0x0B47	Reserved	
0x0B48	MAC Auxiliary Timestamp Nanoseconds register (MAC_AUXILIARY_TIMESTAMP_NANOSECONDS)	*Section 56.2.152*

Table 1022. Ethernet memory map (continued)

Address offset (hex)	Register	Location
0x0B4C	MAC Auxiliary Timestamp Seconds register (MAC_AUXILIARY_TIMESTAMP_SECONDS)	*Section 56.2.153*
0x0B50	MAC Timestamp Ingress Asymmetry Correction register (MAC_TIMESTAMP_INGRESS_ASYM_CORR)	*Section 56.2.154*
0x0B54	MAC Timestamp Egress Asymmetry Correction register (MAC_TIMESTAMP_EGRESS_ASYM_CORR)	*Section 56.2.155*
0x0B58	MAC Timestamp Ingress Correction Nanosecond register (MAC_TIMESTAMP_INGRESS_CORR_NANOSECOND)	*Section 56.2.156*
0x0B5C	MAC Timestamp Egress Correction Nanosecond register (MAC_TIMESTAMP_EGRESS_CORR_NANOSECOND)	*Section 56.2.157*
0x0B60–0x0B6F	Reserved	
0x0B70	MAC_PPS_Control PPS Control register (MAC_PPS_CONTROL)	*Section 56.2.158*
0x0B74–0x0B7F	Reserved	
0x0B80	PPS0 Target Time Seconds register (MAC_PPS0_TARGET_TIME_SECONDS)	*Section 56.2.159*
0x0B84	PPS0 Target Time Nanoseconds register (MAC_PPS0_TARGET_TIME_NANOSECONDS)	*Section 56.2.160*
0x0B88	PPS0 Interval register (MAC_PPS0_INTERVAL)	*Section 56.2.161*
0x0B8C	PPS0 Width register (MAC_PPS0_WIDTH)	*Section 56.2.162*
0x0B90–0x0BFF	Reserved	
MTL registers		
0x0C00	Operation Mode register (MTL_OPERATION_MODE)	*Section 56.2.163*
0x0C04–0x0C07	Reserved	
0x0C08	Debug Access Control register (MTL_DBG_CTL)	*Section 56.2.164*
0x0C0C	Debug Status register (MTL_DBG_STS)	*Section 56.2.165*
0x0C10	FIFO Debug Data register (MTL_FIFO_DEBUG_DATA)	*Section 56.2.166*
0x0C14–0x0C1F	Reserved	
0x0C20	Interrupt Status register (MTL_INTERRUPT_STATUS)	*Section 56.2.167*
0x0C24–0xC2F	Reserved	
0x0C30	Receive Queue and DMA Channel Mapping 0 register (MTL_RXQ_DMA_MAP0)	*Section 56.2.168*
0x0C34–0x0CFF	Reserved	
MTL_Q0 registers		
0x0D00	Queue 0 Transmit Operation Mode register (MTL_TXQ0_OPERATION_MODE)	*Section 56.2.169*
0x0D04	Queue 0 Underflow Counter register (MTL_TXQ0_UNDERFLOW)	*Section 56.2.170*
0x0D08	Queue 0 Transmit Debug register (MTL_TXQ0_DEBUG)	*Section 56.2.171*

Table 1022. Ethernet memory map (continued)

Address offset (hex)	Register	Location
0x0D0C–0x0D13	Reserved	
0x0D14	Queue 0 ETS Status register (MTL_TXQ0_ETS_STATUS)	*Section 56.2.172*
0x0D18	Queue 0 Quantum or Weights register (MTL_TXQ0_QUANTUM_WEIGHT)	*Section 56.2.173*
0x0D1C–0x0D2B	Reserved	
0x0D2C	Interrupt Control Status register (MTL_Q0_IINTERRUPT_CONTROL_STATUS)	*Section 56.2.174*
0x0D30	Queue 0 Receive Operation Mode register (MTL_RXQ0_OPERATION_MODE)	*Section 56.2.175*
0x0D34	Queue 0 Missed Packet and Overflow Counter register (MTL_RXQ0_MISSED_PACKET_OVERFLOW_CNT)	*Section 56.2.176*
0x0D38	Queue 0 Receive Debug register (MTL_RXQ0_DEBUG)	*Section 56.2.177*
0x0D3C	Queue Receive Control register (MTL_RXQ0_CONTROL)	*Section 56.2.178*
MTL_Q*n* registers (for *n* = 1)		
0x0D00 + *n**0x40	Transmit Q*n* Operation Mode register (MTL_TXQ*n*_OPERATION_MODE)	*Section 56.2.179*
0x0D04 + *n**0x40	Transmit Q*n* Underflow register (MTL_TXQ*n*_UNDERFLOW)	*Section 56.2.180*
0x0D08 + *n**0x40	Transmit Q*n* Debug register (MTL_TXQ*n*_DEBUG)	*Section 56.2.181*
0x0D0C + *n**0x40 –0x0D0F + *n**0x40	Reserved	
0x0D10 + *n**0x40	Transmit Q*n* ETS Control register (MTL_TXQ*n*_ETS_CONTROL)	*Section 56.2.182*
0x0D14 + *n**0x40	Transmit Q*n* ETS Status register (MTL_TXQ*n*_ETS_STATUS)	*Section 56.2.183*
0x0D18 + *n**0x40	Transmit Q*n* Quantum Weight register (MTL_TXQ*n*_QUANTUM_WEIGHT)	*Section 56.2.184*
0x0D1C + *n**0x40	Transmit Q*n* Send Slope Credit register (MTL_TXQ*n*_SENDSPLOPECREDIT)	*Section 56.2.185*
0x0D20 + *n**0x40	Transmit Q*n* High Credit register (MTL_TXQ*n*_HICREDIT)	*Section 56.2.186*
0x0D24 + *n**0x40	Transmit Q*n* Low Credit register (MTL_TXQ*n*_LOCREDIT)	*Section 56.2.187*
0x0D2C + *n**0x40	Q*n* Interrupt Control Status register (MTL_Q*n*_INTERRUPT_CONTROL_STATUS)	*Section 56.2.188*
0x0D30 + *n**0x40	Receive Q*n* Operation Mode register (MTL_RXQ*n*_OPERATION_MODE)	*Section 56.2.189*
0x0D34 + *n**0x40	Receive Q*n* Missed Packet and Overflow Counter register (MTL_RXQ*n*_MISSED_PACKET_OVERFLOW_CNT)	*Section 56.2.190*
0x0D38 + *n**0x40	Receive Q*n* Debug register (MTL_RXQ*n*_DEBUG)	*Section 56.2.191*
0x0D3C + *n**0x40	Receive Q*n* Control register (MTL_RXQ*n*_CONTROL)	*Section 56.2.192*
0x0D80–0x0FFF	Reserved	

Table 1022. Ethernet memory map (continued)

Address offset (hex)	Register	Location
DMA registers		
0x1000	DMA_Mode register (DMA_MODE)	*Section 56.2.193*
0x1004	System Bus Mode register (DMA_SYSBUS_MODE)	*Section 56.2.194*
0x1008	Interrupt Status register (DMA_INTERRUPT_STATUS)	*Section 56.2.195*
0x100C	Debug Status 0 register (DMA_DEBUG_STATUS0)	*Section 56.2.196*
DMA_CHn registers (for *n* = 0 to 1)		
0x1100 + *n**0x80	DMA Channel *n* Control register (DMA_CH*n*_CONTROL)	*Section 56.2.197*
0x1104 + *n**0x80	DMA Channel *n* Transmit Control register (DMA_CH*n*_TX_CONTROL)	*Section 56.2.198*
0x1108 + *n**0x80	DMA Channel *n* Receive Control register (DMA_CH*n*_RX_CONTROL)	*Section 56.2.199*
0x1114 + *n**0x80	DMA Channel *n* Transmit Descriptor List Address register (DMA_CH*n*_TXDESC_LIST_ADDRESS)	*Section 56.2.200*
0x111C + *n**0x80	DMA Channel *n* Receive Descriptor List Address register(DMA_CH*n*_RXDESC_LIST_ADDRESS)	*Section 56.2.201*
0x1120 + *n**0x80	DMA Channel *n* Transmit Descriptor Tail Pointer register(DMA_CH*n*_TXDESC_TAIL_POINTER)	*Section 56.2.202*
0x1128 + *n**0x80	DMA Channel *n* Receive Descriptor Tail Pointer (DMA_CH*n*_RXDESC_TAIL_POINTER)	*Section 56.2.203*
0x112C + *n**0x80	DMA Channel *n* Transmit Descriptor Ring Length register (DMA_CH*n*_TXDESC_RING_LENGTH)	*Section 56.2.204*
0x1130 + *n**0x80	DMA Channel *n* Receive Descriptor Ring Length register (DMA_CH*n*_RXDESC_RING_LENGTH)	*Section 56.2.205*
0x1134 + *n**0x80	DMA Channel *n* Interrupt Enable register (DMA_CH*n*_INTERRUPT_ENABLE)	*Section 56.2.206*
0x1138 + *n**0x80	DMA Channel *n* Receive Interrupt Watchdog Timer register (DMA_CH*n*_Rx_INTERRUPT_WATCHDOG_TIMER)	*Section 56.2.207*
0x113C + *n**0x8	DMA Channel *n* Slot Function Control Status register (DMA_CH*n*_SLOT_FUNCTION_CONTROL_STATUS)	*Section 56.2.208*
0x1144 + *n**0x8	DMA Channel *n* Current Application Transmit Descriptor register (DMA_CH*n*_CURRENT_APP_TXDESC)	*Section 56.2.209*
0x114C + *n**0x8	DMA Channel *n* Current Application Transmit Descriptor register (DMA_CH*n*_CURRENT_APP_RXDESC)	*Section 56.2.210*
0x1154 + *n**0x8	DMA Channel *n* Current Application Transmit Buffer register (DMA_CH*n*_CURRENT_APP_TXBUFFER)	*Section 56.2.211*
0x115C + *n**0x8	DMA Channel *n* Current Application Receive Buffer register (DMA_CH*n*_CURRENT_APP_RXBUFFER)	*Section 56.2.212*

Table 1022. Ethernet memory map (continued)

Address offset (hex)	Register	Location
0x1160 + *n**0x8	DMA Channel *n* Status register (DMA_CH*n*_STATUS)	*Section 56.2.213*
0x116C + *n**0x8	DMA Channel *n* Miss Frame Counter register (DMA_CH*n*_MISS_FRAME_CNT)	*Section 56.2.214*

56.2.1 MAC Configuration Register (MAC_CONFIGURATION)

The MAC Configuration Register establishes the operating mode of the MAC.

Offset 0x0000 Access: User Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	ARPEN	SARC			IPC	IPG			GPSLCE	S2KP	CST	ACS	WD	0	JD	JE
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	PS	FES	DM	LM	ECRSFD	DO	DCRS	DR	0	BL		DC	PRELEN		TE	RE
W																
Reset	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 973. MAC Configuration Register (MAC_CONFIGURATION)

Table 1023. MAC_CONFIGURATION field descriptions

Field	Description
0 ARPEN	ARP Offload Enable When this bit is set, the MAC can recognize an incoming ARP request packet and schedules the ARP packet for transmission. It will forward the ARP packet to the application and also indicate the events in the RxStatus. When this bit is reset, the receiver does not recognize any ARP packet and indicates them as Type frame in the RxStatus.
1:3 SARC	Source Address Insertion or Replacement Control This field controls the source address insertion or replacement for all transmitted packets. Bit 1 specifies which MAC Address register (0 or 1) is used for source address insertion or replacement based on the values of Bits[2:3]: 0x If TDES3[SAIC] = 3'b000 then the application provides the correct SA field else insertion logic will work according to the setting of TDES3[SAIC]. 10 If Bit 1 is set to 0, the MAC inserts the content of the MAC Address 0 registers (MAC registers 192 and 193) in the SA field of all transmitted packets. If Bit 1 is set to 1 and the Enable MAC Address Register 1 option is selected while configuring the core, the MAC inserts the content of the MAC Address 1 registers (MAC registers 194 and 195) in the SA field of all transmitted packets. 11 If Bit 1 is set to 0, the MAC replaces the content of the MAC Address 0 registers (MAC registers 192 and 193) in the SA field of all transmitted packets. If Bit 1 is set to 1 and the MAC Address Register 1 is enabled, the MAC replaces the content of the MAC Address 1 registers (MAC registers 194 and 195) in the SA field of all transmitted packets. **Note:** Changes to this field take effect only on the start of a packet. If you write to this register field when a packet is being transmitted, only the subsequent packet can use the updated value, that is, the current packet does not use the updated value. These bits are reserved and RO when the Enable SA and VLAN Insertion on Tx feature is not selected while configuring the core.
4 IPC	Checksum Offload When set, this bit enables the IPv4 header checksum checking and IPv4 or IPv6 TCP, UDP, or ICMP payload checksum checking. When this bit is reset, the COE function in the receiver is disabled.
5:7 IPG	Inter-Packet Gap These bits control the minimum IPG between packets during transmission. 000 96 bit times 001 88 bit times 010 80 bit times ... 111 40 bit times This range of minimum IPG is valid in full-duplex mode. In the half-duplex mode, the minimum IPG can be configured only for 64-bit times (IPG = 100). Lower values are not considered. When a JAM pattern is being transmitted because of backpressure activation, the MAC does not consider the minimum IPG. The above function (IPG less than 96 bit times) is valid only when EIPGEN bit in MAC_Ext_Configuration register is reset. When EIPGEN is set, then the minimum IPG (greater than 96 bit times) is controlled as per the description given in EIPG field in MAC_Ext_Configuration register.

Table 1023. MAC_CONFIGURATION field descriptions (continued)

Field	Description
8 GPSLCE	Giant Packet Size Limit Control Enable When this bit is set, the MAC considers the value in GPSL field in MAC_Ext_Configuration register to declare a received packet as Giant packet. This field must be programed to more than 1,518 bytes. Otherwise, the MAC considers 1,518 bytes as giant packet limit. When this bit is reset, the MAC considers a received packet as Giant packet when its size is greater than 1,518 bytes (1522 bytes for tagged packet). The watchdog timeout limit, Jumbo Packet Enable and 2K Packet Enable have higher precedence over this bit, that is the MAC considers a received packet as Giant packet when its size is greater than 9,018 bytes (9,022 bytes for tagged packet) with Jumbo Packet Enabled and greater than 2,000 bytes with 2K Packet Enabled. The watchdog timeout, if enabled, terminates the received packet when watchdog limit is reached. Therefore, the programed giant packet limit should be less than the watchdog limit to get the giant packet status.
9 S2KP	IEEE 802.3 as Support for 2K Packets When this bit is set, the MAC considers all packets with up to 2,000 bytes length as normal packets. When the JE bit is not set, the MAC considers all received packets of size more than 2K bytes as Giant packets. When this bit is reset and the JE bit is not set, the MAC considers all received packets of size more than 1,518 bytes (1,522 bytes for tagged) as giant packets. For more information about how the setting of this bit and the JE bit impact the Giant packet status, refer to Giant Packet Status based on S2KP and JE Bits. **Note:** When the JE bit is set, setting this bit has no effect on the giant packet status.
10 CST	CRC stripping for Type packets When this bit is set, the last four bytes (FCS) of all packets of Ether type (type field greater than 1,536) are stripped and dropped before forwarding the packet to the application. This function is not valid when the IP Checksum Engine (Type 1) is enabled in the MAC receiver. This function is valid when Type 2 Checksum Offload Engine is enabled. **Note:** For information about how the settings of the ACS bit and this bit impact the packet length, refer to *Table 1024*.
11 ACS	Automatic Pad or CRC Stripping When this bit is set, the MAC strips the Pad or FCS field on the incoming packets only if the value of the length field is less than 1,536 bytes. All received packets with length field greater than or equal to 1,536 bytes are passed to the application without stripping the Pad or FCS field. When this bit is reset, the MAC passes all incoming packets to the application, without any modification. For more details, refer to *Table 1024* **Note:** For information about how the settings of CST bit and this bit impact the packet length, refer to *Table 1024*.
12 WD	Watchdog Disable When this bit is set, the MAC disables the watchdog timer on the receiver. The MAC can receive packets of up to 16,383 bytes. When this bit is reset, the MAC does not allow more than 2,048 bytes (10,240 if JE is set high) of the packet being received. The MAC cuts off any bytes received after 2,048 bytes.

Table 1023. MAC_CONFIGURATION field descriptions (continued)

Field	Description
14 JD	Jabber Disable When this bit is set, the MAC disables the jabber timer on the transmitter. The MAC can transfer packets of up to 16,383 bytes. When this bit is reset, if the application sends more than 2,048 bytes of data (10,240 if JE is set high) during transmission, the MAC does not send rest of the bytes in that packet.
15 JE	Jumbo Packet Enable When this bit is set, the MAC allows jumbo packets of 9,018 bytes (9,022 bytes for VLAN tagged packets) without reporting a giant packet error in the Rx packet status.
16 PS	Port Select 1 For 10 or 100 Mbps operations
17 FES	Speed This bit selects the speed in the 10/100 Mbps mode: 0 10 Mbps 1 100 Mbps
18 DM	Duplex Mode When this bit is set, the MAC operates in the full-duplex mode in which it can transmit and receive simultaneously.
19 LM	Loopback Mode When this bit is set, the MAC operates in the loopback mode at MII. The MII Rx clock input (clk_rx_i) is required for the loopback to work properly. This is because the Tx clock is not internally looped back.
20 ECRSFD	Enable Carrier Sense Before Transmission in Full-Duplex Mode When this bit is set, the MAC transmitter checks the CRS signal before packet transmission in the full-duplex mode. The MAC starts the transmission only when the CRS signal is low. When this bit is reset, the MAC transmitter ignores the status of the CRS signal.
21 DO	Disable Receive Own When this bit is set, the MAC disables the reception of packets when the MII_TX_EN is asserted in the half-duplex mode. When this bit is reset, the MAC receives all packets given by the PHY. This bit is not applicable in the full-duplex mode.
22 DCRS	Disable Carrier Sense During Transmission When this bit is set, the MAC transmitter ignores the MII CRS signal during packet transmission in the half-duplex mode. As a result, no errors are generated because of Loss of Carrier or No Carrier during transmission. When this bit is reset, the MAC transmitter generates errors because of Carrier Sense. The MAC can even abort the transmission.
23 DR	Disable Retry When this bit is set, the MAC attempts only one transmission. When a collision occurs on the MII interface, the MAC ignores the current packet transmission and reports a Packet Abort with excessive collision error in the Tx packet status. When this bit is reset, the MAC retries based on the settings of the BL field. This bit is applicable only in the half-duplex mode.

Table 1023. MAC_CONFIGURATION field descriptions (continued)

Field	Description
25:26 BL	Back-Off Limit The back-off limit determines the random integer number (r) of slot time delays (512 bit times for 10/100 Mbps) for which the MAC waits before rescheduling a transmission attempt during retries after a collision. 00 k = min (n, 10) 01 k = min (n, 8) 10 k = min (n, 4) 11 k = min (n, 1) where - = retransmission attempt The random integer r takes the value in the range 0 <= r < 2^k This bit is applicable only in the half-duplex mode.
27 DC	Deferral Check When this bit is set, the deferral check function is enabled in the MAC. The MAC issues a Packet Abort status, along with the excessive deferral error bit set in the Tx packet status, when the Tx state machine is deferred for more than 24,288 bit times in 10 or 100 Mbps mode. Deferral begins when the transmitter is ready to transmit, but it is prevented because of an active carrier sense signal (CRS) on MII. The defer time is not cumulative. For example, if the transmitter defers for 10,000 bit times because the CRS signal is active and the CRS signal becomes inactive, the transmitter transmits and collision happens. Because of collision, the transmitter needs to back off and then defer again after back off completion. In such a scenario, the deferral timer is reset to 0, and it is restarted. When this bit is reset, the deferral check function is disabled and the MAC defers until the CRS signal goes inactive. This bit is applicable only in the half-duplex mode.
28:29 PRELEN	Preamble Length for Transmit packets These bits control the number of preamble bytes that are added to the beginning of every Tx packet. The preamble reduction occurs only when the MAC is operating in the full-duplex mode. 00 7 bytes of preamble 01 5 bytes of preamble 10 3 bytes of preamble 11 Reserved
30 TE	Transmitter Enable When this bit is set, the Tx state machine of the MAC is enabled for transmission on the MII interface. When this bit is reset, the MAC Tx state machine is disabled after it completes the transmission of the current packet. The Tx state machine does not transmit any more packets.
31 RE	Receiver Enable When this bit is set, the Rx state machine of the MAC is enabled for receiving packets from the MII interface. When this bit is reset, the MAC Rx state machine is disabled after it completes the reception of the current packet. The Rx state machine does not receive any more packets from the MII interface.

Table 1024. Packet length based on the CST and ACS bits

Receive checksum offload engine	Receive packet length	CST	ACS	FCS stripping done
IPCKSUM_EN = 0 and IPC_FULL_OFFLOAD = 0 or IPCKSUM_EN = 1 and IPC_FULL_OFFLOAD = 1	< 1536	x	0	No
		x	1	Yes (for Ethernet packets)
	>= 1536	0	x	No
		1	x	Yes (for Type packets)
IPCKSUM_EN = 1 and IPC_FULL_OFFLOAD = 0	< 1536	x	0	No
		x	1	Yes (for Ethernet packets)
	>= 1536	x	x	No

56.2.2 MAC Extended Configuration Register (MAC_EXT_CONFIGURATION)

The MAC Extended Configuration Register establishes the operating mode of the MAC.

Offset 0x0004 Access: User Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	EIPG					EIPGEN	0	0	0	0	0	USP	SPEN	DCRCC
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	GPSL													
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 974. MAC_Ext_Configuration Register (MAC_EXT_CONFIGURATION)

Table 1025. MAC_EXT_CONFIGURATION field descriptions

Field	Description
2:6 EIPG	Extended Inter-Packet Gap The value in this field is applicable when the EIPGEN bit is set. This field (as Most Significant bits), along with IPG field in MAC_Configuration register, gives the minimum IPG greater than 96 bit times in steps of 8 bit times: {EIPG, IPG} 8'h00 - 104 bit times 8'h01 - 112 bit times 8'h02 - 120 bit times ---------- 8'hFF - 2144 bit times
7 EIPGEN	Extended Inter-Packet Gap Enable When this bit is set, the MAC interprets EIPG field and IPG field in MAC_Configuration register together as minimum IPG greater than 96 bit times in steps of 8 bit times. When this bit is reset, the MAC ignores EIPG field and interprets IPG field in MAC_Configuration register as minimum IPG less than or equal to 96 bit times in steps of 8 bit times. **Note:** The extended Inter-Packet Gap feature must be enabled when operating in Full-Duplex mode only. There may be undesirable effects on back-pressure function and frame transmission if it is enabled in Half-Duplex mode.
13 USP	Unicast Slow Protocol Packet Detect When this bit is set, the MAC detects the Slow Protocol packets with unicast address of the station specified in the MAC_Address0_High and MAC_Address0_Low registers. The MAC also detects the Slow Protocol packets with the Slow Protocols multicast address (01-80-C2-00-00-02). When this bit is reset, the MAC detects only Slow Protocol packets with the Slow Protocol multicast address specified in the IEEE 802.3-2008, Section 5.
14 SPEN	Slow Protocol Detection Enable When this bit is set, MAC processes the Slow Protocol packets (Ether Type 0x8809) and provides the Rx status. The MAC discards the Slow Protocol packets with invalid sub-types. When this bit is reset, the MAC forwards all error-free Slow Protocol packets to the application. The MAC considers such packets as normal Type packets.
15 DCRCC	Disable CRC Checking for Received Packets When this bit is set, the MAC receiver does not check the CRC field in the received packets. When this bit is reset, the MAC receiver always checks the CRC field in the received packets.
18:31 GPSL	Giant Packet Size Limit If the received packet size is greater than the value programed in this field in units of bytes, the MAC declares the received packet as Giant packet. The value programed in this field must be greater than or equal to 1,518 bytes. Any other programed value is considered as 1,518 bytes. For VLAN tagged packets, the MAC adds 4 bytes to the programed value. When the Enable Double VLAN Processing option is selected, the MAC adds 8 bytes to the programed value for double VLAN tagged packets. The value in this field is applicable when the GPSLCE bit is set in MAC_Configuration register.

56.2.3 MAC Packet Filter Register (MAC_PACKET_FILTER)

The MAC Packet Filter register contains the filter controls for receiving packets. Some of the controls from this register go to the address check block of the MAC which performs the first level of address filtering. The second level of filtering is performed on the incoming packet based on other controls such as Pass Bad Packets and Pass Control Packets.

Offset 0x0008 Access: User Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	RA	0	0	0	0	0	0	0	0	0	DNTU	IPFE	0	0	0	VTFE
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	HPF	SAF	SAIF	PCF		DBF	PM	DAIF	HMC	HUC	PR
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 975. MAC Packet Filter Register (MAC_PACKET_FILTER)

Table 1026. MAC_PACKET_FILTER field descriptions

Field	Description
0 RA	Receive All When this bit is set, the MAC Receiver module passes all received packets to the application, irrespective of whether they pass the address filter or not. The result of the SA or DA filtering is updated (pass or fail) in the corresponding bit in the Rx Status Word. When this bit is reset, the Receiver module passes only those packets to the application that pass the SA or DA address filter.
10 DNTU	Drop Non-TCP/UDP over IP Packets When this bit is set, the MAC drops the non-TCP or UDP over IP packets. The MAC forward only those packets that are processed by the Layer 4 filter. When this bit is reset, the MAC forwards all non-TCP or UDP over IP packets.
11 IPFE	Layer 3 and Layer 4 Filter Enable When this bit is set, the MAC drops packets that do not match the enabled Layer 3 and Layer 4 filters. If Layer 3 or Layer 4 filters are not enabled for matching, this bit does not have any effect. When this bit is reset, the MAC forwards all packets irrespective of the match status of the Layer 3 and Layer 4 fields.
15 VTFE	VLAN Tag Filter Enable When this bit is set, the MAC drops the VLAN tagged packets that do not match the VLAN Tag. When this bit is reset, the MAC forwards all packets irrespective of the match status of the VLAN Tag.
21 HPF	Hash or Perfect Filter When this bit is set, the address filter passes a packet if it matches either the perfect filtering or hash filtering as set by the HMC or HUC bit. When this bit is reset and the HUC or HMC bit is set, the packet is passed only if it matches the Hash filter.

Table 1026. MAC_PACKET_FILTER field descriptions (continued)

Field	Description
22 SAF	Source Address Filter Enable When this bit is set, the MAC compares the SA field of the received packets with the values programed in the enabled SA registers. If the comparison fails, the MAC drops the packet. When this bit is reset, the MAC forwards the received packet to the application with updated SAF bit of the Rx Status depending on the SA address comparison. **Note:** According to the IEEE specification, Bit 47 of the SA is reserved. However, in the Ethernet module, the MAC compares all 48 bits. The software driver should take this into consideration while programming the MAC address registers for SA.
23 SAIF	SA Inverse Filtering When this bit is set, the Address Check block operates in the inverse filtering mode for SA address comparison. If the SA of a packet matches the values programed in the SA registers, it is marked as failing the SA Address filter. When this bit is reset, if the SA of a packet does not match the values programed in the SA registers, it is marked as failing the SA Address filter.
24:25 PCF	Pass Control Packets These bits control the forwarding of all control packets (including unicast and multicast Pause packets). 00 The MAC filters all control packets from reaching the application. 01 The MAC forwards all control packets except Pause packets to the application even if they fail the Address filter. 10 The MAC forwards all control packets to the application even if they fail the Address filter. 11 The MAC forwards the control packets that pass the Address filter.
26 DBF	Disable Broadcast Packets When this bit is set, the AFM module blocks all incoming broadcast packets. In addition, it overrides all other filter settings. When this bit is reset, the AFM module passes all received broadcast packets.
27 PM	Pass All Multicast When this bit is set, it indicates that all received packets with a multicast destination address (first bit in the destination address field is '1') are passed. When this bit is reset, filtering of multicast packet depends on HMC bit.
28 DAIF	DA Inverse Filtering When this bit is set, the Address Check block operates in inverse filtering mode for the DA address comparison for both unicast and multicast packets. When this bit is reset, normal filtering of packets is performed.
29 HMC	Hash Multicast When this bit is set, the MAC performs the destination address filtering of received multicast packets according to the hash table. When this bit is reset, the MAC performs the perfect destination address filtering for multicast packets, that is, it compares the DA field with the values programed in DA registers.

Table 1026. MAC_PACKET_FILTER field descriptions (continued)

Field	Description
30 HUC	Hash Unicast When this bit is set, the MAC performs the destination address filtering of unicast packets according to the hash table. When this bit is reset, the MAC performs a perfect destination address filtering for unicast packets, that is, it compares the DA field with the values programed in DA registers.
31 PR	Promiscuous Mode When this bit is set, the Address Filtering module passes all incoming packets irrespective of the destination or source address. The SA or DA Filter Fails status bits of the Rx Status Word are always cleared when PR is set.

56.2.4 Watchdog Timeout Register (MAC_WATCHDOG_TIMEOUT)

The Watchdog Timeout register controls the watchdog timeout for received packets.

Offset 0x000C — Access: User Read/Write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	PWE	0	0	0	0	WTO			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 976. MAC Watchdog Timeout Register (MAC_WATCHDOG_TIMEOUT)

Table 1027. MAC_WATCHDOG_TIMEOUT field descriptions

Field	Description
23 PWE	Programmable Watchdog Enable When this bit is set and the WD bit of the MAC_Configuration register is reset, the WTO field is used as watchdog timeout for a received packet. When this bit is cleared, the watchdog timeout for a received packet is controlled by setting of WD and JE bits in MAC_Configuration register.
28:31 WTO	Watchdog Timeout When the PWE bit is set and the WD bit of the MAC_Configuration register is reset, this field is used as watchdog timeout for a received packet. If the length of a received packet exceeds the value of this field, such packet is terminated and declared as an error packet. Encoding is as follows: 0x0 2 KB 0x1 3 KB 0x2 4 KB 0x3 5 KB ... 0xC 14 KB 0xD 15 KB 0xE 16383 Bytes 0xF Reserved **Note:** When the PWE bit is set, the value in this field should be more than 1,522 (0x05F2). Otherwise, the IEEE 802.3-specified valid tagged packets are declared as error packets and then dropped.

56.2.5 Hash Table Register 0 (MAC_HASH_TABLE_REG0)

The Hash Table Register 0 contains the first 32 bits of the hash table.

The Hash table is used for group address filtering. For hash filtering, the content of the destination address in the incoming packet is passed through the CRC logic and the upper six (seven in 128-bit Hash or eight in 256-bit Hash) bits of the CRC register are used to index the content of the Hash table. The most significant bit determines the register to be used (Hash Table Register X), and the least significant five bits determine the bit within the register. For example, a hash value of 6'b100000 (in 64-bit Hash) selects Bit 0 of the Hash Table Register 1.

The hash value of the destination address is calculated in the following way:

- Calculate the 32-bit CRC for the DA (Refer to IEEE 802.3, Section 3.2.8 for the steps to calculate CRC32).
- Perform bitwise reversal for the value obtained in Step 1.
- Take the upper 6 (or 7 or 8) bits from the value obtained in Step 2.

If the corresponding bit value of the register is 1'b1, the packet is accepted. Otherwise, it is rejected. If the PM bit is set in MAC_Packet_Filter, all multicast packets are accepted regardless of the multicast hash values.

If the Hash Table register is configured to be double-synchronized to the MII clock domain, the synchronization is triggered only when Bits[31:24] (in little-endian mode) or Bits[7:0] (in big-endian mode) of the Hash Table Register X registers are written.

If double-synchronization is enabled, consecutive writes to this register should be performed after at least four clock cycles in the destination clock domain.

Offset 0x0010 Access: User Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	HT31T0															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	HT31T0															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 977. MAC Hash Table Reg0 Register (MAC_HASH_TABLE_REG0)

Table 1028. MAC_HASH_TABLE_REG0 field descriptions

Field	Description
0:31 HT31T0	This field contains the first 32 Bits [0:31] of the Hash table.

56.2.6 Hash Table Register 1 (MAC_HASH_TABLE_REG1)

The Hash Table Register 1 contains the second 32 bits of the hash table.

The Hash table is used for group address filtering. For hash filtering, the content of the destination address in the incoming packet is passed through the CRC logic and the upper six (seven in 128-bit Hash or eight in 256-bit Hash) bits of the CRC register are used to index the content of the Hash table. The most significant bits determine the register to be used (Hash Table Register X), and the least significant five bits determine the bit within the register. For example, a hash value of 6'b100000 (in 64-bit Hash) selects Bit 0 of the Hash Table Register 1.

The hash value of the destination address is calculated in the following way:

- Calculate the 32-bit CRC for the DA (refer to IEEE 802.3, Section 3.2.8 for the steps to calculate CRC32).
- Perform bitwise reversal for the value obtained in Step 1.
- Take the upper 6 (or 7 or 8) bits from the value obtained in Step 2.

If the corresponding bit value of the register is 1'b1, the packet is accepted. Otherwise, it is rejected. If the PM bit is set in MAC_Packet_Filter, all multicast packets are accepted regardless of the multicast hash values.

If the Hash Table register is configured to be double-synchronized to the MII clock domain, the synchronization is triggered only when Bits[31:24] (in little-endian mode) or Bits[7:0] (in big-endian mode) of the Hash Table Register X registers are written.

If double-synchronization is enabled, consecutive writes to this register should be performed after at least four clock cycles in the destination clock domain.

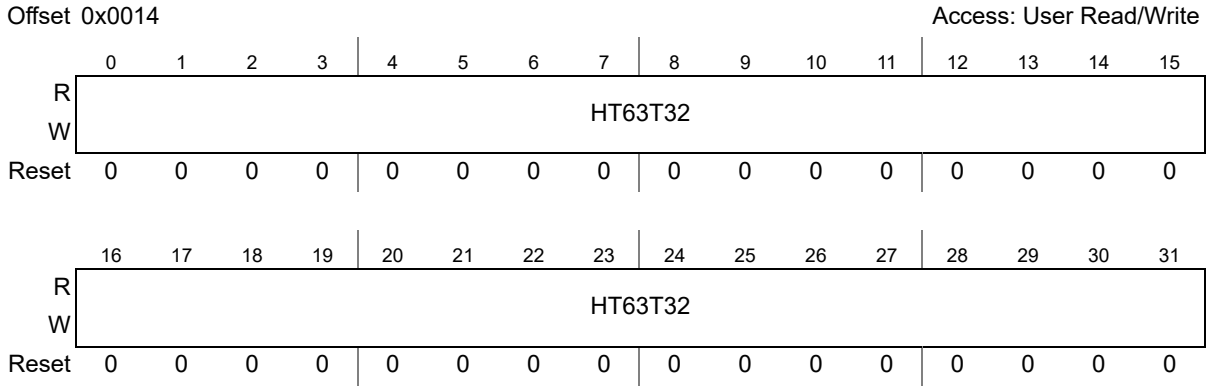


Figure 978. Hash Table Reg1 Register (MAC_HASH_TABLE_REG1)

Table 1029. MAC_HASH_TABLE_REG1 field descriptions

Field	Description
0:31 HT63T32	This field contains the second 32 Bits [31:0] of the Hash table.

56.2.7 VLAN Tag Register (MAC_VLAN_TAG)

The VLAN Tag register identifies the IEEE 802.1Q VLAN type packets.

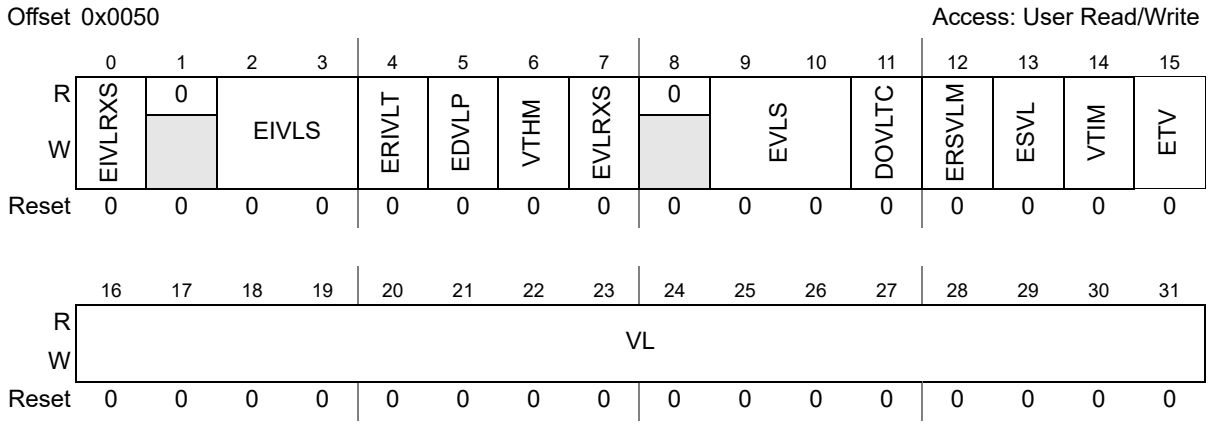


Figure 979. VLAN Tag Register (MAC_VLAN_TAG)

Table 1030. MAC_VLAN_TAG field descriptions

Field	Description
0 EIVLRXS	Enable Inner VLAN Tag in Rx Status When this bit is set, the MAC provides the inner VLAN Tag in the Rx status. When this bit is reset, the MAC does not provide the inner VLAN Tag in Rx status.
2:3 EIVLS	Enable Inner VLAN Tag Stripping on Receive This field indicates the stripping operation on inner VLAN Tag in received packet: 00 Do not strip 01 Strip if VLAN filter passes 10 Strip if VLAN filter fails 11 Always strip
4 ERIVLT	Enable Inner VLAN Tag When this bit and the EDVLP field are set, the MAC receiver enables operation on the inner VLAN Tag (if present). When this bit is reset, the MAC receiver enables operation on the outer VLAN Tag (if present). The ERSVLM bit determines which VLAN type is enabled for filtering or matching.
5 EDVLP	Enable Double VLAN Processing When this bit is set, the MAC enables processing of up to two VLAN Tags on Tx and Rx (if present). When this bit is reset, the MAC enables processing of up to one VLAN Tag on Tx and Rx (if present).
6 VTHM	VLAN Tag Hash Table Match Enable When this bit is set, the most significant four bits of CRC of VLAN Tag are used to index the content of the MAC_VLAN_Hash_Table register. A value of 1 in the VLAN Hash Table register, corresponding to the index, indicates that the packet matched the VLAN hash table. When the ETV bit is set, the CRC of the 12-bit VLAN Identifier (VID) is used for comparison. When the ETV bit is reset, the CRC of the 16-bit VLAN tag is used for comparison. When this bit is reset, the VLAN Hash Match operation is not performed.
7 EVLRXS	Enable VLAN Tag in Rx status When this bit is set, MAC provides the outer VLAN Tag in the Rx status. When this bit is reset, the MAC does not provide the outer VLAN Tag in Rx status.
9:10 EVLS	Enable VLAN Tag Stripping on Receive This field indicates the stripping operation on the outer VLAN Tag in received packet: 00 Do not strip 01 Strip if VLAN filter passes 10 Strip if VLAN filter fails 11 Always strip
11 DOVLTC	Disable VLAN Type Check When this bit is set, the MAC does not check whether the VLAN Tag specified by the ERIVLT bit is of type S-VLAN or C-VLAN. When this bit is reset, the MAC filters or matches the VLAN Tag specified by the ERIVLT bit only when VLAN Tag type is similar to the one specified by the ERSVLM bit.

Table 1030. MAC_VLAN_TAG field descriptions (continued)

Field	Description
12 ERSVLM	Enable Receive S-VLAN Match When this bit is set, the MAC receiver enables filtering or matching for S-VLAN (Type = 0x88A8) packets. When this bit is reset, the MAC receiver enables filtering or matching for C-VLAN (Type = 0x8100) packets. The ERIVLT bit determines the VLAN tag position considered for filtering or matching.
13 ESVL	Enable S-VLAN When this bit is set, the MAC transmitter and receiver consider the S-VLAN packets (Type = 0x88A8) as valid VLAN tagged packets.
14 VTIM	VLAN Tag Inverse Match Enable When this bit is set, this bit enables the VLAN Tag inverse matching. The packets without matching VLAN Tag are marked as matched. When reset, this bit enables the VLAN Tag perfect matching. The packets with matched VLAN Tag are marked as matched.
15 ETV	Enable 12-Bit VLAN Tag Comparison When this bit is set, a 12-bit VLAN identifier is used for comparing and filtering instead of the complete 16-bit VLAN tag. Bits[20:31] of VLAN tag are compared with the corresponding field in the received VLAN-tagged packet. Similarly, when enabled, only 12 bits of the VLAN tag in the received packet are used for hash-based VLAN filtering. When this bit is reset, all 16 bits of the 15th and 16th bytes of the received VLAN packet are used for comparison and VLAN hash filtering.
16:31 VL	VLAN Tag Identifier for Receive Packets This field contains the 802.1Q VLAN tag to identify the VLAN packets. This VLAN tag identifier is compared to the 15th and 16th bytes of the packets being received for VLAN packets. The following list describes the bits of this field: – Bits[16:18]: User Priority – Bit 19: Canonical Format Indicator (CFI) or Drop Eligible Indicator (DEI) – Bits[20:31]: VLAN Identifier (VID) field of VLAN tag When the ETV bit is set, only the VID is used for comparison. If this field ([20:31] if ETV is set) is all zeros, the MAC does not check the 15th and 16th bytes for VLAN tag comparison and declares all packets with Type field value of 0x8100 or 0x88a8 as VLAN packets.

56.2.8 VLAN Hash Table Register (MAC_VLAN_HASH_TABLE)

When the ERSVLM bit of MAC_Hash_Table_Reg1 register is set, the 16-bit VLAN Hash Table register is used for group address filtering based on the VLAN tag. For hash filtering, the content of the 16-bit VLAN tag or 12-bit VLAN ID (based on the ETV bit of MAC_VLAN_Tag Register) in the incoming packet is passed through the CRC logic. The upper four bits of the calculated CRC are used to index the contents of the VLAN Hash table. For example, a hash value of 4b'1000 selects Bit 8 of the VLAN Hash table.

The hash value of the destination address is calculated in the following way:

- Calculate the 32-bit CRC for the VLAN tag or ID (For steps to calculate CRC32, refer to Section 3.2.8 of IEEE 802.3 - 2008).
- Perform bitwise reversal for the value obtained in step 1.
- Take the upper four bits from the value obtained in step 2.

If the VLAN hash Table register is configured to be double-synchronized to the MII clock domain, the synchronization is triggered only when Bits[15:8] (in little-endian mode) or Bits[7:0] (in big-endian mode) of this register are written.

- If double-synchronization is enabled, consecutive writes to this register should be performed after at least four clock cycles in the destination clock domain.

Offset 0x0058 Access: User Read/Write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	VLHT															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 980. VLAN_Hash_Table Register (MAC_VLAN_HASH_TABLE)

Table 1031. MAC_VLAN_HASH_TABLE field descriptions

Field	Description
16:31 VLHT	VLAN Hash Table This field contains the 16-bit VLAN Hash Table.

56.2.9 VLAN Tag Inclusion Register (MAC_VLAN_INCL)

The VLAN Tag Inclusion or Replacement register contains the VLAN tag for insertion or replacement in the Transmit packets. It also contains the VLAN tag insertion controls.

Offset 0x0060 Access: User Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	VLTI	CSVL	VLP	VLC	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	VLT															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 981. VLAN Tag Inclusion Register (MAC_VLAN_INCL)

Table 1032. MAC_VLAN_INCL field descriptions

Field	Description
11 VLTI	VLAN Tag Input When this bit is set, it indicates that the VLAN tag to be inserted or replaced in Tx packet should be taken the Tx descriptor.
12 CSVL	C-VLAN or S-VLAN When this bit is set, S-VLAN type (0x88A8) is inserted or replaced in the 13th and 14th bytes of transmitted packets. When this bit is reset, C-VLAN type (0x8100) is inserted or replaced in the 13th and 14th bytes of transmitted packets.
13 VLP	VLAN Priority Control When this bit is set, the control bits[17:16] are used for VLAN deletion, insertion, or replacement. When this bit is reset, bits[17:16] are ignored.
14:15 VLC	VLAN Tag Control in Transmit Packets 00 No VLAN tag deletion, insertion, or replacement 01 VLAN tag deletion, The MAC removes the VLAN type (bytes 13 and 14) and VLAN tag (bytes 15 and 16) of all transmitted packets with VLAN tags. 10 VLAN tag insertion. The MAC inserts VLT in bytes 15 and 16 of the packet after inserting the Type value (0x8100 or 0x88a8) in bytes 13 and 14. This operation is performed on all transmitted packets, irrespective of whether they already have a VLAN tag. 11 VLAN tag replacement. The MAC replaces VLT in bytes 15 and 16 of all VLAN-type transmitted packets (Bytes 13 and 14 are 0x8100 or 0x88a8). **Note:** Changes to this field take effect only on the start of a packet. If you write this register field when a packet is being transmitted, only the subsequent packet can use the updated value, that is, the current packet does not use the updated value.
16:31 VLT	VLAN Tag for Transmit Packets This field contains the value of the VLAN tag to be inserted or replaced. The value must only be changed when the transmit lines are inactive or during the initialization phase. Bits[16:18] are the User Priority field, Bit 19 is the CFI/DEI field, and Bits[20:31] are the VID field in the VLAN tag. The following list describes the bits of this field: – Bits[16:18] User Priority – Bit 19 Canonical Format Indicator (CFI) or Drop Eligible Indicator (DEI) – Bits[20:31] VLAN Identifier (VID) field of VLAN tag

56.2.10 Inner VLAN Tag Inclusion Register (MAC_INNER_VLAN_INCL)

The Inner VLAN Tag Inclusion or Replacement register contains the inner VLAN tag to be inserted or replaced in the Transmit packet. It also contains the inner VLAN tag insertion controls.

Offset 0x0064 Access: User Read/Write

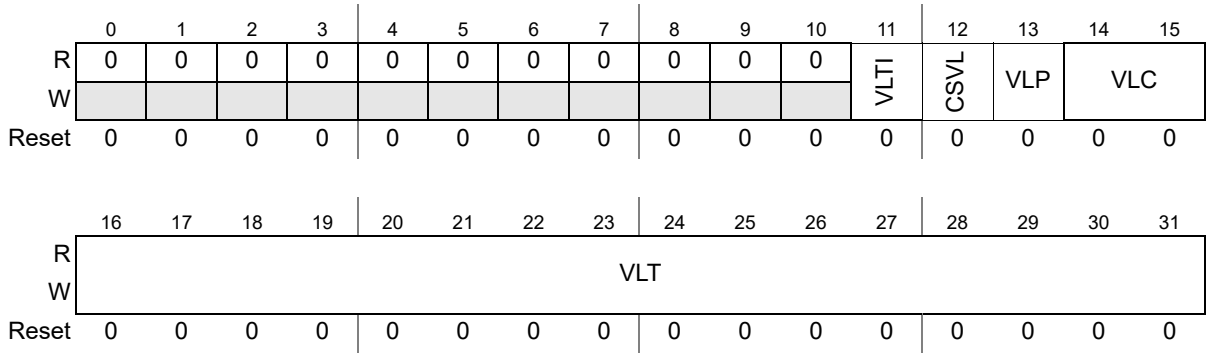


Figure 982. Inner VLAN Tag Inclusion Register (MAC_INNER_VLAN_INCL)

Table 1033. MAC_INNER_VLAN_INCL field descriptions

Name	Description
11 VLTI	VLAN Tag Input When this bit is set, it indicates that the VLAN tag to be inserted or replaced in Tx packet should be taken from the Tx descriptor.
12 CSVL	C-VLAN or S-VLAN When this bit is set, S-VLAN type (0x88A8) is inserted or replaced in the 13th and 14th bytes of transmitted packets. When this bit is reset, C-VLAN type (0x8100) is inserted or replaced in the 13th and 14th bytes of transmitted packets.
13 VLP	VLAN Priority Control When this bit is set, the VLC field is used for VLAN deletion, insertion, or replacement. When this bit is reset, the VLC field is ignored.
14:15 VLC	VLAN Tag Control in Transmit Packets 00 No VLAN tag deletion, insertion, or replacement 01 VLAN tag deletion. The MAC removes the VLAN type (bytes 17 and 18) and VLAN tag (bytes 19 and 20) of all transmitted packets with VLAN tags. 10 VLAN tag insertion. The MAC inserts VLT in bytes 19 and 20 of the packet after inserting the Type value (0x8100 or 0x88a8) in bytes 17 and 18. This operation is performed on all transmitted packets, irrespective of whether they already have a VLAN tag. 11 VLAN tag replacement. The MAC replaces VLT in bytes 19 and 20 of all VLAN-type transmitted packets (Bytes 17 and 18 are 0x8100 or 0x88a8). **Note:** Changes to this field take effect only on the start of a packet. If you write this register field when a packet is being transmitted, only the subsequent packet can use the updated value, that is, the current packet does not use the updated value.
16:31 VLT	VLAN Tag for Transmit Packets This field contains the value of the VLAN tag to be inserted or replaced. The value must only be changed when the transmit lines are inactive or during the initialization phase. Bits[16:18] are the User Priority field, Bit 19 is the CFI/DEI field, and Bits[20:31] are the VID field in the VLAN tag. The following list describes the bits of this field: – Bits[16:18] User Priority – Bit 19 Canonical Format Indicator (CFI) or Drop Eligible Indicator (DEI) – Bits[20:31] VLAN Identifier (VID) field of VLAN tag

56.2.11 MAC Q0 Flow Control Register (MAC_Q0_TX_FLOW_CTRL)

The Flow Control register controls the generation and reception of the Control (Pause Command) packets by the Flow control module of the MAC. A Write to a register with the Busy bit set to 1 triggers the Flow Control block to generate a Pause packet. The fields of the control packet are selected as specified in the 802.3x specification, and the Pause Time value from this register is used in the Pause Time field of the control packet. The Busy bit remains set until the control packet is transferred onto the cable. The application must make sure that the Busy bit is cleared before writing to the register.

Offset 0x0070 Access: User Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	PT															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	DZPQ	PLT			0	0	TFE	FCB_BPA
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 983. MAC Q0 Flow Control Register (MAC_Q0_TX_FLOW_CTRL)

Table 1034. MAC_Q0_TX_FLOW_CTRL field descriptions

Name	Description
0:15 PT	Pause Time This field holds the value to be used in the Pause Time field in the Tx control packet. If the Pause Time bits are configured to be double-synchronized to the MII clock domain, consecutive writes to this register should be performed only after at least four clock cycles in the destination clock domain.
24 DZPQ	Disable Zero-Quanta Pause When this bit is set, it disables the automatic generation of the zero-quanta Pause packets on de-assertion of the flow-control signal from the FIFO layer (MTL or external sideband flow control signal sbd_flowctrl_i or mti_flowctrl_i). When this bit is reset, normal operation with automatic zero-quanta Pause packet generation is enabled.

Table 1034. MAC_Q0_TX_FLOW_CTRL field descriptions (continued)

Name	Description
25:27 PLT	Pause Low Threshold This field configures the threshold of the Pause timer at which the input flow control signal mti_flowctrl_i (or sbd_flowctrl_i) is checked for automatic retransmission of the Pause packet. The threshold values should be always less than the Pause Time configured in Bits[0:15]. For example, if PT = 100H (256 slot times), and PLT = 001, a second Pause packet is automatically transmitted if the mti_flowctrl_i signal is asserted at 228 (256-28) slot times after the first Pause packet is transmitted. The following list provides the threshold values for different values: 000 Pause Time minus 4 Slot Times (PT -4 slot times) 001 Pause Time minus 28 Slot Times (PT -28 slot times) 010 Pause Time minus 36 Slot Times (PT -36 slot times) 011 Pause Time minus 144 Slot Times (PT -144 slot times) 100 Pause Time minus 256 Slot Times (PT -256 slot times) 101 Pause Time minus 512 Slot Times (PT -512 slot times) 110 Reserved 111 Reserved The slot time is defined as the time taken to transmit 512 bits (64 bytes) on the MII interface. This (approximate) computation is based on the packet size (64, 1518, 2000, 9018, 16384, or 32768) + 2 Pause Packet Size + IPG in Slot Times.
30 TFE	Transmit Flow Control Enable Full-Duplex Mode: In the full-duplex mode, when this bit is set, the MAC enables the flow control operation to Tx Pause packets. When this bit is reset, the flow control operation in the MAC is disabled, and the MAC does not transmit any Pause packets. Half-Duplex Mode: In the half-duplex mode, when this bit is set, the MAC enables the backpressure operation. When this bit is reset, the backpressure feature is disabled.
31 FCB_BPA	Flow Control Busy or Backpressure Activate This bit initiates a Pause packet in the full-duplex mode and activates the backpressure function in the half-duplex mode if the TFE bit is set. Full-Duplex Mode: In the full-duplex mode, this bit should be read as 1'b0 before writing to this register. To initiate a Pause packet, the application must set this bit to 1'b1. During Control packet transfer, this bit continues to be set to indicate that a packet transmission is in progress. When Pause packet transmission is complete, the MAC resets this bit to 1'b0. You should not write to this register until this bit is cleared. Half-Duplex Mode: When this bit is set (and TFE bit is set) in the half-duplex mode, the MAC asserts the backpressure. During backpressure, when the MAC receives a new packet, the transmitter starts sending a JAM pattern resulting in a collision. Access restriction applies. Setting 1 sets. Self-cleared. Setting 0 has no effect.

56.2.12 MAC Q1 Flow Control Register (MAC_Q1_TX_FLOW_CTRL)

This register controls the generation of PFC Control packets of priorities mapped as per the PSRQ*i* field in the MAC_RxQ_Ctrl2 registers. This register is present only when the Enable Data Center Bridging feature is selected during configuration.

Offset 0x0074 Access: User Read

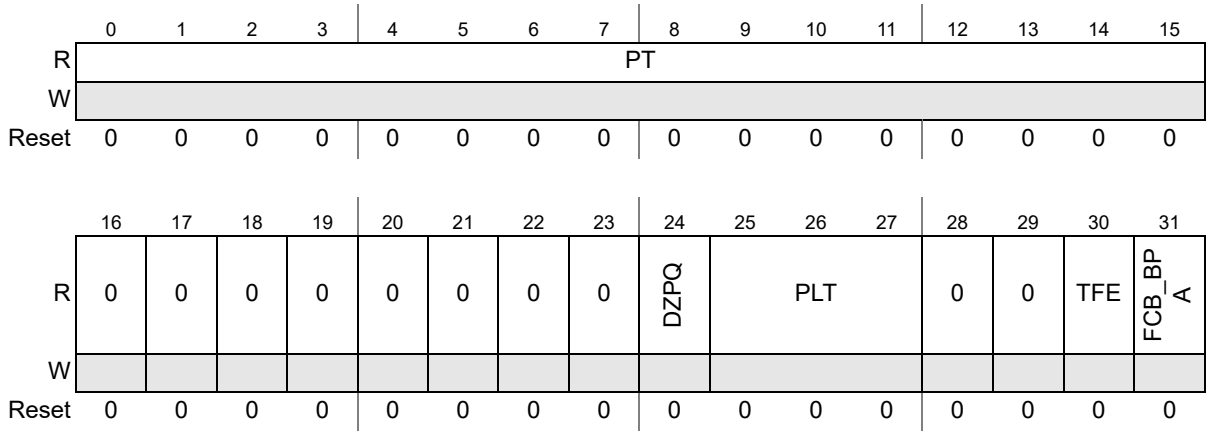


Figure 984. MAC Q1 Flow Control Register (MAC_Q1_TX_FLOW_CTRL)

Table 1035. MAC_Q1_TX_FLOW_CTRL field descriptions

Name	Description
0:15 PT	Pause Time This field holds the value to be used in the Pause Time field in the Tx control packet. If the Pause Time bits are configured to be double-synchronized to the MII clock domain, consecutive writes to this register should be performed only after at least four clock cycles in the destination clock domain.
24 DZPQ	Disable Zero-Quanta Pause When this bit is set, it disables the automatic generation of the zero-quanta Pause packets on de-assertion of the flow-control signal from the FIFO layer (MTL or external sideband flow control signal sbd_flowctrl_i or mti_flowctrl_i). When this bit is reset, normal operation with automatic zero-quanta Pause packet generation is enabled.
25:27 PLT	Pause Low Threshold This field configures the threshold of the Pause timer at which the input flow control signal mti_flowctrl_i (or sbd_flowctrl_i) is checked for automatic retransmission of the Pause packet. The threshold values should be always less than the Pause Time configured in Bits[0:15]. For example, if PT = 100H (256 slot times), and PLT = 001, a second Pause packet is automatically transmitted if the mti_flowctrl_i signal is asserted at 228 (256-28) slot times after the first Pause packet is transmitted. The following list provides the threshold values for different values: 000 Pause Time minus 4 Slot Times (PT -4 slot times) 001 Pause Time minus 28 Slot Times (PT -28 slot times) 010 Pause Time minus 36 Slot Times (PT -36 slot times) 011 Pause Time minus 144 Slot Times (PT -144 slot times) 100 Pause Time minus 256 Slot Times (PT -256 slot times) 101 Pause Time minus 512 Slot Times (PT -512 slot times) 110 Reserved 111 Reserved The slot time is defined as the time taken to transmit 512 bits (64 bytes) on the MII interface. This (approximate) computation is based on the packet size (64, 1518, 2000, 9018, 16384, or 32768) + 2 Pause Packet Size + IPG in Slot Times.

Table 1035. MAC_Q1_TX_FLOW_CTRL field descriptions (continued)

Name	Description
30 TFE	When this bit is set in full-duplex mode, the MAC enables the flow control operation to Tx Pause packets. When this bit is reset, the flow control operation in the MAC is disabled, and the MAC does not transmit any Pause packets.
31 FCB_BPA	This bit initiates a PFC packet if the TFE bit is set. To initiate a PFC packet, the application must set this bit to 1'b1. During Control packet transfer, this bit continues to be set to indicate that a packet transmission is in progress. When PFC packet transmission is complete, the MAC resets this bit to 1'b0. You should not write to this register until this bit is cleared. Access restriction applies. Setting 1 sets. Self-cleared. Setting 0 has no effect.

56.2.13 MAC Receive Flow Control Register (MAC_RX_FLOW_CTRL)

The Receive Flow Control register controls the pausing of MAC Transmit based on the received Pause packet.

Offset 0x0090 Access: User Read/Write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	UP	RFE
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 985. MAC Receive Flow Control Register (MAC_RX_FLOW_CTRL)

Table 1036. MAC_RX_FLOW_CTRL field descriptions

Field	Description
30 UP	Unicast Pause Packet Detect A pause packet is processed when it has the unique multicast address specified in the IEEE 802.3. When this bit is set, the MAC can also detect Pause packets with unicast address of the station. This unicast address should be as specified in MAC_Address0_High and MAC_Address0_Low. When this bit is reset, the MAC only detects Pause packets with unique multicast address. **Note:** The MAC does not process a Pause packet if the multicast address is different from the unique multicast address. This is also applicable to the received PFC packet when the Priority Flow Control (PFC) is enabled. The unique multicast address (0x01_80_C2_00_00_01) is as specified in IEEE 802.1 Qbb-2011.
31 RFE	Receive Flow Control Enable When this bit is set and the MAC is operating in full-duplex mode, the MAC decodes the received Pause packet and disables its transmitter for a specified (Pause) time. When this bit is reset or the MAC is operating in half-duplex mode, the decode function of the Pause packet is disabled. When PFC is enabled, flow control is enabled for PFC packets. The MAC decodes the received PFC packet and disables the Transmit queue, with matching priorities, for a duration of received Pause time.

56.2.14 Transmit Queue Priority Mapping 0 Register (MAC_TXQ_PRTY_MAP0)

The Transmit Queue Priority Mapping 0 register contains the priority values assigned to Tx Queue 0 and Tx Queue1.

Offset 0x0098 Access: User Read/Write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	PSTQ1								PSTQ0							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 986. Transmit Queue Priority Mapping 0 Register (MAC_TXQ_PRTY_MAP0)

Table 1037. MAC_TXQ_PRTY_MAP0 field descriptions

Field	Description
16:23 PSTQ1	Priorities Selected in Transmit Queue 1 This bit is similar to the PSTQ0 bit.
24:31 PSTQ0	Priorities Selected in Transmit Queue 0 This field holds the priorities assigned to Tx Queue 0 by the software. This field determines if Tx Queue 0 should be blocked from transmitting specified pause time when a PFC packet is received with priorities matching the priorities programed in this field. If the content of this field is not mutually exclusive to corresponding fields of other Transmit queues, that is, same priority is mapped to multiple Tx queues, the MAC blocks all queues with matching priority for specified time.

56.2.15 Receive Queue Control 0 Register (MAC_RXQ_CTRL0)

The Receive Queue Control 0 register controls the queue management in the MAC Receiver.

Offset 0x00A0 Access: User Read/Write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	RXQ1EN		RXQ0EN	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 987. Receive Queue Control 0 Register (MAC_RXQ_CTRL0)

Table 1038. MAC_RXQ_CTRL0 field descriptions

Field	Description
28:29 RXQ1EN	Receive Queue 1 Enable This field indicates whether Rx Queue 1 is enabled for audio video (AV) or data center bridging (DCB). 00 Not enabled 01 Queue 1 enabled for AV 10 Queue 1 enabled for DCB or generic 11 Reserved
30:31 RXQ0EN	Receive Queue 0 Enable This field indicates whether Rx Queue 0 is enabled for audio video (AV) or data center bridging (DCB). 00 Not enabled 01 Queue 0 enabled for AV 10 Queue 0 enabled for DCB or generic 11 Reserved

56.2.16 Receive Queue Control 1 Register (MAC_RXQ_CTRL1)

The Receive Queue Control 1 register controls the routing of multicast, broadcast, AV, and untagged packets to the Rx queues.

Offset 0x00A4 Access: User Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	MCBCQEN	0	MCBCQ		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	UPQ			0	0	0	0	0	AVPTPQ			0	AVCPQ		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 988. Receive Queue Control 1 Register (MAC_RXQ_CTRL1)

Table 1039. MAC_RXQ_CTRL1 field descriptions

Field	Description
11 MCBCQEN	Multicast and Broadcast Queue Enable This bit specifies that Multicast or Broadcast packets routing to the Rx Queue is enabled and the Multicast or Broadcast packets must be routed to Rx Queue specified in MCBCQ field.
13:15 MCBCQ	Multicast and Broadcast Queue This field specifies the Rx Queue onto which Multicast or Broadcast Packets are routed. Any Rx Queue enabled for Generic/AV traffic can be used to route the Multicast or Broadcast Packets. 000 Rx Queue 0 001 Rx Queue 1 others Reserved
17:19 UPQ	Untagged Packet Queue This field indicates the Rx Queue to which Untagged Packets are to be routed. Any Rx Queue enabled for Generic/AV traffic can be used to route the Untagged Packets. 000 Rx Queue 0 001 Rx Queue 1 others Reserved

Table 1039. MAC_RXQ_CTRL1 field descriptions (continued)

Field	Description
25:27 AVPTPQ	AV PTP Packets Queue This field specifies the Rx queue on which the PTP packets sent over the Ethernet payload (not over IPv4 or IPv6) are routed. 000 Rx Queue 0 001 Rx Queue 1 others: Reserved When the AV8021ASMEN bit of MAC_Timestamp_Control register is set, only untagged PTP over Ethernet packets are routed on an Rx queue.
29:31 AVCPQ	AV Untagged Control Packets Queue This field specifies the Receive queue on which the received AV untagged control packets are routed: 000 Receive Queue 0 001 Receive Queue 1 others: Reserved The AV tagged control and data packets are routed based on PSRQ field in the Transmit Flow Control Register of corresponding queue.

56.2.17 Receive Queue Control 2 Register (MAC_RXQ_CTRL2)

This register controls the routing of tagged packets based on the USP (user Priority) field of the received packets to the RxQueues 0 to 1.

Offset 0x00A8 Access: User Read/Write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	PSRQ1								PSRQ0							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 989. Receive Queue Control 2 Register (MAC_RXQ_CTRL2)

Table 1040. MAC_RXQ_CTRL2 field descriptions

Name	Description
16:23 PSRQ1	Priorities Selected in the Receive Queue 1 This field decides the priorities assigned to Rx Queue 1. All packets with priorities that match the values set in this field are routed to Rx Queue 1. For example, if PSRQ1[4] is set, packets with USP field equal to 4 are routed to Rx Queue 1. The software must ensure that the content of this field is mutually exclusive to the PSRQ fields for other queues, that is, the same priority is not mapped to multiple Rx queues.
24:31 PSRQ0	Priorities Selected in the Receive Queue 0 This field decides the priorities assigned to Rx Queue 0. All packets with priorities that match the values set in this field are routed to Rx Queue 0. For example, if PSRQ0[5] is set, packets with USP field equal to 5 are routed to Rx Queue 0. The software must ensure that the content of this field is mutually exclusive to the PSRQ fields for other queues, that is, the same priority is not mapped to multiple Rx queues. Value After Reset: 0x0

56.2.18 Interrupt Status Register (MAC_INTERRUPT_STATUS)

The Interrupt Status register contains the status of interrupts.

Offset 0x00B0 Access: User Read

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	RXSTSIS	TXSTSIS	TSIS	MMCRXIPIS	MMCTXIS	MMCRXIS	MMCIS	0	0	LPIIS	PMTIS	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 990. Interrupt Status Register (MAC_INTERRUPT_STATUS)

Table 1041. MAC_INTERRUPT_STATUS field descriptions

Field	Description
17 RXSTSIS	Receive Status Interrupt This bit indicates the status of received packets. This bit is set when the RWT bit is set in the MAC_Rx_Tx_Status register. This bit is cleared when the corresponding interrupt source bit is read in the MAC_Rx_Tx_Status register.
18 TXSTSIS	Transmit Status Interrupt This bit indicates the status of transmitted packets. This bit is set when any of the following bits is set in the MAC_Rx_Tx_Status register: – Excessive Collision (EXCOL) – Late Collision (LCOL) – Excessive Deferral (EXDEF) – Loss of Carrier (LCARR) – No Carrier (NCARR) – Jabber Timeout (TJT) This bit is cleared when the corresponding interrupt source bit is read in the MAC_Rx_Tx_Status register.
19 TSIS	Timestamp Interrupt Status This bit is set when any of the following conditions is true: – The system time value is equal to or exceeds the value specified in the Target Time High and Low registers. – There is an overflow in the Seconds register. – The Target Time Error occurred, that is, programed target time already elapsed. – When the auxiliary snapshot trigger is asserted. – When drop transmit status is enabled in MTL, this bit is set when the captured transmit timestamp is updated in the MAC_TxTimestamp_Status_Nanoseconds and Mac_TxTimestamp_Status_Seconds registers. This bit is cleared when the corresponding interrupt source bit is read in the MAC_Timestamp_Status register.
20 MMCRXIPIS	MMC Receive Checksum Offload Interrupt Status This bit is set high when an interrupt is generated in the MMC Receive Checksum Offload Interrupt Register. This bit is cleared when all bits in this interrupt register are cleared.
21 MMCTXIS	MMC Transmit Interrupt Status This bit is set high when an interrupt is generated in the MMC Transmit Interrupt Register. This bit is cleared when all bits in this interrupt register are cleared.
22 MMCRXIS	MMC Receive Interrupt Status This bit is set high when an interrupt is generated in the MMC Receive Interrupt Register. This bit is cleared when all bits in this interrupt register are cleared.
23 MMCIS	MMC Interrupt Status This bit is set high when Bit 20, Bit 21, or Bit 22 is set high. This bit is cleared only when all these bits are low.

Table 1041. MAC_INTERRUPT_STATUS field descriptions (continued)

Field	Description
26 LPIIS	LPI Interrupt Status This bit is set for any LPI state entry or exit in the MAC Transmitter or Receiver. This bit is cleared when the TLPIEN bit of MAC_LPI_Control_Status register is read.
27 PMTIS	PMT Interrupt Status This bit is set when a Magic packet or Wake-on-LAN packet is received in the power-down mode (RWKPRCVD and MGKPRCVD bits in MAC_PMT_Control_Status register). This bit is cleared when Bits[25:26] are cleared because of a Read operation to the MAC_PMT_Control_Status register.

56.2.19 Interrupt Enable Register (MAC_INTERRUPT_ENABLE)

The Interrupt Enable register contains the masks for generating the interrupts.

Offset 0x00B4 Access: User Read

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	RXSTSIE	TXSTSIE	TSIE	0	0	0	0	0	0	LPIIE	PMTIE	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 991. Interrupt Enable Register (MAC_INTERRUPT_ENABLE)

Table 1042. MAC_INTERRUPT_ENABLE field descriptions

Field	Description
17 RXSTSIE	Receive Status Interrupt Enable When this bit is set, it enables the assertion of the interrupt signal because of the setting of RXSTSIS bit in the MAC_Interrupt_Status register.
18 TXSTSIE	Transmit Status Interrupt Enable When this bit is set, it enables the assertion of the interrupt signal because of the setting of TXSTSIS bit in the MAC_Interrupt_Status register.
19 TSIE	Timestamp Interrupt Enable When this bit is set, it enables the assertion of the interrupt signal because of the setting of TSIS bit in MAC_Interrupt_Status register.

Table 1042. MAC_INTERRUPT_ENABLE field descriptions (continued)

Field	Description
26 LPIIE	LPI Interrupt Enable When this bit is set, it enables the assertion of the interrupt signal because of the setting of LPIIS bit in MAC_Interrupt_Status register.
27 PMTIE	PMT Interrupt Enable When this bit is set, it enables the assertion of the interrupt signal because of the setting of PMTIS bit in MAC_Interrupt_Status register.

56.2.20 Receive Transmit Status Register (MAC_RX_TX_STATUS)

The Receive Transmit Status register contains the Receive and Transmit Error status.

Offset 0x00B8 Access: User Read

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	RWT	0	0	EXCOL	LCOL	EXDEF	LCARR	NCARR	TJT
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 992. Receive Transmit Status Register (MAC_RX_TX_STATUS)

Table 1043. MAC_RX_TX_STATUS field descriptions

Field	Description
23 RWT	Receive Watchdog Timeout This bit is set when a packet with length greater than 2,048 bytes is received (10, 240 bytes when Jumbo Packet mode is enabled) and the WD bit is reset in the MAC_Configuration register. This bit is set when a packet with length greater than 16,383 bytes is received and the WD bit is set in the MAC_Configuration register. Access restriction applies. Clears on read. Self-set to 1 on internal event.
26 EXCOL	Excessive Collisions When the DTXSTS bit is set in the MTL_Operation_Mode register, this bit indicates that the transmission aborted after 16 successive collisions while attempting to transmit the current packet. If the DR bit is set in the MAC_Configuration register, this bit is set after the first collision and the packet transmission is aborted. Access restriction applies. Clears on read. Self-set to 1 on internal event.

Table 1043. MAC_RX_TX_STATUS field descriptions (continued)

Field	Description
27 LCOL	Late Collision When the DTXSTS bit is set in the MTL_Operation_Mode register, this bit indicates that the packet transmission aborted because a collision occurred after the collision window (64 bytes including Preamble in MII mode). This bit is not valid if the Underflow error occurs. Access restriction applies. Clears on read. Self-set to 1 on internal event.
28 EXDEF	Excessive Deferral When the DTXSTS bit is set in the MTL_Operation_Mode register and the DC bit is set in the MAC_Configuration register, this bit indicates that the transmission ended because of excessive deferral of over 24,288 bit times (155,680 when Jumbo packet is enabled). Access restriction applies. Clears on read. Self-set to 1 on internal event.
29 LCARR	Loss of Carrier When the DTXSTS bit is set in the MTL_Operation_Mode register, this bit indicates that the loss of carrier occurred during packet transmission, that is, the phy_crs_i signal was inactive for one or more transmission clock periods during packet transmission. This bit is valid only for packets transmitted without collision. Access restriction applies. Clears on read. Self-set to 1 on internal event.
30 NCARR	No Carrier When the DTXSTS bit is set in the MTL_Operation_Mode register, this bit indicates that the carrier signal from the PHY is not present at the end of preamble transmission. Access restriction applies. Clears on read. Self-set to 1 on internal event.
31 TJT	Transmit Jabber Timeout This bit indicates that the Transmit Jabber Timer expired which happens when the packet size exceeds 2,048 bytes (10,240 bytes when the Jumbo packet is enabled) and JD bit is reset in the MAC_Configuration register. This bit is set when the packet size exceeds 16,383 bytes and the JD bit is set in the MAC_Configuration register. Access restriction applies. Clears on read. Self-set to 1 on internal event.

56.2.21 PMT Control and Status Register (MAC_PMT_CONTROL_STATUS)

This register captures the remote wakeup control and status.

Offset 0x00C0 Access: User Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	RWKFILTRST	0	0	0	0	RWKPTR			0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	RWKPFE	GLBLUCAST	0	0	RWKPRCVD	MGKPRCVD	0	0	RWKPKTEN	MGKPKTEN	PWRDWN
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 993. PMT Control and Status Register (MAC_PMT_CONTROL_STATUS)

Table 1044. MAC_PMT_CONTROL_STATUS field descriptions

Field	Description
0 RWKFILTRST	Remote Wake-Up Packet Filter Register Pointer Reset When this bit is set, the remote wake-up packet filter register pointer is reset to 3'b000. It is automatically cleared after 1 clock cycle. Access restriction applies. Setting 1 sets. Self-cleared. Setting 0 has no effect.
5:7 RWKPTR	Remote Wake-up FIFO Pointer This field gives the current value (0 to 7) of the Remote Wake-up Packet Filter register pointer. When the value of this pointer is equal to 7, the contents of the Remote Wake-up Packet Filter Register are transferred to the clk_rx_i domain when a Write occurs to that register.
21 RWKPFE	Remote Wake-up Packet Forwarding Enable When this bit is set along with RWKPKTEN, the MAC receiver drops all received frames until it receives the expected Wake-up frame. All frames after that event including the received wake-up frame are forwarded to application. This bit is then self-cleared on receiving the wake-up packet. The application can also clear this bit before the expected wake-up frame is received. In such cases, the MAC reverts to the default behavior where packets received are forwarded to the application. This bit must only be set when RWKPKTEN is set high and PWRDWN is set low. The setting of this bit has no effect when PWRDWN is set high. **Note:** If Magic Packet Enable and Wake-Up Frame Enable are both set along with setting of this bit and Magic Packet is received prior to wake-up frame, this bit is self-cleared on receiving Magic Packet, the received Magic packet is dropped, and all frames after received Magic Packet are forwarded to application. Access restriction applies. Setting 1 sets. Self-cleared. Setting 0 has no effect.
22 GLBLUCAST	Global Unicast When this bit set, any unicast packet filtered by the MAC (DAF) address recognition is detected as a remote wake-up packet.

Table 1044. MAC_PMT_CONTROL_STATUS field descriptions (continued)

Field	Description
25 RWKPRCVD	Remote Wake-Up Packet Received When this bit is set, it indicates that the power management event is generated because of the reception of a remote wake-up packet. This bit is cleared when this register is read. Access restriction applies. Clears on read. Self-set to 1 on internal event.
26 MGKPRCVD	Magic Packet Received When this bit is set, it indicates that the power management event is generated because of the reception of a magic packet. This bit is cleared when this register is read. Access restriction applies. Clears on read. Self-set to 1 on internal event.
29 RWKPKTEN	Remote Wake-Up Packet Enable When this bit is set, a power management event is generated when the MAC receives a remote wake-up packet.
30 MGKPKTEN	Magic Packet Enable When this bit is set, a power management event is generated when the MAC receives a magic packet.
31 PWRDWN	Power Down When this bit is set, the MAC receiver drops all received packets until it receives the expected magic packet or remote wake-up packet. This bit is then self-cleared and the power-down mode is disabled. The software can clear this bit before the expected magic packet or remote wake-up packet is received. The packets received by the MAC after this bit is cleared are forwarded to the application. This bit must only be set when the Magic Packet Enable, Global Unicast, or Remote Wake-Up Packet Enable bit is set high. **Note:** You can gate-off the CSR clock during the power-down mode. However, when the CSR clock is gated-off, you cannot perform any read or write operations on this register. Therefore, the Software cannot clear this bit. Access restriction applies. Setting 1 sets. Self-cleared. Setting 0 has no effect.

56.2.22 Wake-up Packet Filter Registers

The wkuppktfilter_reg register at address 0x00C4 loads the Wake-up Packet Filter register.

To load values in a Wake-up Packet Filter register, the entire register (wkuppktfilter_reg) must be written. The wkuppktfilter_reg register is loaded by sequentially loading the eight register values in address (0x00C4) for wkuppktfilter_reg0, wkuppktfilter_reg1, …, wkuppktfilter_reg7, respectively. The wkuppktfilter_reg register is read in a similar way.

The Ethernet module updates the wkuppktfilter_reg register current pointer value in Bits[5:7] of MAC_PMT_Control_Status register.

- Filter i Byte Mask: The filter i byte mask register defines the bytes of the packet that are examined by filter i (0, 1, 2, 3) to determine whether or not a packet is a wake-up packet.
  - The MSB (bit 0) must be zero.
  - Bit j[1:31] is the byte mask.
  - If Bit j (byte number) of the byte mask is set, the CRC block processes the Filter i Offset + j of the incoming packet; otherwise Filter i Offset + j is ignored.
- Filter i Command: The 4-bit filter i command controls the filter i operation.
  - Bit 28 specifies the address type, defining the destination address type of the pattern. When the bit is set, the pattern applies to only multicast packets; when the bit is reset, the pattern applies only to unicast packet.
  - Bit 29 (Inverse Mode), when set, reverses the logic of the CRC16 hash function signal, to reject a packet with matching CRC_16 value.
  - Bit 29, along with Bit 30, allows a MAC to reject a subset of remote wake-up packets by creating filter logic such as "Pattern 1 AND NOT Pattern 2".
  - Bit 30 (And_Previous) implements the Boolean logic. When set, the result of the current entry is logically ANDed with the result of the previous filter. This AND logic allows a filter pattern longer than 32 bytes by splitting the mask among two, three, or four filters. This depends on the number of filters that have the And_Previous bit set.
  - Bit 31 is the enable for filter i. If Bit 31 is not set, filter i is disabled.
- Filter i Offset: This filter i offset register defines the offset (within the packet) from which the filter i examines the packets.
  - This 8-bit pattern-offset is the offset for the filter i first byte to be examined.
  - The minimum allowed offset is 12, which refers to the 13th byte of the packet.
  - The offset value 0 refers to the first byte of the packet.
- Filter i CRC-16: This filter i CRC-16 register contains the CRC_16 value calculated from the pattern and also the byte mask programed to the wake-up filter register block.
  - The 16-bit CRC calculation uses the following polynomial:
    $G(x) = x^{16} + x^{15} + x^2 + 1$
    Each mask, used in the hash function calculation, is compared with a 16-bit value associated with that mask. Each filter has the following:
  - 32-bit Mask: Each bit in this mask corresponds to one byte in the detected packet. If the bit is 1', the corresponding byte is taken into the CRC16 calculation.
  - 8-bit Offset Pointer: Specifies the byte to start the CRC16 computation.

The pointer and the mask are used together to locate the bytes to be used in the CRC16 calculations.

Note: If you are accessing these registers in byte or half-word mode, the internal counter to access the appropriate wkuppktfilter_reg is incremented when CPU accesses Lane 3 (or Lane 0 in big-endian mode).

When any Register content is being transferred to a different clock domain after a write operation, there should not be any further writes to the same location until the first write is updated. Otherwise, the second write operation does not get updated to the destination clock domain. Therefore, the delay between two writes to the same register location should

be at least 4 cycles of the destination clock (PHY receive clock, PHY transmit clock, or PTP clock).

Table 1045. Remote Wake-Up Packet Filter Register

wkuppktfilter_reg0	Filter 0 Byte Mask							
wkuppktfilter_reg1	Filter 1 Byte Mask							
wkuppktfilter_reg2	Filter 2 Byte Mask							
wkuppktfilter_reg3	Filter 3 Byte Mask							
wkuppktfilter_reg4	RSVD	Filter 3 Command	RSVD	Filter 2 Command	RSVD	Filter 1 Command	RSVD	Filter 0 Command
wkuppktfilter_reg5	Filter 3 Offset		Filter 2 Offset		Filter 1 Offset		Filter 0 Offset	
wkuppktfilter_reg6	Filter 1 CRC - 16				Filter 0 CRC - 16			
wkuppktfilter_reg7	Filter 3 CRC - 16				Filter 2 CRC - 16			

56.2.22.1 Wakeup Filter0 Byte Mask Register (RWK_FILTER0_BYTE_MASK)

Corresponds to registers wkuppktfilter_reg0 described in *Table 1045*.

Offset 0x00C4 Access: User Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	Filter0_Byte_Mask															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	Filter0_Byte_Mask															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 994. Wakeup Filter0 Byte Mask Register (RWK_FILTER0_BYTE_MASK)

Table 1046. RWK_FILTER0_BYTE_MASK field descriptions

Field	Description
0:31 Filter0_Byte_Mask	The filter 0 byte mask register defines the bytes of the packet that are examined by filter 0 to determine whether or not a packet is a wake-up packet. The MSB (bit 0) must be zero. Bit j[1:31] is the byte mask. If bit j (byte number) of the byte mask is set, the CRC block processes the filter 0 offset + j of the incoming packet; otherwise filter 0 offset + j is ignored.

56.2.22.2 Wakeup Filter 1 Byte Mask Register (RWK_FILTER1_BYTE_MASK)

Corresponds to registers wkuppktfilter_reg1 described in *Table 1045*.

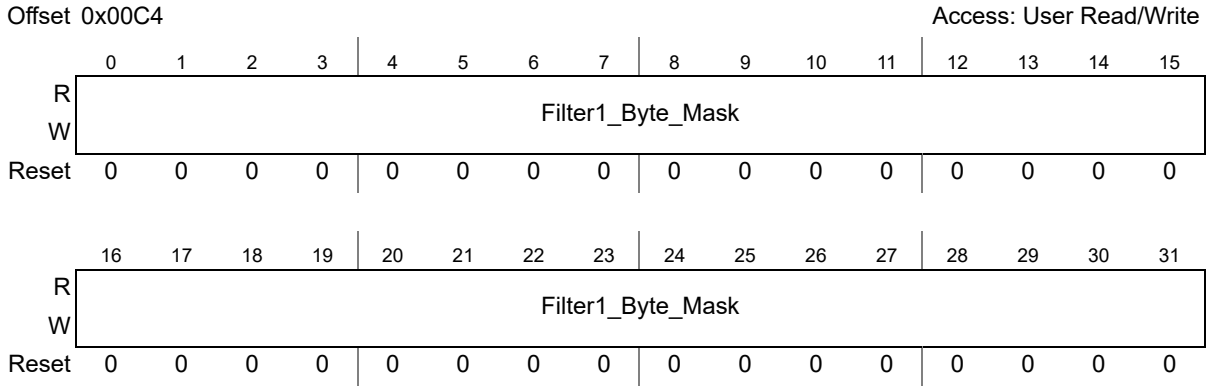


Figure 995. Wakeup Filter 1 Byte Mask Register (RWK_FILTER1_BYTE_MASK)

Table 1047. RWK_FILTER1_BYTE_MASK field descriptions

Field	Description
0:31 Filter1_Byte_Mask	The filter 1 byte mask register defines the bytes of the packet that are examined by filter 1 to determine whether or not a packet is a wake-up packet. The MSB (bit 0) must be zero. Bit j[1:31] is the byte mask. If bit j (byte number) of the byte mask is set, the CRC block processes the filter 1 offset + j of the incoming packet; otherwise filter 1 offset + j is ignored.

56.2.22.3 Wakeup Filter 2 Byte Mask Register (RWK_FILTER2_BYTE_MASK)

Corresponds to registers wkuppktfilter_reg2 described in *Table 1045*.

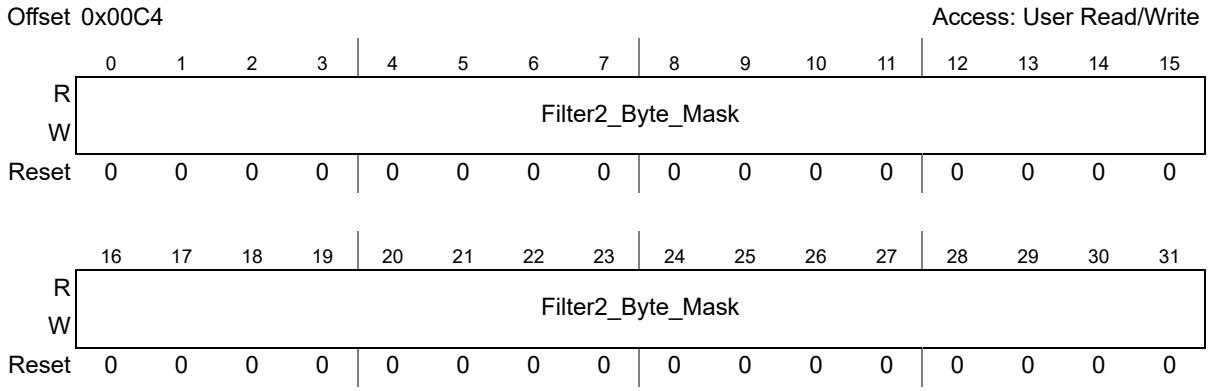


Figure 996. Wakeup Filter 2 Byte Mask Register (RWK_FILTER2_BYTE_MASK)

Table 1048. RWK_FILTER2_BYTE_MASK field descriptions

Field	Description
0:31 Filter2_Byte_Mask	The filter 2 byte mask register defines the bytes of the packet that are examined by filter 2 to determine whether or not a packet is a wake-up packet. The MSB (bit 0) must be zero. Bit j[1:31] is the byte mask. If bit j (byte number) of the byte mask is set, the CRC block processes the filter 2 offset + j of the incoming packet; otherwise filter 2 offset + j is ignored.

56.2.22.4 Wakeup Filter 3 Byte Mask Register (RWK_FILTER3_BYTE_MASK)

Corresponds to registers wkuppktfilter_reg3 described in *Table 1045*.

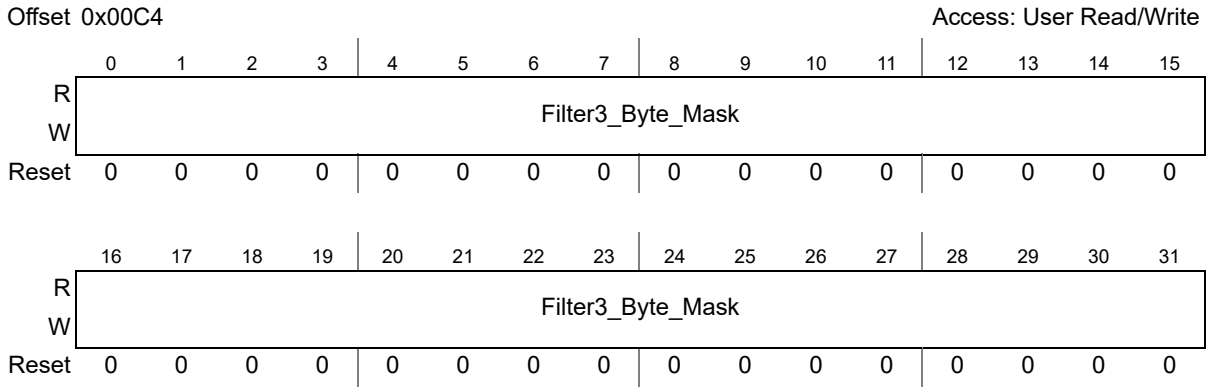


Figure 997. Wakeup Filter 3 Byte Mask Register (RWK_FILTER3_BYTE_MASK)

Table 1049. RWK_FILTER3_BYTE_MASK field descriptions

Field	Description
0:31 Filter3_Byte_Mask	The filter 3 byte mask register defines the bytes of the packet that are examined by filter 3 to determine whether or not a packet is a wake-up packet. The MSB (bit 0) must be zero. Bit j[1:31] is the byte mask. If bit j (byte number) of the byte mask is set, the CRC block processes the filter 3 offset + j of the incoming packet; otherwise filter 3 offset + j is ignored.

56.2.22.5 Wakeup Filter 0123 Command Register (RWK_FILTER0123_COMMAND)

Corresponds to registers wkuppktfilter_reg4 described in *Table 1045*.

Offset 0x00C4 Access: User Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	Filter3_Command				0	0	0	0	Filter2_Command			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	Filter1_Command				0	0	0	0	Filter0_Command			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 998. Wakeup Filter 0123 Command Register (RWK_FILTER0123_COMMAND)

Table 1050. RWK_FILTER0123_COMMAND field descriptions

Field	Description
4:7 Filter3_Command	Refer to the description in *Section 56.2.22*.
12:15 Filter2_Command	Refer to the description in *Section 56.2.22*.
20:23 Filter1_Command	Refer to the description of *Section 56.2.22*.
28:31 Filter0_Command	Refer to the description of *Section 56.2.22*.

56.2.22.6 Wakeup Filter 0123 Offset Register (RWK_FILTER0123_OFFSET)

Corresponds to registers wkuppktfilter_reg5 described in *Table 1045*.

Offset 0x00C4 Access: User Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	Filter3_Offset								Filter2_Offset							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	Filter1_Offset								Filter0_Offset							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 999. Wakeup Filter 0123 Offset Register (RWK_FILTER0123_OFFSET)

Table 1051. RWK_FILTER0123_OFFSET field descriptions

Field	Description
0:7 Filter3_Offset	Refer to the description of *Section 56.2.22*.
8:15 Filter2_Offset	Refer to the description of *Section 56.2.22*.
16:23 Filter1_Offset	Refer to the description of *Section 56.2.22*.
24:31 Filter0_Offset	Refer to the description of *Section 56.2.22*.

56.2.22.7 Wakeup Filter01 CRC Register (RWK_FILTER01_CRC)

Corresponds to registers wkuppktfilter_reg6 described in *Table 1045*.

Offset 0x00C4 Access: User Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	Filter1_CRC															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	Filter0_CRC															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1000. Wakeup Filter01 CRC Register (RWK_FILTER01_CRC)

Table 1052. RWK_FILTER01_CRC field descriptions

Field	Description
0:15 Filter1_CRC	Refer to the description in *Section 56.2.22*.
16:31 Filter0_CRC	Refer to the description in *Section 56.2.22*.

56.2.22.8 Wakeup Filter23 CRC Register (RWK_FILTER23_CRC)

Corresponds to registers wkuppktfilter_reg7 described in *Table 1045*.

Offset 0x00C4 Access: User Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	Filter3_CRC															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	Filter2_CRC															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1001. Wakeup Filter23 CRC Register (RWK_FILTER23_CRC)

Table 1053. RWK_FILTER23_CRC field descriptions

Field	Description
0:15 Filter3_CRC	Refer to the description of *Section 56.2.22*.
16:31 Filter2_CRC	Refer to the description of *Section 56.2.22*.

56.2.23 LPI Control and Status Register (MAC_LPI_CONTROL_STATUS)

The LPI Control and Status Register controls the LPI functions and provides the LPI interrupt status. The status bits are cleared when this register is read.

Offset 0x00D0 Access: User Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	LPITCSE	LPIATE	LPITXA	0	PLS	LPIEN
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	RLPIST	TLPIST	0	0	0	0	RLPIEX	RLPIEN	TLPIEX	TLPIEN
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1002. LPI Control and Status Register (MAC_LPI_CONTROL_STATUS)

Table 1054. MAC_LPI_CONTROL_STATUS field descriptions

Field	Description
10 LPITCSE	LPI Tx Clock Stop Enable When this bit is set, the MAC asserts sbd_tx_clk_gating_ctrl_o signal high after it enters Tx LPI mode to indicate that the Tx clock to MAC can be stopped. When this bit is reset, the MAC does not assert sbd_tx_clk_gating_ctrl_o signal high after it enters Tx LPI mode.
11 LPIATE	LPI Timer Enable This bit controls the automatic entry of the MAC Transmitter into and exit out of the LPI state. When LPITE, LPITXA and LPITXEN bits are set, the MAC Transmitter enters LPI state only when the complete MAC TX data path is IDLE for a period indicated by the MAC_LPI_Entry_Timer register. After entering LPI state, if the data path becomes non-IDLE (due to a new packet being accepted for transmission), the Transmitter exits LPI state but does not clear LPITXEN bit. This enables the re-entry into LPI state when it is IDLE again. When LPITE is 0, the LPI Auto timer is disabled and MAC Transmitter enters LPI state based on the settings of LPITXA and LPITXEN bit descriptions.
12 LPITXA	LPI Tx Automate This bit controls the behavior of the MAC when it is entering or coming out of the LPI mode on the Transmit side. If the LPITXA and LPIEN bits are set to 1, the MAC enters the LPI mode only after all outstanding packets (in the core) and pending packets (in the application interface) have been transmitted. The MAC comes out of the LPI mode when the application sends any packet for transmission or the application issues a Tx FIFO Flush command. In addition, the MAC automatically clears the LPIEN bit when it exits the LPI state. If Tx FIFO Flush is set in the FTQ bit of the MTL_TxQ0_Operation_Mode register, when the MAC is in the LPI mode, it exits the LPI mode. When this bit is 0, the LPIEN bit directly controls behavior of the MAC when it is entering or coming out of the LPI mode.
14 PLS	PHY Link Status This bit indicates the link status of the PHY. The MAC Transmitter asserts the LPI pattern only when the link status is up (OKAY) at least for the time indicated by the LPI LS TIMER. When this bit is set, the link is considered to be okay (UP) and when this bit is reset, the link is considered to be down. Software should get the PHY link status by reading the PHY register and accordingly update the PLS bit
15 LPIEN	LPI Enable When this bit is set, it instructs the MAC Transmitter to enter the LPI state. When this bit is reset, it instructs the MAC to exit the LPI state and resume normal transmission. This bit is cleared when the LPITXA bit is set and the MAC exits the LPI state because of the arrival of a new packet for transmission.
22 RLPIST	Receive LPI State When this bit is set, it indicates that the MAC is receiving the LPI pattern on the MII interface.
23 TLPIST	Transmit LPI State When this bit is set, it indicates that the MAC is transmitting the LPI pattern on the MII interface.

Table 1054. MAC_LPI_CONTROL_STATUS field descriptions (continued)

Field	Description
28 RLPIEX	Receive LPI Exit When this bit is set, it indicates that the MAC Receiver has stopped receiving the LPI pattern on the MII interface, exited the LPI state, and resumed the normal reception. This bit is cleared by a read into this register. **Note:** This bit may not be set if the MAC stops receiving the LPI pattern for a very short duration, such as, less than three clock cycles of CSR clock.
29 RLPIEN	Receive LPI Entry When this bit is set, it indicates that the MAC Receiver has received an LPI pattern and entered the LPI state. This bit is cleared by a read into this register. **Note:** This bit may not be set if the MAC stops receiving the LPI pattern for a very short duration, such as, less than three clock cycles of CSR clock.
30 TLPIEX	Transmit LPI Exit When this bit is set, it indicates that the MAC transmitter exited the LPI state after the application cleared the LPIEN bit and the LPI TW Timer has expired. This bit is cleared by a read into this register.
31 TLPIEN	Transmit LPI Entry When this bit is set, it indicates that the MAC Transmitter has entered the LPI state because of the setting of the LPIEN bit. This bit is cleared by a read into this register.

56.2.24 LPI Timers Control Register (MAC_LPI_TIMERS_CONTROL)

The LPI Timers Control register controls the timeout values in the LPI states. It specifies the time for which the MAC transmits the LPI pattern and also the time for which the MAC waits before resuming the normal transmission.

Offset 0x00D4 Access: User Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	LST									
W																
Reset	0	0	0	0	0	0	1	1	1	1	1	0	1	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	TWT															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1003. LPI Timers Control Register (MAC_LPI_TIMERS_CONTROL)

Table 1055. MAC_LPI_TIMERS_CONTROL field descriptions

Field	Description
6:15 LST	LPI LS TIMER This field specifies the minimum time (in milliseconds) for which the link status from the PHY should be up (OKAY) before the LPI pattern can be transmitted to the PHY. The MAC does not transmit the LPI pattern even when the LPIEN bit is set unless the LPI LS Timer reaches the programed terminal count. The default value of the LPI LS Timer is 1000 (1 sec) as defined in the IEEE standard.
16:31 TWT	LPI TW TIMER This field specifies the minimum time (in microseconds) for which the MAC waits after it stops transmitting the LPI pattern to the PHY and before it resumes the normal transmission. The TLPIEX status bit is set after the expiry of this timer.

56.2.25 LPI Entry Timer Register (MAC_LPI_ENTRY_TIMER)

This register controls the Tx LPI entry timer. This counter is enabled only when bit[11](LPITE) bit of MAC_LPI_Control_Status is set to 1.

Offset 0x00D8 Access: User Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	LPIET			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	LPIET													0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1004. LPI Entry Timer Register (MAC_LPI_ENTRY_TIMER)

Table 1056. MAC_LPI_ENTRY_TIMER field descriptions

Field	Description
12:28 LPIET	LPI Entry Timer This field specifies the time in microseconds the MAC will wait to enter LPI mode, after it has transmitted all the frames. This field is valid and used only when LPITE and LPITXA are set to 1. Bits [29:31] are read-only so that the granularity of this timer is in steps of 8 micro-seconds.

56.2.26 1US Tic Counter Register (MAC_1US_TIC_COUNTER)

This register controls the generation of the Reference time (1 microsecond tic) for all the LPI timers. This timer has to be programed by the software initially.

Offset 0x00DC Access: User Read/Write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	TIC_1US_CNTR											
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1005. 1US Tic Counter Register (MAC_1US_TIC_COUNTER)

Table 1057. MAC_1US_TIC_COUNTER field descriptions

Field	Description
20:31 TIC_1US_CNTR	1US TIC Counter The application must program this counter so that the number of clock cycles of CSR clock is 1us. (Subtract 1 from the value before programming). For example if the CSR clock is 100MHz then this field needs to be programed to value 100 - 1 = 99 (which is 0x63). This is required to generate the 1US events that are used to update some of the EEE related counters.

56.2.27 Version Register (MAC_VERSION)

The version register identifies the version of the Ethernet module.

Offset 0x0110 Access: User Read

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	USERVER								VENDORVER							
W																
Reset	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0	1

Figure 1006. Version Register (MAC_VERSION)

Table 1058. MAC_VERSION field descriptions

Field	Description
16:23 USERVER	User IP Version Number
24:31 VENDORVER	Vendor IP Version Number

56.2.28 Debug Register (MAC_DEBUG)

The Debug register provides the debug status of various MAC blocks.

Offset 0x0114 Access: User Read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	TFCSTS		TPESTS
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	RFCFCSTS		RPESTS
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1007. Debug Register (MAC_DEBUG)

Table 1059. MAC_DEBUG field descriptions

Field	Description
13:14 TFCSTS	MAC Transmit Packet Controller Status This field indicates the state of the MAC Transmit Packet Controller module: 00 Idle state 01 Waiting for one of the following: Status of the previous packet OR IPG or backoff period to be over 10 Generating and transmitting a Pause control packet (in full-duplex mode) 11 Transferring input packet for transmission
15 TPESTS	MAC MII Transmit Protocol Engine Status When this bit is set, it indicates that the MAC MII transmit protocol engine is actively transmitting data, and it is not in the Idle state.
29:30 RFCFCSTS	MAC Receive Packet Controller FIFO Status When this bit is set, this field indicates the active state of the small FIFO Read and Write controllers of the MAC Receive Packet Controller module.
31 RPESTS	MAC MII Receive Protocol Engine Status When this bit is set, it indicates that the MAC MII receive protocol engine is actively receiving data, and it is not in the Idle state.

56.2.29 Hardware Feature0 Register (MAC_HW_FEATURE0)

This register indicates the presence of the optional features or functions of the Ethernet module. The software driver can use this register to dynamically enable or disable the programs related to the optional blocks.

Offset 0x011C Access: User Read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	ACTPHYSEL			SAVLANINS	TSSTSSEL		MACADR64SEL	MACADR32SEL	ADDMACADRSEL					0	RXCOESEL
W																
Reset	0	1	0	0	1	0	1	1	1	0	0	0	0	0	0	1

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	TXCOESEL	EEESEL	TSSEL	0	0	ARPOFFSEL	MMCSEL	MGKSEL	RWKSEL	SMASEL	VLHASH	0	HDSEL	0	MIISEL
W																
Reset	0	1	1	1	0	0	1	1	1	1	1	1	0	1	0	1

Figure 1008. Hardware Feature0 Register (MAC_HW_FEATURE0)

Table 1060. MAC_HW_FEATURE0 field descriptions

Field	Description
1:3 ACTPHYSEL	Active PHY Selected When you have multiple PHY interfaces in your configuration, this field indicates the sampled value of phy_intf_sel_i during reset de-assertion: 000 MII 100 RMII All Others Reserved
4 SAVLANINS	Source Address or VLAN Insertion Enable
5:6 TSSTSSEL	Timestamp System Time Source This bit indicates the source of the Timestamp system time: 01 Internal
7 MACADR64SEL	MAC Addresses 64-127 Selected This bit is set to 1 when the Additional 64 MAC Address Registers (64-127) are present.
8 MACADR32SEL	MAC Addresses 32-63 Selected This bit is set to 1 when the Additional 32 MAC Address Registers (32-63) are present.
9:13 ADDMACADRSEL	MAC Addresses 1-31 Selected This bit is set to 1 when the Additional 1-31 MAC Address Registers are present
15 RXCOESEL	Receive Checksum Offload present This bit is set to 1 when the Receive TCP/IP Checksum Check is present

Table 1060. MAC_HW_FEATURE0 field descriptions (continued)

Field	Description
17 TXCOESEL	Transmit Checksum Offload present
18 EEESEL	Energy Efficient Ethernet present
19 TSSEL	IEEE 1588-2008 Timestamp present
22 ARPOFFSEL	ARP Offload present
23 MMCSEL	RMON Module Present
24 MGKSEL	PMT Magic Packet present
25 RWKSEL	PMT Remote Wake-up Packet Enable
26 SMASEL	SMA (MDIO) Interface present
27 VLHASH	VLAN Hash Filter enabled
29 HDSEL	Half-duplex Support
31 MIISEL	10 or 100 Mbps Support

56.2.30 Hardware Feature1 Register (MAC_HW_FEATURE1)

This register indicates the presence of the optional features or functions of the Ethernet module. The software driver can use this register to dynamically enable or disable the programs related to the optional blocks.

Offset 0x0120 Access: User Read

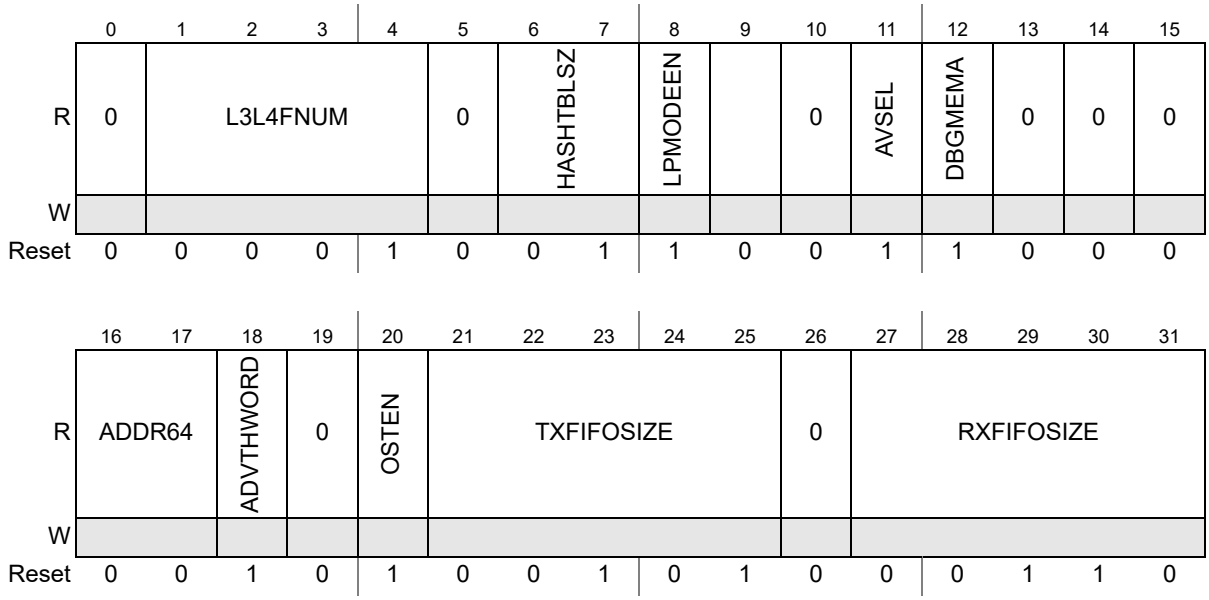


Figure 1009. Hardware Feature1 Register (MAC_HW_FEATURE1)

Table 1061. MAC_HW_FEATURE1 field descriptions

Field	Description
1:4 L3L4FNUM	Total number of L3 or L4 Filters This field indicates the total number of L3 or L4 filters: 0001 1 L3 or L4 Filter
6:7 HASHTBLSZ	Hash Table Size This field indicates the size of the hash table: 01 64 bit hash table
8 LPMODEEN	Low Power Mode Enabled This bit is set to 1 when the Enable UPF-Based Low-Power Support is present.
11 AVSEL	AV Feature Enabled This bit is set to 1 when the Enable Audio Video Bridging is present.
12 DBGMEMA	DMA Debug Registers Enabled This bit is set to 1 when the Debug Mode is present
16:17 ADDR64	Address Width. This field indicates the configured address width: 00 32-bit address width
18 ADVTHWORD	IEEE 1588 High Word Register Present
20 OSTEN	One-Step Timestamping Present

Table 1061. MAC_HW_FEATURE1 field descriptions (continued)

Field	Description
21:25 TXFIFOSIZE	MTL Transmit FIFO Size This field contains the configured value of MTL Tx FIFO in bytes expressed as Log to base 2 minus 7, that is, Log2(TXFIFO_SIZE) -7: 00101 4,096 bytes
27:31 RXFIFOSIZE	MTL Receive FIFO Size This field contains the configured value of MTL Rx FIFO in bytes expressed as Log to base 2 minus 7, that is, Log2(RXFIFO_SIZE) -7: 00110 8,192 bytes

56.2.31 Hardware Feature2 Register (MAC_HW_FEATURE2)

This register indicates the presence of the optional features or functions of the Ethernet module. The software driver can use this register to dynamically enable or disable the programs related to the optional blocks.

Offset 0x0124 Access: User Read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	AUXSNAPNUM			0	PPSOUTNUM			0	0	TXCHCNT				0	0
W																
Reset	0	0	1	0	0	0	0	1	0	0	0	0	0	1	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	RXCHCNT				0	0	TXQCNT				0	0	RXQCNT			
W																
Reset	0	0	0	1	0	0	0	0	0	1	0	0	0	0	0	1

Figure 1010. Hardware Feature2 Register (MAC_HW_FEATURE2)

Table 1062. MAC_HW_FEATURE2 field descriptions

Field	Description
1:3 AUXSNAPNUM	Number of Auxiliary Snapshot Inputs This field indicates the number of auxiliary snapshot inputs: 010 2 auxiliary inputs others Reserved
5:7 PPSOUTNUM	Number of PPS Outputs This field indicates the number of PPS outputs: 001 1 PPS output others Reserved
10:13 TXCHCNT	Number of DMA Transmit Channels This field indicates the number of DMA Transmit channels: 0001 2 DMA Tx Channels
16:19 RXCHCNT	Number of DMA Receive Channels This field indicates the number of DMA Receive channels: 0001 2 DMA Rx Channels

Table 1062. MAC_HW_FEATURE2 field descriptions (continued)

Field	Description
22:25 TXQCNT	Number of MTL Transmit Queues This field indicates the number of MTL Transmit queues: 0001 2 MTL Tx Queues
28:31 RXQCNT	Number of MTL Receive Queues This field indicates the number of MTL Receive queues: 0001 2 MTL Rx Queues

56.2.32 MDIO Address Register (MAC_MDIO_ADDRESS)

The MDIO Address register controls the management cycles to external PHY through a management interface.

Offset 0x0200 Access: User Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	PSE	BTB	PA						RDA			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	NTC			CR				0	0	0	SKAP	GOC		C45E	GB
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1011. MDIO Address Register (MAC_MDIO_ADDRESS)

Table 1063. MAC_MDIO_ADDRESS field description

Field	Description
4 PSE	Preamble Suppression Enable When this bit is set, the SMA will suppress the 32-bit preamble and transmit MDIO frames with only 1 preamble bit. When this bit is 0, the MDIO frame always has 32 bits of preamble as defined in the IEEE specifications.
5 BTB	Back to Back transactions When this bit is set and the NTC has value greater than 0, then the MAC will inform the completion of a read or write command at the end of frame transfer (before the trailing clocks are transmitted). The software can thus initiate the next command which will be executed immediately irrespective of the number trailing clocks generated for the previous frame. When this bit is reset, then the read/write command completion (GB is cleared) only after the trailing clocks are generated. In this mode, it is ensured that the NTC is always generated after each frame. This bit must not be set when NTC=0.
6:10 PA	Physical Layer Address This field indicates which Clause 22 PHY devices (out of 32 devices) the MAC is accessing. This field indicates which Clause 45 capable PHYs (out of 32 PHYs) the MAC is accessing.

Table 1063. MAC_MDIO_ADDRESS field description (continued)

Field	Description
11:15 RDA	Register/Device Address These bits select the PHY register in selected Clause 22 PHY device. These bits select the Device (MMD) in selected Clause 45 capable PHY.
17:19 NTC	Number of Training Clocks This field controls the number of trailing clock cycles generated on sma_mdc_o (MDC) after the end of transmission of MDIO frame. The valid values can be from 0 to 7. Programming the value to 3'h3 indicates that there are additional three clock cycles on the MDC line after the end of MDIO frame transfer.
20:23 CR	CSR Clock Range The CSR Clock Range selection determines the frequency of the MDC clock according to the CSR clock frequency used in your design: 0000 CSR clock = 60-100 MHz; MDC clock = CSR clock/42 0001 CSR clock = 100-150 MHz; MDC clock = CSR clock/62 0010 CSR clock = 20-35 MHz; MDC clock = CSR clock/16 0011 CSR clock = 35-60 MHz; MDC clock = CSR clock/26 0100 CSR clock = 150-250 MHz; MDC clock = CSR clock/102 0101 CSR clock = 250-300 MHz; MDC clock = CSR clock/124 0110 Reserved 0111 Reserved The suggested range of CSR clock frequency applicable for each value (when Bit 20 = 0) ensures that the MDC clock is approximately between 1.0 MHz to 2.5 MHz frequency range. When Bit 20 is set, you can achieve a higher frequency of the MDC clock than the frequency limit of 2.5 MHz (specified in the IEEE 802.3) and program a clock divider of lower value. For example, when CSR clock is of 100 MHz frequency and you program these bits as 1010, the resultant MDC clock is of 12.5 MHz which is above the range specified in IEEE 802.3. Program the following values only if the interfacing chips support faster MDC clocks: 1000 CSR clock/4 1001 CSR clock/6 1010 CSR clock/8 1011 CSR clock/10 1100 CSR clock/12 1101 CSR clock/14 1110 CSR clock/16 1111 CSR clock/18
27 SKAP	Skip Address Packet When this bit is set, the SMA does not send the address packets before read, write, or post-read increment address packets. This bit is valid only when C45E is set.
28:29 GOC	PHY SMA interface Operation Command This field indicates the operation command to the PHY: 00 Reserved 01 Write 10 Post Read Increment Address for Clause 45 PHY 11 Read When Clause 22 PHY is enabled, only Write (01) and Read (11) commands are valid.

Table 1063. MAC_MDIO_ADDRESS field description (continued)

Field	Description
30 C45E	Clause 45 PHY Enable When this bit is set, Clause 45 capable PHY is connected to MDIO. When this bit is reset, Clause 22 capable PHY is connected to MDIO.
31 GB	PHY SMA interface Busy The application sets this bit to instruct the SMA to initiate a Read or Write access to the MDIO slave. The MAC clears this bit after the MDIO frame transfer is completed. Hence the software must not write or change any of the fields in MAC_MDIO_Address and MAC_MDIO_Data registers as long as this bit is set. For write transfers, the application must first write 16-bit data in the GDI field (and also RA field when C45E is set) in MAC_MDIO_Data register before setting this bit. When C45E is set, it should also write into the RA field of MAC_MDIO_Data register before initiating a read transfer. When a read transfer is completed (GB=0), the data read from the PHY register is valid in the GD field of the MAC_MDIO_Data register. **Note:** Even if the addressed PHY is not present, there is no change in the functionality of this bit. Access restriction applies. Setting 1 sets. Self-cleared. Setting 0 has no effect.

56.2.33 MDIO Data Register (MAC_MDIO_DATA)

The MDIO Data register stores the Write data to be written to the PHY register located at the address specified in MAC_MDIO_Address. This register also stores the Read data from the PHY register located at the address specified by MDIO Address register.

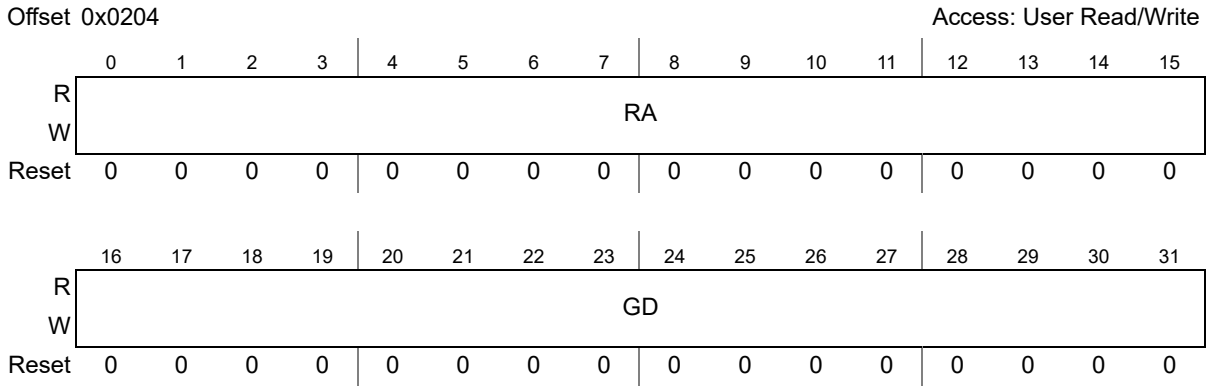


Figure 1012. MDIO Data Register (MAC_MDIO_DATA)

Table 1064. MAC_MDIO_DATA field descriptions

Name	Description
0:15 RA	Register Address This field is valid only when C45E is set. It contains the Register Address in the PHY to which the MDIO frame is intended for.
16:31 GD	PHY management Data This field contains the 16-bit data value read from the PHY after a Management Read operation or the 16-bit data value to be written to the PHY before a Management Write operation.

56.2.34 ARP Address Register (MAC_ARP_ADDRESS)

The ARP Address register contains the IPv4 Destination Address of the MAC.

Offset 0x0210 Access: User Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	ARPPA															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	ARPPA															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1013. ARP Address Register (MAC_ARP_ADDRESS)

Table 1065. MAC_ARP_ADDRESS field descriptions

Name	Description
0:31 ARPPA	ARP Protocol Address This field contains the IPv4 Destination Address of the MAC. This address is used for perfect match with the Protocol Address of Target field in the received ARP packet.

56.2.35 Address0 High Register (MAC_ADDRESS0_HIGH)

The MAC Address0 High register holds the upper 16 bits of the first 6-byte MAC address of the station. The first DA byte that is received on the MII interface corresponds to the LS byte (Bits [24:31]) of the MAC Address Low register. For example, if 0x112233445566 is received (0x11 in lane 0 of the first column) on the MII as the destination address, then the MacAddress0 Register [0:47] is compared with 0x665544332211.

Offset 0x0300 Access: User Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	AE	0	0	0	0	0	0	0	0	0	0	0	0	0	0	DCS
W																
Reset	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	ADDRHI															
W																
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Figure 1014. Address0 High Register (MAC_ADDRESS0_HIGH)

Table 1066. MAC_ADDRESS0_HIGH field descriptions

Field	Description
0 AE	Address Enable This bit is always set to 1.
15 DCS	DMA Channel Select This field contains the DMA Channel number to which the Rx packet whose DA matches the MAC Address0 content is routed. 0 Route RX packet to DMA channel0 1 Route RX packet to DMA channel1
16:31 ADDRHI	MAC Address0[0:15] This field contains the upper 16 bits [0:15] of the first 6-byte MAC address. The MAC uses this field for filtering the received packets and inserting the MAC address in the Transmit Flow Control (Pause) Packets.

56.2.36 Address0 Low Register (MAC_ADDRESS0_LOW)

The MAC Address0 Low register holds the lower 32 bits of the 6-byte first MAC address of the station.



Figure 1015. Address0 Low Register (MAC_ADDRESS0_LOW)

Table 1067. MAC_ADDRESS0_LOW field descriptions

Name	Description
0:31 ADDRLO	MAC Address0[16:47] This field contains the lower 32 bits of the first 6-byte MAC address. The MAC uses this field for filtering the received packets and inserting the MAC address in the Transmit Flow Control (Pause) Packets.

56.2.37 MAC Address High register 1 to 31 (MAC_ADDRESS*n*_HIGH31_1)

The MAC Address1 High register holds the upper 16 bits of the second 6-byte MAC address of the station.

Offset 0x0300 + *n**0x8 (n=1 to 31) Access: User Read/Write

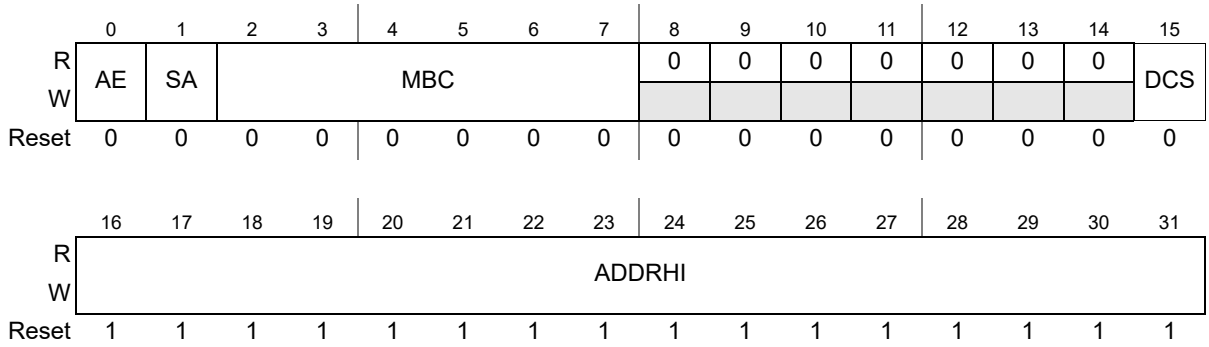


Figure 1016. MAC Address High register 1 to 31 (MAC_ADDRESS*n*_HIGH31_1)

Table 1068. MAC_ADDRESS*n*_HIGH31_1 field descriptions

Field	Description
0 AE	Address Enable When this bit is set, the address filter module uses the second MAC address for perfect filtering. When this bit is reset, the address filter module ignores the address for filtering.
1 SA	Source Address When this bit is set, the MAC Address1[0:47] is used to compare with the SA fields of the received packet. When this bit is reset, the MAC Address1[0:47] is used to compare with the DA fields of the received packet.
2:7 MBC	Mask Byte Control These bits are mask control bits for comparing each of the MAC Address bytes. When set high, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address1 registers. Each bit controls the masking of the bytes as follows: Bit 2: Register 194[16:23] Bit 3: Register 194[24:31] Bit 4: Register 195[0:7] … Bit 7: Register 195[24:31] You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address.
15 DCS	DMA Channel Select This field contains the DMA Channel number to which the Rx packet whose DA matches the MAC Address1 content is routed. 0 Route RX packet to DMA channel0 1 Route RX packet to DMA channel1
16:31 ADDRHI	MAC Address1 [0:15] This field contains the upper 16 bits[0:15] of the second 6-byte MAC address.

56.2.38 MAC Address Low register 1 to 31 (MAC_ADDRESS*n*_LOW31_1)

The MAC Address1 Low register holds the lower 32 bits of the second 6-byte MAC address of the station.

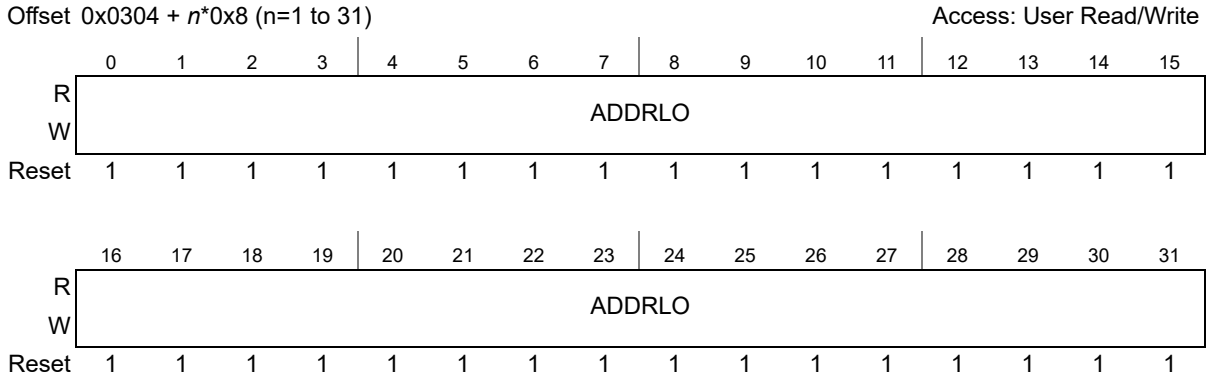


Figure 1017. MAC Address Low register 1 to 31 (MAC_ADDRESS*n*_LOW31_1)

Table 1069. MAC_ADDRESS*n*_LOW31_1 field descriptions

Field	Description
0:31 ADDRLO	MAC Address1 [31:0] This field contains the lower 32 bits of second 6-byte MAC address. The content of this field is undefined until loaded by the application after the initialization process.

56.2.39 MAC Address High register 32 to 63 (MAC_ADDRESS*n*_HIGH63_32)

The MAC Address32 High register holds the upper 16 bits of the 33rd 6-byte MAC address of the station.

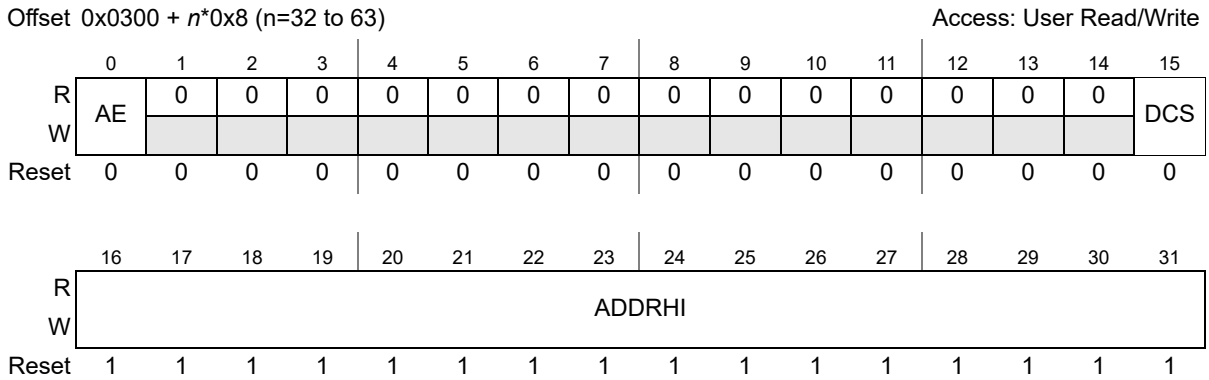


Figure 1018. MAC Address High register 32 to 63 (MAC_ADDRESS*n*_HIGH63_32)

Table 1070. MAC_ADDRESS*n*_HIGH63_32 field descriptions

Field	Description
0 AE	Address Enable When this bit is set, the Address filter module uses the 33rd MAC address for perfect filtering. When this bit is reset, the address filter module ignores the address for filtering.
15 DCS	DMA Channel Select This field contains the DMA Channel number to which an Rx packet whose DA matches the MAC Address32 content is routed. 0 Route RX packet to DMA channel0 1 Route RX packet to DMA channel1
16:31 ADDRHI	MAC Address32 [0:15] This field contains the upper 16 bits (0:15) of the 33rd 6-byte MAC address.

56.2.40 MAC Address Low register 32 to 63 (MAC_ADDRESS*n*_LOW63_32)

The MAC Address32 Low register holds the lower 16 bits of the 33rd 6-byte MAC address of the station.

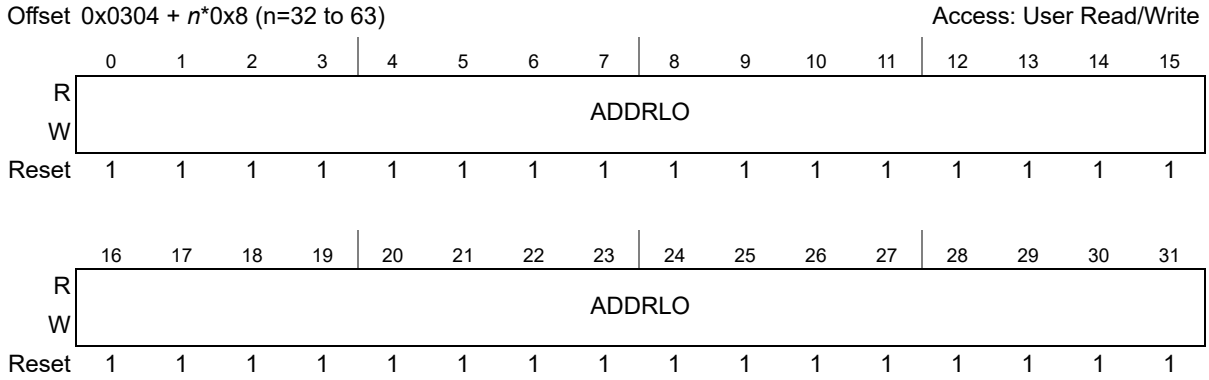


Figure 1019. MAC Address Low register 32 to 63 (MAC_ADDRESS*n*_LOW63_32)

Table 1071. MAC_ADDRESS*n*_LOW63_32 field descriptions

Field	Description
0:31 ADDRLO	MAC Address32 [16:47] This field contains the lower 32 bits of the 33rd 6-byte MAC address. The content of this field is undefined until loaded by the Application after the initialization process

56.2.41 MAC Address High register 64 to 127 (MAC_ADDRESS*n*_HIGH127_64)

The MAC Address32 High register holds the upper 16 bits of the 33rd 6-byte MAC address of the station.

Offset 0x0300 + *n**0x8 (n=64 to 127) Access: User Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	AE	0	0	0	0	0	0	0	0	0	0	0	0	0	0	DCS
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	ADDRHI															
W																
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Figure 1020. MAC Address High register 64 to 127 (MAC_ADDRESS*n*_HIGH127_64)

Table 1072. MAC_ADDRESS*n*_HIGH127_64 field descriptions

Field	Description
0 AE	Address Enable When this bit is set, the Address filter module uses the 33rd MAC address for perfect filtering. When this bit is reset, the address filter module ignores the address for filtering.
15 DCS	DMA Channel Select This field contains the DMA Channel number to which an Rx packet whose DA matches the MAC Address32 content is routed. 0 Route RX packet to DMA channel0 1 Route RX packet to DMA channel1
16:31 ADDRHI	MAC Address32 [0:15] This field contains the upper 16 bits (0:15) of the 33rd 6-byte MAC address.

56.2.42 MAC Address Low register 64 to 127 (MAC_ADDRESS*n*_LOW127_64)

The MAC Address32 Low register holds the lower 16 bits of the 33rd 6-byte MAC address of the station.

Offset 0x0304 + *n**0x8 (n=64 to 127) Access: User Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	ADDRLO															
W																
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	ADDRLO															
W																
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Figure 1021. MAC Address Low register 64 to 127 (MAC_ADDRESS*n*_LOW127_64)

Table 1073. MAC_ADDRESS*n*_LOW127_64 field descriptions

Field	Description
0:31 ADDRLO	MAC Address32 [16:47] This field contains the lower 32 bits of the 33rd 6-byte MAC address. The content of this field is undefined until loaded by the Application after the initialization process

56.2.43 MMC Control Register (MMC_CONTROL)

The MMC Control register establishes the operating mode of the management counters.

Offset 0x0700 Access: User Read/Write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0		0	0						
W								UCDBC			CNTPRSTLVL	CNTPRST	CNTFREEZ	RSTONRD	CNTSTOPRO	CNTRST
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1022. MMC Control Register (MMC_CONTROL)

Table 1074. MMC_CONTROL field descriptions

Field	Description
23 UCDBC	Update MMC Counters for Dropped Broadcast Packets When set, the MAC updates all related MMC Counters for Broadcast packets that are dropped because of the setting of the DBF bit of MAC_Packet_Filter register. When reset, the MMC Counters are not updated for dropped Broadcast packets.
26 CNTPRSTLVL	Full-Half Preset When this bit is low and the CNTPRST bit is set, all MMC counters get preset to almost-half value. All octet counters get preset to 0x7FFF_F800 (Half 2KBytes) and all packet-counters gets preset to 0x7FFF_FFF0 (Half 16). When this bit is high and the CNTPRST bit is set, all MMC counters get preset to almost-full value. All octet counters get preset to 0xFFFF_F800 (Full 2KBytes) and all packet-counters gets preset to 0xFFFF_FFF0 (Full 16).

Table 1074. MMC_CONTROL field descriptions (continued)

Field	Description
27 CNTPRST	Counters Preset When this bit is set, all counters are initialized or preset to almost full or almost half according to the CNTPRSTLVL bit. This bit is cleared automatically after 1 clock cycle. This bit, along with the CNTPRSTLVL bit, is useful for debugging and testing the assertion of interrupts because of MMC counter becoming half-full or full. Access restriction applies. Self-cleared. Setting 0 clears. Setting 1 sets. The CNTRST bit has a higher priority than the CNTPRST bit. Therefore, when the software tries to set both bits in the same write cycle, all counters are cleared and the CNTPRST bit is not set.
28 CNTFREEZ	MMC Counter Freeze When this bit is set, it freezes all MMC counters to their current value. Until this bit is reset to 0, no MMC counter is updated because of any transmitted or received packet. If any MMC counter is read with the Reset on Read bit set, then that counter is also cleared in this mode.
29 RSTONRD	Reset on Read When this bit is set, the MMC counters are reset to zero after Read (self-clearing after reset). The counters are cleared when the least significant byte lane (Bits[0:7]) is read.
30 CNTSTOPRO	Counter Stop Rollover When this bit is set, the counter does not roll over to zero after reaching the maximum value.
31 CNTRST	Counters Reset When this bit is set, all counters are reset. This bit is cleared automatically after 1 clock cycle. Access restriction applies. Self-cleared. Setting 0 clears. Setting 1 sets. The CNTRST bit has a higher priority than the CNTPRST bit. Therefore, when the software tries to set both bits in the same write cycle, all counters are cleared and the CNTPRST bit is not set.

56.2.44 MMC Receive Interrupt Register (MMC_RX_INTERRUPT)

The MMC Receive Interrupt register maintains the interrupts that are generated when the following happens:

- Receive statistic counters reach half of their maximum values, that is 0x8000_0000.
- Receive statistic counters cross their maximum values, that is 0xFFFF_FFFF.
- When the Counter Stop Rollover is set, interrupts are set but the counter remains at all-ones. The MMC Receive Interrupt register is a 32-bit wide register. An interrupt bit is cleared when the respective MMC counter that caused the interrupt is read. The least significant byte lane (Bits[0:7]) of the respective counter must be read in order to clear the interrupt bit.

Offset 0x0704 Access: User Read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	RXLPITRCIS	RXLPIUSCIS	RXCTRLPIS	RXRCVERRPIS	RXWDOGPIS	RXVLANGBPIS	RXFOVPIS	RXPAUSPIS	RXORANGEPIS	RXLENERPIS	RXUCGPIS	RX1024TMAXOCTGBPIS
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	RX512T1023OCTGBPIS	RX256T511OCTGBPIS	RX128T255OCTGBPIS	RX65T127OCTGBPIS	RX64OCTGBPIS	RXOSIZEGPIS	RXUSIZEGPIS	RXJABERPIS	RXRUNTPIS	RXALGNERPIS	RXCRCERPIS	RXMCGPIS	RXBCGPIS	RXGOCTIS	RXGBOCTIS	RXGBPKTIS
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1023. MMC Receive Interrupt Register (MMC_RX_INTERRUPT)

Table 1075. MMC_RX_INTERRUPT field descriptions

Field	Description
4 RXLPITRCIS	MMC Receive LPI Transition Counter Interrupt Status This bit is set when the RX_LPI_TRAN_CNTR counter reaches half of the maximum value or the maximum value. Access restriction applies. Clears on read. Self-set to 1 on internal event.
5 RXLPIUSCIS	MMC Receive LPI Microsecond Counter Interrupt Status This bit is set when the RX_LPI_USEC_CNTR counter reaches half of the maximum value or the maximum value. Access restriction applies. Clears on read. Self-set to 1 on internal event.
6 RXCTRLPIS	MMC Receive Control Packet Counter Interrupt Status This bit is set when the rxctrlpackets_g counter reaches half of the maximum value or the maximum value. Access restriction applies. Clears on read. Self-set to 1 on internal event.

Table 1075. MMC_RX_INTERRUPT field descriptions (continued)

Field	Description
7 RXRCVERRPIS	MMC Receive Error Packet Counter Interrupt Status This bit is set when the rxrcverror counter reaches half of the maximum value or the maximum value. Access restriction applies. Clears on read. Self-set to 1 on internal event.
8 RXWDOGPIS	MMC Receive Watchdog Error Packet Counter Interrupt Status This bit is set when the rxwatchdog error counter reaches half of the maximum value or the maximum value. Access restriction applies. Clears on read. Self-set to 1 on internal event.
9 RXVLANGBPIS	MMC Receive VLAN Good Bad Packet Counter Interrupt Status This bit is set when the rxvlanpackets_gb counter reaches half of the maximum value or the maximum value. Access restriction applies. Clears on read. Self-set to 1 on internal event.
10 RXFOVPIS	MMC Receive FIFO Overflow Packet Counter Interrupt Status This bit is set when the rxfifooverflow counter reaches half of the maximum value or the maximum value. Access restriction applies. Clears on read. Self-set to 1 on internal event.
11 RXPAUSPIS	MMC Receive Pause Packet Counter Interrupt Status This bit is set when the rxpausepackets counter reaches half of the maximum value or the maximum value. Access restriction applies. Clears on read. Self-set to 1 on internal event.
12 RXORANGEPIS	MMC Receive Out Of Range Error Packet Counter Interrupt Status. This bit is set when the rxoutofrangetype counter reaches half of the maximum value or the maximum value. Access restriction applies. Clears on read. Self-set to 1 on internal event.
13 RXLENERPIS	MMC Receive Length Error Packet Counter Interrupt Status This bit is set when the rxlengtherror counter reaches half of the maximum value or the maximum value. Access restriction applies. Clears on read. Self-set to 1 on internal event.
14 RXUCGPIS	MMC Receive Unicast Good Packet Counter Interrupt Status This bit is set when the rxunicastpackets_g counter reaches half of the maximum value or the maximum value. Access restriction applies. Clears on read. Self-set to 1 on internal event.
15 RX1024TMAXOCTGBPIS	MMC Receive 1024 to Maximum Octet Good Bad Packet Counter Interrupt Status This bit is set when the rx1024tomaxoctets_gb counter reaches half of the maximum value or the maximum value. Access restriction applies. Clears on read. Self-set to 1 on internal event.

Table 1075. MMC_RX_INTERRUPT field descriptions (continued)

Field	Description
16 RX512T1023OCTGBPIS	MMC Receive 512 to 1023 Octet Good Bad Packet Counter Interrupt Status This bit is set when the rx512to1023octets_gb counter reaches half of the maximum value or the maximum value. Access restriction applies. Clears on read. Self-set to 1 on internal event.
17 RX256T511OCTGBPIS	MMC Receive 256 to 511 Octet Good Bad Packet Counter Interrupt Status This bit is set when the rx256to511octets_gb counter reaches half of the maximum value or the maximum value. Access restriction applies. Clears on read. Self-set to 1 on internal event.
18 RX128T255OCTGBPIS	MMC Receive 128 to 255 Octet Good Bad Packet Counter Interrupt Status This bit is set when the rx128to255octets_gb counter reaches half of the maximum value or the maximum value. Access restriction applies. Clears on read. Self-set to 1 on internal event.
19 RX65T127OCTGBPIS	MMC Receive 65 to 127 Octet Good Bad Packet Counter Interrupt Status This bit is set when the rx65to127octets_gb counter reaches half of the maximum value or the maximum value. Access restriction applies. Clears on read. Self-set to 1 on internal event.
20 RX64OCTGBPIS	MMC Receive 64 Octet Good Bad Packet Counter Interrupt Status This bit is set when the rx64octets_gb counter reaches half of the maximum value or the maximum value. Access restriction applies. Clears on read. Self-set to 1 on internal event.
21 RXOSIZEGPIS	MMC Receive Oversize Good Packet Counter Interrupt Status This bit is set when the rxoversize_g counter reaches half of the maximum value or the maximum value. Access restriction applies. Clears on read. Self-set to 1 on internal event.
22 RXUSIZEGPIS	MMC Receive Undersize Good Packet Counter Interrupt Status This bit is set when the rxundersize_g counter reaches half of the maximum value or the maximum value. Access restriction applies. Clears on read. Self-set to 1 on internal event.
23 RXJABERPIS	MMC Receive Jabber Error Packet Counter Interrupt Status This bit is set when the rxjabbererror counter reaches half of the maximum value or the maximum value. Access restriction applies. Clears on read. Self-set to 1 on internal event.

Table 1075. MMC_RX_INTERRUPT field descriptions (continued)

Field	Description
24 RXRUNTPIS	MMC Receive Runt Packet Counter Interrupt Status This bit is set when the rxrunterror counter reaches half of the maximum value or the maximum value. Access restriction applies. Clears on read. Self-set to 1 on internal event.
25 RXALGNERPIS	MMC Receive Alignment Error Packet Counter Interrupt Status This bit is set when the rxalignmenterror counter reaches half of the maximum value or the maximum value. Access restriction applies. Clears on read. Self-set to 1 on internal event.
26 RXCRCERPIS	MMC Receive CRC Error Packet Counter Interrupt Status This bit is set when the rxcrcerror counter reaches half of the maximum value or the maximum value. Access restriction applies. Clears on read. Self-set to 1 on internal event.
27 RXMCGPIS	MMC Receive Multicast Good Packet Counter Interrupt Status This bit is set when the rxmulticastpackets_g counter reaches half of the maximum value or the maximum value. Access restriction applies. Clears on read. Self-set to 1 on internal event.
28 RXBCGPIS	MMC Receive Broadcast Good Packet Counter Interrupt Status This bit is set when the rxbroadcastpackets_g counter reaches half of the maximum value or the maximum value. Access restriction applies. Clears on read. Self-set to 1 on internal event.
29 RXGOCTIS	MMC Receive Good Octet Counter Interrupt Status This bit is set when the rxoctetcount_g counter reaches half of the maximum value or the maximum value. Access restriction applies. Clears on read. Self-set to 1 on internal event.
30 RXGBOCTIS	MMC Receive Good Bad Octet Counter Interrupt Status This bit is set when the rxoctetcount_gb counter reaches half of the maximum value or the maximum value. Access restriction applies. Clears on read. Self-set to 1 on internal event.
31 RXGBPKTIS	MMC Receive Good Bad Packet Counter Interrupt Status This bit is set when the rxpacketcount_gb counter reaches half of the maximum value or the maximum value. Access restriction applies. Clears on read. Self-set to 1 on internal event.

56.2.45 MMC Tx Interrupt Register (MMC_TX_INTERRUPT)

The MMC Transmit Interrupt register maintains the interrupts generated when transmit statistic counters reach half their maximum values, that is 0x8000_0000, and when they cross their maximum values, that is 0xFFFF_FFFF. When Counter Stop Rollover is set, then

interrupts are set but the counter remains at all-ones. The MMC Transmit Interrupt register is a 32-bit wide register. An interrupt bit is cleared when the respective MMC counter that caused the interrupt is read. The least significant byte lane (Bits[0:7]) of the respective counter must be read in order to clear the interrupt bit.

Offset 0x0708 Access: User Read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	TXLPITRCIS	TXLPIUSCIS	TXOSIZEGPIS	TXVLANGPIS	TXPAUSPIS	TXEXDEFPIS	TXGPKTIS	TXGOCTIS	TXCARERPIS	TXEXCOLPIS	TXLATCOLPIS	TXDEFPIS
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	TXMCOLGPIS	TXSCOLGPIS	TXUFLOWERPIS	TXBCGBPIS	TXMCGBPIS	TXUCGBPIS	TX1024TMAXOCTGBPIS	TX512T1023OCTGBPIS	TX256T511OCTGBPIS	TX128T255OCTGBPIS	TX65T127OCTGBPIS	TX64OCTGBPIS	TXMCGPIS	TXBCGPIS	TXGBPKTIS	TXGBOCTIS
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1024. MMC Tx Interrupt Register (MMC_TX_INTERRUPT)

Table 1076. MMC_TX_INTERRUPT field descriptions

Field	Description
4 TXLPITRCIS	MMC Transmit LPI Transition Counter Interrupt Status This bit is set when the TX_LPI_TRAN_CNTR counter reaches half of the maximum value or the maximum value. Access restriction applies. Clears on read. Self-set to 1 on internal event.
5 TXLPIUSCIS	MMC Transmit LPI Microsecond Counter Interrupt Status This bit is set when the TX_LPI_USEC_CNTR counter reaches half of the maximum value or the maximum value. Access restriction applies. Clears on read. Self-set to 1 on internal event.
6 TXOSIZEGPIS	MMC Transmit Oversize Good Packet Counter Interrupt Status This bit is set when the txoversize_g counter reaches half of the maximum value or the maximum value. Access restriction applies. Clears on read. Self-set to 1 on internal event.

Table 1076. MMC_TX_INTERRUPT field descriptions (continued)

Field	Description
7 TXVLANGPIS	MMC Transmit VLAN Good Packet Counter Interrupt Status This bit is set when the txvlanpackets_g counter reaches half of the maximum value or the maximum value. Access restriction applies. Clears on read. Self-set to 1 on internal event.
8 TXPAUSPIS	MMC Transmit Pause Packet Counter Interrupt Status This bit is set when the txpausepacketserror counter reaches half of the maximum value or the maximum value. Access restriction applies. Clears on read. Self-set to 1 on internal event.
9 TXEXDEFPIS	MMC Transmit Excessive Deferral Packet Counter Interrupt Status This bit is set when the txexcessdef counter reaches half of the maximum value or the maximum value. Access restriction applies. Clears on read. Self-set to 1 on internal event.
10 TXGPKTIS	MMC Transmit Good Packet Counter Interrupt Status This bit is set when the txpacketcount_g counter reaches half of the maximum value or the maximum value. Access restriction applies. Clears on read. Self-set to 1 on internal event.
11 TXGOCTIS	MMC Transmit Good Octet Counter Interrupt Status This bit is set when the txoctetcount_g counter reaches half of the maximum value or the maximum value. Access restriction applies. Clears on read. Self-set to 1 on internal event.
12 TXCARERPIS	MMC Transmit Carrier Error Packet Counter Interrupt Status This bit is set when the txcarriererror counter reaches half of the maximum value or the maximum value. Access restriction applies. Clears on read. Self-set to 1 on internal event.
13 TXEXCOLPIS	MMC Transmit Excessive Collision Packet Counter Interrupt Status This bit is set when the txexesscol counter reaches half of the maximum value or the maximum value. Access restriction applies. Clears on read. Self-set to 1 on internal event.
14 TXLATCOLPIS	MMC Transmit Late Collision Packet Counter Interrupt Status This bit is set when the txlatecol counter reaches half of the maximum value or the maximum value. Access restriction applies. Clears on read. Self-set to 1 on internal event.
15 TXDEFPIS	MMC Transmit Deferred Packet Counter Interrupt Status This bit is set when the txdeferred counter reaches half of the maximum value or the maximum value. Access restriction applies. Clears on read. Self-set to 1 on internal event.

Table 1076. MMC_TX_INTERRUPT field descriptions (continued)

Field	Description
16 TXMCOLGPIS	MMC Transmit Multiple Collision Good Packet Counter Interrupt Status This bit is set when the txmulticol_g counter reaches half of the maximum value or the maximum value. Access restriction applies. Clears on read. Self-set to 1 on internal event.
17 TXSCOLGPIS	MMC Transmit Single Collision Good Packet Counter Interrupt Status This bit is set when the txsinglecol_g counter reaches half of the maximum value or the maximum value. Access restriction applies. Clears on read. Self-set to 1 on internal event.
18 TXUFLOWERPIS	MMC Transmit Underflow Error Packet Counter Interrupt Status This bit is set when the txunderflowerror counter reaches half of the maximum value or the maximum value. Access restriction applies. Clears on read. Self-set to 1 on internal event.
19 TXBCGBPIS	MMC Transmit Broadcast Good Bad Packet Counter Interrupt Status This bit is set when the txbroadcastpackets_gb counter reaches half of the maximum value or the maximum value. Access restriction applies. Clears on read. Self-set to 1 on internal event.
20 TXMCGBPIS	MMC Transmit Multicast Good Bad Packet Counter Interrupt Status The bit is set when the txmulticastpackets_gb counter reaches half of the maximum value or the maximum value. Access restriction applies. Clears on read. Self-set to 1 on internal event.
21 TXUCGBPIS	MMC Transmit Unicast Good Bad Packet Counter Interrupt Status This bit is set when the txunicastpackets_gb counter reaches half of the maximum value or the maximum value. Access restriction applies. Clears on read. Self-set to 1 on internal event.
22 TX1024TMAXOCTGBPIS	MMC Transmit 1024 to Maximum Octet Good Bad Packet Counter Interrupt Status This bit is set when the tx1024tomaxoctets_gb counter reaches half of the maximum value or the maximum value. Access restriction applies. Clears on read. Self-set to 1 on internal event.
23 TX512T1023OCTGBPIS	MMC Transmit 512 to 1023 Octet Good Bad Packet Counter Interrupt Status This bit is set when the tx512to1023octets_gb counter reaches half of the maximum value or the maximum value. Access restriction applies. Clears on read. Self-set to 1 on internal event.

Table 1076. MMC_TX_INTERRUPT field descriptions (continued)

Field	Description
24 TX256T511OCTGBPIS	MMC Transmit 256 to 511 Octet Good Bad Packet Counter Interrupt Status This bit is set when the tx256to511octets_gb counter reaches half of the maximum value or the maximum value. Access restriction applies. Clears on read. Self-set to 1 on internal event.
25 TX128T255OCTGBPIS	MMC Transmit 128 to 255 Octet Good Bad Packet Counter Interrupt Status This bit is set when the tx128to255octets_gb counter reaches half of the maximum value or the maximum value. Access restriction applies. Clears on read. Self-set to 1 on internal event.
26 TX65T127OCTGBPIS	MMC Transmit 65 to 127 Octet Good Bad Packet Counter Interrupt Status This bit is set when the tx65to127octets_gb counter reaches half the maximum value, and also when it reaches the maximum value. Access restriction applies. Clears on read. Self-set to 1 on internal event.
27 TX64OCTGBPIS	MMC Transmit 64 Octet Good Bad Packet Counter Interrupt Status This bit is set when the tx64octets_gb counter reaches half of the maximum value or the maximum value. Access restriction applies. Clears on read. Self-set to 1 on internal event.
28 TXMCGPIS	MMC Transmit Multicast Good Packet Counter Interrupt Status This bit is set when the txmulticastpackets_g counter reaches half of the maximum value or the maximum value. Access restriction applies. Clears on read. Self-set to 1 on internal event.
29 TXBCGPIS	MMC Transmit Broadcast Good Packet Counter Interrupt Status This bit is set when the txbroadcastpackets_g counter reaches half of the maximum value or the maximum value. Access restriction applies. Clears on read. Self-set to 1 on internal event.
30 TXGBPKTIS	MMC Transmit Good Bad Packet Counter Interrupt Status This bit is set when the txpacketcount_gb counter reaches half of the maximum value or the maximum value. Access restriction applies. Clears on read. Self-set to 1 on internal event.
31 TXGBOCTIS	MMC Transmit Good Bad Octet Counter Interrupt Status This bit is set when the txoctetcount_gb counter reaches half of the maximum value or the maximum value. Access restriction applies. Clears on read. Self-set to 1 on internal event.

56.2.46 MMC Rx Interrupt mask Register (MMC_RX_INTERRUPT_MASK)

This register maintains the masks for interrupts generated from all Receive statistics counters.

Offset 0x070C

Access: Software read/write

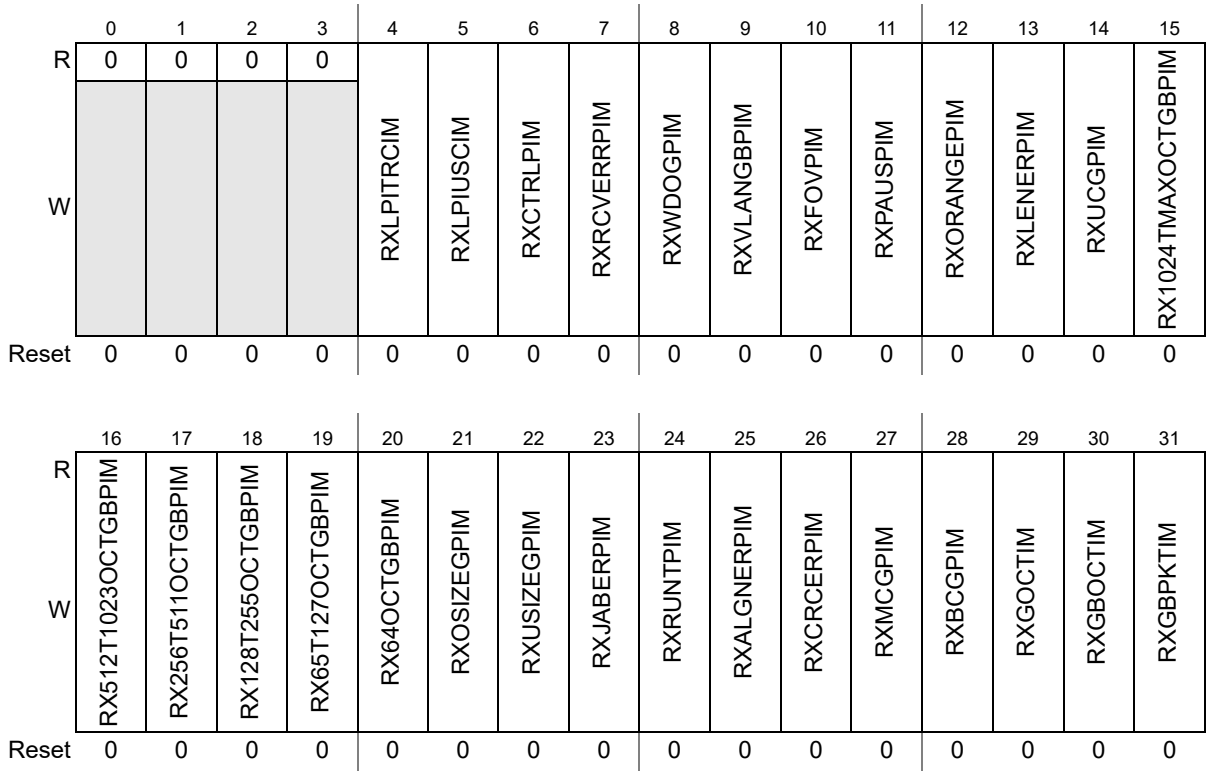


Figure 1025. MMC Rx Interrupt mask Register (MMC_RX_INTERRUPT_MASK)

Table 1077. MMC_RX_INTERRUPT_MASK field descriptions

Field	Description
4 RXLPITRCIM	MMC Receive LPI Transition Counter Interrupt Mask Setting this bit masks the interrupt when RX_LPI_TRAN_CNTR reaches half the maximum value or the maximum value.
5 RXLPIUSCIM	MMC Receive LPI Microsecond Counter Interrupt Mask Setting this bit masks the interrupt when RX_LPI_USEC_CNTR reaches half the maximum value or the maximum value.
6 RXCTRLPIM	MMC Receive Control Packet Counter Interrupt Mask Setting this bit masks the interrupt when the rxctrlpackets_g counter reaches half of the maximum value or the maximum value.
7 RXRCVERRPIM	MMC Receive Error Packet Counter Interrupt Mask Setting this bit masks the interrupt when the rxrcverror counter reaches half of the maximum value or the maximum value.

Table 1077. MMC_RX_INTERRUPT_MASK field descriptions (continued)

Field	Description
8 RXWDOGPIM	MMC Receive Watchdog Error Packet Counter Interrupt Mask Setting this bit masks the interrupt when the rxwatchdog counter reaches half of the maximum value or the maximum value.
9 RXVLANGBPIM	MMC Receive VLAN Good Bad Packet Counter Interrupt Mask Setting this bit masks the interrupt when the rxvlanpackets_gb counter reaches half of the maximum value or the maximum value.
10 RXFOVPIM	MMC Receive FIFO Overflow Packet Counter Interrupt Mask Setting this bit masks the interrupt when the rxfifooverflow counter reaches half of the maximum value or the maximum value.
11 RXPAUSPIM	MMC Receive Pause Packet Counter Interrupt Mask Setting this bit masks the interrupt when the rxpausepackets counter reaches half of the maximum value or the maximum value.
12 RXORANGEPIM	MMC Receive Out Of Range Error Packet Counter Interrupt Mask Setting this bit masks the interrupt when the rxoutofrangetype counter reaches half of the maximum value or the maximum value.
13 RXLENERPIM	MMC Receive Length Error Packet Counter Interrupt Mask Setting this bit masks the interrupt when the rxlengtherror counter reaches half of the maximum value or the maximum value.
14 RXUCGPIM	MMC Receive Unicast Good Packet Counter Interrupt Mask Setting this bit masks the interrupt when the rxunicastpackets_g counter reaches half of the maximum value or the maximum value.
15 RX1024TMAXOCTGBPIM	MMC Receive 1024 to Maximum Octet Good Bad Packet Counter Interrupt Mask. Setting this bit masks the interrupt when the rx1024tomaxoctets_gb counter reaches half of the maximum value or the maximum value.
16 RX512T1023OCTGBPIM	MMC Receive 512 to 1023 Octet Good Bad Packet Counter Interrupt Mask Setting this bit masks the interrupt when the rx512to1023octets_gb counter reaches half of the maximum value or the maximum value.
17 RX256T511OCTGBPIM	MMC Receive 256 to 511 Octet Good Bad Packet Counter Interrupt Mask Setting this bit masks the interrupt when the rx256to511octets_gb counter reaches half of the maximum value or the maximum value.
18 RX128T255OCTGBPIM	MMC Receive 128 to 255 Octet Good Bad Packet Counter Interrupt Mask Setting this bit masks the interrupt when the rx128to255octets_gb counter reaches half of the maximum value or the maximum value.
19 RX65T127OCTGBPIM	MMC Receive 65 to 127 Octet Good Bad Packet Counter Interrupt Mask Setting this bit masks the interrupt when the rx65to127octets_gb counter reaches half of the maximum value or the maximum value.
20 RX64OCTGBPIM	MMC Receive 64 Octet Good Bad Packet Counter Interrupt Mask Setting this bit masks the interrupt when the rx64octets_gb counter reaches half of the maximum value or the maximum value.

Table 1077. MMC_RX_INTERRUPT_MASK field descriptions (continued)

Field	Description
21 RXOSIZEGPIM	MMC Receive Oversize Good Packet Counter Interrupt Mask Setting this bit masks the interrupt when the rxoversize_g counter reaches half of the maximum value or the maximum value.
22 RXUSIZEGPIM	MMC Receive Undersize Good Packet Counter Interrupt Mask Setting this bit masks the interrupt when the rxundersize_g counter reaches half of the maximum value or the maximum value.
23 RXJABERPIM	MMC Receive Jabber Error Packet Counter Interrupt Mask Setting this bit masks the interrupt when the rxjabbererror counter reaches half of the maximum value or the maximum value.
24 RXRUNTPIM	MMC Receive Runt Packet Counter Interrupt Mask Setting this bit masks the interrupt when the rxrunterror counter reaches half of the maximum value or the maximum value.
25 RXALGNERPIM	MMC Receive Alignment Error Packet Counter Interrupt Mask Setting this bit masks the interrupt when the rxalignmenterror counter reaches half of the maximum value or the maximum value.
26 RXCRCERPIM	MMC Receive CRC Error Packet Counter Interrupt Mask Setting this bit masks the interrupt when the rxcrcerror counter reaches half of the maximum value or the maximum value.
27 RXMCGPIM	MMC Receive Multicast Good Packet Counter Interrupt Mask Setting this bit masks the interrupt when the rxmulticastpackets_g counter reaches half of the maximum value or the maximum value.
28 RXBCGPIM	MMC Receive Broadcast Good Packet Counter Interrupt Mask Setting this bit masks the interrupt when the rxbroadcastpackets_g counter reaches half of the maximum value or the maximum value.
29 RXGOCTIM	MMC Receive Good Octet Counter Interrupt Mask Setting this bit masks the interrupt when the rxoctetcount_g counter reaches half of the maximum value or the maximum value.
30 RXGBOCTIM	MMC Receive Good Bad Octet Counter Interrupt Mask Setting this bit masks the interrupt when the rxoctetcount_gb counter reaches half of the maximum value or the maximum value.
31 RXGBPKTIM	MMC Receive Good Bad Packet Counter Interrupt Mask Setting this bit masks the interrupt when the rxpacketcount_gb counter reaches half of the maximum value or the maximum value.

56.2.47 MMC Tx Interrupt Mask Register (MMC_TX_INTERRUPT_MASK)

This register maintains the masks for interrupts generated from all Transmit statistics counters.

Offset 0x0710

Access: Software read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	TXLPITRCIM	TXLPIUSCIM	TXOSIZEGPIM	TXVLANGPIM	TXPAUSPIM	TXEXDEFPIM	TXGPKTIM	TXGOCTIM	TXCARERPIM	TXEXCOLPIM	TXLATCOLPIM	TXDEFPIM
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	TXMCOLGPIM	TXSCOLGPIM	TXUFLOWERPIM	TXBCGBPIM	TXMCGBPIM	TXUCGBPIM	TX1024TMAXOCTGBPIM	TX512T1023OCTGBPIM	TX256T511OCTGBPIM	TX128T255OCTGBPIM	TX65T127OCTGBPIM	TX64OCTGBPIM	TXMCGPIM	TXBCGPIM	TXGBPKTIM	TXGBOCTIM
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1026. MMC Tx Interrupt Mask Register (MMC_TX_INTERRUPT_MASK)

Table 1078. MMC_TX_INTERRUPT_MASK field descriptions

Field	Description
4 TXLPITRCIM	MMC Transmit LPI Transition Counter Interrupt Mask Setting this bit masks the interrupt when TX_LPI_TRAN_CNTR reaches half the maximum value or the maximum value.
5 TXLPIUSCIM	MMC Transmit LPI Microsecond Counter Interrupt Mask Setting this bit masks the interrupt when TX_LPI_USEC_CNTR reaches half the maximum value or the maximum value.
6 TXOSIZEGPIM	MMC Transmit Oversize Good Packet Counter Interrupt Mask Setting this bit masks the interrupt when the txoversize_g counter reaches half of the maximum value or the maximum value.
7 TXVLANGPIM	MMC Transmit VLAN Good Packet Counter Interrupt Mask Setting this bit masks the interrupt when the txvlanpackets_g counter reaches half of the maximum value or the maximum value.
8 TXPAUSPIM	MMC Transmit Pause Packet Counter Interrupt Mask Setting this bit masks the interrupt when the txpausepackets counter reaches half of the maximum value or the maximum value.
9 TXEXDEFPIM	MMC Transmit Excessive Deferral Packet Counter Interrupt Mask Setting this bit masks the interrupt when the txexcessdef counter reaches half of the maximum value or the maximum value.
10 TXGPKTIM	MMC Transmit Good Packet Counter Interrupt Mask Setting this bit masks the interrupt when the txpacketcount_g counter reaches half of the maximum value or the maximum value.

Table 1078. MMC_TX_INTERRUPT_MASK field descriptions (continued)

Field	Description
11 TXGOCTIM	MMC Transmit Good Octet Counter Interrupt Mask Setting this bit masks the interrupt when the txoctetcount_g counter reaches half of the maximum value or the maximum value.
12 TXCARERPIM	MMC Transmit Carrier Error Packet Counter Interrupt Mask Setting this bit masks the interrupt when the txcarriererror counter reaches half of the maximum value or the maximum value.
13 TXEXCOLPIM	MMC Transmit Excessive Collision Packet Counter Interrupt Mask Setting this bit masks the interrupt when the txexcesscol counter reaches half of the maximum value or the maximum value.
14 TXLATCOLPIM	MMC Transmit Late Collision Packet Counter Interrupt Mask Setting this bit masks the interrupt when the txlatecol counter reaches half of the maximum value or the maximum value.
15 TXDEFPIM	MMC Transmit Deferred Packet Counter Interrupt Mask Setting this bit masks the interrupt when the txdeferred counter reaches half of the maximum value or the maximum value.
16 TXMCOLGPIM	MMC Transmit Multiple Collision Good Packet Counter Interrupt Mask Setting this bit masks the interrupt when the txmulticol_g counter reaches half of the maximum value or the maximum value.
17 TXSCOLGPIM	MMC Transmit Single Collision Good Packet Counter Interrupt Mask Setting this bit masks the interrupt when the txsinglecol_g counter reaches half of the maximum value or the maximum value.
18 TXUFLOWERPIM	MMC Transmit Underflow Error Packet Counter Interrupt Mask Setting this bit masks the interrupt when the txunderflowerror counter reaches half of the maximum value or the maximum value.
19 TXBCGBPIM	MMC Transmit Broadcast Good Bad Packet Counter Interrupt Mask Setting this bit masks the interrupt when the txbroadcastpackets_gb counter reaches half of the maximum value or the maximum value.
20 TXMCGBPIM	MMC Transmit Multicast Good Bad Packet Counter Interrupt Mask Setting this bit masks the interrupt when the txmulticastpackets_gb counter reaches half of the maximum value or the maximum value.
21 TXUCGBPIM	MMC Transmit Unicast Good Bad Packet Counter Interrupt Mask Setting this bit masks the interrupt when the txunicastpackets_gb counter reaches half of the maximum value or the maximum value.
22 TX1024TMAXOCTGBPIM	MMC Transmit 1024 to Maximum Octet Good Bad Packet Counter Interrupt Mask Setting this bit masks the interrupt when the tx1024tomaxoctets_gb counter reaches half of the maximum value or the maximum value.
23 TX512T1023OCTGBPIM	MMC Transmit 512 to 1023 Octet Good Bad Packet Counter Interrupt Mask Setting this bit masks the interrupt when the tx512to1023octets_gb counter reaches half of the maximum value or the maximum value.

Table 1078. MMC_TX_INTERRUPT_MASK field descriptions (continued)

Field	Description
24 TX256T511OCTGBPIM	MMC Transmit 256 to 511 Octet Good Bad Packet Counter Interrupt Mask Setting this bit masks the interrupt when the tx256to511octets_gb counter reaches half of the maximum value or the maximum value.
25 TX128T255OCTGBPIM	MMC Transmit 128 to 255 Octet Good Bad Packet Counter Interrupt Mask Setting this bit masks the interrupt when the tx128to255octets_gb counter reaches half of the maximum value or the maximum value.
26 TX65T127OCTGBPIM	MMC Transmit 65 to 127 Octet Good Bad Packet Counter Interrupt Mask Setting this bit masks the interrupt when the tx65to127octets_gb counter reaches half of the maximum value or the maximum value.
27 TX64OCTGBPIM	MMC Transmit 64 Octet Good Bad Packet Counter Interrupt Mask Setting this bit masks the interrupt when the tx64octets_gb counter reaches half of the maximum value or the maximum value.
28 TXMCGPIM	MMC Transmit Multicast Good Packet Counter Interrupt Mask Setting this bit masks the interrupt when the txmulticastpackets_g counter reaches half of the maximum value or the maximum value.
29 TXBCGPIM	MMC Transmit Broadcast Good Packet Counter Interrupt Mask Setting this bit masks the interrupt when the txbroadcastpackets_g counter reaches half of the maximum value or the maximum value.
30 TXGBPKTIM	MMC Transmit Good Bad Packet Counter Interrupt Mask Setting this bit masks the interrupt when the txpacketcount_gb counter reaches half of the maximum value or the maximum value.
31 TXGBOCTIM	MMC Transmit Good Bad Octet Counter Interrupt Mask Setting this bit masks the interrupt when the txoctetcount_gb counter reaches half of the maximum value or the maximum value.

56.2.48 Transmit Octet Count Good Bad Register (TX_OCTET_COUNT_GOOD_BAD)

This register provides the number of bytes transmitted by the Ethernet module, exclusive of preamble and retried bytes, in good and bad packets.

Offset 0x0714 Access: User Read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	TXOCTGB															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	TXOCTGB															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1027. Transmit Octet Count Good Bad Register (TX_OCTET_COUNT_GOOD_BAD)

Table 1079. TX_OCTET_COUNT_GOOD_BAD field descriptions

Field	Description
0:31 TXOCTGB	This field indicates the number of bytes transmitted, exclusive of preamble and retried bytes, in good and bad packets.

56.2.49 Transmit Packet Count Good Bad Register (TX_PACKET_COUNT_GOOD_BAD)

This register provides the number of good and bad packets transmitted by the Ethernet module, exclusive of retried packets.

Offset 0x0718 Access: User Read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	TXPKTGB															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	TXPKTGB															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1028. Transmit Packet Count Good Bad Register (TX_PACKET_COUNT_GOOD_BAD)

Table 1080. TX_PACKET_COUNT_GOOD_BAD field descriptions

Field	Description
0:31 TXPKTGB	This field indicates the number of good and bad packets transmitted, exclusive of retried packets.

56.2.50 Transmit Broadcast Packets Good Register (TX_BROADCAST_PACKETS_GOOD)

This register provides the number of good broadcast packets transmitted by the Ethernet module.

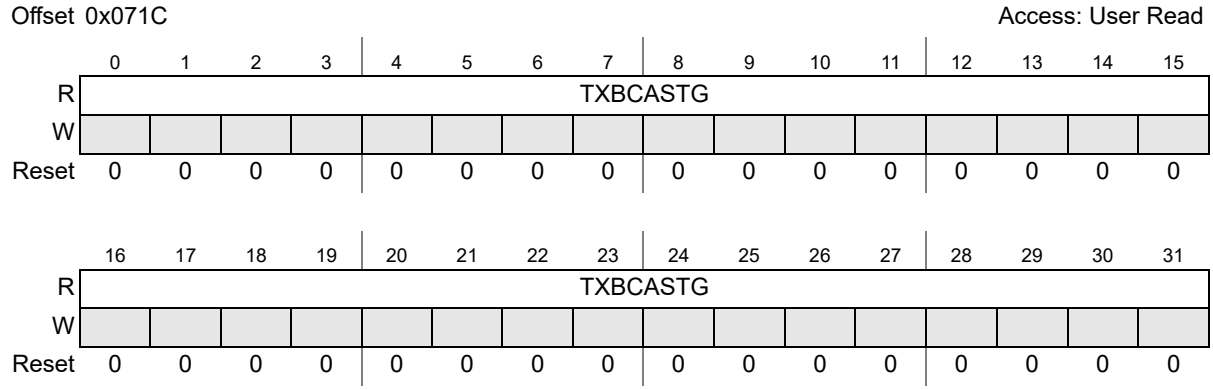


Figure 1029. Transmit Broadcast Packets Good Register (TX_BROADCAST_PACKETS_GOOD)

Table 1081. TX_BROADCAST_PACKETS_GOOD field descriptions

Field	Description
0:31 TXBCASTG	This field indicates the number of good broadcast packets transmitted.

56.2.51 Transmit Multicast Packets Good Register (TX_MULTICAST_PACKETS_GOOD)

This register provides the number of good multicast packets transmitted by the Ethernet module.

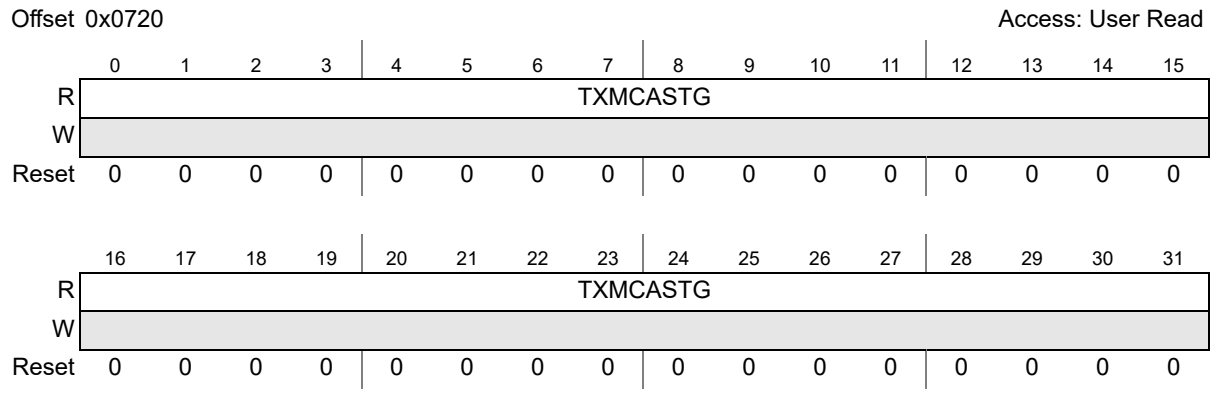


Figure 1030. Transmit Multicast Packets Good Register (TX_MULTICAST_PACKETS_GOOD)

Table 1082. TX_MULTICAST_PACKETS_GOOD field descriptions

Field	Description
0:31 TXMCASTG	This field indicates the number of good multicast packets transmitted.

56.2.52 Transmit 64Octets Packets Good Bad Register (TX_64OCTETS_PACKETS_GOOD_BAD)

This register provides the number of good and bad packets transmitted by the Ethernet module with length 64 bytes, exclusive of preamble and retried packets.

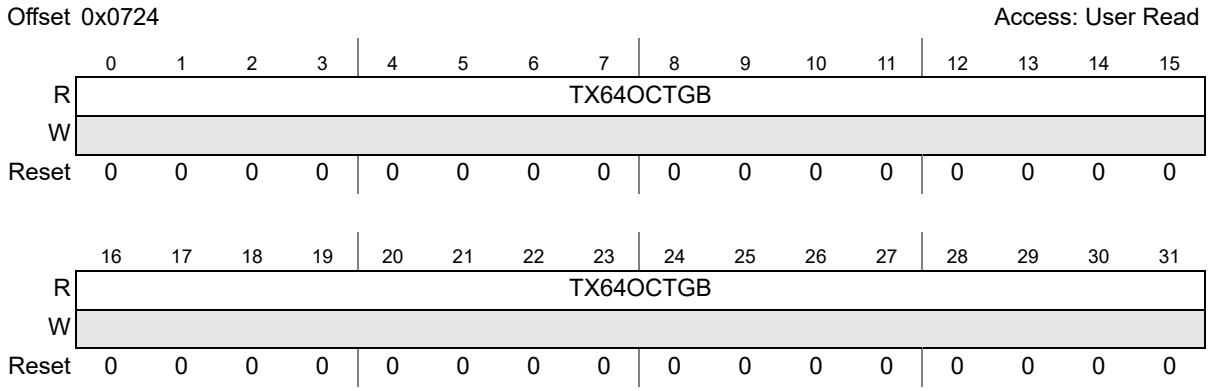


Figure 1031. Transmit 64Octets Packets Good Bad Register (TX_64OCTETS_PACKETS_GOOD_BAD)

Table 1083. TX_64OCTETS_PACKETS_GOOD_BAD field descriptions

Field	Description
0:31 TX64OCTGB	This field indicates the number of good and bad packets transmitted with length 64 bytes, exclusive of preamble and retried packets.

56.2.53 Transmit 65to127Octets Packets Good Bad Register (TX_65TO127OCTETS_PACKETS_GOOD_BAD)

This register provides the number of good and bad packets transmitted by the Ethernet module with length between 65 and 127 (inclusive) bytes, exclusive of preamble and retried packets.

Offset 0x0728 Access: User Read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	TX65_127OCTGB															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	TX65_127OCTGB															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1032. Transmit 65to127Octets Packets Good Bad Register (TX_65TO127OCTETS_PACKETS_GOOD_BAD)

Table 1084. TX_65TO127OCTETS_PACKETS_GOOD_BAD field descriptions

Field	Description
0:31 TX65_127OCTGB	This field indicates the number of good and bad packets transmitted with length between 65 and 127 (inclusive) bytes, exclusive of preamble and retried packets.

56.2.54 Transmit 128To255Octets Packets Good Bad Register (TX_128TO255OCTETS_PACKETS_GOOD_BAD)

This register provides the number of good and bad packets transmitted by the Ethernet module with length between 128 to 255 (inclusive) bytes, exclusive of preamble and retried packets.

Offset 0x072C Access: User Read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	TX128_255OCTGB															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	TX128_255OCTGB															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1033. Transmit 128To255Octets Packets Good Bad Register (TX_128TO255OCTETS_PACKETS_GOOD_BAD)

Table 1085. TX_128TO255OCTETS_PACKETS_GOOD_BAD field descriptions

Field	Description
0:31 TX128_255OCTGB	This field indicates the number of good and bad packets transmitted with length between 128 and 255 (inclusive) bytes, exclusive of preamble and retried packets.

56.2.55 Transmit 256To511Octets Packets Good Bad Register (TX_256TO511OCTETS_PACKETS_GOOD_BAD)

This register provides the number of good and bad packets transmitted by the Ethernet module with length between 256 to 511 (inclusive) bytes, exclusive of preamble and retried packets.

Offset 0x0730 Access: User Read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	TX256_511OCTGB															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	TX256_511OCTGB															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1034. Transmit 256To511Octets Packets Good Bad Register (TX_256TO511OCTETS_PACKETS_GOOD_BAD)

Table 1086. TX_256TO511OCTETS_PACKETS_GOOD_BAD field descriptions

Field	Description
0:31 TX256_511OCTGB	This field indicates the number of good and bad packets transmitted with length between 256 and 511 (inclusive) bytes, exclusive of preamble and retried packets.

56.2.56 Transmit 512To1023Octets Packets Good Bad Register (TX_512TO1023OCTETS_PACKETS_GOOD_BAD)

This register provides the number of good and bad packets transmitted by the Ethernet module with length 512 to 1023 (inclusive) bytes, exclusive of preamble and retried packets.

Offset 0x0734 Access: User Read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	TX512_1023OCTGB															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	TX512_1023OCTGB															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1035. Transmit 512To1023Octets Packets Good Bad Register (TX_512TO1023OCTETS_PACKETS_GOOD_BAD)

Table 1087. TX_512TO1023OCTETS_PACKETS_GOOD_BAD field descriptions

Field	Description
0:31 TX512_1023OCTGB	This field indicates the number of good and bad packets transmitted with length between 512 and 1023 (inclusive) bytes, exclusive of preamble and retried packets.

56.2.57 Transmit 1024ToMaxOctets Packets Good Bad Register (TX_1024TOMAXOCTETS_PACKETS_GOOD_BAD)

This register provides the number of good and bad packets transmitted by the Ethernet module with length 1024 to maxsize (inclusive) bytes, exclusive of preamble and retried packets.

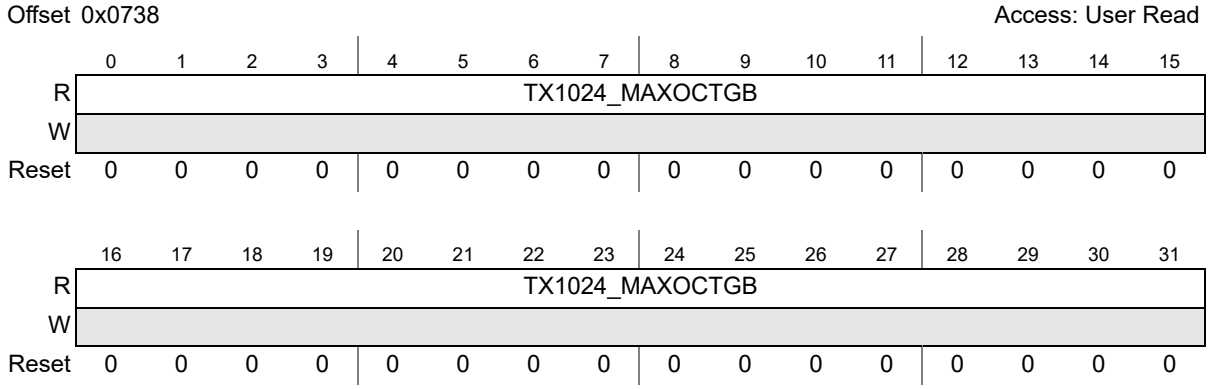


Figure 1036. Transmit 1024ToMaxOctets Packets Good Bad Register (TX_1024TOMAXOCTETS_PACKETS_GOOD_BAD)

Table 1088. TX_1024TOMAXOCTETS_PACKETS_GOOD_BAD field descriptions

Field	Description
0:31 TX1024_MAXOCTGB	This field indicates the number of good and bad packets transmitted with length between 1024 and maxsize (inclusive) bytes, exclusive of preamble and retried packets.

56.2.58 Transmit Unicast Packets Good Bad Register (TX_UNICAST_PACKETS_GOOD_BAD)

This register provides the number of good and bad unicast packets transmitted by the Ethernet module.

Offset 0x073C Access: User Read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	TXUCASTGB															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	TXUCASTGB															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1037. Transmit Unicast Packets Good Bad Register (TX_UNICAST_PACKETS_GOOD_BAD)

Table 1089. TX_UNICAST_PACKETS_GOOD_BAD field descriptions

Field	Description
0:31 TXUCASTGB	This field indicates the number of good and bad unicast packets transmitted.

56.2.59 Transmit Multicast Packets Good Bad Register (TX_MULTICAST_PACKETS_GOOD_BAD)

This register provides the number of good and bad multicast packets transmitted by the Ethernet module.

Offset 0x0740 Access: User Read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	TXMCASTGB															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	TXMCASTGB															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1038. Transmit Multicast Packets Good Bad Register (TX_MULTICAST_PACKETS_GOOD_BAD)

Table 1090. TX_MULTICAST_PACKETS_GOOD_BAD field descriptions

Field	Description
0:31 TXMCASTGB	This field indicates the number of good and bad multicast packets transmitted.

56.2.60 Transmit Broadcast Packets Good Bad Register (TX_BROADCAST_PACKETS_GOOD_BAD)

This register provides the number of good and bad broadcast packets transmitted by the Ethernet module.

Offset 0x0744 Access: User Read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	TXBCASTGB															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	TXBCASTGB															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1039. Transmit Broadcast Packets Good Bad Register (TX_BROADCAST_PACKETS_GOOD_BAD)

Table 1091. TX_BROADCAST_PACKETS_GOOD_BAD field descriptions

Field	Description
0:31 TXBCASTGB	This field indicates the number of good and bad broadcast packets transmitted

56.2.61 Transmit Underflow Error Packets Register (TX_UNDERFLOW_ERROR_PACKETS)

This register provides the number of packets aborted by the Ethernet module because of packets underflow error.

Offset 0x0748 Access: User Read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	TXUNDRFLW															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	TXUNDRFLW															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1040. Transmit Underflow Error Packets Register (TX_UNDERFLOW_ERROR_PACKETS)

Table 1092. TX_UNDERFLOW_ERROR_PACKETS field descriptions

Field	Description
0:31 TXUNDRFLW	This field indicates the number of packets aborted because of packets underflow error.

56.2.62 Transmit Single Collision Good Packets Register (TX_SINGLE_COLLISION_GOOD_PACKETS)

This register provides the number of successfully transmitted packets by the Ethernet module after a single collision in the half-duplex mode.

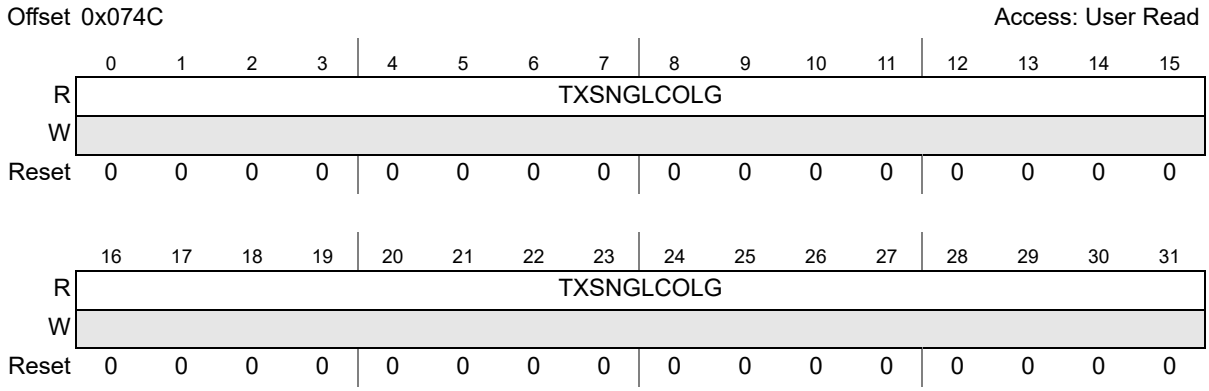


Figure 1041. Transmit Single Collision Good Packets Register (TX_SINGLE_COLLISION_GOOD_PACKETS)

Table 1093. TX_SINGLE_COLLISION_GOOD_PACKETS field descriptions

Field	Description
0:31 TXSNGLCOLG	This field indicates the number of successfully transmitted packets after a single collision in the half-duplex mode.

56.2.63 Transmit Multiple Collision Good Packets Register (TX_MULTIPLE_COLLISION_GOOD_PACKETS)

This register provides the number of successfully transmitted packets by the Ethernet module after multiple collisions in the half-duplex mode.

Offset 0x0750 Access: User Read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	TXMULTLCOLG															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	TXMULTLCOLG															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1042. Transmit Multiple Collision Good Packets Register (TX_MULTIPLE_COLLISION_GOOD_PACKETS)

Table 1094. TX_MULTIPLE_COLLISION_GOOD_PACKETS field descriptions

Field	Description
0:31 TXMULTLCOLG	This field indicates the number of successfully transmitted packets after multiple collisions in the half-duplex mode.

56.2.64 Transmit Deferred Packets Register (TX_DEFERRED_PACKETS)

This register provides the number of successfully transmitted by the Ethernet module after a deferral in the half-duplex mode.

Offset 0x0754 Access: User Read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	TXDEFRD															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	TXDEFRD															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1043. Transmit Deferred Packets Register (TX_DEFERRED_PACKETS)

Table 1095. TX_DEFERRED_PACKETS field descriptions

Field	Description
0:31 TXDEFRD	This field indicates the number of successfully transmitted after a deferral in the half-duplex mode.

56.2.65 Transmit Late Collision Packets Register (TX_LATE_COLLISION_PACKETS)

This register provides the number of packets aborted by the Ethernet module because of late collision error.

Offset 0x0758 Access: User Read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	TXLATECOL															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	TXLATECOL															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1044. Transmit Late Collision Packets Register (TX_LATE_COLLISION_PACKETS)

Table 1096. TX_LATE_COLLISION_PACKETS field descriptions

Field	Description
0:31 TXLATECOL	This field indicates the number of packets aborted because of late collision error.

56.2.66 Transmit Excessive Collision Packets Register (TX_EXCESSIVE_COLLISION_PACKETS)

This register provides the number of packets aborted by the Ethernet module because of excessive (16) collision errors.

Offset 0x075C Access: User Read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	TXEXSCOL															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	TXEXSCOL															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1045. Transmit Excessive Collision Packets Register (TX_EXCESSIVE_COLLISION_PACKETS)

Table 1097. TX_EXCESSIVE_COLLISION_PACKETS field descriptions

Field	Description
0:31 TXEXSCOL	This field indicates the number of packets aborted because of excessive (16) collision errors.

56.2.67 Transmit Carrier Error Packets Register (TX_CARRIER_ERROR_PACKETS)

This register provides the number of packets aborted by the Ethernet module because of carrier sense error (no carrier or loss of carrier).

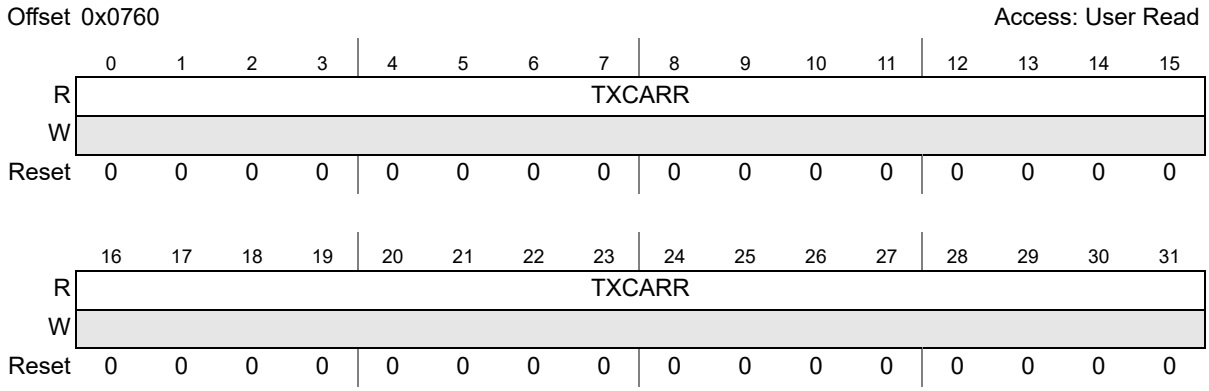


Figure 1046. Transmit Carrier Error Packets Register (TX_CARRIER_ERROR_PACKETS)

Table 1098. TX_CARRIER_ERROR_PACKETS field descriptions

Field	Description
0:31 TXCARR	This field indicates the number of packets aborted because of carrier sense error (no carrier or loss of carrier).

56.2.68 Transmit Octet Count Good Register (TX_OCTET_COUNT_GOOD)

This register provides the number of bytes transmitted by the Ethernet module, exclusive of preamble, only in good packets.

Offset 0x0764 Access: User Read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	TXOCTG															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	TXOCTG															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1047. Transmit Octet Count Good Register (TX_OCTET_COUNT_GOOD)

Table 1099. TX_OCTET_COUNT_GOOD field descriptions

Field	Description
0:31 TXOCTG	This field indicates the number of bytes transmitted, exclusive of preamble, only in good packets.

56.2.69 Transmit Packet Count Good Register (TX_PACKET_COUNT_GOOD)

This register provides the number of good packets transmitted by the Ethernet module.

Offset 0x0768 Access: User Read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	TXPKTG															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	TXPKTG															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1048. Transmit Packet Count Good Register (TX_PACKET_COUNT_GOOD)

Table 1100. TX_PACKET_COUNT_GOOD field descriptions

Field	Description
0:31 TXPKTG	This field indicates the number of good packets transmitted.

56.2.70 Transmit Excessive Deferral Error Register (TX_EXCESSIVE_DEFERRAL_ERROR)

This register provides the number of packets aborted by the Ethernet module because of excessive deferral error (deferred for more than two max-sized packet times).

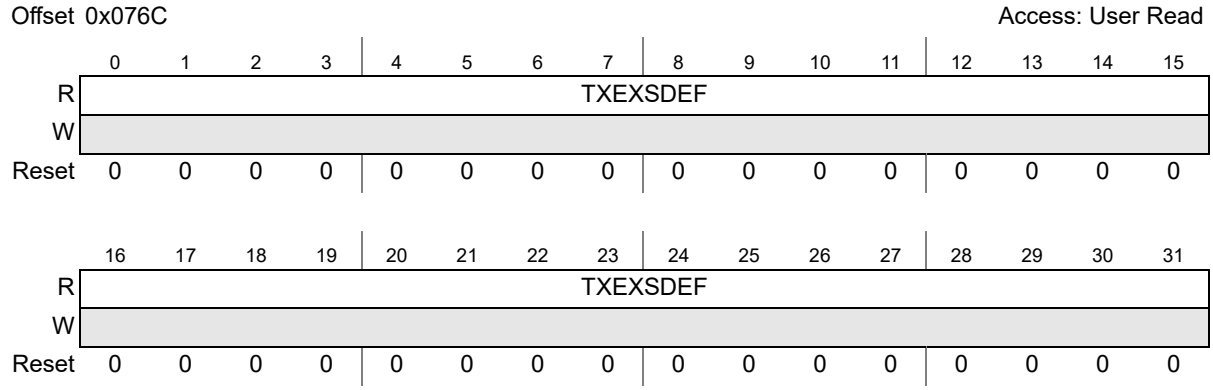


Figure 1049. Transmit Excessive Deferral Error Register (TX_EXCESSIVE_DEFERRAL_ERROR)

Table 1101. TX_EXCESSIVE_DEFERRAL_ERROR field descriptions

Field	Description
0:31 TXEXSDEF	This field indicates the number of packets aborted because of excessive deferral error (deferred for more than two max-sized packet times).

56.2.71 Transmit Pause Packets Register (TX_PAUSE_PACKETS)

This register provides the number of good Pause packets transmitted by the Ethernet module.

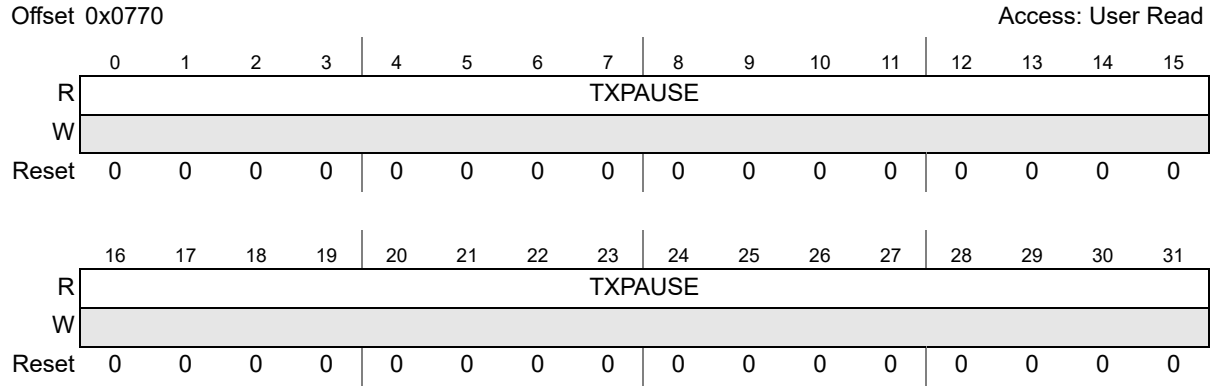


Figure 1050. Transmit Pause Packets Register (TX_PAUSE_PACKETS)

Table 1102. TX_PAUSE_PACKETS field descriptions

Field	Description
0:31 TXPAUSE	This field indicates the number of good Pause packets transmitted.

56.2.72 Transmit VLAN Packets Good Register (TX_VLAN_PACKETS_GOOD)

This register provides the number of good VLAN packets transmitted by the Ethernet module.

Offset 0x0774 Access: User Read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	TXVLANG															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	TXVLANG															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1051. Transmit VLAN Packets Good Register (TX_VLAN_PACKETS_GOOD)

Table 1103. TX_VLAN_PACKETS_GOOD field descriptions

Field	Description
0:31 TXVLANG	This field provides the number of good VLAN packets transmitted.

56.2.73 Transmit OSize Packets Good Register (TX_OSIZE_PACKETS_GOOD)

This register provides the number of packets transmitted by the Ethernet module without errors and with length greater than the maxsize (1,518 or 1,522 bytes for VLAN tagged packets; 2000 bytes if enabled in S2KP bit of the MAC_Configuration register).

Offset 0x0778 Access: User Read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	TXOSIZG															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	TXOSIZG															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1052. Transmit OSize Packets Good Register (TX_OSIZE_PACKETS_GOOD)

Table 1104. TX_OSIZE_PACKETS_GOOD field descriptions

Field	Description
0:31 TXOSIZG	This field indicates the number of packets transmitted without errors and with length greater than the maxsize (1,518 or 1,522 bytes for VLAN tagged packets; 2000 bytes if enabled in S2KP bit of the MAC_Configuration register).

56.2.74 Receive Packets Count Good Bad Register (RX_PACKETS_COUNT_GOOD_BAD)

This register provides the number of good and bad packets received by the Ethernet module.

Offset 0x0780 Access: User Read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	RXPKTGB															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	RXPKTGB															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1053. Receive Packets Count Good Bad Register (RX_PACKETS_COUNT_GOOD_BAD)

Table 1105. RX_PACKETS_COUNT_GOOD_BAD field descriptions

Field	Description
0:31 RXPKTGB	This field indicates the number of good and bad packets received.

56.2.75 Receive Octet Count Good Bad Register (RX_OCTET_COUNT_GOOD_BAD)

This register provides the number of bytes received by DWC_ther_qos, exclusive of preamble, in good and bad packets.

Offset 0x0784 Access: User Read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	RXOCTGB															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	RXOCTGB															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1054. Receive Octet Count Good Bad Register (RX_OCTET_COUNT_GOOD_BAD)

Table 1106. RX_OCTET_COUNT_GOOD_BAD field descriptions

Field	Description
0:31 RXOCTGB	This field indicates the number of bytes received, exclusive of preamble, in good and bad packets.

56.2.76 Receive Octet Count Good Register (RX_OCTET_COUNT_GOOD)

This register provides the number of bytes received by the Ethernet module, exclusive of preamble, only in good packets.

Offset 0x0788 Access: User Read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	RXOCTG															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	RXOCTG															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1055. Receive Octet Count Good Register (RX_OCTET_COUNT_GOOD)

Table 1107. RX_OCTET_COUNT_GOOD field descriptions

Field	Description
0:31 RXOCTG	This field indicates the number of bytes received, exclusive of preamble, in good and bad packets.

56.2.77 Receive Broadcast Packets Good Register (RX_BROADCAST_PACKETS_GOOD)

This register provides the number of good broadcast packets received by the Ethernet module.

Offset 0x078C Access: User Read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	RXBCASTG															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	RXBCASTG															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1056. Receive Broadcast Packets Good Register (RX_BROADCAST_PACKETS_GOOD)

Table 1108. RX_BROADCAST_PACKETS_GOOD field descriptions

Field	Description
0:31 RXBCASTG	This field indicates the number of good broadcast packets received.

56.2.78 Receive Multicast Packets Good Register (RX_MULTICAST_PACKETS_GOOD)

This register provides the number of good multicast packets received by the Ethernet module.

Offset 0x0790 Access: User Read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	RXMCASTG															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	RXMCASTG															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1057. Receive Multicast Packets Good Register (RX_MULTICAST_PACKETS_GOOD)

Table 1109. RX_MULTICAST_PACKETS_GOOD field descriptions

Field	Description
0:31 RXMCASTG	This field indicates the number of good multicast packets received.

56.2.79 Receive CRC Error Packets Register (RX_CRC_ERROR_PACKETS)

This register provides the number of packets received by the Ethernet module with CRC error.

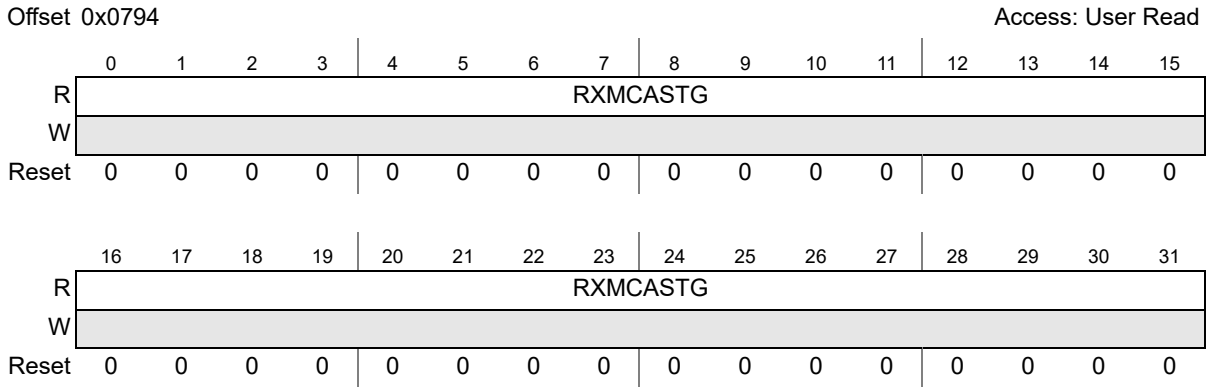


Figure 1058. Receive CRC Error Packets Register (RX_CRC_ERROR_PACKETS)

Table 1110. RX_CRC_ERROR_PACKETS field descriptions

Field	Description
0:31 RXMCASTG	This field indicates the number of good multicast packets received.

56.2.80 Receive Alignment Error Packets Register (RX_ALIGNMENT_ERROR_PACKETS)

This register provides the number of packets received by the Ethernet module with alignment (dribble) error. It is valid only in 10/100 mode.

Offset 0x0798 Access: User Read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	RXALGNERR															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	RXALGNERR															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1059. Receive Alignment Error Packets Register (RX_ALIGNMENT_ERROR_PACKETS)

Table 1111. RX_ALIGNMENT_ERROR_PACKETS field descriptions

Field	Description
0:31 RXALGNERR	This field indicates the number of packets received with alignment (dribble) error. It is valid only in 10/100 mode.

56.2.81 Receive Runt Error Packets Register (RX_RUNT_ERROR_PACKETS)

This register provides the number of packets received by the Ethernet module with runt (length less than 64 bytes and CRC error) error.

Offset 0x079C Access: User Read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	RXRUNTERR															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	RXRUNTERR															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1060. Receive Runt Error Packets Register (RX_RUNT_ERROR_PACKETS)

Table 1112. RX_RUNT_ERROR_PACKETS field descriptions

Field	Description
0:31 RXRUNTERR	This field indicates the number of packets received with runt (length less than 64 bytes and CRC error) error.

56.2.82 Receive Jabber Error Packets Register (RX_JABBER_ERROR_PACKETS)

This register provides the number of giant packets received by the Ethernet module with length (including CRC) greater than 1,518 bytes (1,522 bytes for VLAN tagged) and with CRC error. If Jumbo Packet mode is enabled, packets of length greater than 9,018 bytes (9,022 bytes for VLAN tagged) are considered as giant packets.

Offset 0x07A0 Access: User Read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	RXJABERR															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	RXJABERR															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1061. Receive Jabber Error Packets Register (RX_JABBER_ERROR_PACKETS)

Table 1113. RX_JABBER_ERROR_PACKETS field descriptions

Field	Description
0:31 RXJABERR	This field indicates the number of giant packets received with length (including CRC) greater than 1,518 bytes (1,522 bytes for VLAN tagged) and with CRC error. If Jumbo Packet mode is enabled, packets of length greater than 9,018 bytes (9,022 bytes for VLAN tagged) are considered as giant packets.

56.2.83 Receive Undersize Packets Good Register (RX_UNDERSIZE_PACKETS_GOOD)

This register provides the number of packets received by the Ethernet module with length less than 64 bytes, without any errors.

Offset 0x07A4 Access: User Read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	RXUNDERSZG															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	RXUNDERSZG															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1062. Receive Undersize Packets Good Register (RX_UNDERSIZE_PACKETS_GOOD)

Table 1114. RX_UNDERSIZE_PACKETS_GOOD field descriptions

Field	Description
0:31 RXUNDERSZG	This field indicates the number of giant packets received with length (including CRC) greater than 1,518 bytes (1,522 bytes for VLAN tagged) and with CRC error. If Jumbo Packet mode is enabled, packets of length greater than 9,018 bytes (9,022 bytes for VLAN tagged) are considered as giant packets.

56.2.84 Receive Oversize Packets Good Register (RX_OVERSIZE_PACKETS_GOOD)

This register provides the number of packets received by the Ethernet module without errors, with length greater than the maxsize (1,518 bytes or 1,522 bytes for VLAN tagged packets; 2000 bytes if enabled in the S2KP bit of the MAC_Configuration register).

Offset 0x07A8 Access: User Read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	RXOVERSZG															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	RXOVERSZG															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1063. Receive Oversize Packets Good Register (RX_OVERSIZE_PACKETS_GOOD)

Table 1115. RX_OVERSIZE_PACKETS_GOOD field descriptions

Field	Description
0:31 RXOVERSZG	This field indicates the number of packets received without errors, with length greater than the maxsize (1,518 bytes or 1,522 bytes for VLAN tagged packets; 2000 bytes if enabled in the S2KP bit of the MAC_Configuration register).

56.2.85 Receive 64Octets Packets Good Bad Register (RX_64OCTETS_PACKETS_GOOD_BAD)

This register provides the number of good and bad packets received by the Ethernet module with length 64 bytes, exclusive of the preamble.

Offset 0x07AC Access: User Read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	RX64OCTGB															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	RX64OCTGB															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1064. Receive 64Octets Packets Good Bad Register (RX_64OCTETS_PACKETS_GOOD_BAD)

Table 1116. RX_64OCTETS_PACKETS_GOOD_BAD field descriptions

Field	Description
0:31 RX64OCTGB	This field indicates the number of good and bad packets received with length 64 bytes, exclusive of the preamble.

56.2.86 Receive 65To127Octets Packets Good Bad Register (RX_65TO127OCTETS_PACKETS_GOOD_BAD)

This register provides the number of good and bad packets received by the Ethernet module with length between 65 and 127 (inclusive) bytes, exclusive of the preamble.

Offset 0x07B0 Access: User Read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	RX65_127OCTGB															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	RX65_127OCTGB															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1065. Receive 65To127Octets Packets Good Bad Register (RX_65TO127OCTETS_PACKETS_GOOD_BAD)

Table 1117. RX_65TO127OCTETS_PACKETS_GOOD_BAD field descriptions

Field	Description
0:31 RX65_127OCTGB	This field indicates the number of good and bad packets received with length between 65 and 127 (inclusive) bytes, exclusive of the preamble.

56.2.87 Receive 128To255Octets Packets Good Bad Register (RX_128TO255OCTETS_PACKETS_GOOD_BAD)

This register provides the number of good and bad packets received by the Ethernet module with length between 128 and 255 (inclusive) bytes, exclusive of the preamble.

Offset 0x07B4 Access: User Read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	RX128_255OCTGB															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	RX128_255OCTGB															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1066. Receive 128To255Octets Packets Good Bad Register (RX_128TO255OCTETS_PACKETS_GOOD_BAD)

Table 1118. RX_128TO255OCTETS_PACKETS_GOOD_BAD field descriptions

Field	Description
0:31 RX128_255OCTGB	This field indicates the number of good and bad packets received with length between 128 and 255 (inclusive) bytes, exclusive of the preamble.

56.2.88 Receive 256To511Octets Packets Good Bad Register (RX_256TO511OCTETS_PACKETS_GOOD_BAD)

This register provides the number of good and bad packets received by the Ethernet module with length between 256 and 511 (inclusive) bytes, exclusive of the preamble.

Offset 0x07B8 Access: User Read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	RX256_511OCTGB															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	RX256_511OCTGB															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1067. Receive 256To511Octets Packets Good Bad Register (RX_256TO511OCTETS_PACKETS_GOOD_BAD)

Table 1119. RX_256TO511OCTETS_PACKETS_GOOD_BAD field descriptions

Field	Description
0:31 RX256_511OCTGB	This field indicates the number of good and bad packets received with length between 256 and 511 (inclusive) bytes, exclusive of the preamble.

56.2.89 Receive 512To1023Octets Packets Good Bad Register (RX_512TO1023OCTETS_PACKETS_GOOD_BAD)

This register provides the number of good and bad packets received by the Ethernet module with length between 512 and 1023 (inclusive) bytes, exclusive of the preamble.

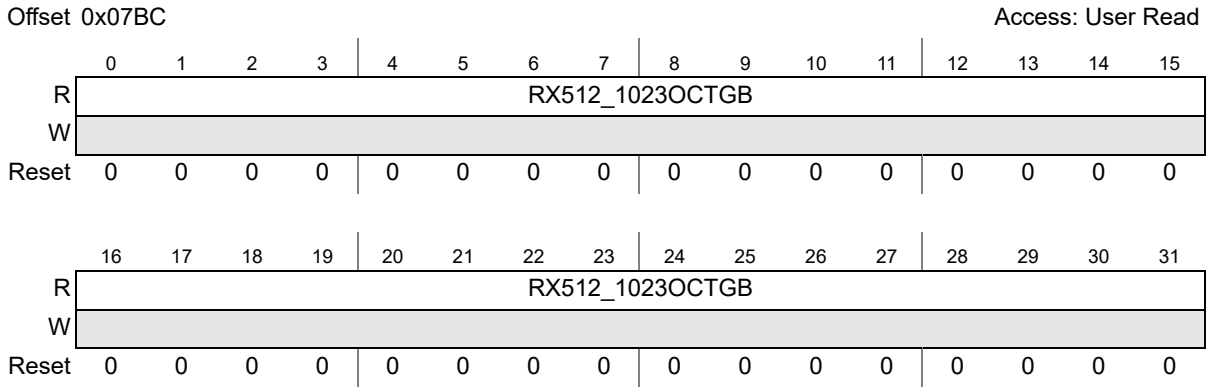


Figure 1068. Receive 512To1023Octets Packets Good Bad Register (RX_512TO1023OCTETS_PACKETS_GOOD_BAD)

Table 1120. RX_512TO1023OCTETS_PACKETS_GOOD_BAD field descriptions

Field	Description
0:31 RX512_1023OCTGB	This field indicates the number of good and bad packets received with length between 512 and 1023 (inclusive) bytes, exclusive of the preamble.

56.2.90 Receive 1024ToMax Octets Packets Good bad Register (RX_1024TOMAXOCTETS_PACKETS_GOOD_BAD)

This register provides the number of good and bad packets received by the Ethernet module with length between 1024 and maxsize (inclusive) bytes, exclusive of the preamble.

Offset 0x07C0 Access: User Read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	RX1024_MAXOCTGB															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	RX1024_MAXOCTGB															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1069. Receive 1024ToMax Octets Packets Good bad Register (RX_1024TOMAXOCTETS_PACKETS_GOOD_BAD)

Table 1121. RX_1024TOMAXOCTETS_PACKETS_GOOD_BAD field descriptions

Field	Description
0:31 RX1024_MAXOCTGB	This field indicates the number of good and bad packets received with length between 1024 and maxsize (inclusive) bytes, exclusive of the preamble.

56.2.91 Receive Unicast Packets Good Register (RX_UNICAST_PACKETS_GOOD)

This register provides the number of good unicast packets received by the Ethernet module.

Offset 0x07C4 Access: User Read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	RXUCASTG															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	RXUCASTG															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1070. Receive Unicast Packets Good Register (RX_UNICAST_PACKETS_GOOD)

Table 1122. RX_UNICAST_PACKETS_GOOD field descriptions

Field	Description
0:31 RXUCASTG	This field indicates the number of good unicast packets received.

56.2.92 Receive Length Error Packets (RX_LENGTH_ERROR_PACKETS)

This register provides the number of packets received by the Ethernet module with length error (Length Type field not equal to packet size), for all packets with valid length field.

Offset 0x07C8 Access: User Read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	RXLENERR															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	RXLENERR															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1071. Receive Length Error Packets (RX_LENGTH_ERROR_PACKETS)

Table 1123. RX_LENGTH_ERROR_PACKETS field descriptions

Field	Description
0:31 RXLENERR	This field indicates the number of packets received with length error (Length Type field not equal to packet size), for all packets with valid length field.

56.2.93 Receive Out Of Range Type Packets (RX_OUT_OF_RANGE_TYPE_PACKETS)

This register provides the number of packets received by the Ethernet module with length field not equal to the valid packet size (greater than 1,500 but less than 1,536).

Offset 0x07CC Access: User Read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	RXOUTOFRNG															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	RXOUTOFRNG															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1072. Receive Out Of Range Type Packets (RX_OUT_OF_RANGE_TYPE_PACKETS)

Table 1124. RX_OUT_OF_RANGE_TYPE_PACKETS field descriptions

Field	Description
0:31 RXOUTOFRNG	This field indicates the number of packets received with length field not equal to the valid packet size (greater than 1,500 but less than 1,536).

56.2.94 Receive Pause Packets Register (RX_PAUSE_PACKETS)

This register provides the number of good and valid Pause packets received by the Ethernet module.

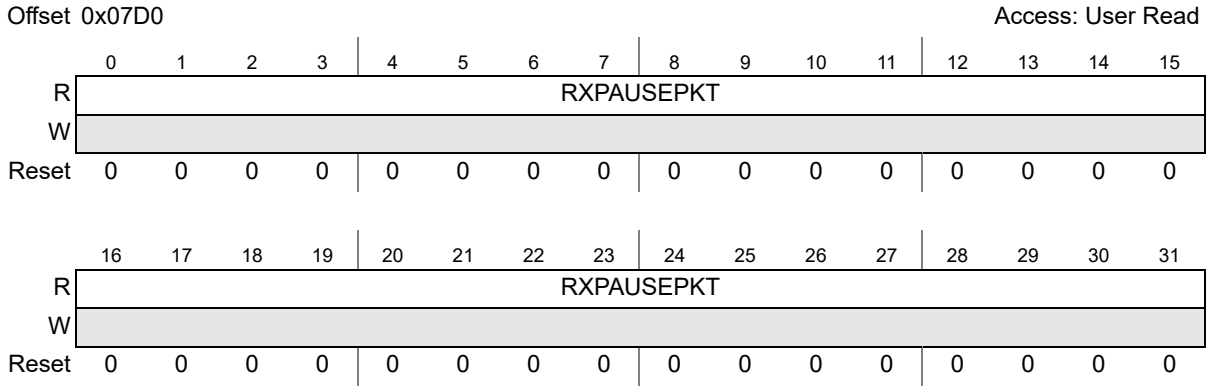


Figure 1073. Receive Pause Packets Register (RX_PAUSE_PACKETS)

Table 1125. RX_PAUSE_PACKETS field descriptions

Field	Description
0:31 RXPAUSEPKT	This field indicates the number of good and valid Pause packets received.

56.2.95 Receive FIFO Overflow Packets Register (RX_FIFO_OVERFLOW_PACKETS)

This register provides the number of missed received packets because of FIFO overflow in the Ethernet module.

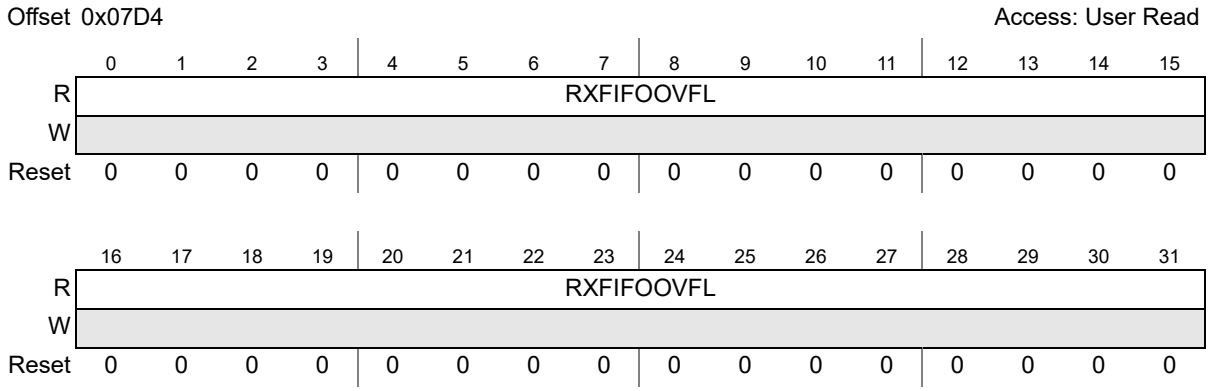


Figure 1074. Receive FIFO Overflow Packets Register (RX_FIFO_OVERFLOW_PACKETS)

Table 1126. RX_FIFO_OVERFLOW_PACKETS field descriptions

Field	Description
0:31 RXFIFOOVFL	This field indicates the number of missed received packets because of FIFO overflow.

56.2.96 Receive VLAN Packets Good Bad Register (RX_VLAN_PACKETS_GOOD_BAD)

This register provides the number of good and bad VLAN packets received by the Ethernet module.

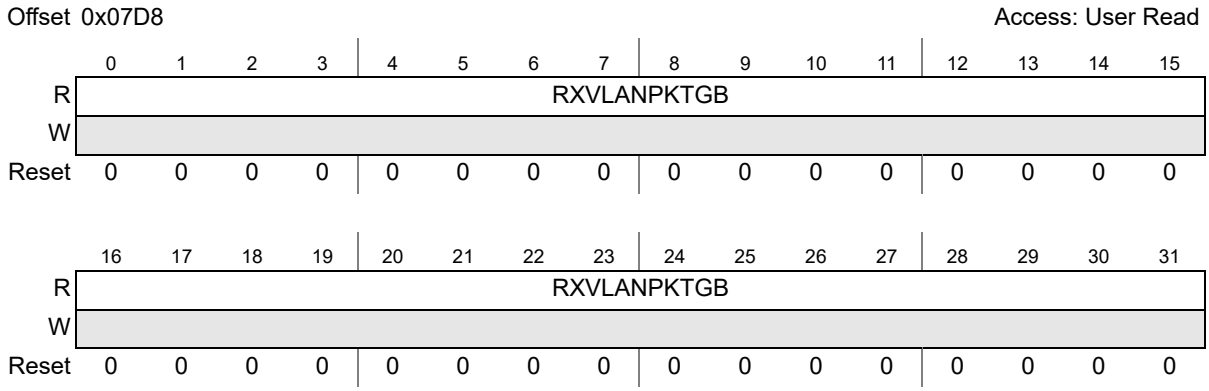


Figure 1075. Receive VLAN Packets Good Bad Register (RX_VLAN_PACKETS_GOOD_BAD)

Table 1127. RX_VLAN_PACKETS_GOOD_BAD field descriptions

Field	Description
0:31 RXVLANPKTGB	This field indicates the number of good and bad VLAN packets received.

56.2.97 Receive Watchdog Error Packets Register (RX_WATCHDOG_ERROR_PACKETS)

This register provides the number of packets received by the Ethernet module with error because of watchdog timeout error (packets with a data load larger than 2,048 bytes (when JE and WD bits are reset in MAC_Configuration register), 10,240 bytes (when JE bit is set and WD bit is reset in MAC_Configuration register), 16,384 bytes (when WD bit is set in MAC_Configuration register) or the value programed in the MAC_Watchdog_Timeout register).

Offset 0x07DC Access: User Read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	RXWDGERR															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	RXWDGERR															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1076. Receive Watchdog Error Packets Register (RX_WATCHDOG_ERROR_PACKETS)

Table 1128. RX_WATCHDOG_ERROR_PACKETS field descriptions

Field	Description
0:31 RXWDGERR	This field indicates the number of packets received with error because of watchdog timeout error (packets with a data load larger than 2,048 bytes (when JE and WD bits are reset in MAC_Configuration register), 10,240 bytes (when JE bit is set and WD bit is reset in MAC_Configuration register), 16,384 bytes (when WD bit is set in MAC_Configuration register) or the value programed in the MAC_Watchdog_Timeout register).

56.2.98 Receive Error Packets Register (RX_RECEIVE_ERROR_PACKETS)

This register provides the number of packets received by the Ethernet module with Receive error or Packet Extension error on the MII interface.

Offset 0x07E0 Access: User Read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	RXRCVERR															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	RXRCVERR															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1077. Receive Error Packets Register (RX_RECEIVE_ERROR_PACKETS)

Table 1129. RX_RECEIVE_ERROR_PACKETS field descriptions

Field	Description
0:31 RXRCVERR	This field indicates the number of packets received with Receive error or Packet Extension error on the MII interface.

56.2.99 Receive Control Packets Good Register (RX_CONTROL_PACKETS_GOOD)

This register provides the number of good control packets received by the Ethernet module.

Offset 0x07E4 Access: User Read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	RXCTRLG															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	RXCTRLG															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1078. Receive Control Packets Good Register (RX_CONTROL_PACKETS_GOOD)

Table 1130. RX_CONTROL_PACKETS_GOOD field descriptions

Field	Description
0:31 RXCTRLG	This field indicates the number of good control packets received.

56.2.100 Transmit LPI USEC Counter (TX_LPI_USEC_CNTR)

This register provides the number of microseconds Tx LPI is asserted by the Ethernet module.

Offset 0x07EC Access: User Read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	TXLPIUSC															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	TXLPIUSC															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1079. Transmit LPI USEC Counter (TX_LPI_USEC_CNTR)

Table 1131. TX_LPI_USEC_CNTR field descriptions

Field	Description
0:31 TXLPIUSC	This field indicates the number of microseconds Tx LPI is asserted. For every Tx LPI Entry and Exit, the Timer value can have an error of +/- 1 microsecond.

56.2.101 Transmit LPI Transaction Counter Register (TX_LPI_TRAN_CNTR)

This register provides the number of times the Ethernet module has entered Tx LPI.

Offset 0x07F0 Access: User Read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	TXLPITRC															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	TXLPITRC															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1080. Transmit LPI Transaction Counter Register (TX_LPI_TRAN_CNTR)

Table 1132. TX_LPI_TRAN_CNTR descriptions

Field	Description
0:31 TXLPITRC	This field indicates the number of times Tx LPI Entry has occurred. Even if Tx LPI Entry occurs in Automate Mode (because of LPITXA bit set in the LPI Control and Status register), the counter will increment.

56.2.102 Receive LPI USEC Counter (RX_LPI_USEC_CNTR)

This register provides the number of microseconds Rx LPI is sampled by the Ethernet module.

Offset 0x07F4 Access: User Read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	RXLPIUSC															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	RXLPIUSC															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1081. Receive LPI USEC Counter (RX_LPI_USEC_CNTR)

Table 1133. RX_LPI_USEC_CNTR field descriptions

Field	Description
0:31 RXLPIUSC	This field indicates the number of microseconds Rx LPI is asserted. For every Rx LPI Entry and Exit, the Timer value can have an error of +/- 1 microsecond.

56.2.103 Receive LPI Transaction Counter Register (RX_LPI_TRAN_CNTR)

This register provides the number of times the Ethernet module has entered Rx LPI.

Offset 0x07F8 Access: User Read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	RXLPITRC															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	RXLPITRC															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1082. Receive LPI Transaction Counter Register (RX_LPI_TRAN_CNTR)

Table 1134. RX_LPI_TRAN_CNTR field descriptions

Field	Description
0:31 RXLPITRC	This field indicates the number of times Rx LPI Entry has occurred.

56.2.104 MMC IPC Receive Interrupt Mask Register (MMC_IPC_RX_INTERRUPT_MASK)

This register maintains the mask for the interrupt generated from the receive IPC statistic counters.

Offset 0x0800 Access: User Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	RXICMPEROIM	RXICMPGOIM	RXTCPEROIM	RXTCPGOIM	RXUDPEROIM	RXUDPGOIM	RXIPV6NOPAYOIM	RXIPV6HEROIM	RXIPV6GOIM	RXIPV4UDSBLOIM	RXIPV4FRAGOIM	RXIPV4NOPAYOIM	RXIPV4HEROIM	RXIPV4GOIM
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	RXICMPERPIM	RXICMPGPIM	RXTCPERPIM	RXTCPGPIM	RXUDPERPIM	RXUDPGPIM	RXIPV6NOPAYPIM	RXIPV6HERPIM	RXIPV6GPIM	RXIPV4UDSBLPIM	RXIPV4FRAGPIM	RXIPV4NOPAYPIM	RXIPV4HERPIM	RXIPV4GPIM
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1083. MMC IPC Receive Interrupt Mask Register (MMC_IPC_RX_INTERRUPT_MASK)

Table 1135. MMC_IPC_RX_INTERRUPT_MASK field descriptions

Field	Description
2 RXICMPEROIM	MMC Receive ICMP Error Octet Counter Interrupt Mask Setting this bit masks the interrupt when the rxicmp_err_octets counter reaches half of the maximum value or the maximum value.
3 RXICMPGOIM	MMC Receive ICMP Good Octet Counter Interrupt Mask Setting this bit masks the interrupt when the rxicmp_gd_octets counter reaches half of the maximum value or the maximum value.
4 RXTCPEROIM	MMC Receive TCP Error Octet Counter Interrupt Mask Setting this bit masks the interrupt when the rxtcp_err_octets counter reaches half of the maximum value or the maximum value.
5 RXTCPGOIM	MMC Receive TCP Good Octet Counter Interrupt Mask Setting this bit masks the interrupt when the rxtcp_gd_octets counter reaches half of the maximum value or the maximum value.
6 RXUDPEROIM	MMC Receive UDP Good Octet Counter Interrupt Mask Setting this bit masks the interrupt when the rxudp_err_octets counter reaches half of the maximum value or the maximum value.
7 RXUDPGOIM	MMC Receive IPV6 No Payload Octet Counter Interrupt Mask Setting this bit masks the interrupt when the rxudp_gd_octets counter reaches half of the maximum value or the maximum value.
8 RXIPV6NOPAYOIM	MMC Receive IPV6 Header Error Octet Counter Interrupt Mask Setting this bit masks the interrupt when the rxipv6_nopay_octets counter reaches half of the maximum value or the maximum value.

Table 1135. MMC_IPC_RX_INTERRUPT_MASK field descriptions (continued)

Field	Description
9 RXIPV6HEROIM	MMC Receive IPV6 Good Octet Counter Interrupt Mask Setting this bit masks the interrupt when the rxipv6_hdrerr_octets counter reaches half of the maximum value or the maximum value.
10 RXIPV6GOIM	MMC Receive IPV6 Good Octet Counter Interrupt Mask Setting this bit masks the interrupt when the rxipv6_gd_octets counter reaches half of the maximum value or the maximum value.
11 RXIPV4UDSBLOIM	MMC Receive IPV4 UDP Checksum Disabled Octet Counter Interrupt Mask Setting this bit masks the interrupt when the rxipv4_udsbl_octets counter reaches half of the maximum value or the maximum value.
12 RXIPV4FRAGOIM	MMC Receive IPV4 Fragmented Octet Counter Interrupt Mask Setting this bit masks the interrupt when the rxipv4_frag_octets counter reaches half of the maximum value or the maximum value.
13 RXIPV4NOPAYOIM	MMC Receive IPV4 No Payload Octet Counter Interrupt Mask Setting this bit masks the interrupt when the rxipv4_nopay_octets counter reaches half of the maximum value or the maximum value.
14 RXIPV4HEROIM	MMC Receive IPV4 Header Error Octet Counter Interrupt Mask Setting this bit masks the interrupt when the rxipv4_hdrerr_octets counter reaches half of the maximum value or the maximum value.
15 RXIPV4GOIM	MMC Receive IPV4 Good Octet Counter Interrupt Mask Setting this bit masks the interrupt when the rxipv4_gd_octets counter reaches half of the maximum value or the maximum value.
18 RXICMPERPIM	MMC Receive ICMP Error Packet Counter Interrupt Mask Setting this bit masks the interrupt when the rxicmp_err_pkts counter reaches half of the maximum value or the maximum value.
19 RXICMPGPIM	MMC Receive ICMP Good Packet Counter Interrupt Mask Setting this bit masks the interrupt when the rxicmp_gd_pkts counter reaches half of the maximum value or the maximum value.
20 RXTCPERPIM	MMC Receive TCP Error Packet Counter Interrupt Mask Setting this bit masks the interrupt when the rxtcp_err_pkts counter reaches half of the maximum value or the maximum value.
21 RXTCPGPIM	MMC Receive TCP Good Packet Counter Interrupt Mask Setting this bit masks the interrupt when the rxtcp_gd_pkts counter reaches half of the maximum value or the maximum value.
22 RXUDPERPIM	MMC Receive UDP Error Packet Counter Interrupt Mask Setting this bit masks the interrupt when the rxudp_err_pkts counter reaches half of the maximum value or the maximum value.
23 RXUDPGPIM	MMC Receive UDP Good Packet Counter Interrupt Mask Setting this bit masks the interrupt when the rxudp_gd_pkts counter reaches half of the maximum value or the maximum value.
24 RXIPV6NOPAYPIM	MMC Receive IPV6 No Payload Packet Counter Interrupt Mask Setting this bit masks the interrupt when the rxipv6_nopay_pkts counter reaches half of the maximum value or the maximum value.

Table 1135. MMC_IPC_RX_INTERRUPT_MASK field descriptions (continued)

Field	Description
25 RXIPV6HERPIM	MMC Receive IPV6 Header Error Packet Counter Interrupt Mask Setting this bit masks the interrupt when the rxipv6_hdrerr_pkts counter reaches half of the maximum value or the maximum value.
26 RXIPV6GPIM	MMC Receive IPV6 Good Packet Counter Interrupt Mask Setting this bit masks the interrupt when the rxipv6_gd_pkts counter reaches half of the maximum value or the maximum value.
27 RXIPV4UDSBLPIM	MMC Receive IPV4 UDP Checksum Disabled Packet Counter Interrupt Mask Setting this bit masks the interrupt when the rxipv4_udsbl_pkts counter reaches half of the maximum value or the maximum value.
28 RXIPV4FRAGPIM	MMC Receive IPV4 Fragmented Packet Counter Interrupt Mask Setting this bit masks the interrupt when the rxipv4_frag_pkts counter reaches half of the maximum value or the maximum value.
29 RXIPV4NOPAYPIM	MMC Receive IPV4 No Payload Packet Counter Interrupt Mask Setting this bit masks the interrupt when the rxipv4_nopay_pkts counter reaches half of the maximum value or the maximum value.
30 RXIPV4HERPIM	MMC Receive IPV4 Header Error Packet Counter Interrupt Mask Setting this bit masks the interrupt when the rxipv4_hdrerr_pkts counter reaches half of the maximum value or the maximum value.
31 RXIPV4GPIM	MMC Receive IPV4 Good Packet Counter Interrupt Mask Setting this bit masks the interrupt when the rxipv4_gd_pkts counter reaches half of the maximum value or the maximum value.

56.2.105 MMC IPC Receive Interrupt Register (MMC_IPC_RX_INTERRUPT)

This register maintains the interrupt that the receive IPC statistic counters generate.

Offset 0x0808 Access: User Read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	RXICMPEROIS	RXICMPGOIS	RXTCPEROIS	RXTCPGOIS	RXUDPEROIS	RXUDPGOIS	RXIPV6NOPAYOIS	RXIPV6HEROIS	RXIPV6GOIS	RXIPV4UDSBLOIS	RXIPV4FRAGOIS	RXIPV4NOPAYOIS	RXIPV4HEROIS	RXIPV4GOIS
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	RXICMPERPIS	RXICMPGPIS	RXTCPERPIS	RXTCPGPIS	RXUDPERPIS	RXUDPGPIS	RXIPV6NOPAYPIS	RXIPV6HERPIS	RXIPV6GPIS	RXIPV4UDSBLPIS	RXIPV4FRAGPIS	RXIPV4NOPAYPIS	RXIPV4HERPIS	RXIPV4GPIS
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1084. MMC IPC Receive Interrupt Register (MMC_IPC_RX_INTERRUPT)

Table 1136. MMC_IPC_RX_INTERRUPT field descriptions

Name	Description
2 RXICMPEROIS	MMC Receive ICMP Error Octet Counter Interrupt Status This bit is set when the rxicmp_err_octets counter reaches half of the maximum value or the maximum value. Access restriction applies. Clears on read. Self-set to 1 on internal event.
3 RXICMPGOIS	MMC Receive ICMP Good Octet Counter Interrupt Status This bit is set when the rxicmp_gd_octets counter reaches half of the maximum value or the maximum value. Access restriction applies. Clears on read. Self-set to 1 on internal event.
4 RXTCPEROIS	MMC Receive TCP Error Octet Counter Interrupt Status This bit is set when the rxtcp_err_octets counter reaches half of the maximum value or the maximum value. Access restriction applies. Clears on read. Self-set to 1 on internal event.
5 RXTCPGOIS	MMC Receive TCP Good Octet Counter Interrupt Status This bit is set when the rxtcp_gd_octets counter reaches half of the maximum value or the maximum value. Access restriction applies. Clears on read. Self-set to 1 on internal event.
6 RXUDPEROIS	MMC Receive UDP Error Octet Counter Interrupt Status This bit is set when the rxudp_err_octets counter reaches half of the maximum value or the maximum value. Access restriction applies. Clears on read. Self-set to 1 on internal event.

Table 1136. MMC_IPC_RX_INTERRUPT field descriptions (continued)

Name	Description
7 RXUDPGOIS	MMC Receive UDP Good Octet Counter Interrupt Status This bit is set when the rxudp_gd_octets counter reaches half of the maximum value or the maximum value. Access restriction applies. Clears on read. Self-set to 1 on internal event.
8 RXIPV6NOPAYOIS	MMC Receive IPV6 No Payload Octet Counter Interrupt Status This bit is set when the rxipv6_nopay_octets counter reaches half of the maximum value or the maximum value. Access restriction applies. Clears on read. Self-set to 1 on internal event.
9 RXIPV6HEROIS	MMC Receive IPV6 Header Error Octet Counter Interrupt Status This bit is set when the rxipv6_hdrerr_octets counter reaches half of the maximum value or the maximum value. Access restriction applies. Clears on read. Self-set to 1 on internal event.
10 RXIPV6GOIS	MMC Receive IPV6 Good Octet Counter Interrupt Status This bit is set when the rxipv6_gd_octets counter reaches half of the maximum value or the maximum value. Access restriction applies. Clears on read. Self-set to 1 on internal event.
11 RXIPV4UDSBLOIS	MMC Receive IPV4 UDP Checksum Disabled Octet Counter Interrupt Status This bit is set when the rxipv4_udsbl_octets counter reaches half of the maximum value or the maximum value. Access restriction applies. Clears on read. Self-set to 1 on internal event.
12 RXIPV4FRAGOIS	MMC Receive IPV4 Fragmented Octet Counter Interrupt Status This bit is set when the rxipv4_frag_octets counter reaches half of the maximum value or the maximum value. Access restriction applies. Clears on read. Self-set to 1 on internal event.
13 RXIPV4NOPAYOIS	MMC Receive IPV4 No Payload Octet Counter Interrupt Status This bit is set when the rxipv4_nopay_octets counter reaches half of the maximum value or the maximum value. Access restriction applies. Clears on read. Self-set to 1 on internal event.
14 RXIPV4HEROIS	MMC Receive IPV4 Header Error Octet Counter Interrupt Status This bit is set when the rxipv4_hdrerr_octets counter reaches half of the maximum value or the maximum value. Access restriction applies. Clears on read. Self-set to 1 on internal event.
15 RXIPV4GOIS	MMC Receive IPV4 Good Octet Counter Interrupt Status This bit is set when the rxipv4_gd_octets counter reaches half of the maximum value or the maximum value. Access restriction applies. Clears on read. Self-set to 1 on internal event.
18 RXICMPERPIS	MMC Receive ICMP Error Packet Counter Interrupt Status This bit is set when the rxicmp_err_pkts counter reaches half of the maximum value or the maximum value. Access restriction applies. Clears on read. Self-set to 1 on internal event.

Table 1136. MMC_IPC_RX_INTERRUPT field descriptions (continued)

Name	Description
19 RXICMPGPIS	MMC Receive ICMP Good Packet Counter Interrupt Status This bit is set when the rxicmp_gd_pkts counter reaches half of the maximum value or the maximum value. Access restriction applies. Clears on read. Self-set to 1 on internal event.
20 RXTCPERPIS	MMC Receive TCP Error Packet Counter Interrupt Status This bit is set when the rxtcp_err_pkts counter reaches half of the maximum value or the maximum value. Access restriction applies. Clears on read. Self-set to 1 on internal event.
21 RXTCPGPIS	MMC Receive TCP Good Packet Counter Interrupt Status This bit is set when the rxtcp_gd_pkts counter reaches half of the maximum value or the maximum value. Access restriction applies. Clears on read. Self-set to 1 on internal event.
22 RXUDPERPIS	MMC Receive UDP Error Packet Counter Interrupt Status This bit is set when the rxudp_err_pkts counter reaches half of the maximum value or the maximum value. Access restriction applies. Clears on read. Self-set to 1 on internal event.
23 RXUDPGPIS	MC Receive UDP Good Packet Counter Interrupt Status This bit is set when the rxudp_gd_pkts counter reaches half of the maximum value or the maximum value. Access restriction applies. Clears on read. Self-set to 1 on internal event.
24 RXIPV6NOPAYPIS	MMC Receive IPV6 No Payload Packet Counter Interrupt Status This bit is set when the rxipv6_nopay_pkts counter reaches half of the maximum value or the maximum value. Access restriction applies. Clears on read. Self-set to 1 on internal event.
25 RXIPV6HERPIS	MMC Receive IPV6 Header Error Packet Counter Interrupt Status This bit is set when the rxipv6_hdrerr_pkts counter reaches half of the maximum value or the maximum value. Access restriction applies. Clears on read. Self-set to 1 on internal event.
26 RXIPV6GPIS	MMC Receive IPV6 Good Packet Counter Interrupt Status This bit is set when the rxipv6_gd_pkts counter reaches half of the maximum value or the maximum value. Access restriction applies. Clears on read. Self-set to 1 on internal event.
27 RXIPV4UDSBLPIS	MMC Receive IPV4 UDP Checksum Disabled Packet Counter Interrupt Status This bit is set when the rxipv4_udsbl_pkts counter reaches half of the maximum value or the maximum value.
28 RXIPV4FRAGPIS	MMC Receive IPV4 Fragmented Packet Counter Interrupt Status This bit is set when the rxipv4_frag_pkts counter reaches half of the maximum value or the maximum value. Access restriction applies. Clears on read. Self-set to 1 on internal event.
29 RXIPV4NOPAYPIS	MMC Receive IPV4 No Payload Packet Counter Interrupt Status This bit is set when the rxipv4_nopay_pkts counter reaches half of the maximum value or the maximum value. Access restriction applies. Clears on read. Self-set to 1 on internal event.

Table 1136. MMC_IPC_RX_INTERRUPT field descriptions (continued)

Name	Description
30 RXIPV4HERPIS	MMC Receive IPV4 Header Error Packet Counter Interrupt Status This bit is set when the rxipv4_hdrerr_pkts counter reaches half of the maximum value or the maximum value. Access restriction applies. Clears on read. Self-set to 1 on internal event.
31 RXIPV4GPIS	MMC Receive IPV4 Good Packet Counter Interrupt Status This bit is set when the rxipv4_gd_pkts counter reaches half of the maximum value or the maximum value. Access restriction applies. Clears on read. Self-set to 1 on internal event.

56.2.106 Receive IPv4 Good Packets Register (RXIPV4_GOOD_PACKETS)

This register provides the number of good IPv4 datagrams received by the Ethernet module with the TCP, UDP, or ICMP payload.

Offset 0x0810 Access: User Read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	RXIPV4GDPKT															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	RXIPV4GDPKT															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1085. Receive IPv4 Good Packets Register (RXIPV4_GOOD_PACKETS)

Table 1137. RXIPV4_GOOD_PACKETS field descriptions

Field	Description
0:31 RXIPV4GDPKT	This field indicates the number of good IPv4 datagrams received with the TCP, UDP, or ICMP payload.

56.2.107 Receive IPv4 Header Error Packets (RXIPV4_HEADER_ERROR_PACKETS)

This register provides the number of IPv4 datagrams received by the Ethernet module with header (checksum, length, or version mismatch) errors.

Offset 0x0814 Access: User Read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	RXIPV4HDRERRPKT															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	RXIPV4HDRERRPKT															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1086. Receive IPv4 Header Error Packets (RXIPV4_HEADER_ERROR_PACKETS)

Table 1138. RXIPV4_HEADER_ERROR_PACKETS field descriptions

Field	Description
0:31 RXIPV4HDRERRPKT	This field indicates the number of IPv4 datagrams received with header (checksum, length, or version mismatch) errors.

56.2.108 Receive IPv4 No Payload Packets Register (RXIPV4_NO_PAYLOAD_PACKETS)

This register provides the number of IPv4 datagram packets received by the Ethernet module that did not have a TCP, UDP, or ICMP payload.

Offset 0x0818 Access: User Read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	RXIPV4NOPAYPKT															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	RXIPV4NOPAYPKT															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1087. Receive IPv4 No Payload Packets Register (RXIPV4_NO_PAYLOAD_PACKETS)

Table 1139. RXIPV4_NO_PAYLOAD_PACKETS field descriptions

Field	Description
0:31 RXIPV4NOPAYPKT	This field indicates the number of IPv4 datagram packets received that did not have a TCP, UDP, or ICMP payload.

56.2.109 Receive IPv4 Fragmented Packets Register (RXIPV4_FRAGMENTED_PACKETS)

This register provides the number of good IPv4 datagrams received by the Ethernet module with fragmentation.

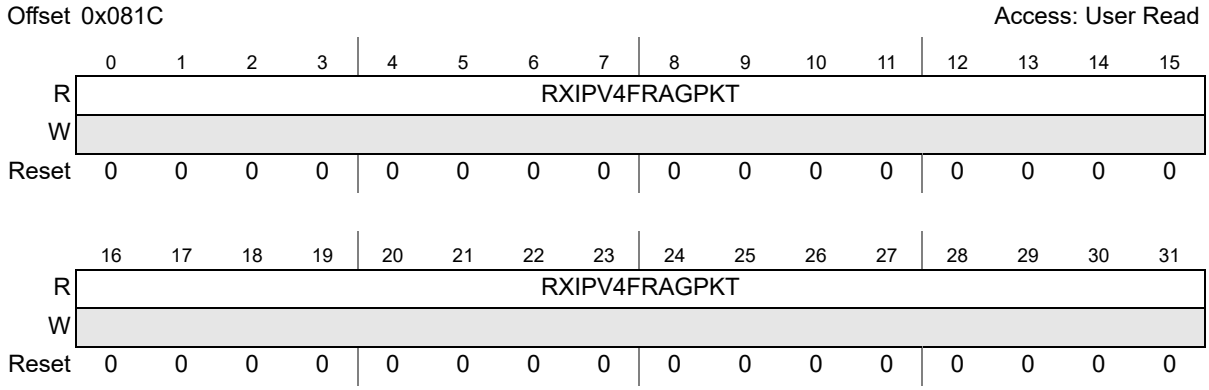


Figure 1088. Receive IPv4 Fragmented Packets Register (RXIPV4_FRAGMENTED_PACKETS)

Table 1140. RXIPV4_FRAGMENTED_PACKETS field descriptions

Field	Description
0:31 RXIPV4FRAGPKT	This field indicates the number of good IPv4 datagrams received with fragmentation.

56.2.110 Receive IPv4 UDP Checksum Disabled Packets Register (RXIPV4_UDP_CHECKSUM_DISABLED_PACKETS)

This register provides the number of good IPv4 datagrams received by the Ethernet module that had a UDP payload with checksum disabled.

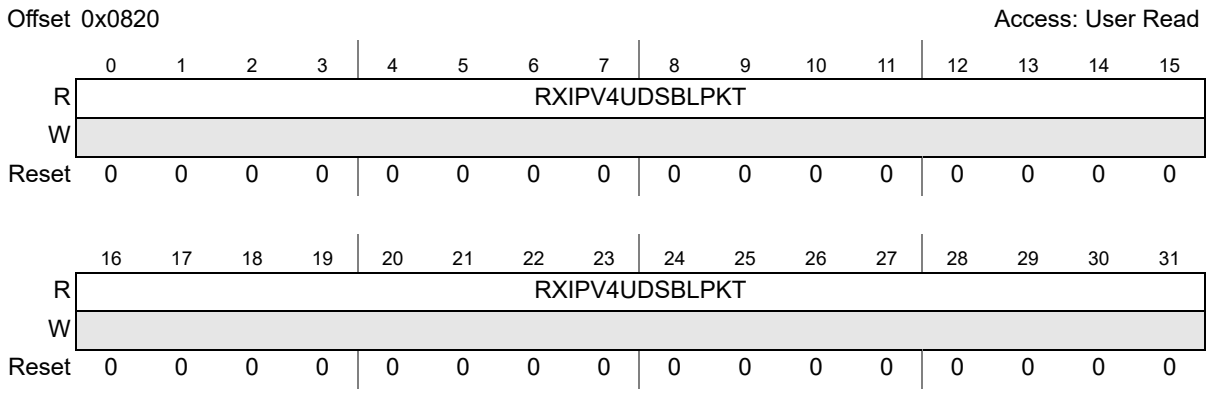


Figure 1089. Receive IPv4 UDP Checksum Disabled Packets Register (RXIPV4_UDP_CHECKSUM_DISABLED_PACKETS)

Table 1141. RXIPV4_UDP_CHECKSUM_DISABLED_PACKETS field descriptions

Field	Description
0:31 RXIPV4UDSBLPKT	This field indicates the number of good IPv4 datagrams received that had a UDP payload with checksum disabled.

56.2.111 Receive IPv6 Good Packets Register (RXIPV6_GOOD_PACKETS)

This register provides the number of good IPv6 datagrams received by the Ethernet module with the TCP, UDP, or ICMP payload.

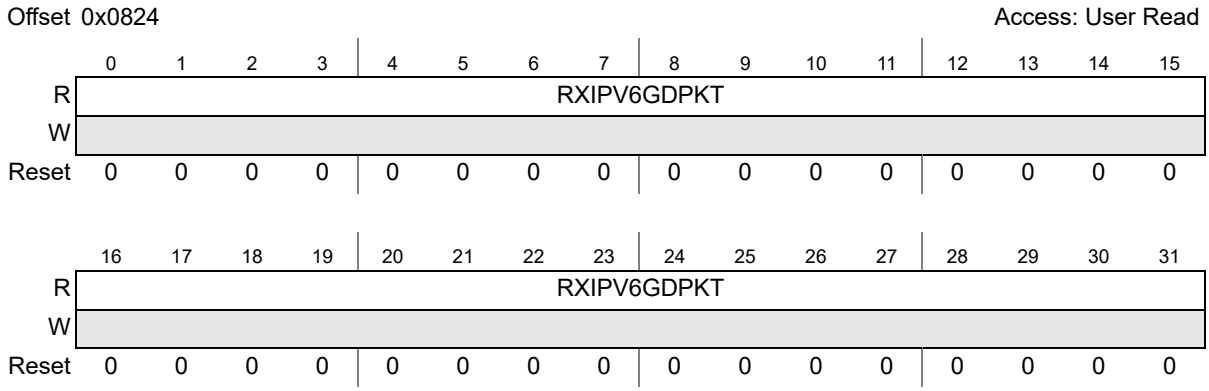


Figure 1090. Receive IPv6 Good Packets Register (RXIPV6_GOOD_PACKETS)

Table 1142. RXIPV6_GOOD_PACKETS field descriptions

Field	Description
0:31 RXIPV6GDPKT	This field indicates the number of good IPv6 datagrams received with the TCP, UDP, or ICMP payload.

56.2.112 Receive IPv6 Header Error Packets (RXIPV6_HEADER_ERROR_PACKETS)

This register provides the number of IPv6 datagrams received by the Ethernet module with header (length or version mismatch) errors.

Offset 0x0828 Access: User Read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	RXIPV6HDRERRPKT															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	RXIPV6HDRERRPKT															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1091. Receive IPv6 Header Error Packets (RXIPV6_HEADER_ERROR_PACKETS)

Table 1143. RXIPV6_HEADER_ERROR_PACKETS field descriptions

Field	Description
0:31 RXIPV6HDRERRPKT	This field indicates the number of IPv6 datagrams received with header (length or version mismatch) errors.

56.2.113 Receive IPv6 Payload Packets Register (RXIPV6_NO_PAYLOAD_PACKETS)

This register provides the number of IPv6 datagram packets received by the Ethernet module that did not have a TCP, UDP, or ICMP payload. This includes all IPv6 datagrams with fragmentation or security extension headers.

Offset 0x082C Access: User Read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	RXIPV6NOPAYPKT															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	RXIPV6NOPAYPKT															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1092. Receive IPv6 Payload Packets Register (RXIPV6_NO_PAYLOAD_PACKETS)

Table 1144. RXIPV6_NO_PAYLOAD_PACKETS field descriptions

Field	Description
0:31 RXIPV6NOPAYPKT	This field indicates the number of IPv6 datagram packets received that did not have a TCP, UDP, or ICMP payload. This includes all IPv6 datagrams with fragmentation or security extension headers.

56.2.114 Receive UDP Good Packets Register (RXUDP_GOOD_PACKETS)

This register provides the number of good IP datagrams received by the Ethernet module with a good UDP payload. This counter is not updated when the RxIPv4_UDP_Checksum_Disabled_Packets counter is incremented.

Offset 0x0830 Access: User Read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	RXUDPGDPKT															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	RXUDPGDPKT															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1093. Receive UDP Good Packets Register (RXUDP_GOOD_PACKETS)

Table 1145. RXUDP_GOOD_PACKETS field descriptions

Field	Description
0:31 RXUDPGDPKT	This field indicates the number of good IP datagrams received with a good UDP payload. This counter is not updated when the RxIPv4_UDP_Checksum_Disabled_Packets counter is incremented.

56.2.115 Receive UDP Error Packets Register (RXUDP_ERROR_PACKETS)

This register provides the number of good IP datagrams received by the Ethernet module whose UDP payload has a checksum error.

Offset 0x0834 Access: User Read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	RXUDPERRPKT															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	RXUDPERRPKT															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1094. Receive UDP Error Packets Register (RXUDP_ERROR_PACKETS)

Table 1146. RXUDP_ERROR_PACKETS field descriptions

Field	Description
0:31 RXUDPERRPKT	This field indicates the number of good IP datagrams received whose UDP payload has a checksum error.

56.2.116 Receive TCP Good Packets Register (RXTCP_GOOD_PACKETS)

This register provides the number of good IP datagrams received by the Ethernet module with a good TCP payload.

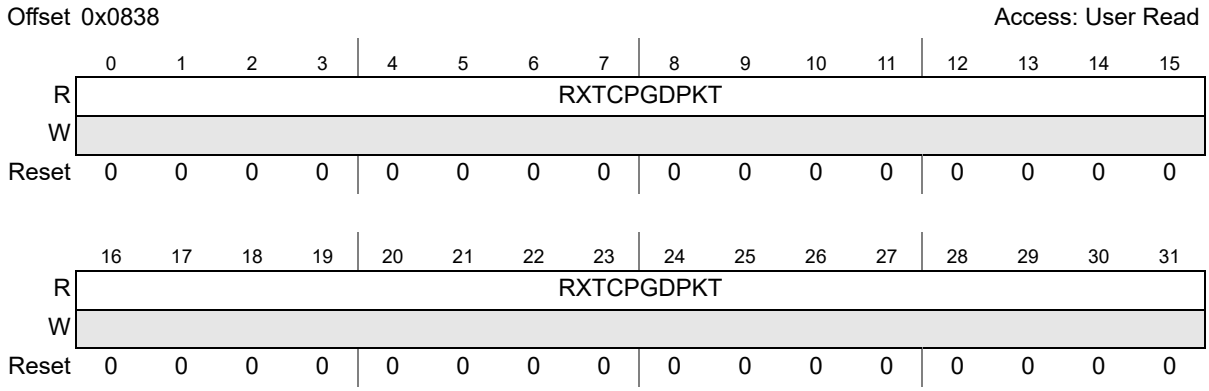


Figure 1095. Receive TCP Good Packets Register (RXTCP_GOOD_PACKETS)

Table 1147. RXTCP_GOOD_PACKETS field descriptions

Field	Description
0:31 RXTCPGDPKT	This field indicates the number of good IP datagrams received with a good TCP payload.

56.2.117 Receive TCP Error Packets Register (RXTCP_ERROR_PACKETS)

This register provides the number of good IP datagrams received by the Ethernet module whose TCP payload has a checksum error.

Offset 0x083C Access: User Read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	RXTCPERRPKT															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	RXTCPERRPKT															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1096. Receive TCP Error Packets Register (RXTCP_ERROR_PACKETS)

Table 1148. RXTCP_ERROR_PACKETS field descriptions

Field	Description
0:31 RXTCPERRPKT	This field indicates the number of good IP datagrams received whose TCP payload has a checksum error.

56.2.118 Receive ICMP Good Packets Register (RXICMP_GOOD_PACKETS)

This register provides the number of good IP datagrams received by the Ethernet module with a good ICMP payload.

Offset 0x0840 Access: User Read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	RXICMPGDPKT															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	RXICMPGDPKT															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1097. Receive ICMP Good Packets Register (RXICMP_GOOD_PACKETS)

Table 1149. RXICMP_GOOD_PACKETS field descriptions

Field	Description
0:31 RXICMPGDPKT	This field indicates the number of good IP datagrams received with a good ICMP payload

56.2.119 Receive ICMP Error Packets Register (RXICMP_ERROR_PACKETS)

This register provides the number of good IP datagrams received by the Ethernet module whose ICMP payload has a checksum error.

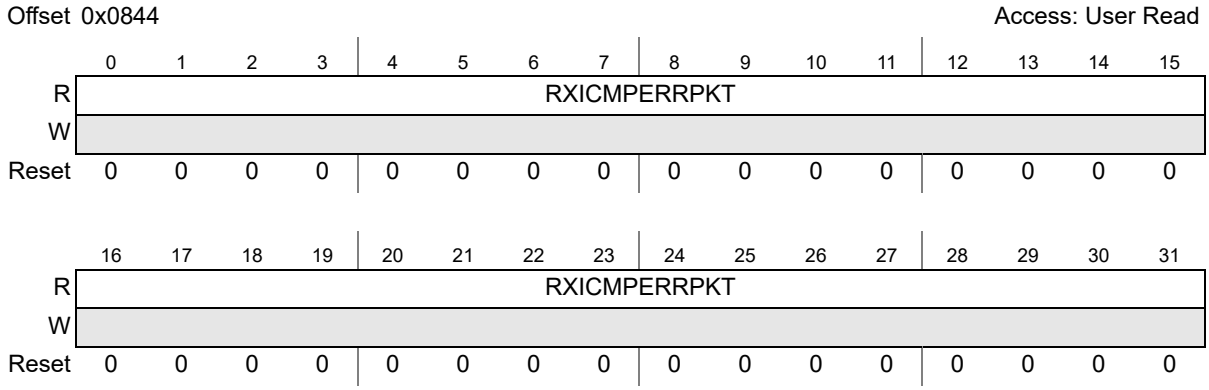


Figure 1098. Receive ICMP Error Packets Register (RXICMP_ERROR_PACKETS)

Table 1150. RXICMP_ERROR_PACKETS field descriptions

Field	Description
0:31 RXICMPERRPKT	This field indicates the number of good IP datagrams received whose ICMP payload has a checksum error.

56.2.120 Receive IPv4 Good Octets Register (RXIPV4_GOOD_OCTETS)

This register provides the number of bytes received by the Ethernet module in good IPv4 datagrams encapsulating TCP, UDP, or ICMP data. Ethernet header, FCS, pad, or IP pad bytes are not included in this counter.

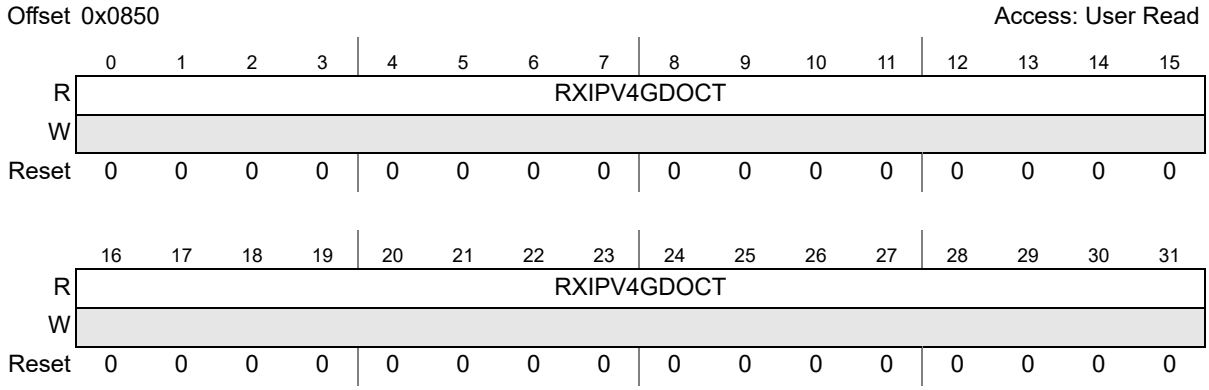


Figure 1099. Receive IPv4 Good Octets Register (RXIPV4_GOOD_OCTETS)

Table 1151. RXIPV4_GOOD_OCTETS field descriptions

Field	Description
0:31 RXIPV4GDOCT	This field indicates the number of bytes received in good IPv4 datagrams encapsulating TCP, UDP, or ICMP data. Ethernet header, FCS, pad, or IP pad bytes are not included in this counter.

56.2.121 Receive IPv4 Header Error Octets Register (RXIPV4_HEADER_ERROR_OCTETS)

This register provides the number of bytes received by the Ethernet module in IPv4 datagrams with header errors (checksum, length, version mismatch). The value in the Length field of IPv4 header is used to update this counter. Ethernet header, FCS, pad, or IP pad bytes are not included in this counter.

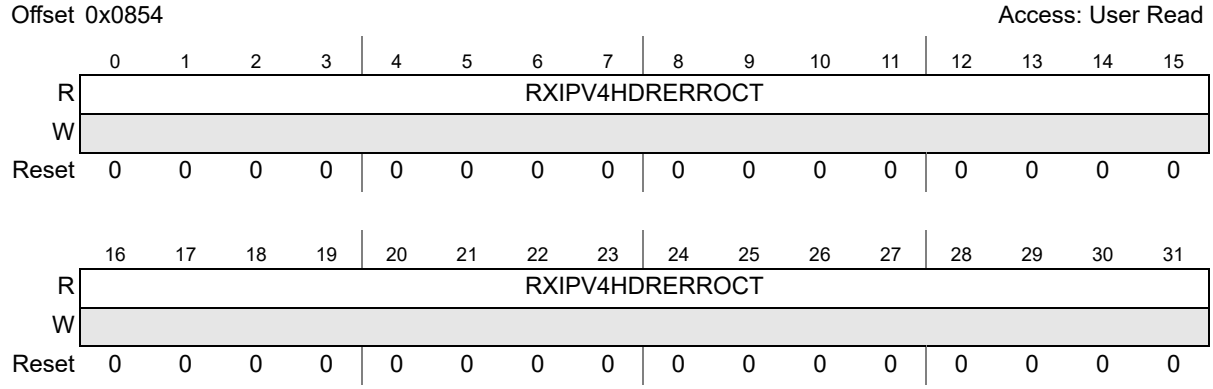


Figure 1100. Receive IPv4 Header Error Octets Register (RXIPV4_HEADER_ERROR_OCTETS)

Table 1152. RXIPV4_HEADER_ERROR_OCTETS field descriptions

Field	Description
0:31 RXIPV4HDRERROCT	This field indicates the number of bytes received in IPv4 datagrams with header errors (checksum, length, version mismatch). The value in the Length field of IPv4 header is used to update this counter. Ethernet header, FCS, pad, or IP pad bytes are not included in this counter.

56.2.122 Receive IPv4 No Payload Octets Register (RXIPV4_NO_PAYLOAD_OCTETS)

This register provides the number of bytes received by the Ethernet module in IPv4 datagrams that did not have a TCP, UDP, or ICMP payload. The value in the Length field of IPv4 header is used to update this counter. Ethernet header, FCS, pad, or IP pad bytes are not included in this counter.

Offset 0x0858 Access: User Read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	RXIPV4NOPAYOCT															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	RXIPV4NOPAYOCT															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1101. Receive IPv4 No Payload Octets Register (RXIPV4_NO_PAYLOAD_OCTETS)

Table 1153. RXIPV4_NO_PAYLOAD_OCTETS field descriptions

Field	Description
0:31 RXIPV4NOPAYOCT	This field indicates the number of bytes received in IPv4 datagrams that did not have a TCP, UDP, or ICMP payload. The value in the Length field of IPv4 header is used to update this counter. Ethernet header, FCS, pad, or IP pad bytes are not included in this counter.

56.2.123 Receive IPv4 Fragmented Octets Register (RXIPV4_FRAGMENTED_OCTETS)

This register provides the number of bytes received by the Ethernet module in fragmented IPv4 datagrams. The value in the Length field of IPv4 header is used to update this counter. Ethernet header, FCS, pad, or IP pad bytes are not included in this counter.

Offset 0x085C Access: User Read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	RXIPV4FRAGOCT															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	RXIPV4FRAGOCT															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1102. Receive IPv4 Fragmented Octets Register (RXIPV4_FRAGMENTED_OCTETS)

Table 1154. RXIPV4_FRAGMENTED_OCTETS field descriptions

Field	Description
0:31 RXIPV4FRAGOCT	This field indicates the number of bytes received in fragmented IPv4 datagrams. The value in the Length field of IPv4 header is used to update this counter. Ethernet header, FCS, pad, or IP pad bytes are not included in this counter.

56.2.124 Receive IPv4 UDP Checksum Disable Octets Register (RXIPV4_UDP_CHECKSUM_DISABLE_OCTETS)

This register provides the number of bytes received by the Ethernet module in a UDP segment that had the UDP checksum disabled. This counter does not count IP Header bytes. Ethernet header, FCS, pad, or IP pad bytes are not included in this counter.

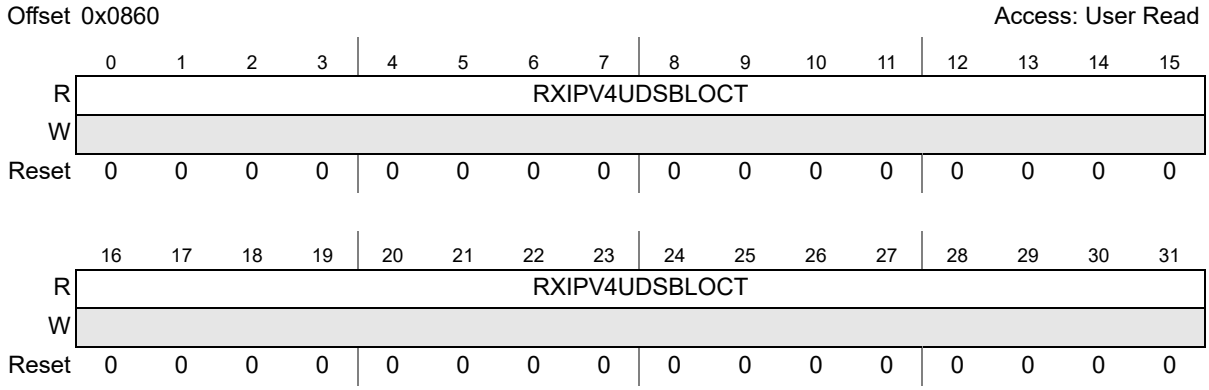


Figure 1103. Receive IPv4 UDP Checksum Disable Octets Register (RXIPV4_UDP_CHECKSUM_DISABLE_OCTETS)

Table 1155. RXIPV4_UDP_CHECKSUM_DISABLE_OCTETS field descriptions

Field	Description
0:31 RXIPV4UDSBLOCT	This field indicates the number of bytes received in a UDP segment that had the UDP checksum disabled. This counter does not count IP Header bytes. Ethernet header, FCS, pad, or IP pad bytes are not included in this counter.

56.2.125 Receive IPv6 Good Octets Register (RXIPV6_GOOD_OCTETS)

This register provides the number of bytes received by the Ethernet module in good IPv6 datagrams encapsulating TCP, UDP, or ICMP data. Ethernet header, FCS, pad, or IP pad bytes are not included in this counter.

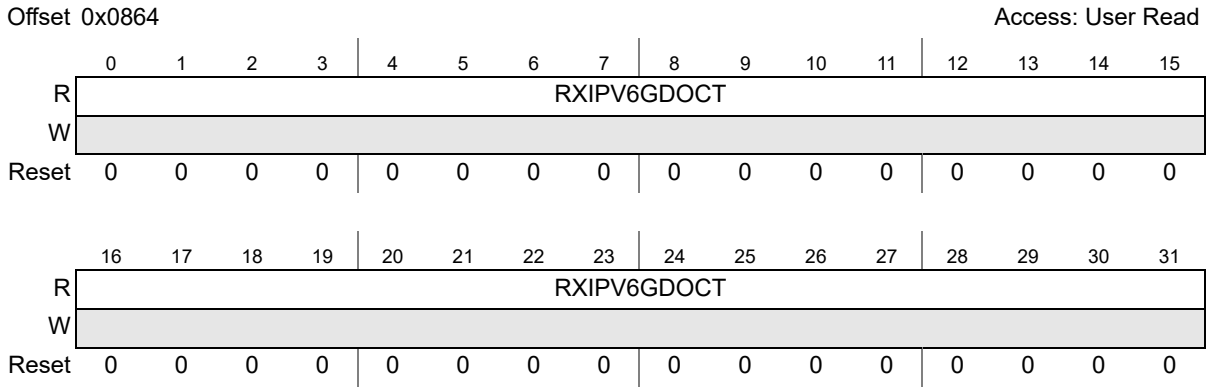


Figure 1104. Receive IPv6 Good Octets Register (RXIPV6_GOOD_OCTETS)

Table 1156. RXIPV6_GOOD_OCTETS field descriptions

Field	Description
0:31 RXIPV6GDOCT	This field indicates the number of bytes received in good IPv6 datagrams encapsulating TCP, UDP, or ICMP data. Ethernet header, FCS, pad, or IP pad bytes are not included in this counter.

56.2.126 Receive Header Error Octets Register (RXIPV6_HEADER_ERROR_OCTETS)

This register provides the number of bytes received by the Ethernet module in IPv6 datagrams with header errors (length, version mismatch). The value in the Length field of IPv6 header is used to update this counter. Ethernet header, FCS, pad, or IP pad bytes are not included in this counter.

Offset 0x0868 Access: User Read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	RXIPV6HDRERROCT															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	RXIPV6HDRERROCT															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1105. Receive Header Error Octets Register (RXIPV6_HEADER_ERROR_OCTETS)

Table 1157. RXIPV6_HEADER_ERROR_OCTETS field descriptions

Field	Description
0:31 RXIPV6HDRERROCT	This field indicates the number of bytes received in IPv6 datagrams with header errors (length, version mismatch). The value in the Length field of IPv6 header is used to update this counter. Ethernet header, FCS, pad, or IP pad bytes are not included in this counter.

56.2.127 Receive IPv6 No Payload Octets Register (RXIPV6_NO_PAYLOAD_OCTETS)

This register provides the number of bytes received by the Ethernet module in IPv6 datagrams that did not have a TCP, UDP, or ICMP payload. The value in the Length field of IPv6 header is used to update this counter. Ethernet header, FCS, pad, or IP pad bytes are not included in this counter.

Offset 0x086C Access: User Read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	RXIPV6NOPAYOCT															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	RXIPV6NOPAYOCT															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1106. Receive IPv6 No Payload Octets Register (RXIPV6_NO_PAYLOAD_OCTETS)

Table 1158. RXIPV6_NO_PAYLOAD_OCTETS field descriptions

Field	Description
0:31 RXIPV6NOPAYOCT	This field indicates the number of bytes received in IPv6 datagrams that did not have a TCP, UDP, or ICMP payload. The value in the Length field of IPv6 header is used to update this counter. Ethernet header, FCS, pad, or IP pad bytes are not included in this counter.

56.2.128 Receive UDP Good Octets Register (RXUDP_GOOD_OCTETS)

This register provides the number of bytes received by the Ethernet module in a good UDP segment. This counter does not count IP header bytes.

Offset 0x0870 Access: User Read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	RXUDPGDOCT															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	RXUDPGDOCT															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1107. Receive UDP Good Octets Register (RXUDP_GOOD_OCTETS)

Table 1159. RXUDP_GOOD_OCTETS field descriptions

Field	Description
0:31 RXUDPGDOCT	This field indicates the number of bytes received in a good UDP segment. This counter does not count IP header bytes.

56.2.129 Receive UDP Error Octets Register (RXUDP_ERROR_OCTETS)

This register provides the number of bytes received by the Ethernet module in a UDP segment that had checksum errors. This counter does not count IP header bytes.

Offset 0x0874 Access: User Read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	RXUDPERROCT															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	RXUDPERROCT															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1108. Receive UDP Error Octets Register (RXUDP_ERROR_OCTETS)

Table 1160. RXUDP_ERROR_OCTETS field descriptions

Field	Description
0:31 RXUDPERROCT	This field indicates the number of bytes received in a UDP segment that had checksum errors. This counter does not count IP header bytes.

56.2.130 Receive TCP Good Octets Register (RXTCP_GOOD_OCTETS)

This register provides the number of bytes received by the Ethernet module in a good TCP segment. This counter does not count IP header bytes.

Offset 0x0878 Access: User Read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	RXTCPGDOCT															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	RXTCPGDOCT															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1109. Receive TCP Good Octets Register (RXTCP_GOOD_OCTETS)

Table 1161. RXTCP_GOOD_OCTETS field descriptions

Field	Description
0:31 RXTCPGDOCT	This field indicates the number of bytes received in a good TCP segment. This counter does not count IP header bytes.

56.2.131 Receive TCP Error Octets Register (RXTCP_ERROR_OCTETS)

This register provides the number of bytes received by the Ethernet module in a TCP segment that had checksum errors. This counter does not count IP header bytes.

Offset 0x087C Access: User Read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	RXTCPERROCT															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	RXTCPERROCT															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1110. Receive TCP Error Octets Register (RXTCP_ERROR_OCTETS)

Table 1162. RXTCP_ERROR_OCTETS field descriptions

Field	Description
0:31 RXTCPERROCT	This field indicates the number of bytes received in a TCP segment that had checksum errors. This counter does not count IP header bytes.

56.2.132 Receive ICMP Good Octets Register (RXICMP_GOOD_OCTETS)

This register provides the number of bytes received by the Ethernet module in a good ICMP segment. This counter does not count IP header bytes.

Offset 0x0880 Access: User Read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	RXICMPGDOCT															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	RXICMPGDOCT															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1111. Receive ICMP Good Octets Register (RXICMP_GOOD_OCTETS)

Table 1163. RXICMP_GOOD_OCTETS field descriptions

Field	Description
0:31 RXICMPGDOCT	This field indicates the number of bytes received in a good ICMP segment. This counter does not count IP header bytes.

56.2.133 Receive ICMP Error Octets Register (RXICMP_ERROR_OCTETS)

This register provides the number of bytes received by the Ethernet module in a ICMP segment that had checksum errors. This counter does not count IP header bytes.

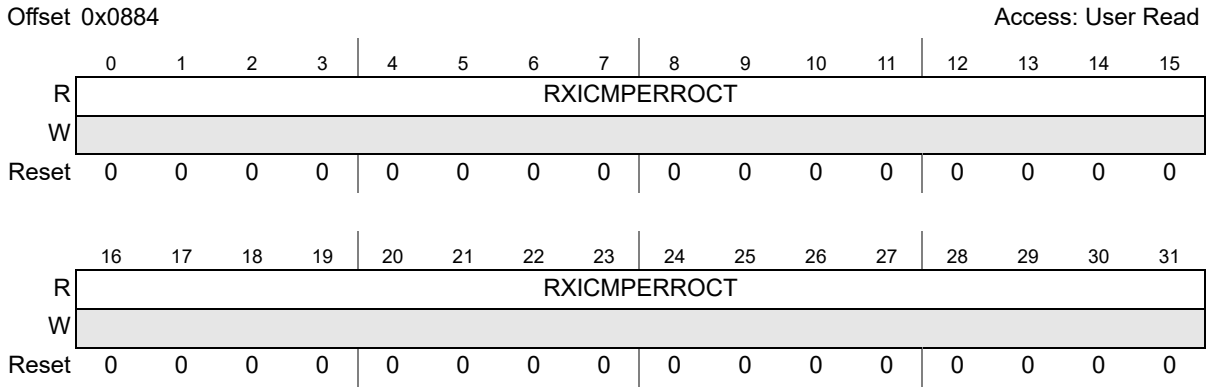


Figure 1112. Receive ICMP Error Octets Register (RXICMP_ERROR_OCTETS)

Table 1164. RXICMP_ERROR_OCTETS field descriptions

Field	Description
0:31 RXICMPERROCT	This field indicates the number of bytes received in a ICMP segment that had checksum errors. This counter does not count IP header bytes.

56.2.134 Layer 3 and Layer 4 Control 0 Register (MAC_L3_L4_CONTROL0)

The Layer 3 and Layer 4 Control register controls the operations of filter 0 of Layer 3 and Layer 4.

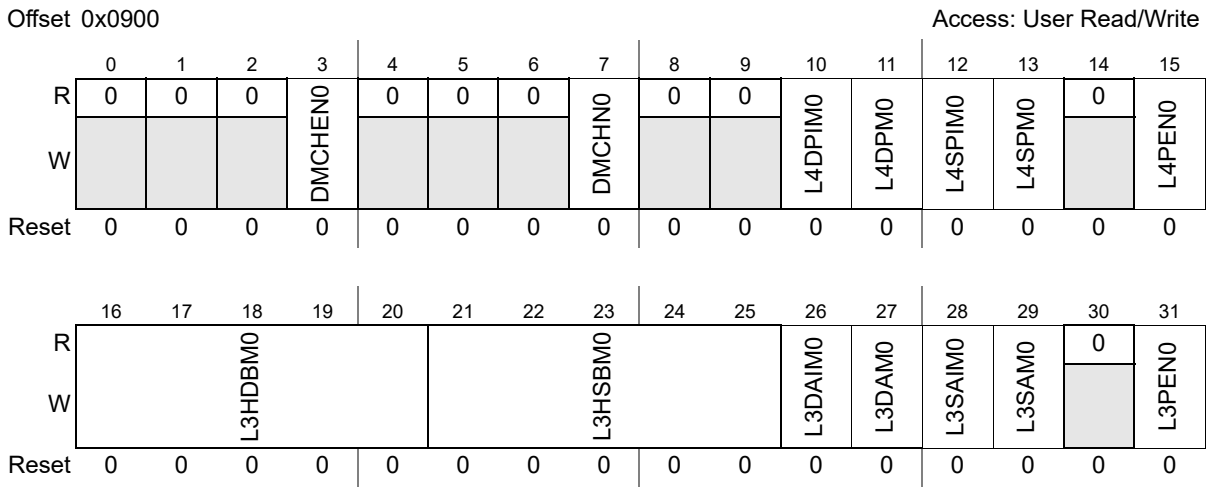


Figure 1113. Layer 3 and Layer 4 Control 0 Register (MAC_L3_L4_CONTROL0)

Table 1165. MAC_L3_L4_CONTROL0 field descriptions

Field	Description
3 DMCHEN0	DMA Channel Select Enable When set, this bit enables the selection of the DMA channel number for the packet that is passed by this L3_L4 filter. The DMA channel is indicated by the DMCHN bits. When this bit is reset, the DMA channel is not decided by this filter.
7 DMCHN0	DMA Channel Number When DMCHEN is set high, this field selects the DMA Channel number to which the packet passed by this filter is routed. 0 Packet is passed by this filter to DMA CH 0 1 Packet is passed by this filter to DMA CH 1
10 L4DPIM0	Layer 4 Destination Port Inverse Match Enable When this bit is set, the Layer 4 Destination Port number field is enabled for inverse matching. When this bit is reset, the Layer 4 Destination Port number field is enabled for perfect matching. This bit is valid and applicable only when the L4DPM0 bit is set high.
11 L4DPM0	Layer 4 Destination Port Match Enable When this bit is set, the Layer 4 Destination Port number field is enabled for matching. When this bit is reset, the MAC ignores the Layer 4 Destination Port number field for matching.
12 L4SPIM0	Layer 4 Source Port Inverse Match Enable When this bit is set, the Layer 4 Source Port number field is enabled for inverse matching. When this bit is reset, the Layer 4 Source Port number field is enabled for perfect matching. This bit is valid and applicable only when the L4SPM0 bit is set high.
13 L4SPM0	Layer 4 Source Port Match Enable When this bit is set, the Layer 4 Source Port number field is enabled for matching. When this bit is reset, the MAC ignores the Layer 4 Source Port number field for matching.
15 L4PEN0	Layer 4 Protocol Enable When this bit is set, the Source and Destination Port number fields of UDP packets are used for matching. When this bit is reset, the Source and Destination Port number fields of TCP packets are used for matching. The Layer 4 matching is done only when the L4SPM0 or L4DPM0 bit is set.

Table 1165. MAC_L3_L4_CONTROL0 field descriptions (continued)

Field	Description
16:20 L3HDBM0	Layer 3 IP DA Higher Bits Match **IPv4 Packets:** This field contains the number of higher bits of IP Destination Address that are matched in the IPv4 packets. The following list describes the values of this field: 0x00 No bits are masked. 0x01 1 LSb is masked 0x02 Two LSbs are masked ... 0x1F All bits except MSb are masked. **IPv6 Packets:** Bits[29:30] of this field correspond to Bits[25:26] of L3HSBM0 which indicate the number of lower bits of IP Source or Destination Address that are masked in the IPv6 packets. The following list describes the concatenated values of the L3HDBM0[30:31] and L3HSBM0 bits: - 0x00 No bits are masked. - 0x01 1 LSb is masked. - 0x02 Two LSbs are masked - … - 0x7F All bits except MSb are masked. This field is valid and applicable only when the L3DAM0 or L3SAM0 bit is set.
21:25 L3HSBM0	Layer 3 IP SA Higher Bits Match **IPv4 Packets:** This field contains the number of lower bits of IP Source Address that are masked for matching in the IPv4 packets. The following list describes the values of this field: 0x00 No bits are masked. 0x01 1 LSb is masked 0x02 Two LSbs are masked … 0x1F All bits except MSb are masked. **IPv6 Packets:** This field contains Bits[27:31] of L3HSBM0. These bits indicate the number of higher bits of IP Source or Destination Address matched in the IPv6 packets. This field is valid and applicable only when the L3DAM0 or L3SAM0 bit is set high.
26 L3DAIM0	Layer 3 IP DA Inverse Match Enable When this bit is set, the Layer 3 IP Destination Address field is enabled for inverse matching. When this bit is reset, the Layer 3 IP Destination Address field is enabled for perfect matching. This bit is valid and applicable only when the L3DAM0 bit is set high.
27 L3DAM0	Layer 3 IP DA Match Enable When this bit is set, the Layer 3 IP Destination Address field is enabled for matching. When this bit is reset, the MAC ignores the Layer 3 IP Destination Address field for matching. **Note:** When the L3PEN0 bit is set, you should set either this bit or the L3SAM0 bit because either IPv6 DA or SA can be checked for filtering.

Table 1165. MAC_L3_L4_CONTROL0 field descriptions (continued)

Field	Description
28 L3SAIM0	Layer 3 IP SA Inverse Match Enable When this bit is set, the Layer 3 IP Source Address field is enabled for inverse matching. When this bit reset, the Layer 3 IP Source Address field is enabled for perfect matching. This bit is valid and applicable only when the L3SAM0 bit is set.
29 L3SAM0	Layer 3 IP SA Match Enable When this bit is set, the Layer 3 IP Source Address field is enabled for matching. When this bit is reset, the MAC ignores the Layer 3 IP Source Address field for matching. **Note:** When the L3PEN0 bit is set, you should set either this bit or the L3DAM0 bit because either IPv6 SA or DA can be checked for filtering.
31 L3PEN0	Layer 3 Protocol Enable When this bit is set, the Layer 3 IP Source or Destination Address matching is enabled for IPv6 packets. When this bit is reset, the Layer 3 IP Source or Destination Address matching is enabled for IPv4 packets. The Layer 3 matching is done only when the L3SAM0 or L3DAM0 bit is set. Value After Reset: 0x0

56.2.135 Layer 4 Address 0 Register (MAC_LAYER4_ADDRESS0)

This register holds the TCP source and destination port numbers against which a match is performed.

Offset 0x0904 Access: User Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	L4DP0															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	L4SP0															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1114. Layer4 Address 0 Register (MAC_LAYER4_ADDRESS0)

Table 1166. MAC_LAYER4_ADDRESS0 field descriptions

Field	Description
0:15 L4DP0	Layer 4 Destination Port Number Field When the L4PEN0 bit is reset and the L4DPM0 bit is set in the MAC_L3_L4_Control0 register, this field contains the value to be matched with the TCP Destination Port Number field in the IPv4 or IPv6 packets. When the L4PEN0 and L4DPM0 bits are set in MAC_L3_L4_Control0 register, this field contains the value to be matched with the UDP Destination Port Number field in the IPv4 or IPv6 packets.
16:31 L4SP0	Layer 4 Source Port Number Field When the L4PEN0 bit is reset and the L4DPM0 bit is set in the MAC_L3_L4_Control0 register, this field contains the value to be matched with the TCP Source Port Number field in the IPv4 or IPv6 packets. When the L4PEN0 and L4DPM0 bits are set in MAC_L3_L4_Control0 register, this field contains the value to be matched with the UDP Source Port Number field in the IPv4 or IPv6 packets.

56.2.136 Layer 3 Address 0 Register 0 Register (MAC_LAYER3_ADDRESS0_REG0)

For IPv4 packets, the Layer 3 Address 0 Register 0 register contains the 32-bit IP Source Address field. For IPv6 packets, it contains Bits[96:127] of the 128-bit IP Source Address or Destination Address field.

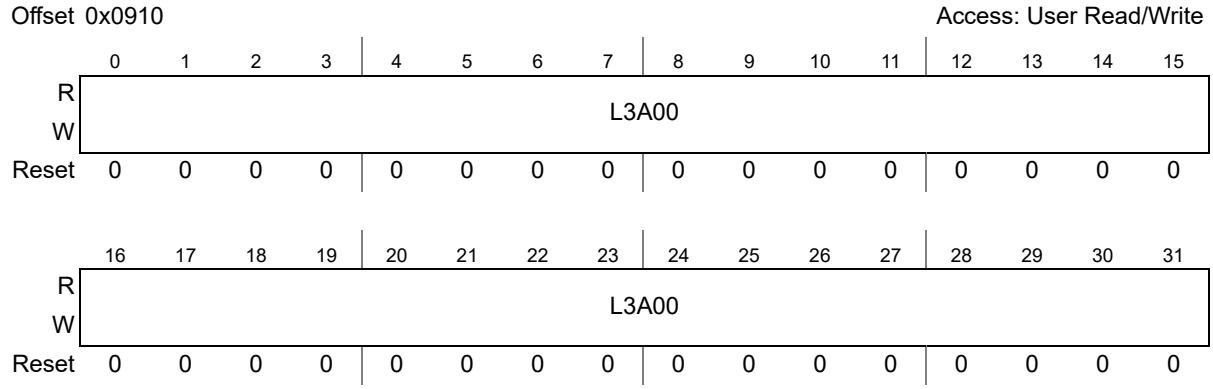


Figure 1115. Layer 3 Address 0 Register 0 Register (MAC_LAYER3_ADDRESS0_REG0)

Table 1167. MAC_LAYER3_ADDRESS0_REG0 field descriptions

Field	Description
0:31 L3A00	Layer 3 Address 0 Field When the L3PEN0 and L3SAM0 bits are set in the MAC_L3_L4_Control0 register, this field contains the value to be matched with Bits[96:127] of the IP Source Address field in the IPv6 packets. When the L3PEN0 and L3DAM0 bits are set in the MAC_L3_L4_Control0 register, this field contains the value to be matched with Bits[96:127] of the IP Destination Address field in the IPv6 packets. When the L3PEN0 bit is reset and the L3SAM0 bit is set in the MAC_L3_L4_Control0 register, this field contains the value to be matched with the IP Source Address field in the IPv4 packets.

56.2.137 Layer3 Address1 Register 0 Register (MAC_LAYER3_ADDRESS1_REG0)

For IPv4 packets, the Layer 3 Address 1 Register 0 register contains the 32-bit IP Destination Address field. For IPv6 packets, it contains Bits[64:95] of the 128-bit IP Source Address or Destination Address field.

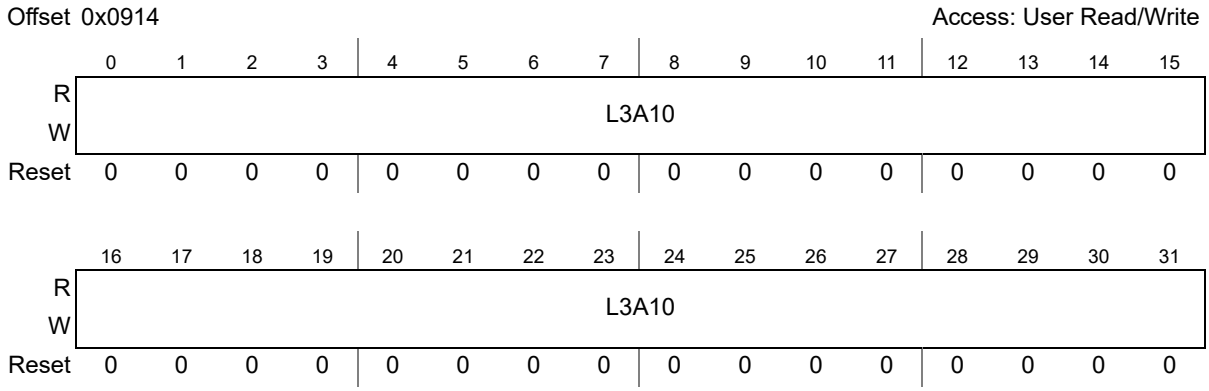


Figure 1116. Layer3 Address1 Register 0 Register (MAC_LAYER3_ADDRESS1_REG0)

Table 1168. MAC_LAYER3_ADDRESS1_REG0 field descriptions

Field	Description
0:31 L3A10	Layer 3 Address 1 Field When the L3PEN0 and L3SAM0 bits are set in the MAC_L3_L4_Control0 register, this field contains the value to be matched with Bits[64:95] of the IP Source Address field in the IPv6 packets. When the L3PEN0 and L3DAM0 bits are set in the MAC_L3_L4_Control0 register, this field contains the value to be matched with Bits[64:95] of the IP Destination Address field in the IPv6 packets. When the L3PEN0 bit is reset and the L3SAM0 bit is set in the MAC_L3_L4_Control0 register, this field contains the value to be matched with the IP Destination Address field in the IPv4 packets.

56.2.138 Layer3 Address2 Register 0 Register (MAC_LAYER3_ADDRESS2_REG0)

The Layer 3 Address 2 Register 0 register is reserved for IPv4 packets. For IPv6 packets, it contains Bits[32:63] of 128-bit IP Source Address or Destination Address field.

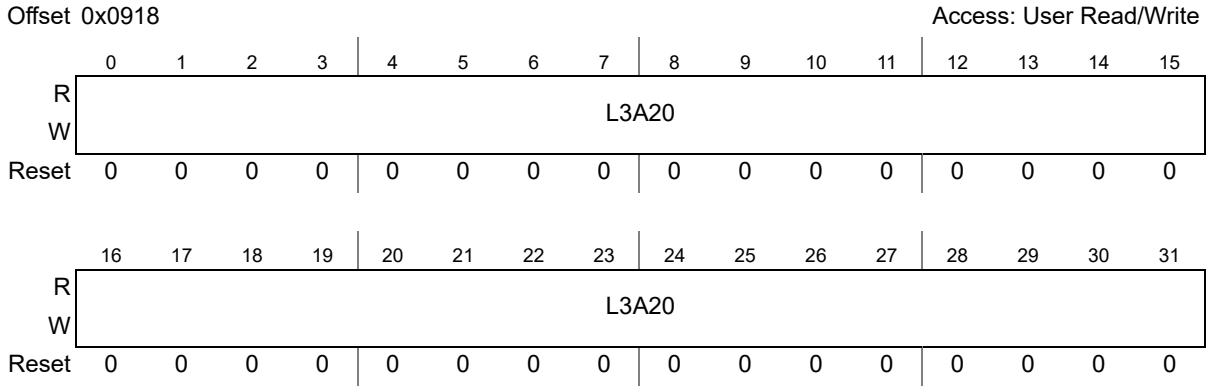


Figure 1117. Layer3 Address2 Register 0 Register (MAC_LAYER3_ADDRESS2_REG0)

Table 1169. MAC_LAYER3_ADDRESS2_REG0 field descriptions

Field	Description
0:31 L3A20	Layer 3 Address 2 Field When the L3PEN0 and L3SAM0 bits are set in the MAC_L3_L4_Control0 register, this field contains the value to be matched with Bits[32:63] of the IP Source Address field in the IPv6 packets. When the L3PEN0 and L3DAM0 bits are set in the MAC_L3_L4_Control0 register, this field contains the value to be matched with Bits[32:63] of the IP Destination Address field in the IPv6 packets. When the L3PEN0 bit is reset in the MAC_L3_L4_Control0 register, this field is not used.

56.2.139 Layer3 Address3 Register 0 Register (MAC_LAYER3_ADDRESS3_REG0)

The Layer 3 Address 3 Register 0 register is reserved for IPv4 packets. For IPv6 packets, it contains Bits[0:31] of 128-bit IP Source Address or Destination Address field.

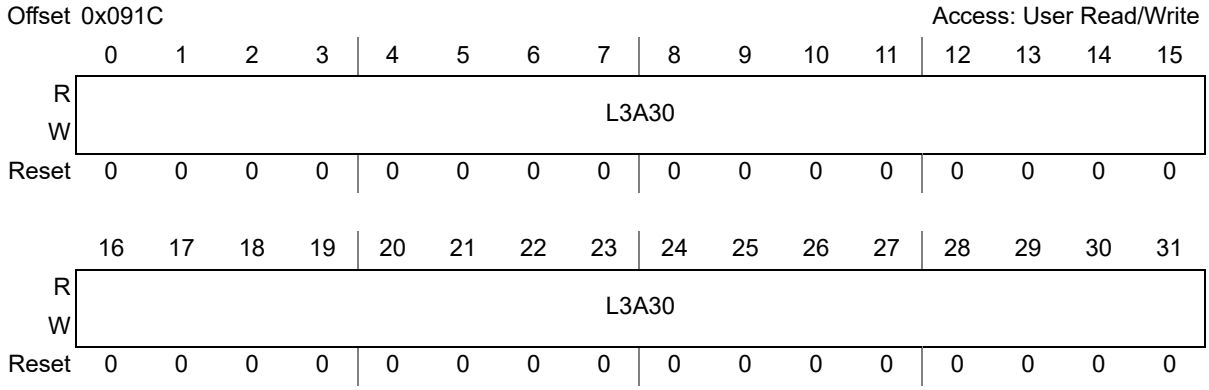


Figure 1118. Layer3 Address3 Register 0 Register (MAC_LAYER3_ADDRESS3_REG0)

Table 1170. MAC_LAYER3_ADDRESS3_REG0 field descriptions

Field	Description
0:31 L3A30	Layer 3 Address 3 Field When the L3PEN0 and L3SAM0 bits are set in the MAC_L3_L4_Control0 register, this field contains the value to be matched with Bits[0:31] of the IP Source Address field in the IPv6 packets. When the L3PEN0 and L3DAM0 bits are set in the MAC_L3_L4_Control0 register, this field contains the value to be matched with Bits[0:31] of the IP Destination Address field in the IPv6 packets. When the L3PEN0 bit is reset in the MAC_L3_L4_Control0 register, this field is not used.

56.2.140 Timestamp Control Register (MAC_TIMESTAMP_CONTROL)

This register controls the operation of the System Time generator and processing of PTP packets for timestamping in the Receiver.

Offset 0x0B00 Access: User Read/Write

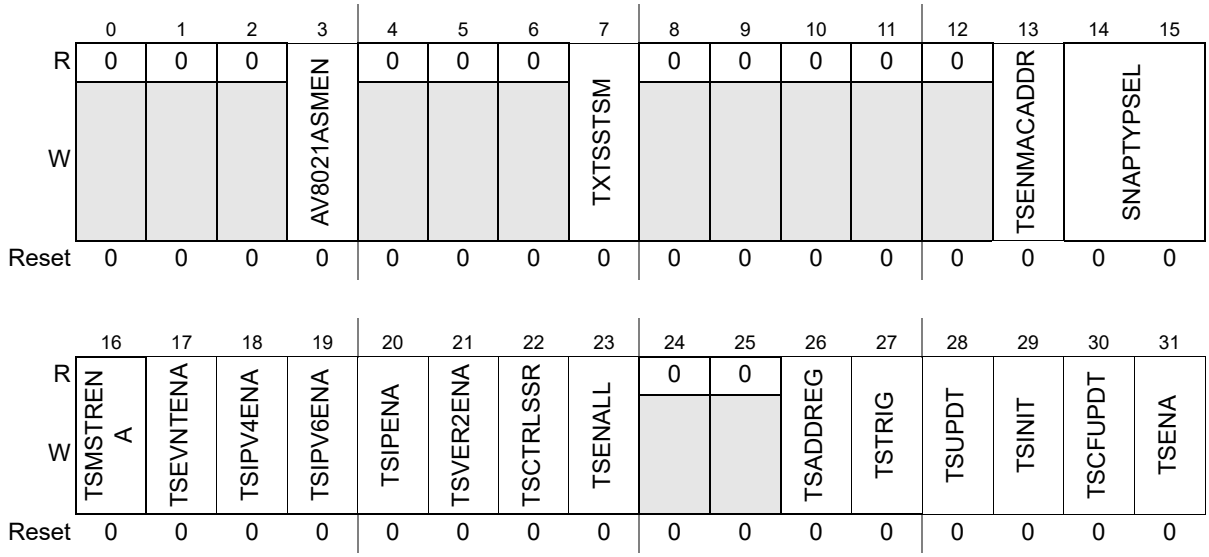


Figure 1119. Timestamp Control Register (MAC_TIMESTAMP_CONTROL)

Table 1171. MAC_TIMESTAMP_CONTROL field descriptions

Field	Description
3 AV8021ASMEN	AV 802.1AS Mode Enable When this bit is set, the MAC processes only untagged PTP over Ethernet packets for providing PTP status and capturing timestamp snapshots, that is, IEEE 802.1AS mode of operation.
7 TXTSSTSM	Transmit Timestamp Status Mode When this bit is set, the MAC overwrites the earlier transmit timestamp status even if it is not read by the software. The MAC indicates this by setting the TXTSSTSMIS bit of the MAC_TxTimestamp_Status_Nanoseconds register. When this bit is reset, the MAC ignores the timestamp status of current packet if the timestamp status of previous packet is not read by the software. The MAC indicates this by setting the TXTSSTSHI bit of the MAC_TxTimestamp_Status_Seconds register.
13 TSENMACADDR	Enable MAC Address for PTP Packet Filtering When this bit is set, the DA MAC address (that matches any MAC Address register) is used to filter the PTP packets when PTP is directly sent over Ethernet. When this bit is set, received PTP packets with DA containing a special multicast or unicast address that matches the one programed in MAC address registers are considered for processing as indicated below, when PTP is directly sent over Ethernet. MAC address registers 0 to 31 is considered for unicast destination address matching.
14:15 SNAPTYPSEL	Select PTP packets for Taking Snapshots These bits, along with Bits 16 and 17, decide the set of PTP packet types for which snapshot needs to be taken. The encoding is given in Timestamp Snapshot Dependency on Register Bits Table.

Table 1171. MAC_TIMESTAMP_CONTROL field descriptions (continued)

Field	Description
16 TSMSTRENA	Enable Snapshot for Messages Relevant to Master When this bit is set, the snapshot is taken only for the messages that are relevant to the master node. Otherwise, the snapshot is taken for the messages relevant to the slave node.
17 TSEVNTENA	Enable Timestamp Snapshot for Event Messages When this bit is set, the timestamp snapshot is taken only for event messages (SYNC, Delay_Req, Pdelay_Req, or Pdelay_Resp). When this bit is reset, the snapshot is taken for all messages except Announce, Management, and Signaling. For more information about the timestamp snapshots, refer to Timestamp Snapshot Dependency on Register Bits Table.
18 TSIPV4ENA	Enable Processing of PTP Packets Sent over IPv4-UDP When this bit is set, the MAC receiver processes the PTP packets encapsulated in IPv4-UDP packets. When this bit is reset, the MAC ignores the PTP transported over IPv4-UDP packets. This bit is set by default.
19 TSIPV6ENA	Enable Processing of PTP Packets Sent over IPv6-UDP When this bit is set, the MAC receiver processes the PTP packets encapsulated in IPv6-UDP packets. When this bit is clear, the MAC ignores the PTP transported over IPv6-UDP packets.
20 TSIPENA	Enable Processing of PTP over Ethernet Packets When this bit is set, the MAC receiver processes the PTP packets encapsulated directly in the Ethernet packets. When this bit is reset, the MAC ignores the PTP over Ethernet packets.
21 TSVER2ENA	Enable PTP Packet Processing for Version 2 Format When this bit is set, the IEEE 1588 version 2 format is used to process the PTP packets. When this bit is reset, the IEEE 1588 version 1 format is used to process the PTP packets. The IEEE 1588 formats are described in 'PTP Processing and Control'.
22 TSCTRLSSR	Timestamp Digital or Binary Rollover Control When this bit is set, the Timestamp Low register rolls over after 0x3B9A_C9FF value (that is, 1 nanosecond accuracy) and increments the timestamp (High) seconds. When this bit is reset, the rollover value of sub-second register is 0x7FFF_FFFF. The sub-second increment must be programed correctly depending on the PTP reference clock frequency and the value of this bit.
23 TSENALL	Enable Timestamp for All Packets When this bit is set, the timestamp snapshot is enabled for all packets received by the MAC.
26 TSADDREG	Update Addend Register When this bit is set, the content of the Timestamp Addend register is updated in the PTP block for fine correction. This bit is cleared when the update is complete. This bit should be zero before it is set. Access restriction applies. Setting 1 sets. Self-cleared. Setting 0 has no effect.
27 TSTRIG	Enable Timestamp Interrupt Trigger When this bit is set, the timestamp interrupt is generated when the System Time becomes greater than the value written in the Target Time register. This bit is reset after the Timestamp Trigger Interrupt is generated. Access restriction applies. Setting 1 sets. Self-cleared. Setting 0 has no effect.

Table 1171. MAC_TIMESTAMP_CONTROL field descriptions (continued)

Field	Description
28 TSUPDT	Update Timestamp When this bit is set, the system time is updated (added or subtracted) with the value specified in MAC_System_Time_Seconds_Update and MAC_System_Time_Nanoseconds_Update. This bit should be zero before updating it. This bit is reset when the update is complete in hardware. The Timestamp Higher Word register (if enabled during core configuration) is not updated. Access restriction applies. Setting 1 sets. Self-cleared. Setting 0 has no effect.
29 TSINIT	Initialize Timestamp When this bit is set, the system time is initialized (overwritten) with the value specified in the MAC Register 80 (System Time Seconds Update Register) and MAC Register 81 (System Time Nanoseconds Update Register). This bit should be zero before it is updated. This bit is reset when the initialization is complete. The Timestamp Higher Word register (if enabled during core configuration) can only be initialized. Access restriction applies. Setting 1 sets. Self-cleared. Setting 0 has no effect.
30 TSCFUPDT	Fine or Coarse Timestamp Update When this bit is set, the Fine method is used to update system timestamp. When this bit is reset, Coarse method is used to update the system timestamp.
31 TSENA	Enable Timestamp When this bit is set, the timestamp is added for Transmit and Receive packets. When disabled, timestamp is not added for transmit and receive packets and the Timestamp Generator is also suspended. You need to initialize the Timestamp (system time) after enabling this mode. On the Receive side, the MAC processes the 1588 packets only if this bit is set.

56.2.141 Sub Second Increment Register (MAC_SUB_SECOND_INCREMENT)

In Coarse Update mode [Bit 30 in MAC_Timestamp_Control], the value in this register is added to the system time every clock cycle of clk_ptp_ref_i. In Fine Update mode, the value in this register is added to the system time whenever the Accumulator gets an overflow.

Offset 0x0B04 Access: User Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	SSINC							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	SNSINC								0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1120. Sub Second Increment Register (MAC_SUB_SECOND_INCREMENT)

Table 1172. MAC_SUB_SECOND_INCREMENT field descriptions

Name	Description
8:15 SSINC	Sub-second Increment Value The value programed in this field is accumulated every clock cycle (of clk_ptp_i) with the contents of the sub-second register. For example, when the PTP clock is 50 MHz (period is 20 ns), you should program 20 (0x14) when the System Time Nanoseconds register has an accuracy of 1 ns [Bit 22 (TSCTRLSSR) is set in MAC_Timestamp_Control]. When TSCTRLSSR is clear, the Nanoseconds register has a resolution of ~0.465 ns. In this case, you should program a value of 43 (0x2B) which is derived by 20 ns/0.465.
16:23 SNSINC	Sub-nanosecond Increment Value This field contains the sub-nanosecond increment value, represented in nanoseconds multiplied by 2 ^ 8. This value is accumulated with the sub-nanoseconds field of the subsecond register. For example, when TSCTRLSSR field in the MAC_Timestamp_Control register is set, and if the required increment is 5.3 ns, then SSINC should be 0x05 and SNSINC should be 0x4C. This field is reserved and read-only when Enable One-Step Timestamp Feature and Enable IEEE 1588 Sub-Nanoseconds Timestamp Support features are not selected.

56.2.142 System Time Seconds Register (MAC_SYSTEM_TIME_SECONDS)

The System Time Seconds register, along with System Time Nanoseconds register, indicates the current value of the system time maintained by the MAC. Though it is updated on a continuous basis, there is some delay from the actual time because of clock domain transfer latencies (from clk_ptp_ref_i to CSR clock).

Offset 0x0B08 Access: User Read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	TSS															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	TSS															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1121. System Time Seconds Register (MAC_SYSTEM_TIME_SECONDS)

Table 1173. MAC_SYSTEM_TIME_SECONDS field descriptions

Name	Description
0:31 TSS	Timestamp Second The value in this field indicates the current value in seconds of the System Time maintained by the MAC.

56.2.143 System Time Nanoseconds Register (MAC_SYSTEM_TIME_NANOSECONDS)

The System Time Nanoseconds register, along with System Time Seconds register, indicates the current value of the system time maintained by the MAC.

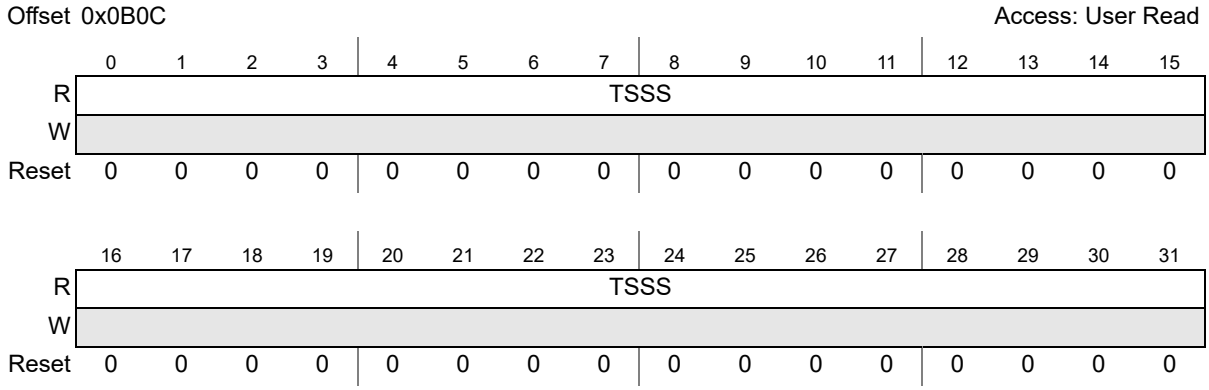


Figure 1122. System Time Nanoseconds Register (MAC_SYSTEM_TIME_NANOSECONDS)

Table 1174. MAC_SYSTEM_TIME_NANOSECONDS field descriptions

Name	Description
0:31 TSSS	Timestamp Sub Seconds The value in this field has the sub-second representation of time, with an accuracy of 0.46 ns. When Bit 22 is set in MAC_Timestamp_Control, each bit represents 1 ns. The maximum value is 0x3B9A_C9FF after which it rolls-over to zero.

56.2.144 System Time Seconds Update Register (MAC_SYSTEM_TIME_SECONDS_UPDATE)

The System Time Seconds Update register, along with the System Time Nanoseconds Update register, initializes or updates the system time maintained by the MAC. You must write both registers before setting the TSINIT or TSUPDT bits in MAC_Timestamp_Control.

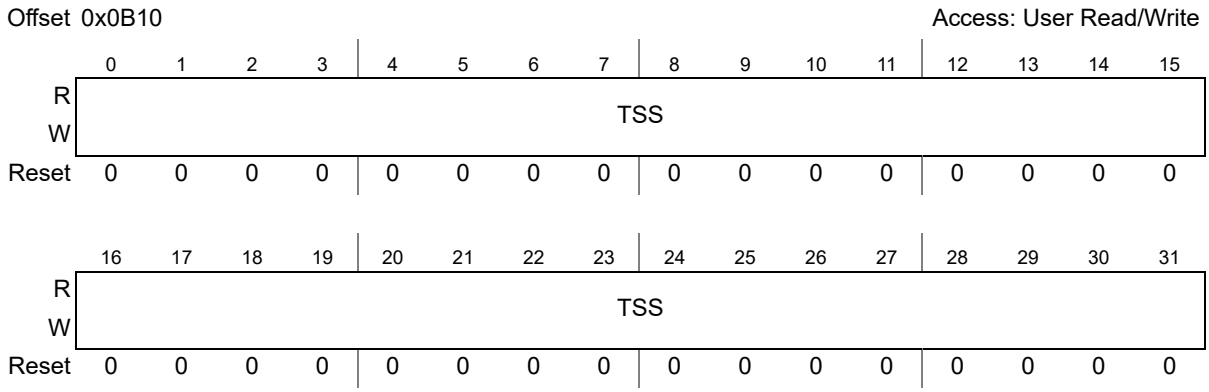


Figure 1123. System Time Seconds Update Register (MAC_SYSTEM_TIME_SECONDS_UPDATE)

Table 1175. MAC_SYSTEM_TIME_SECONDS_UPDATE field descriptions

Field	Description
0:31 TSS	Timestamp Seconds The value in this field is the sub-second part of the update. When ADDSUB is reset, this field must be programed with the subsecond part of the update value, with an accuracy based on the TSCTRLSSR bit of the MAC_Timestamp_Control register. When ADDSUB is set, then this field must be programed with the complement of the sub-second part of the update value as described below. When TSCTRLSSR is set, then the programed value must be 10^9 - <sub-second value>. When TSCTRLSSR is reset, then the programed value must be 2^31 - <sub-second_value> For example, when TSCTRLSSR bit is set and if 2.000000001 seconds need to be subtracted from the system time, then the TSS field in the MAC_Timestamp Seconds update register must be 0xFFFF_FFFE (that is, 2^32 - 2), ADDSUB bit in this register should be set, and the TSSS field must be 0x3B9A_C9FF (that is, 10^9 - 1).

56.2.145 System Time Nanoseconds Update Register (MAC_SYSTEM_TIME_NANOSECONDS_UPDATE)

Offset 0x0B14 Access: User Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	ADDSUB	TSS														
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	TSS															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1124. System Time Nanoseconds Update Register (MAC_SYSTEM_TIME_NANOSECONDS_UPDATE)

Table 1176. MAC_SYSTEM_TIME_NANOSECONDS_UPDATE field descriptions

Field	Description
0 ADDSUB	Add or Subtract Time When this bit is set, the time value is subtracted with the contents of the update register. When this bit is reset, the time value is added with the contents of the update register.
1:31 TSS	Timestamp Sub Seconds The value in this field has the sub second representation of time, with an accuracy of 0.46 ns. When the TSCTRLSSR bit is set in the MAC_Timestamp_Control register, each bit represents 1 ns and the programed value should not exceed 0x3B9A_C9FF.

56.2.146 Timestamp Addend Register (MAC_TIMESTAMP_ADDEND)

This register value is used only when the system time is configured for Fine Update mode (TSCFUPDT bit in the MAC_Timestamp_Control register). The content of this register is added to a 32-bit accumulator in every clock cycle (of clk_ptp_ref_i) and the system time is updated whenever the accumulator overflows.

Offset 0x0B18 Access: User Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	TSAR															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	TSAR															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1125. Timestamp Addend Register(MAC_TIMESTAMP_ADDEND)

Table 1177. MAC_TIMESTAMP_ADDEND field descriptions

Field	Description
0:31 TSAR	Timestamp Addend Register This field indicates the 32-bit time value to be added to the Accumulator register to achieve time synchronization.

56.2.147 System Time Higher Word Seconds Register (MAC_SYSTEM_TIME_HIGHER_WORD_SECONDS)

This register contains the most-significant 16-bits of timestamp seconds value.

Offset 0x0B1C Access: User Read/Write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	TSHWR															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1126. System Time Higher Word Seconds Register (MAC_SYSTEM_TIME_HIGHER_WORD_SECONDS)

Table 1178. MAC_SYSTEM_TIME_HIGHER_WORD_SECONDS field descriptions

Field	Description
16:31 TSHWR	Timestamp Higher Word Register This field contains the most-significant 16-bits of timestamp seconds value. This register is directly written to initialize the value and it is incremented when there is an overflow from 32-bits of the System Time Seconds register. Access restriction applies. Updated based on the event. Setting 1 sets. Setting 0 clears.

56.2.148 Timestamp Status Register (MAC_TIMESTAMP_STATUS)

All bits except Bits[4:6] gets cleared when the application reads this register.

Offset 0x0B20 Access: User Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	ATSNS					ATSSTM	0	0	0	0	0	0	ATSSTN	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	TXTSSIS	0	0	0	0	0	0	0	0	0	0	0	TSTRGTERR0	AUXTSTRIG	TSTARGT0	TSSOVF
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1127. Timestamp Status Register (MAC_TIMESTAMP_STATUS)

Table 1179. MAC_TIMESTAMP_STATUS field descriptions

Field	Description
2:6 ATSNS	Number of Auxiliary Timestamp Snapshots This field indicates the number of Snapshots available in the FIFO. value equal to 4 indicates that the Auxiliary Snapshot FIFO is full. These bits are cleared (to 00000) when the Auxiliary snapshot FIFO clear bit is set.
7 ATSSTM	Auxiliary Timestamp Snapshot Trigger Missed This bit is set when the Auxiliary timestamp snapshot FIFO is full and external trigger was set. This indicates that the latest snapshot is not stored in the FIFO.

Table 1179. MAC_TIMESTAMP_STATUS field descriptions (continued)

Field	Description
14:15 ATSSTN	Auxiliary Timestamp Snapshot Trigger Identifier These bits identify the Auxiliary trigger inputs for which the timestamp available in the Auxiliary Snapshot Register is applicable. When more than one bit is set at the same time, it means that corresponding auxiliary triggers were sampled at the same clock. Bit 15: Auxiliary trigger 0 Bit 14: Auxiliary trigger 1 The software can read this register to find the triggers that are set when the timestamp is taken. Access restriction applies. Clears on read. Self-set to 1 on internal event.
16 TXTSSIS	Tx Timestamp Status Interrupt Status When drop transmit status is enabled in MTL, this bit is set when the captured transmit timestamp is updated in the MAC_TxTimestamp_Status_Nanoseconds and MAC_TxTimestamp_Status_Seconds registers. This bit is cleared when the MAC_TxTimestamp_Status_Seconds register is read.
28 TSTRGTERR0	Timestamp Target Time Error This bit is set when the latest target time programed in the MAC_PPS0_Target_Time_Seconds and MAC_PPS0_Target_Time_Nanoseconds registers elapses. This bit is cleared when the application reads this bit. Access restriction applies. Clears on read. Self-set to 1 on internal event.
29 AUXTSTRIG	Auxiliary Timestamp Trigger Snapshot This bit is set high when the auxiliary snapshot is written to the FIFO. Access restriction applies. Clears on read. Self-set to 1 on internal event.
30 TSTARGT0	Timestamp Target Time Reached When set, this bit indicates that the value of system time is greater than or equal to the value specified in the MAC_PPS0_Target_Time_Seconds and MAC_PPS0_Target_Time_Nanoseconds registers. Access restriction applies. Clears on read. Self-set to 1 on internal event.
31 TSSOVF	Timestamp Seconds Overflow When this bit is set, it indicates that the seconds value of the timestamp (when supporting version 2 format) has overflowed beyond 32'hFFFF_FFFF. Access restriction applies. Clears on read. Self-set to 1 on internal event.

56.2.149 Transmit Timestamp Status Nanoseconds Register (MAC_TX_TIMESTAMP_STATUS_NANOSECONDS)

This register contains the nanosecond part of timestamp captured for Transmit when Tx status is disabled.

The MAC_TxTimestamp_Status_Nanoseconds register, along with MAC_TxTimestamp_Status_Seconds, gives the 64-bit timestamp captured for the PTP packet successfully transmitted by the MAC. This value is considered to be read by the application when the last byte of MAC_TxTimestamp_Status_Nanoseconds is read. In the little-endian mode, this means when bits[31:24] are read; in big-endian mode, bits[7:0] are read.

If the application does not read these registers and timestamp of another packet is captured, then either the current timestamp is lost (overwritten) or the new timestamp is lost (dropped), depending on the setting of the TXTSSTSM bit of the MAC_Timestamp_Control register. The status bit TXTSC bit [16] in MAC_Timestamp_Status register is set whenever the MAC transmitter captures the timestamp.

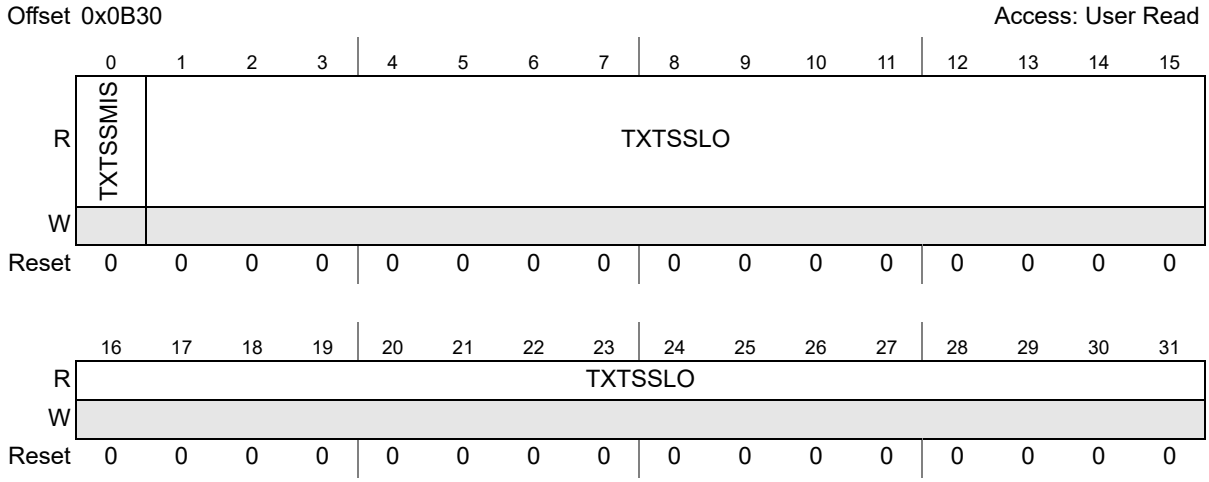


Figure 1128. Transmit Timestamp Status Nanoseconds Register (MAC_TX_TIMESTAMP_STATUS_NANOSECONDS)

Table 1180. MAC_TX_TIMESTAMP_STATUS_NANOSECONDS field descriptions

Field	Description
0 TXTSSMIS	Transmit Timestamp Status Missed When this bit is set, it indicates one of the following: The timestamp of the current packet is ignored if TXTSSTSM bit of the MAC_Timestamp_Control register is reset The timestamp of the previous packet is overwritten with timestamp of the current packet if TXTSSTSM bit of the MAC_Timestamp_Control register is set. Access restriction applies. Clears on read. Self-set to 1 on internal event.
1:31 TXTSSLO	Transmit Timestamp Status Low This field contains the 31 bits of the Nanoseconds field of the Transmit packet's captured timestamp.

56.2.150 Transmit Timestamp Status Seconds Register (MAC_TX_TIMESTAMP_STATUS_SECONDS)

The register contains the higher 32 bits of the timestamp (in seconds) captured when a PTP packet is transmitted.

Offset 0x0B34 Access: User Read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	TXTSSHI															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	TXTSSHI															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1129. Transmit Timestamp Status Seconds Register (MAC_TX_TIMESTAMP_STATUS_SECONDS)

Table 1181. MAC_TX_TIMESTAMP_STATUS_SECONDS field descriptions

Field	Description
0:31 TXTSSHI	Transmit Timestamp Status High This field contains the lower 32 bits of the Seconds field of Transmit packet's captured timestamp.

56.2.151 Auxiliary Control Register (MAC_AUXILIARY_CONTROL)

The Auxiliary Timestamp Control register controls the Auxiliary Timestamp snapshot.

Offset 0x0B40 Access: User Read/Write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	ATSEN1	ATSEN0	0	0	0	ATSFC
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1130. Auxiliary Control Register (MAC_AUXILIARY_CONTROL)

Table 1182. MAC_AUXILIARY_CONTROL field descriptions

Field	Description
26 ATSEN1	Auxiliary Snapshot 1 Enable This bit controls the capturing of Auxiliary Snapshot Trigger 1. When this bit is set, the auxiliary snapshot of the event on ptp_aux_trig_i[1] input is enabled. When this bit is reset, the events on this input are ignored.
27 ATSEN0	Auxiliary Snapshot 0 Enable This bit controls the capturing of Auxiliary Snapshot Trigger 0. When this bit is set, the auxiliary snapshot of the event on ptp_aux_trig_i[0] input is enabled. When this bit is reset, the events on this input are ignored.
31 ATSFC	Auxiliary Snapshot FIFO Clear When set, this bit resets the pointers of the Auxiliary Snapshot FIFO. This bit is cleared when the pointers are reset and the FIFO is empty. When this bit is high, the auxiliary snapshots are stored in the FIFO. Access restriction applies. Setting 1 sets. Self-cleared. Setting 0 has no effect.

56.2.152 Auxiliary Timestamp Nanoseconds Register (MAC_AUXILIARY_TIMESTAMP_NANOSECONDS)

The Auxiliary Timestamp Nanoseconds register, along with MAC_Auxiliary_Timestamp_Seconds, gives the 64-bit timestamp stored as auxiliary snapshot. These two registers form the read port of a 64-bit wide FIFO with a depth of 4.

You can store multiple snapshots in this FIFO. Bits[2:6] in MAC_Timestamp_Status indicate the fill-level of the FIFO. The top of the FIFO is removed only when the last byte of MAC Register 91 (Auxiliary Timestamp - Seconds Register) is read. In the little-endian mode, this means when Bits[31:24] are read and in big-endian mode, Bits[7:0] are read.

Offset 0x0B48 Access: User Read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	AUXTSLO														
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	AUXTSLO															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1131. Auxiliary Timestamp Nanoseconds Register (MAC_AUXILIARY_TIMESTAMP_NANOSECONDS)

Table 1183. MAC_AUXILIARY_TIMESTAMP_NANOSECONDS field descriptions

Field	Description
1:31 AUXTSLO	Contains the lower 31 bits (nanoseconds field) of the auxiliary timestamp.

56.2.153 Auxiliary Timestamp Seconds Register (MAC_AUXILIARY_TIMESTAMP_SECONDS)

The Auxiliary Timestamp - Seconds register contains the lower 32 bits of the Seconds field of the auxiliary timestamp register.

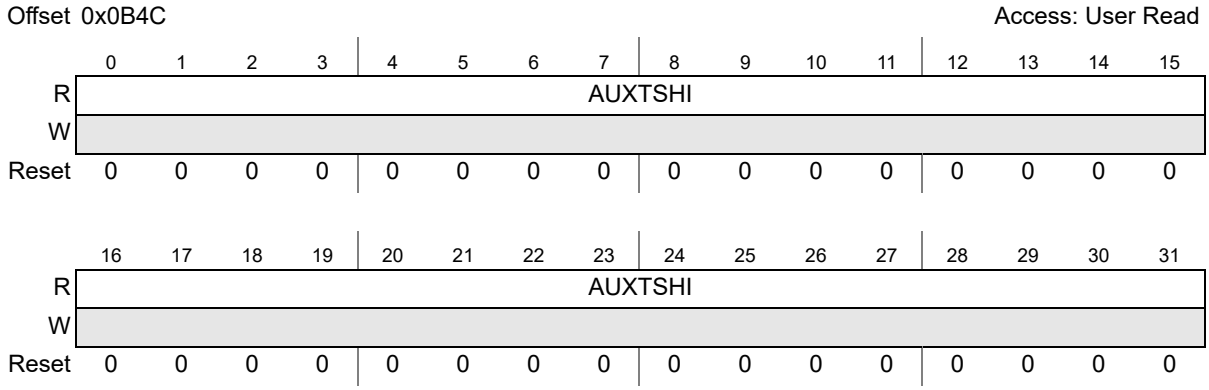


Figure 1132. Auxiliary Timestamp Seconds Register (MAC_AUXILIARY_TIMESTAMP_SECONDS)

Table 1184. MAC_AUXILIARY_TIMESTAMP_SECONDS field descriptions

Field	Description
0:31 AUXTSHI	Contains the lower 32 bits of the Seconds field of the auxiliary timestamp.

56.2.154 Timestamp Ingress Asymmetry Correction Register (MAC_TIMESTAMP_INGRESS_ASYM_CORR)

The MAC Timestamp Ingress Asymmetry Correction register contains the Ingress Asymmetry Correction value to be used while updating correction field in PDelay_Resp PTP messages.

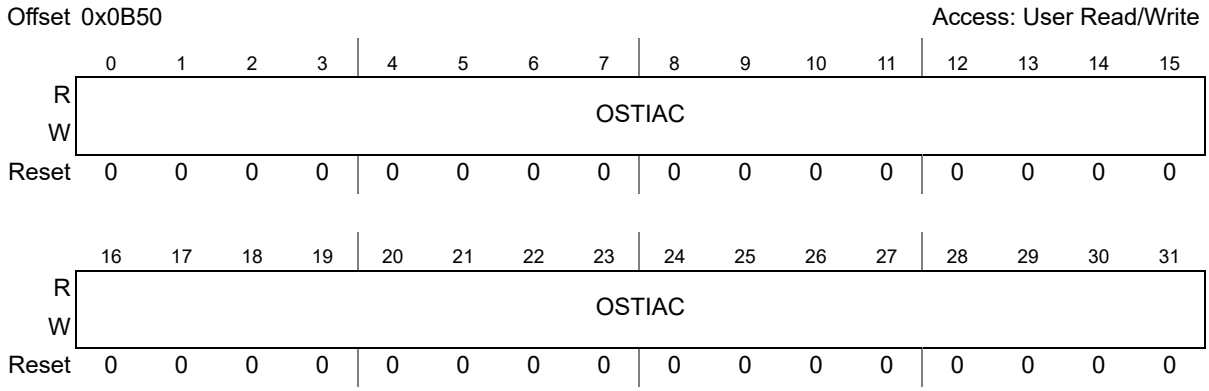


Figure 1133. Timestamp Ingress Asymmetry Correction Register (MAC_TIMESTAMP_INGRESS_ASYM_CORR)

Table 1185. MAC_TIMESTAMP_INGRESS_ASYM_CORR field descriptions

Field	Description
0:31 OSTIAC	One-Step Timestamp Ingress Asymmetry Correction This field contains the ingress path asymmetry value to be added to correctionField of Pdelay_Resp PTP packet. The programed value should be in units of nanoseconds and multiplied by 2^16. For example, 2.5 ns is represented as 0x00028000. The value can also be negative, which is represented in 2's complement form with bit 0 representing the sign bit.

56.2.155 Timestamp Egress Asymmetry Correction Register (MAC_TIMESTAMP_EGRESS_ASYM_CORR)

The MAC Timestamp Egress Asymmetry Correction register contains the Egress Asymmetry Correction value to be used while updating the correction field in PDelay_Req PTP messages.

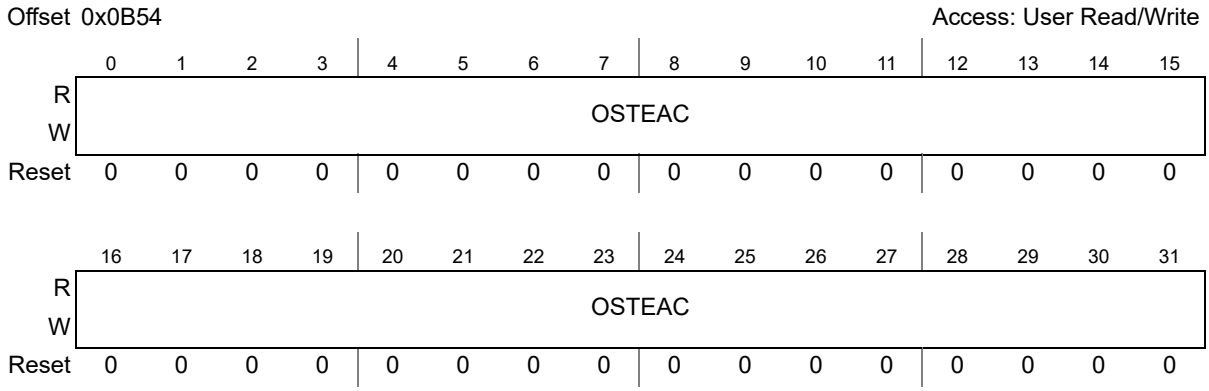


Figure 1134. Timestamp Egress Asymmetry Correction Register (MAC_TIMESTAMP_EGRESS_ASYM_CORR)

Table 1186. MAC_TIMESTAMP_EGRESS_ASYM_CORR field descriptions

Field	Description
0:31 OSTEAC	One-Step Timestamp Egress Asymmetry Correction This field contains the egress path asymmetry value to be subtracted from correctionField of Pdelay_Resp PTP packet. The programed value must be the negated value in units of nanoseconds multiplied by 2^16. For example, if the required correction is +2.5 ns, the programed value must be 0xFFFD_8000, which is the 2's complement of 0x0002_8000 (2.5 * 216). Similarly, if the required correction is -3.3 ns, the programed value is 0x0003_4CCC (3.3 * 216).

56.2.156 Timestamp Ingress Correction Nanosecond Register (MAC_TIMESTAMP_INGRESS_CORR_NANOSECOND)

This register contains the correction value in nanoseconds to be used with the captured timestamp value in the ingress path.

Offset 0x0B58 Access: User Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	TSIC															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	TSIC															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1135. Timestamp Ingress Correction Nanosecond Register (MAC_TIMESTAMP_INGRESS_CORR_NANOSECOND)

Table 1187. MAC_TIMESTAMP_INGRESS_CORR_NANOSECOND field descriptions

Field	Description
0:31 TSIC	Timestamp Ingress Correction This field contains the ingress path correction value as defined by the Ingress Correction expression.

56.2.157 Timestamp Egress Correction Nanosecond Register (MAC_TIMESTAMP_EGRESS_CORR_NANOSECOND)

This register contains the correction value in nanoseconds to be used with the captured timestamp value in the egress path.

Offset 0x0B5C Access: User Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	TSEC															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	TSEC															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1136. Timestamp Egress Correction Nanosecond Register (MAC_TIMESTAMP_EGRESS_CORR_NANOSECOND)

Table 1188. MAC_TIMESTAMP_EGRESS_CORR_NANOSECOND field descriptions

Field	Description
0:31 TSEC	Timestamp Egress Correction This field contains the nanoseconds part of the egress path correction value as defined by the Egress Correction expression.

56.2.158 PPS Control Register (MAC_PPS_CONTROL)

Offset 0x0B70 Access: User Read/Write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	TRGTMODSEL0		PPSEN0	PPSCTRL_PPSCMD			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1137. PPS Control Register (MAC_PPS_CONTROL)

Table 1189. MAC_PPS_CONTROL field descriptions

Field	Description
25:26 TRGTMODSEL0	Target Time Register Mode for PPS0 Output This field indicates the Target Time registers (MAC registers 96 and 97) mode for PPS0 output signal: 00 Target Time registers are programed only for generating the interrupt event. 01 Reserved 10 Target Time registers are programed for generating the interrupt event and starting or stopping the PPS0 output signal generation. 11 Target Time registers are programed only for starting or stopping the PPS0 output signal generation. No interrupt is asserted.

Table 1189. MAC_PPS_CONTROL field descriptions (continued)

Field	Description
27 PPSEN0	Flexible PPS Output Mode Enable When this bit is set, Bits[28:31] function as PPSCMD. When this bit is reset, Bits[28:31] function as PPSCTRL (Fixed PPS mode).
28:31 PPSCTRL _PPSCMD	**PPS Output Frequency Control** This field controls the frequency of the PPS0 output (ptp_pps_o[0]) signal. The default value of PPSCTRL is 0000, and the PPS output is 1 pulse (of width clk_ptp_i) every second. For other values of PPSCTRL, the PPS output becomes a generated clock of following frequencies: 0001 The binary rollover is 2 Hz, and the digital rollover is 1 Hz 0010 The binary rollover is 4 Hz, and the digital rollover is 2 Hz 0011 The binary rollover is 8 Hz, and the digital rollover is 4 Hz 0100 The binary rollover is 16 Hz, and the digital rollover is 8 Hz ... 1111 The binary rollover is 32.768 KHz and the digital rollover is 16.384 KHz **Note:** In the binary rollover mode, the PPS output (ptp_pps_o) has a duty cycle of 50 percent with these frequencies. In the digital rollover mode, the PPS output frequency is an average number. The actual clock is of different frequency that gets synchronized every second.(1) or **Flexible PPS Output (ptp_pps_o[0]) Control** Programming these bits with a non-zero value instructs the MAC to initiate an event. When the command is transferred or synchronized to the PTP clock domain, these bits get cleared automatically. The software should ensure that these bits are programed only when they are 'all-zero'. The following list describes the values of PPSCMD0: - 0000 No Command - 0001 START Single Pulse. This command generates single pulse rising at the start point defined in Target Time Registers (register 455 and 456) and of a duration defined in the PPS0 Width Register. - 0010 START Pulse Train. This command generates the train of pulses rising at the start point defined in the Target Time Registers and of a duration defined in the PPS0 Width Register and repeated at interval defined in the PPS Interval Register. By default, the PPS pulse train is free-running unless stopped by the 'Stop Pulse train at time' or 'Stop Pulse Train immediately' commands. - 0011 Cancel START. This command cancels the START Single Pulse and START Pulse Train commands if the system time has not crossed the programed start time. - 0100 STOP Pulse train at time. This command stops the train of pulses initiated by the START Pulse Train command (PPSCMD = 0010) after the time programed in the Target Time registers elapses. - 0101 STOP Pulse Train immediately. This command immediately stops the train of pulses initiated by the START Pulse Train command (PPSCMD = 0010). - 0110 Cancel STOP Pulse train. This command cancels the STOP pulse train at time command if the programed stop time has not elapsed. The PPS pulse train becomes free-running on the successful execution of this command. - 0111 Reserved - 1111 Reserved Access restriction applies. Setting 1 sets. Self-cleared. Setting 0 has no effect.

1. For example:
When PPSCTRL = 0001, the PPS (1 Hz) has a low period of 537 ms and a high period of 463 ms
When PPSCTRL = 0010, the PPS (2 Hz) is a sequence of one clock of 50 percent duty cycle and 537 ms period. Second clock of 463 ms period (268 ms low and 195 ms high)
When PPSCTRL = 0011, the PPS (4 Hz) is a sequence of three clocks of 50 percent duty cycle and 268 ms period. Fourth clock of 195 ms period (134 ms low and 61 ms high)
This behavior is because of the non-linear toggling of bits in the digital rollover mode in the MAC_System_Time_Nanoseconds register.

56.2.159 PPS0 Target Time Seconds Register (MAC_PPS0_TARGET_TIME_SECONDS)

The PPS Target Time Seconds register, along with PPS Target Time Nanoseconds register, is used to schedule an interrupt event [Bit 30 of MAC_Timestamp_Status] when the system time exceeds the value programed in these registers

Offset 0x0B80 Access: User Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	TSTRH0															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	TSTRH0															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1138. PPS0 Target Time Seconds Register (MAC_PPS0_TARGET_TIME_SECONDS)

Table 1190. MAC_PPS0_TARGET_TIME_SECONDS field descriptions

Field	Description
0:31 TSTRH0	PPS Target Time Seconds Register This field stores the time in seconds. When the timestamp value matches or exceeds both Target Timestamp registers, the MAC starts or stops the PPS signal output and generates an interrupt (if enabled) based on Target Time mode selected for the corresponding PPS output in the MAC_PPS_Control register.

56.2.160 PPS0 Target Time Nanoseconds Register (MAC_PPS0_TARGET_TIME_NANOSECONDS)

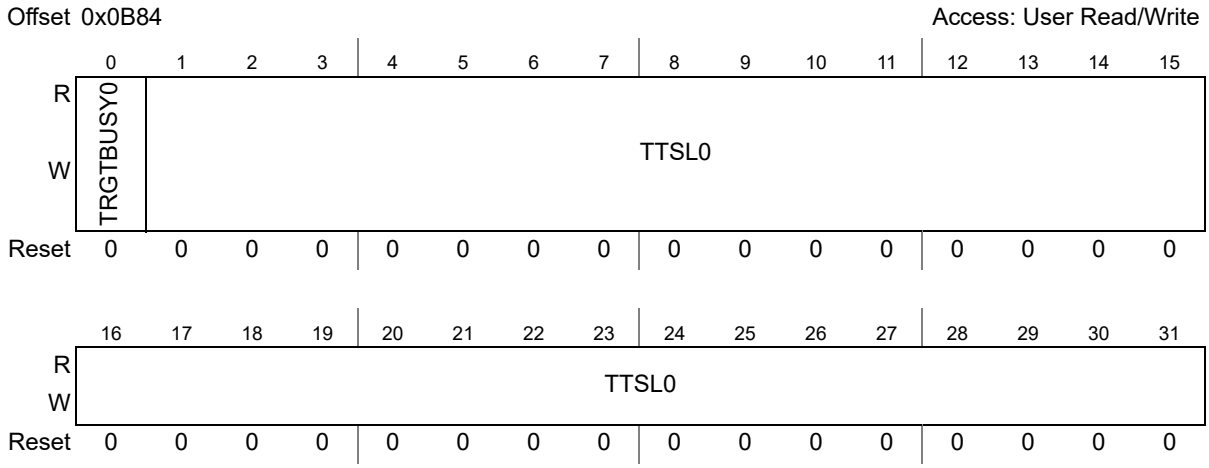


Figure 1139. PPS0 Target Time Nanoseconds Register (MAC_PPS0_TARGET_TIME_NANOSECONDS)

Table 1191. MAC_PPS0_TARGET_TIME_NANOSECONDS field descriptions

Field	Description
0 TRGTBUSY0	PPS Target Time Register Busy The MAC sets this bit when the PPSCMD0 field in the MAC_PPS_Control register is programed to 010 or 011. Programming the PPSCMD0 field to 010 or 011 instructs the MAC to synchronize the Target Time Registers to the PTP clock domain. The MAC clears this bit after synchronizing the Target Time Registers to the PTP clock domain The application must not update the Target Time Registers when this bit is read as 1. Otherwise, the synchronization of the previous programed time gets corrupted.
1:31 TTSL0	Target Time Low for PPS Register This register stores the time in (signed) nanoseconds. When the value of the timestamp matches the value in both Target Timestamp registers, the MAC starts or stops the PPS signal output and generates an interrupt (if enabled) based on the TRGTMODSEL0 field (Bits [25:26]) in MAC_PPS_Control. When the TSCTRLSSR bit is set in the MAC_Timestamp_Control register, this value should not exceed 0x3B9A_C9FF. The actual start or stop time of the PPS signal output may have an error margin up to one unit of sub-second increment value. Access restriction applies. Setting 1 sets. Self-cleared. Setting 0 has no effect.

56.2.161 PPS0_Interval Register (MAC_PPS0_INTERVAL)

The PPS0 Interval register contains the number of units of sub-second increment value between the rising edges of PPS0 signal output (ptp_pps_o[0]).

Offset 0x0B88 Access: User Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	PPSINT0															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	PPSINT0															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1140. PPS0_Interval Register (MAC_PPS0_INTERVAL)

Table 1192. MAC_PPS0_INTERVAL field descriptions

Field	Description
0:31 PPSINT0	PPS Output Signal Interval These bits store the interval between the rising edges of PPS0 signal output. The interval is stored in terms of number of units of sub-second increment value. You need to program one value less than the required interval. For example, if the PTP reference clock is 50 MHz (period of 20 ns), and desired interval between the rising edges of PPS0 signal output is 100 ns (that is, 5 units of sub-second increment value), you should program value 4 (5-1) in this register.

56.2.162 PPS0 Width Register (MAC_PPS0_WIDTH)

The PPS0 Width register contains the number of units of sub-second increment value between the rising and corresponding falling edges of PPS0 signal output (ptp_pps_o[0]).

Offset 0x0B8C Access: User Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	PPSWIDTH0															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	PPSWIDTH0															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1141. PPS0 Width Register (MAC_PPS0_WIDTH)

Table 1193. MAC_PPS0_WIDTH field descriptions

Field	Description
0:31 PPSWIDTH0	PPS Output Signal Width These bits store the width between the rising edge and corresponding falling edge of PPS0 signal output. The width is stored in terms of number of units of sub-second increment value. You need to program one value less than the required interval. For example, if PTP reference clock is 50 MHz (period of 20 ns), and width between the rising and corresponding falling edges of PPS0 signal output is 80 ns (that is, four units of sub-second increment value), you should program value 3 (4-1) in this register. **Note:** The value programed in this register must be lesser than the value programed in MAC_PPS0_Interval.

56.2.163 Operation Mode Register (MTL_OPERATION_MODE)

The Operation Mode register establishes the Transmit and Receive operating modes and commands.

Offset 0x0C00 Access: User Read/Write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	CNTCLR	CNTPRST	0	SCHALG		0	0	RAA	DTXSTS	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1142. Operation Mode Register (MTL_OPERATION_MODE)

Table 1194. MTL_OPERATION_MODE field descriptions

Field	Description
22 CNTCLR	Counters Reset When this bit is set, all counters are reset. This bit is cleared automatically after 1 clock cycle. If this bit is set along with CNT_PRESET bit, CNT_PRESET has precedence. Access restriction applies. Setting 1 sets. Self-cleared. Setting 0 has no effect.
23 CNTPRST	Counters Preset When this bit is set, MTL_TxQ[0-7]_Underflow register is initialized/preset to 12'h7F0. Missed Packet and Overflow Packet counters in MTL_RxQ[0-7]_Missed_Packet_Overflow_Cnt register is initialized/preset to 12'h7F0. Access restriction applies. Setting 1 sets. Self-cleared. Setting 0 has no effect.

Table 1194. MTL_OPERATION_MODE field descriptions (continued)

Field	Description
25:26 SCHALG	Tx Scheduling Algorithm This field indicates the algorithm for Tx scheduling: 00 WRR algorithm 01 Reserved 10 Reserved. 11 Strict priority algorithm.
29 RAA	Receive Arbitration Algorithm This field is used to select the arbitration algorithm for the Rx side. 0 Strict priority (SP) Queue 0 has the lowest priority and the last queue has the highest priority. 1 Weighted Strict Priority (WSP)
30 DTXSTS	Drop Transmit Status When this bit is set, the Tx packet status received from the MAC is dropped in the MTL. When this bit is reset, the Tx packet status received from the MAC is forwarded to the application.

56.2.164 Debug Access Control Register (MTL_DBG_CTL)

The FIFO Debug Access Control and Status register controls the operation mode of FIFO debug access.

Offset 0x0C08 Access: User Read/Write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R									0			0				
W	STSIE	PKTIE	FIFOSEL		FIFOWREN	FIFORDEN	RSTSEL	RSTALL		PKTSTATE			BYTEEN		DBGMOD	FDBGEN
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1143. Debug Access Control Register (MTL_DBG_CTL)

Table 1195. MTL_DBG_CTL field descriptions

Field	Description
16 STSIE	Transmit Status Available Interrupt Status Enable When this bit is set, an interrupt is generated when Transmit status is available in slave mode.
17 PKTIE	Receive Packet Available Interrupt Status Enable When this bit is set, an interrupt is generated when EOP of received packet is written to the Rx FIFO.
18:19 FIFOSEL	FIFO Selected for Access This field indicates the FIFO selected for debug access: 00 Tx FIFO 01 Tx Status FIFO (only read access when SLVMOD is set) 10 Reserved 11 Rx FIFO
20 FIFOWREN	FIFO Write Enable When this bit is set, it enables the Write operation on selected FIFO. Access restriction applies. Self-cleared. Setting 0 clears. Setting 1 sets.
21 FIFORDEN	FIFO Read Enable When this bit is set, it enables the Read operation on selected FIFO. Access restriction applies. Self-cleared. Setting 0 clears. Setting 1 sets.
22 RSTSEL	Reset Pointers of Selected FIFO When this bit is set, the pointers of the currently-selected FIFO are reset. Access restriction applies. Self-cleared. Setting 0 clears. Setting 1 sets.
23 RSTALL	Reset All Pointers When this bit is set, the pointers of all FIFOs are reset. Access restriction applies. Self-cleared. Setting 0 clears. Setting 1 sets.
25:26 PKTSTATE	Encoded Packet State This field is used to write the control information to the Tx FIFO or Rx FIFO. Tx FIFO / Rx FIFO: 00 Packet Data / Packet Data 01 Control Word/ Normal Status 10 SOP Data / Last Status 11 EOP Data / EOP
28:29 BYTEEN	Byte Enables This field indicates the number of data bytes valid in the data register during Write operation. This is valid only when PKTSTATE is 2'b10 (EOP) and Tx FIFO or Rx FIFO is selected. 00 Byte 0 valid 01 Byte 0 and Byte 1 are valid 10 Byte 0, Byte 1, and Byte 2 are valid 11 All four bytes are valid

Table 1195. MTL_DBG_CTL field descriptions (continued)

Field	Description
30 DBGMOD	Debug Mode Access to FIFO When this bit is set, it indicates that the current access to the FIFO is read, write, and debug access. In this mode, the following access types are allowed: Read and Write access to Tx FIFO, TSO FIFO, and Rx FIFO Read access is allowed to Tx Status FIFO. When this bit is reset, it indicates that the current access to the FIFO is slave access bypassing the DMA. In this mode, the following access are allowed: Write access to the Tx FIFO Read access to the Rx FIFO and Tx Status FIFO
31 FDBGEN	FIFO Debug Access Enable When this bit is set, it indicates that the debug mode access to the FIFO is enabled. When this bit is reset, it indicates that the FIFO can be accessed only through a master interface.

56.2.165 Debug Status Register (MTL_DBG_STS)

The FIFO Debug Status register contains the status of FIFO debug access.

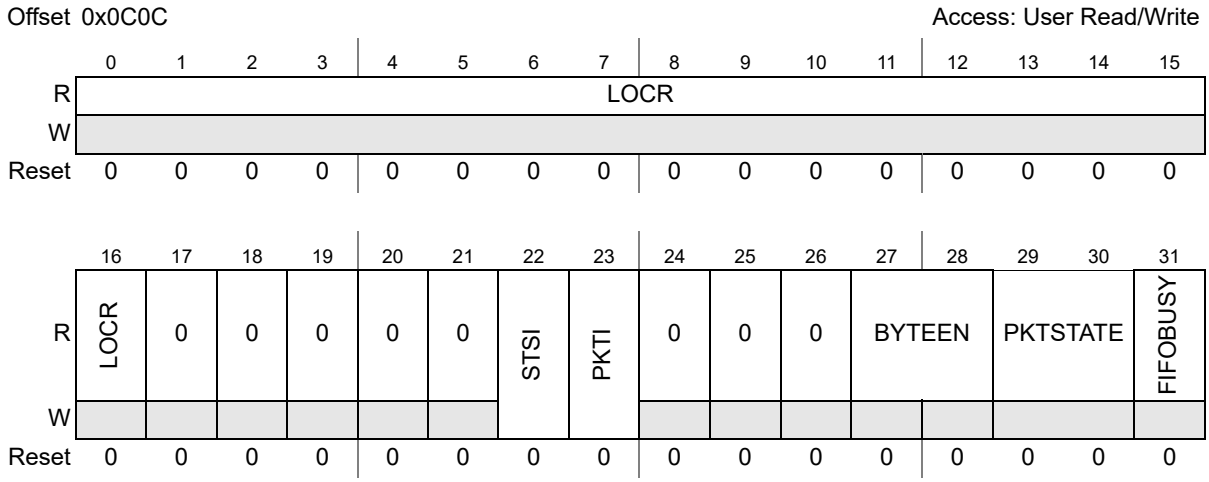


Figure 1144. Debug Status Register (MTL_DBG_STS)

Table 1196. MTL_DBG_STS field descriptions

Field	Description
0:16 LOCR	Remaining Locations in the FIFO Slave Access Mode: This field indicates the space available in selected FIFO. Debug Access Mode: This field contains the Write or Read pointer value of the selected FIFO during Write or Read operation, respectively.
22 STSI	Transmit Status Available Interrupt Status When set, this bit indicates that the Slave mode Tx packet is transmitted, and the status is available in Tx Status FIFO. This bit is reset when 1 is written to this bit.
23 PKTI	Receive Packet Available Interrupt Status When set, this bit indicates that MAC layer has written the EOP of received packet to the Rx FIFO. This bit is reset when 1 is written to this bit.
27:28 BYTEEN	Byte Enables This field indicates the number of data bytes valid in the data register during Read operation. This is valid only when PKTSTATE is 2'b10 (EOP) and Tx FIFO or Rx FIFO is selected. 00 Byte 0 valid 01 Byte 0 and Byte 1 are valid 10 Byte 0, Byte 1, and Byte 2 are valid 11 All four bytes are valid
29:30 PKTSTATE	Encoded Packet State This field is used to get the control or status information of the selected FIFO. Tx FIFO: 00 Packet Data 01 Control Word 10 SOP Data 11 EOP Data Rx FIFO: 00 Packet Data 01 Normal Status 10 Last Status 11 EOP This field is applicable only for Tx FIFO and Rx FIFO during Read operation.
31 FIFOBUSY	FIFO Busy When set, this bit indicates that a FIFO operation is in progress in the MAC and content of the following fields is not valid: All other fields of this register All fields of the MTL_FIFO_Debug_Data register

56.2.166 FIFO Debug Data register (MTL_FIFO_DEBUG_DATA)

The FIFO Debug Data register contains the data to be written to or read from the FIFOs.

Offset 0x0C10 Access: User Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	FDBGDATA															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	FDBGDATA															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1145. FIFO Debug Data register (MTL_FIFO_DEBUG_DATA)

Table 1197. MTL_FIFO_DEBUG_DATA field descriptions

Field	Description
0:31 FDBGDATA	FIFO Debug Data During debug or slave access write operation, this field contains the data to be written to the Tx FIFO, Rx FIFO,. During debug or slave access read operation, this field contains the data read from the Tx FIFO, Rx FIFO, or Tx Status FIFO.

56.2.167 Interrupt Status Register (MTL_INTERRUPT_STATUS)

The software driver (application) reads this register during interrupt service routine or polling to determine the interrupt status of MTL queues and the MAC.

Offset 0x0C20 Access: User Read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	DBGIS	MACIS
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	Q1IS	Q0IS
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1146. Interrupt Status Register (MTL_INTERRUPT_STATUS)

Table 1198. MTL_INTERRUPT_STATUS field descriptions

Field	Description
14 DBGIS	Debug Interrupt status This bit indicates an interrupt event during the slave access. To reset this bit, the application must read the FIFO Debug Access Status register to get the exact cause of the interrupt and clear its source.
15 MACIS	MAC Interrupt Status This bit indicates an interrupt event in the MAC. To reset this bit, the application must read the corresponding register in the MAC to get the exact cause of the interrupt and clear its source.
30 Q1IS	Queue 1 Interrupt status This bit indicates that there is an interrupt from Queue 1. To reset this bit, the application must read the MTL_Q1_Interrupt_Control_Status register to get the exact cause of the interrupt and clear its source.
31 Q0IS	Queue 0 Interrupt status This bit indicates that there is an interrupt from Queue 0. To reset this bit, the application must read Queue 0 Interrupt Control and Status register to get the exact cause of the interrupt and clear its source.

56.2.168 Receive Queue and DMA Channel Mapping 0 Register (MTL_RXQ_DMA_MAP0)

The Receive Queue and DMA Channel Mapping 0 register.

Offset 0x0C30 Access: User Read/Write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	Q1DDMACH	0	0	0	Q1MDMACH	0	0	0	Q0DDMACH	0	0	0	Q0MDMACH
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1147. Receive Queue and DMA Channel Mapping 0 Register (MTL_RXQ_DMA_MAP0)

Table 1199. MTL_RXQ_DMA_MAP0 field descriptions

Field	Description
19 Q1DDMACH	Queue 1 Enabled for DA-based DMA Channel Selection When set, this bit indicates that the packets received in Queue 1 are routed to a particular DMA channel as decided in the MAC Receiver based on the DMA channel number programed in the L3-L4 filter registers, or the Ethernet DA address. When reset, this bit indicates that the packets received in Queue 1 are routed to the DMA Channel programed in the Q1MDMACH field (Bits[21:23]).
23 Q1MDMACH	Queue 1 Mapped to DMA Channel This field controls the routing of the received packet in Queue 1 to the DMA channel: 0 DMA Channel 0 1 DMA Channel 1 This field is valid when the Q1DDMACH field is reset.
27 Q0DDMACH	Queue 0 Enabled for DA-based DMA Channel Selection When set, this bit indicates that the packets received in Queue 0 are routed to a particular DMA channel as decided in the MAC Receiver based on the DMA channel number programed in the L3-L4 filter registers, or the Ethernet DA address. When reset, this bit indicates that the packets received in Queue 0 are routed to the DMA Channel programed in the Q0MDMACH field.
31 Q0MDMACH	Queue 0 Mapped to DMA Channel This field controls the routing of the packet received in Queue 0 to the DMA channel: 0 DMA Channel 0 1 DMA Channel 1 This field is valid when the Q0DDMACH field is reset.

56.2.169 Queue 0 Transmit Operation Mode Register (MTL_TXQ0_OPERATION_MODE)

The Queue 0 Transmit Operation Mode register establishes the Transmit queue operating modes and commands.

Offset 0x0D00 Access: User Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	TQS			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	TTC			TXQEN		TSF	FTQ
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1148. Queue 0 Transmit Operation Mode Register (MTL_TXQ0_OPERATION_MODE)

Table 1200. MTL_TXQ0_OPERATION_MODE field descriptions

Field	Description
12:15 TQS	Transmit Queue Size This field indicates the size of the allocated Transmit queues in blocks of 256 bytes. The sixteenth bit is the starting bit of this field. The width of this field depends on the Tx memory size selected in your configuration. For example, if the memory size is 4096, the width of this field is 4 bits: LOG2(4096/256) = LOG2(16) = 4 bits A value of 0 indicates 256 bytes
25:27 TTC	Transmit Threshold Control These bits control the threshold level of the MTL Tx Queue. The transmission starts when the packet size within the MTL Tx Queue is larger than the threshold. In addition, full packets with length less than the threshold are also transmitted. These bits are used only when the TSF bit is reset. 000 32 001 64 010 96 011 128 100 192 101 256 110 384 111 512
28:29 TXQEN	Transmit Queue Enable This field is used to enable/disable the transmit queue 0. 00 Not enabled 01 Reserved 10 Enabled 11 Reserved
30 TSF	Transmit Store and Forward When this bit is set, the transmission starts when a full packet resides in the MTL Tx queue. When this bit is set, the TTC values specified in Bits[25:27] of this register are ignored. This bit should be changed only when the transmission is stopped.
31 FTQ	Flush Transmit Queue When this bit is set, the Tx queue controller logic is reset to its default values. Therefore, all the data in the Tx queue is lost or flushed. This bit is internally reset when the flushing operation is complete. Until this bit is reset, you should not write to the MTL_TxQ1_Operation_Mode register. The data which is already accepted by the MAC transmitter is not flushed. It is scheduled for transmission and results in underflow and runt packet transmission. **Note:** The flush operation is complete only when the Tx queue is empty and the application has accepted the pending Tx Status of all transmitted packets. To complete this flush operation, the PHY Tx clock (clk_tx_i) should be active. Access restriction applies. Setting 1 sets. Self-cleared. Setting 0 has no effect.

56.2.170 Queue 0 Underflow Counter Register (MTL_TXQ0_UNDERFLOW)

The Queue 0 Underflow Counter register contains the counter for packets aborted because of Transmit queue underflow and packets missed because of Receive queue packet flush.

Offset 0x0D04 Access: User Read

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	UFCNTOVF	UFFRMCNT										
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1149. Queue 0 Underflow Counter Register (MTL_TXQ0_UNDERFLOW)

Table 1201. MTL_TXQ0_UNDERFLOW field descriptions

Field	Description
20 UFCNTOVF	Overflow Bit for Underflow Packet Counter This bit is set every time the Tx queue Underflow Packet Counter field overflows, that is, it has crossed the maximum count. In such a scenario, the overflow packet counter is reset to all-zeros and this bit indicates that the rollover happened. Access restriction applies. Clears on read. Self-set to 1 on internal event.
21:31 UFFRMCNT	Underflow Packet Counter This field indicates the number of packets aborted by the controller because of Tx Queue Underflow. This counter is incremented each time the MAC aborts outgoing packet because of underflow. The counter is cleared when this register is read with mci_be_i[0] at 1'b1. Access restriction applies. Clears on read. Self-set to 1 on internal event.

56.2.171 Queue 0 Transmit Debug Register (MTL_TXQ0_DEBUG)

The Queue 0 Transmit Debug register gives the debug status of various blocks related to the Transmit queue.

Offset 0x0D08 Access: User Read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	STXSTSF			0	PTXQ		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	TXSTSFSTS	TXQSTS	TWCSTS	TRCSTS		TXQPAUSED
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1150. Queue 0 Transmit Debug Register (MTL_TXQ0_DEBUG)

Table 1202. MTL_TXQ0_DEBUG field descriptions

Field	Description
9:11 STXSTSF	Number of Status Words in Tx Status FIFO of Queue This field indicates the current number of status in the Tx Status FIFO of this queue. When the DTXSTS bit of MTL_Operation_Mode register is set to 1, this field does not reflect the number of status words in Tx Status FIFO.
13:15 PTXQ	Number of Packets in the Transmit Queue This field indicates the current number of packets in the Tx Queue. When the DTXSTS bit of MTL_Operation_Mode register is set to 1, this field does not reflect the number of packets in the Transmit queue.
26 TXSTSFSTS	MTL Tx Status FIFO Full Status When high, this bit indicates that the MTL Tx Status FIFO is full. Therefore, the MTL cannot accept any more packets for transmission.
27 TXQSTS	MTL Tx Queue Not Empty Status When this bit is high, it indicates that the MTL Tx Queue is not empty and some data is left for transmission.
28 TWCSTS	MTL Tx Queue Write Controller Status When high, this bit indicates that the MTL Tx Queue Write Controller is active, and it is transferring the data to the Tx Queue.

Table 1202. MTL_TXQ0_DEBUG field descriptions (continued)

Field	Description
29:30 TRCSTS	MTL Tx Queue Read Controller Status This field indicates the state of the Tx Queue Read Controller: 00 Idle state 01 Read state (transferring data to the MAC transmitter) 10 Waiting for pending Tx Status from the MAC transmitter 11 Flushing the Tx queue because of the Packet Abort request from the MAC
31 TXQPAUSED	Transmit Queue in Pause When this bit is high and the Rx flow control is enabled, it indicates that the Tx Queue is in the Pause condition (in the full-duplex only mode) because of the following: Reception of the PFC packet for the priorities assigned to the Tx Queue when PFC is enabled Reception of 802.3x Pause packet when PFC is disabled

56.2.172 Queue 0 ETS Status Register (MTL_TXQ0_ETS_STATUS)

The Queue 0 ETS Status register provides the average traffic transmitted in Queue 0.

Offset 0x0D14 Access: User Read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	ABS							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	ABS															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1151. Queue 0 ETS Status Register (MTL_TXQ0_ETS_STATUS)

Table 1203. MTL_TXQ0_ETS_STATUS field descriptions

Field	Description
8:31 ABS	Average Bits per Slot This field contains the average transmitted bits per slot.

56.2.173 Queue 0 Quantum or Weights Register (MTL_TXQ0_QUANTUM_WEIGHT)

The Queue 0 Quantum or Weights register contains weights for the Weighted Round Robin (WRR), for Queue 0.

Offset 0x0D18 Access: User Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	ISCQW				
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	ISCQW															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1152. Queue 0 Quantum or Weights Register (MTL_TXQ0_QUANTUM_WEIGHT)

Table 1204. MTL_TXQ0_QUANTUM_WEIGHT field descriptions

Field	Description
11:31 ISCQW	When DWRR algorithm is enabled for Queue 0 traffic, this field contains the quantum value in bytes to be added to credit during every queue scanning cycle. The maximum value is 0x1312D0 bytes. When WRR algorithm is enabled for Queue 0 traffic, this field contains the weight for this queue. The maximum value is 0x64. Bits [11:24] must be written to zero.

56.2.174 Interrupt Control Status Register (MTL_Q0_INTERRUPT_CONTROL_STATUS)

This register contains the interrupt enable and status bits for the queue 0 interrupts.

Offset 0x0D2C Access: User Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	RXOIE	0	0	0	0	0	0	0	RXOVFIS
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	ABPSIE	TXUIE	0	0	0	0	0	0	ABPSIS	TXUNFIS
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1153. Interrupt Control Status Register (MTL_Q0_INTERRUPT_CONTROL_STATUS)

Table 1205. MTL_Q0_INTERRUPT_CONTROL_STATUS field descriptions

Field	Description
7 RXOIE	Receive Queue Overflow Interrupt Enable When this bit is set, the Receive Queue Overflow interrupt is enabled. When this bit is reset, the Receive Queue Overflow interrupt is disabled.
15 RXOVFIS	Receive Queue Overflow Interrupt Status This bit indicates that the Receive Queue had an overflow while receiving the packet. If a partial packet is transferred to the application, the overflow status is set in RDES3[10]. This bit is cleared when the application writes 1 to this bit. Access restriction applies. Self-set to 1 on internal event. Setting 1 clears. Setting 0 has no effect.
22 ABPSIE	Average Bits Per Slot Interrupt Enable When this bit is set, the MAC asserts the interrupt or mci_intr_o interrupt when the average bits per slot status is updated. When this bit is cleared, the interrupt is not asserted for such an event.
23 TXUIE	Transmit Queue Underflow Interrupt Enable When this bit is set, the Transmit Queue Underflow interrupt is enabled. When this bit is reset, the Transmit Queue Underflow interrupt is disabled.
30 ABPSIS	Average Bits Per Slot Interrupt Status When set, this bit indicates that the MAC has updated the ABS value. This bit is cleared when the application writes 1 to this bit. Access restriction applies. Self-set to 1 on internal event. Setting 1 clears. Setting 0 has no effect.
31 TXUNFIS	Transmit Queue Underflow Interrupt Status This bit indicates that the Transmit Queue had an underflow while transmitting the packet. Transmission is suspended and an Underflow Error TDES3[2] is set. This bit is cleared when the application writes 1 to this bit. Access restriction applies. Self-set to 1 on internal event. Setting 1 clears. Setting 0 has no effect.

56.2.175 Queue 0 Receive Operation Mode Register (MTL_RXQ0_OPERATION_MODE)

The Queue 0 Receive Operation Mode register establishes the Receive queue operating modes and command.

Offset 0x0D30 Access: User Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	RQS					0	0	RFD	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	RFD		0	0	RFA				EHFC	DIS_TCP_EF	RSF	FEP	FUP	0	RTC	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1154. Queue 0 Receive Operation Mode Register (MTL_RXQ0_OPERATION_MODE)

Table 1206. MTL_RXQ0_OPERATION_MODE field descriptions

Bits	Description
7:11 RQS	Receive Queue Size This field indicates the size of the allocated Receive queues in blocks of 256 bytes. The RQS field is read-write only if the number of Rx Queues more than one and the reset value is 0x0. When the number of Rx Queues is one, the field is read-only and the configured RX FIFO size in blocks of 256 bytes is reflected in the reset value. The width of this field depends on the Rx memory size. For example, if the memory size is 8192, the width of this field is 5 bits: LOG2(8192/256) = LOG2(32) = 5 bits
14:17 RFD	Threshold for Deactivating Flow Control (in half-duplex and full-duplex modes) These bits control the threshold (fill-level of Rx queue) at which the flow control is de-asserted after activation: 0 Full minus 1 KB, that is, FULL 1 KB 1 Full minus 1.5 KB, that is, FULL 1.5 KB 2 Full minus 2 KB, that is, FULL 2 KB 3 Full minus 2.5 KB, that is, FULL 2.5 KB ... 62 Full minus 32 KB, that is, FULL 32 KB 63 Full minus 32.5 KB, that is, FULL 32.5 KB The de-assertion is effective only after flow control is asserted. Note: The value must be programed in such a way to make sure that the threshold is a positive number. When the EHFC is set high, these values are applicable only when the Rx queue size determined by the RQS field of this register, is equal to or greater than 4 KB. For a given queue size, the values ranges between 0 and the encoding for FULL minus (QSIZE - 0.5 KB) and all other values are illegal. Here the term FULL and QSIZE refers to the queue size determined by the RQS field of this register.

Table 1206. MTL_RXQ0_OPERATION_MODE field descriptions (continued)

Bits	Description
20:23 RFA	Threshold for Activating Flow Control (in half-duplex and full-duplex These bits control the threshold (fill-level of Rx queue) at which the flow control is activated: For more information on encoding for this field, refer to RFD.
24 EHFC	Enable Hardware Flow Control When this bit is set, the flow control signal operation, based on the fill-level of Rx queue, is enabled. When reset, the flow control operation is disabled.
25 DIS_TCP_EF	Disable Dropping of TCP/IP Checksum Error Packets When this bit is set, the MAC does not drop the packets which only have the errors detected by the Receive Checksum Offload engine. Such packets have errors only in the encapsulated payload. There are no errors (including FCS error) in the Ethernet packet received by the MAC. When this bit is reset, all error packets are dropped if the FEP bit is reset.
26 RSF	Receive Queue Store and Forward When this bit is set, the Ethernet module reads a packet from the Rx queue only after the complete packet has been written to it, ignoring the RTC field of this register. When this bit is reset, the Rx queue operates in the Threshold (cut-through) mode, subject to the threshold specified by the RTC field of this register.
27 FEP	Forward Error Packets When this bit is reset, the Rx queue drops packets with error status (CRC error, MII_ER, watchdog timeout, or overflow). However, if the start byte (write) pointer of a packet is already transferred to the read controller side (in Threshold mode), the packet is not dropped. When this bit is set, all packets except the runt error packets are forwarded to the application or DMA. If the RSF bit is set and the Rx queue overflows when a partial packet is written, the packet is dropped irrespective of the setting of this bit. However, if the RSF bit is reset and the Rx queue overflows when a partial packet is written, a partial packet may be forwarded to the application or DMA.
28 FUP	Forward Undersized Good Packets When this bit is set, the Rx queue forwards the undersized good packets (packets with no error and length less than 64 bytes), including pad-bytes and CRC. When this bit is reset, the Rx queue drops all packets of less than 64 bytes, unless a packet is already transferred because of the lower value of Rx Threshold, for example, RTC = 01.
30:31 RTC	Receive Queue Threshold Control These bits control the threshold level of the MTL Rx queue (in bytes): 00 64 01 32 10 96 11 128 The received packet is transferred to the application or DMA when the packet size within the MTL Rx queue is larger than the threshold. In addition, full packets with length less than the threshold are automatically transferred. This field is valid only when the RSF bit is zero. This field is ignored when the RSF bit is set to 1.

56.2.176 Queue 0 Missed Packet and Overflow Counter Register (MTL_RXQ0_MISSED_PACKET_OVERFLOW_CNT)

The Queue 0 Missed Packet and Overflow Counter register contains the counter for packets missed because of Receive queue packet flush and packets discarded because of Receive queue overflow.

Offset 0x0D34 Access: User Read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	MISCNTOVF	MISPKTCNT										
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	OVFCNTOVF	OVFPKTCNT										
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1155. Queue 0 Missed Packet and Overflow Counter Register (MTL_RXQ0_MISSED_PACKET_OVERFLOW_CNT)

Table 1207. MTL_RXQ0_MISSED_PACKET_OVERFLOW_CNT field descriptions

Field	Description
4 MISCNTOVF	Missed Packet Counter Overflow Bit When set, this bit indicates that the Rx Queue Missed Packet Counter crossed the maximum limit. Access restriction applies. Clears on read. Self-set to 1 on internal event.
5:15 MISPKTCNT	Missed Packet Counter This field indicates the number of packets missed by the Ethernet module because the application asserted ari_pkt_flush_i[] for this queue. This counter is incremented each time the application issues ari_pkt_flush_i[] for this queue. This counter is reset when this register is read with mci_be_i[0] at 1b1. This counter is incremented by 1 when the DMA discards the packet because of buffer unavailability. Access restriction applies. Clears on read. Self-set to 1 on internal event.

Table 1207. MTL_RXQ0_MISSED_PACKET_OVERFLOW_CNT field descriptions

Field	Description
20 OVFCNTOVF	Overflow Counter Overflow Bit When set, this bit indicates that the Rx Queue Overflow Packet Counter field crossed the maximum limit. Access restriction applies. Clears on read. Self-set to 1 on internal event.
21:31 OVFPKTCNT	Overflow Packet Counter This field indicates the number of packets discarded by the Ethernet module because of Receive queue overflow. This counter is incremented each time the Ethernet module discards an incoming packet because of overflow. This counter is reset when this register is read with mci_be_i[0] at 1'b1. Access restriction applies. Clears on read. Self-set to 1 on internal event.

56.2.177 Queue 0 Receive Debug Register (MTL_RXQ0_DEBUG)

The Queue 0 Receive Debug register gives the debug status of various blocks related to the Receive queue.

Offset 0x0D38 Access: User Read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	PRXQ													
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	RXQSTS		0	RRCSTS		RWCSTS
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1156. Queue 0 Receive Debug Register (MTL_RXQ0_DEBUG)

Table 1208. MTL_RXQ0_DEBUG field descriptions

Field	Description
2:15 PRXQ	Number of Packets in Receive Queue This field indicates the current number of packets in the Rx Queue. The theoretical maximum value for this field is 256KB/16B = 16K Packets, that is, Max_Queue_Size/Min_Packet_Size.
26:27 RXQSTS	MTL Rx Queue Fill-Level Status This field gives the status of the fill-level of the Rx Queue: 00 Rx Queue empty 01 Rx Queue fill-level below flow-control deactivate threshold 10 Rx Queue fill-level above flow-control activate threshold 11 Rx Queue full

Table 1208. MTL_RXQ0_DEBUG field descriptions (continued)

Field	Description
29:30 RRCSTS	MTL Rx Queue Read Controller State This field gives the state of the Rx queue Read controller: 00 Idle state 01 Reading packet data 10 Reading packet status (or timestamp) 11 Flushing the packet data and status
31 RWCSTS	MTL Rx Queue Write Controller Active Status When high, this bit indicates that the MTL Rx queue Write controller is active, and it is transferring a received packet to the Rx Queue.

56.2.178 Queue Receive Control Register (MTL_RXQ0_CONTROL)

The Queue Receive Control register controls the receive arbitration and passing of received packets to the application.

Offset 0x0D3C — Access: User Read/Write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	RXQ_FRM_ARBIT	RXQ_WEGT		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1157. Queue Receive Control Register (MTL_RXQ0_CONTROL)

Table 1209. MTL_RXQ0_CONTROL field descriptions

Field	Description
28 RXQ_FRM_ARBIT	Receive Queue Packet Arbitration When this bit is set, the Ethernet module drives the packet data to the ARI interface such that the entire packet data of currently-selected queue is transmitted before switching to other queue. When this bit is reset, the Ethernet module drives the packet data to the ARI interface such that the following amount of data of currently-selected queue is transmitted before switching to other queue: PBL amount of data (indicated by ari_qN_pbl_i[]) or complete data of a packet The status and the timestamp are not a part of the PBL data. Therefore, the Ethernet module drives the complete status (including timestamp status) during first PBL request for the packet (in store-and-forward mode) or the last PBL request for the packet (in Threshold mode).
29:31 RXQ_WEGT	Receive Queue Weight This field indicates the weight assigned to the Rx Queue 0. The weight is used as the number of continuous PBL requests or contiguous packets (depending on the RXQ_PKT_ARBIT) allocated to the queue in one arbitration cycle.

56.2.179 Transmit Q*n* Operation Mode register (MTL_TXQ*n*_OPERATION_MODE)

The Queue 1 Transmit Operation Mode register establishes the Transmit queue operating modes and commands.

Offset 0x0D00 + *n**0x40 (n=1) Access: User Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	TQS			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	TTC			TXQEN		TSF	FTQ
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1158. Transmit Q*n* Operation Mode register (MTL_TXQ*n*_OPERATION_MODE)

Table 1210. MTL_TXQ*n*_OPERATION_MODE field descriptions

Field	Description
12:15 TQS	Transmit Queue Size This field indicates the size of the allocated Transmit queues in blocks of 256 bytes. The sixteenth bit is the starting bit of this field. The width of this field depends on the Tx memory size selected in your configuration. For example, if the memory size is 2048, the width of this field is 3 bits: LOG2(4096/256) = LOG2(16) = 4 bits
25:27 TTC	Transmit Threshold Control These bits control the threshold level of the MTL Tx Queue. The transmission starts when the packet size within the MTL Tx Queue is larger than the threshold. In addition, full packets with length less than the threshold are also transmitted. These bits are used only when the TSF bit is reset. 000 32 001 64 010 96 011 128 100 192 101 256 110 384 111 512
28:29 TXQEN	Transmit Queue Enable This field is used to enable/disable the transmit queue n. 00 Not enabled 01 Enable in AV mode 10 Enabled 11 Reserved
30 TSF	Transmit Store and Forward When this bit is set, the transmission starts when a full packet resides in the MTL Tx queue. When this bit is set, the TTC values specified in Bits[25:27] of this register are ignored. This bit should be changed only when the transmission is stopped.
31 FTQ	Flush Transmit Queue When this bit is set, the Tx queue controller logic is reset to its default values. Therefore, all the data in the Tx queue is lost or flushed. This bit is internally reset when the flushing operation is complete. Until this bit is reset, you should not write to the MTL_TxQ1_Operation_Mode register. The data which is already accepted by the MAC transmitter is not flushed. It is scheduled for transmission and results in underflow and runt packet transmission. **Note:** The flush operation is complete only when the Tx queue is empty and the application has accepted the pending Tx Status of all transmitted packets. To complete this flush operation, the PHY Tx clock (clk_tx_i) should be active. Access restriction applies. Setting 1 sets. Self-cleared. Setting 0 has no effect.

56.2.180 Transmit Q*n* Underflow register (MTL_TXQ*n*_UNDERFLOW)

The Queue 1 Underflow Counter register contains the counter for packets aborted because of Transmit queue underflow and packets missed because of Receive queue packet flush.

Offset 0x0D04 + *n**0x40 (n=1) Access: User Read

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	UFCNTOVF	UFFRMCNT										
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1159. Transmit Q*n* Underflow register (MTL_TXQ*n*_UNDERFLOW)

Table 1211. MTL_TXQ*n*_UNDERFLOW field descriptions

Field	Description
20 UFCNTOVF	Overflow Bit for Underflow Packet Counter This bit is set every time the Tx queue Underflow Packet Counter field overflows, that is, it has crossed the maximum count. In such a scenario, the overflow packet counter is reset to all-zeros and this bit indicates that the rollover happened. Access restriction applies. Clears on read. Self-set to 1 on internal event.
21:31 UFFRMCNT	Underflow Packet Counter This field indicates the number of packets aborted by the controller because of Tx Queue Underflow. This counter is incremented each time the MAC aborts outgoing packet because of underflow. The counter is cleared when this register is read with mci_be_i[0] at 1'b1. Access restriction applies. Clears on read. Self-set to 1 on internal event.

56.2.181 Transmit Q*n* Debug register (MTL_TXQ*n*_DEBUG)

The Queue 1 Transmit Debug register gives the debug status of various blocks related to the Transmit queue.

Offset 0x0D08 + *n**0x40 (*n*=1) Access: User Read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	STXSTSF			0	PTXQ		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	TXSTSFSTS	TXQSTS	TWCSTS	TRCSTS		TXQPAUSED
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1160. Transmit Q Debug register (MTL_TXQ*n*_DEBUG)

Table 1212. MTL_TXQ*n*_DEBUG field descriptions

Field	Description
9:11 STXSTSF	Number of Status Words in Tx Status FIFO of Queue This field indicates the current number of status in the Tx Status FIFO of this queue. When the DTXSTS bit of MTL_Operation_Mode register is set to 1, this field does not reflect the number of status words in Tx Status FIFO.
13:15 PTXQ	Number of Packets in the Transmit Queue This field indicates the current number of packets in the Tx Queue. When the DTXSTS bit of MTL_Operation_Mode register is set to 1, this field does not reflect the number of packets in the Transmit queue.
26 TXSTSFSTS	MTL Tx Status FIFO Full Status When high, this bit indicates that the MTL Tx Status FIFO is full. Therefore, the MTL cannot accept any more packets for transmission.
27 TXQSTS	MTL Tx Queue Not Empty Status When this bit is high, it indicates that the MTL Tx Queue is not empty and some data is left for transmission.
28 TWCSTS	MTL Tx Queue Write Controller Status When high, this bit indicates that the MTL Tx Queue Write Controller is active, and it is transferring the data to the Tx Queue.

Table 1212. MTL_TXQ*n*_DEBUG field descriptions (continued)

Field	Description
29:30 TRCSTS	MTL Tx Queue Read Controller Status This field indicates the state of the Tx Queue Read Controller: 00 Idle state 01 Read state (transferring data to the MAC transmitter) 10 Waiting for pending Tx Status from the MAC transmitter 11 Flushing the Tx queue because of the Packet Abort request from the MAC
31 TXQPAUSED	Transmit Queue in Pause When this bit is high and the Rx flow control is enabled, it indicates that the Tx Queue is in the Pause condition (in the full-duplex only mode) because of the following: Reception of the PFC packet for the priorities assigned to the Tx Queue when PFC is enabled Reception of 802.3x Pause packet when PFC is disabled

56.2.182 Transmit Q*n* ETS Control register (MTL_TXQ*n*_ETS_CONTROL)

The Queue ETS Control register controls the enhanced transmission selection operation.

Offset 0x0D10 + *n**0x40 (*n*=1) Access: User Read/Write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	SLC			CC	AVALG	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1161. MTL_Transmit Q*n* ETS Control register (MTL_TXQ*n*_ETS_CONTROL)

Table 1213. MTL_TXQ*n*_ETS_CONTROL field descriptions

Field	Description
25:27 SLC	Slot Count If the credit-based shaper algorithm is enabled, the software can program the number of slots (of 125 µs duration) over which the average transmitted bits per slot, provided in the MTL_TxQ1_ETS_Status register, need to be computed for Queue 1. The encoding is as follows: 000 1 Slot 001 2 Slots 010 4 Slots 011 8 Slots 100 16 Slots 101 Reserved 110 Reserved 111 Reserved
28 CC	Credit Control When this bit is set, the accumulated credit parameter in the credit-based shaper algorithm logic is not reset to zero when there is positive credit and no packet to transmit in Channel 1. The credit accumulates even when there is no packet waiting in Channel 1 and another channel is transmitting. When this bit is reset, the accumulated credit parameter in the credit-based shaper algorithm logic is set to zero when there is positive credit and no packet to transmit in Channel 1. When there is no packet waiting in Channel 1 and other channel is transmitting, no credit is accumulated.
29 AVALG	AV Algorithm When Queue 1 is programed for AV, this field configures the scheduling algorithm for this queue: This bit when set, indicates credit based shaper algorithm (CBS) is selected for Queue 1 traffic. When reset, strict priority is selected.

56.2.183 Transmit Q*n* ETS Status register (MTL_TXQ*n*_ETS_STATUS)

The Queue 1 ETS Status register provides the average traffic transmitted in Queue 1.

Offset 0x0D14 + *n**0x40 (n=1) Access: User Read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	ABS							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	ABS															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1162. Transmit Q*n* ETS Status register (MTL_TXQ*n*_ETS_STATUS)

Table 1214. MTL_TXQ*n*_ETS_STATUS field descriptions

Field	Description
8:31 ABS	Average Bits per Slot This field contains the average transmitted bits per slot. For AV operation for Queue 1, this field is computed over number of slots, programed in the SLC field of MTL_TxQ1_ETS_CONTROL register, for Queue 1 traffic. The maximum value is 0x30D4 for 100 Mbps and 0x1E848 for 1000 Mbps. Bits [8:14] are always read as zero.

56.2.184 Transmit Q*n* Quantum Weight register (MTL_TXQ*n*_QUANTUM_WEIGHT)

The Queue 1 idleSlopeCredit, Quantum or Weights register provides the average traffic transmitted in Queue 1.

Offset 0x0D18 + *n**0x40 (n=1) Access: User Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	ISCQW				
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	ISCQW															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1163. Transmit Q*n* Quantum Weight register (MTL_TXQ*n*_QUANTUM_WEIGHT)

Table 1215. MTL_TXQ*n*_QUANTUM_WEIGHT field descriptions

Filed	Description
11:31 ISCQW	idleSlopeCredit, Quantum or Weights **idleSlopeCredit** When AV feature is enabled, this field contains the idleSlopeCredit value required for the credit-based shaper algorithm for Queue 1. This is the rate of change of credit in bits per cycle (40 ns for 100 Mbps; 8 ns for 1000 Mbps) when the credit is increasing. The software should program this field with computed credit in bits per cycle scaled by 1,024. The maximum value is portTransmitRate, that is, 0x2000 in 1000 Mbps mode and 0x1000 in 100 Mbps mode. Bits[20:14] must be written to zero. **Weights** When enabled with WRR algorithm for Queue 1 traffic, this field contains the weight for this queue. The maximum value is 0x64. Bits [11:24] must be written to zero.

56.2.185 Transmit Q*n* Send Slope Credit register (MTL_TXQ*n*_SENDSPLOPECREDIT)

The sendSlopeCredit register contains the sendSlope credit value required for the credit-based shaper algorithm for the Queue.

Offset 0x0D1C + *n**0x40 (n=1) Access: User Read/Write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	SSC													
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1164. Transmit Q*n* Send Slope Credit register (MTL_TXQ*n*_SENDSPLOPECREDIT)

Table 1216. MTL_TXQ*n*_SENDSPLOPECREDIT field descriptions

Field	Description
18:31 SSC	sendSlopeCredit With AV operation enabled, this field contains the sendSlopeCredit value required for credit-based shaper algorithm for Queue 1. This is the rate of change of credit in bits per cycle (40 ns for 100 Mbps) when the credit is decreasing. The software should program this field with computed credit in bits per cycle scaled by 1,024. The maximum value is portTransmitRate, that is, 0x1000 in 100 Mbps mode. This field should be programed with absolute sendSlopeCredit value. The credit-based shaper logic subtracts it from the accumulated credit when Channel 1 is selected for transmission.

56.2.186 Transmit Q*n* High Credit register (MTL_TXQ*n*_HICREDIT)

The hiCredit register contains the hiCredit value required for the credit-based shaper algorithm for the Queue.

Offset 0x0D20 + *n**0x40 (*n*=1) Access: User Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	HC												
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	HC															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1165. MTL_Transmit Q*n* High Credit register (MTL_TXQ*n*_HICREDIT)

Table 1217. MTL_TXQ*n*_HICREDIT field descriptions

Field	Description
3:31 HC	hiCredit When the AV feature is enabled, this field contains the hiCredit value required for the credit-based shaper algorithm. This is the maximum value that can be accumulated in the credit parameter. This is specified in bits scaled by 1,024. The maximum value is maxInterferenceSize, that is, best-effort maximum packet size (16,384 bytes or 131,072 bits). The value to be specified is 131,072 * 1,024 = 134,217,728 or 0x0800_0000.

56.2.187 Transmit Q*n* Low Credit register (MTL_TXQ*n*_LOCREDIT)

The loCredit register contains the loCredit value required for the credit-based shaper algorithm for the Queue.

Offset 0x0D24 + *n**0x40 (*n*=1) Access: User Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	LC												
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	LC															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1166. Transmit Q*n* Low Credit register (MTL_TXQ*n*_LOCREDIT)

Table 1218. MTL_TXQ*n*_LOCREDIT field descriptions

Field	Description
3:31 LC	loCredit When AV operation is enabled, this field contains the loCredit value required for the credit-based shaper algorithm. This is the minimum value that can be accumulated in the credit parameter. This is specified in bits scaled by 1,024. The maximum value is maxFrameSize transmitted from this queue, that is, 16384 * 8 * 1024 = 134,217,728 or 0x0800_0000. Because it is a negative value, the programed value is 2's complement of the value, that is, 0x1800_0000.

56.2.188 Q*n* Interrupt Control Status register (MTL_Q*n*_INTERRUPT_CONTROL_STATUS)

This register contains the interrupt enable and status bits for the queue 1 interrupts.

Offset 0x0D2C + *n**0x40 (n=1) Access: User Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	RXOIE								RXOVFIS
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	ABPSIE	TXUIE							ABPSIS	TXUNFIS
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1167. Q*n* Interrupt Control Status register (MTL_Q*n*_INTERRUPT_CONTROL_STATUS)

Table 1219. MTL_Q*n*_INTERRUPT_CONTROL_STATUS field descriptions

Field	Description
7 RXOIE	Receive Queue Overflow Interrupt Enable When this bit is set, the Receive Queue Overflow interrupt is enabled. When this bit is reset, the Receive Queue Overflow interrupt is disabled.
15 RXOVFIS	Receive Queue Overflow Interrupt Status This bit indicates that the Receive Queue had an overflow while receiving the packet. If a partial packet is transferred to the application, the overflow status is set in RDES3[21]. This bit is cleared when the application writes 1 to this bit. Access restriction applies. Self-set to 1 on internal event. Setting 1 clears. Setting 0 has no effect.
22 ABPSIE	Average Bits Per Slot Interrupt Enable When this bit is set, the MAC asserts the interrupt or mci_intr_o interrupt when the average bits per slot status is updated. When this bit is cleared, the interrupt is not asserted for such an event.
23 TXUIE	Transmit Queue Underflow Interrupt Enable When this bit is set, the Transmit Queue Underflow interrupt is enabled. When this bit is reset, the Transmit Queue Underflow interrupt is disabled.
30 ABPSIS	Average Bits Per Slot Interrupt Status When set, this bit indicates that the MAC has updated the ABS value. This bit is cleared when the application writes 1 to this bit. Access restriction applies. Self-set to 1 on internal event. Setting 1 clears. Setting 0 has no effect.
31 TXUNFIS	Transmit Queue Underflow Interrupt Status This bit indicates that the Transmit Queue had an underflow while transmitting the packet. Transmission is suspended and an Underflow Error TDES3[2] is set. This bit is cleared when the application writes 1 to this bit. Access restriction applies. Self-set to 1 on internal event. Setting 1 clears. Setting 0 has no effect.

56.2.189 Receive Q*n* Operation Mode register (MTL_RXQ*n*_OPERATION_MODE)

The Queue 1 Receive Operation Mode register establishes the Receive queue operating modes and command.

Offset 0x0D30 + *n**0x40 (*n*=1) Access: User Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	RQS							RFD	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	RFD		0	0	RFA				EHFC	DIS_TCP_EF	RSF	FEP	FUP	0	RTC	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1168. MTL_Receive Q*n* Operation Mode register (MTL_RXQ*n*_OPERATION_MODE)

Table 1220. MTL_RXQ*n*_OPERATION_MODE field descriptions

Field	Description
7:11 RQS	Receive Queue Size This field indicates the size of the allocated Receive queues in blocks of 256 bytes. The RQS field is read-write only if the number of Rx Queues more than one and the reset value is 0x0. When the number of Rx Queues is one, the field is read-only and the configured RX FIFO size in blocks of 256 bytes is reflected in the reset value. The width of this field depends on the Rx memory size selected in your configuration. For example, if the memory size is 8192, the width of this field is 5 bits: LOG2(8192/256) = LOG2(32) = 5 bits
14:17 RFD	Threshold for Deactivating Flow Control (in half-duplex and full-duplex modes) These bits control the threshold (fill-level of Rx queue) at which the flow control is de-asserted after activation: 0 Full minus 1 KB, that is, FULL 1 KB 1 Full minus 1.5 KB, that is, FULL 1.5 KB 2 Full minus 2 KB, that is, FULL 2 KB 3 Full minus 2.5 KB, that is, FULL 2.5 KB ... 62 Full minus 32 KB, that is, FULL 32 KB 63 Full minus 32.5 KB, that is, FULL 32.5 KB The de-assertion is effective only after flow control is asserted. Note: The value must be programed in such a way to make sure that the threshold is a positive number. When the EHFC is set high, these values are applicable only when the Rx queue size determined by the RQS field of this register, is equal to or greater than 4 KB. For a given queue size, the values ranges between 0 and the encoding for FULL minus (QSIZE - 0.5 KB) and all other values are illegal. Here the term FULL and QSIZE refers to the queue size determined by the RQS field of this register.
20:23 RFA	Threshold for Activating Flow Control (in half-duplex and full-duplex These bits control the threshold (fill-level of Rx queue) at which the flow control is activated: For more information on encoding for this field, refer to RFD.
24 EHFC	Enable Hardware Flow Control When this bit is set, the flow control signal operation, based on the fill-level of Rx queue, is enabled. When reset, the flow control operation is disabled.
25 DIS_TCP_EF	Disable Dropping of TCP/IP Checksum Error Packets When this bit is set, the MAC does not drop the packets which only have the errors detected by the Receive Checksum Offload engine. Such packets have errors only in the encapsulated payload. There are no errors (including FCS error) in the Ethernet packet received by the MAC. When this bit is reset, all error packets are dropped if the FEP bit is reset.
26 RSF	Receive Queue Store and Forward When this bit is set, the Ethernet module reads a packet from the Rx queue only after the complete packet has been written to it, ignoring the RTC field of this register. When this bit is reset, the Rx queue operates in the Threshold (cut-through) mode, subject to the threshold specified by the RTC field of this register.

Table 1220. MTL_RXQ*n*_OPERATION_MODE field descriptions (continued)

Field	Description
27 FEP	Forward Error Packets When this bit is reset, the Rx queue drops packets with error status (CRC error, MII_RX_ERR, watchdog timeout, or overflow). However, if the start byte (write) pointer of a packet is already transferred to the read controller side (in Threshold mode), the packet is not dropped. When this bit is set, all packets except the runt error packets are forwarded to the application or DMA. If the RSF bit is set and the Rx queue overflows when a partial packet is written, the packet is dropped irrespective of the setting of this bit. However, if the RSF bit is reset and the Rx queue overflows when a partial packet is written, a partial packet may be forwarded to the application or DMA.
28 FUP	Forward Undersized Good Packets When this bit is set, the Rx queue forwards the undersized good packets (packets with no error and length less than 64 bytes), including pad-bytes and CRC. When this bit is reset, the Rx queue drops all packets of less than 64 bytes, unless a packet is already transferred because of the lower value of Rx Threshold, for example, RTC = 01.
30:31 RTC	Receive Queue Threshold Control These bits control the threshold level of the MTL Rx queue (in bytes): 00 64 01 32 10 96 11 128 The received packet is transferred to the application or DMA when the packet size within the MTL Rx queue is larger than the threshold. In addition, full packets with length less than the threshold are automatically transferred. This field is valid only when the RSF bit is zero. This field is ignored when the RSF bit is set to 1.

56.2.190 Receive Q*n* Missed Packet and Overflow Counter register (MTL_RXQ*n*_MISSED_PACKET_OVERFLOW_CNT)

The Queue 1 Missed Packet and Overflow Counter register contains the counter for packets missed because of Receive queue packet flush and packets discarded because of Receive queue overflow.

Offset 0x0D34 + *n**0x40 (*n*=1) Access: User Read

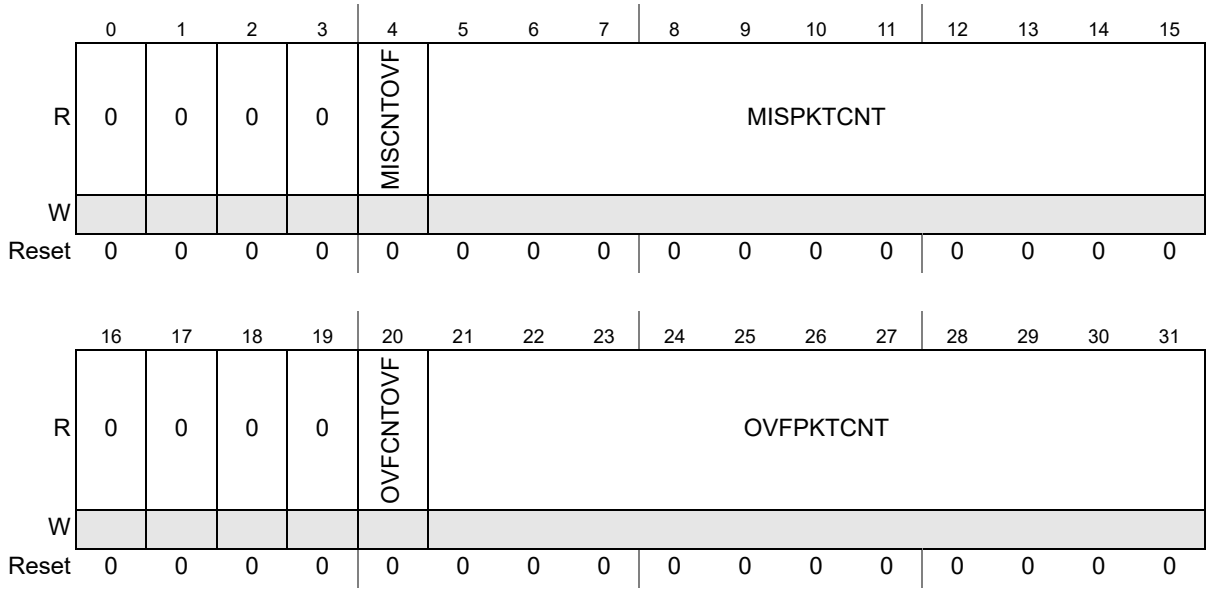


Figure 1169. Receive Q*n* Missed Packet and Overflow Counter register (MTL_RXQ*n*_MISSED_PACKET_OVERFLOW_CNT)

Table 1221. MTL_RXQ*n*_MISSED_PACKET_OVERFLOW_CNT field descriptions

Field	Description
4 MISCNTOVF	Missed Packet Counter Overflow Bit When set, this bit indicates that the Rx Queue Missed Packet Counter crossed the maximum limit. Access restriction applies. Clears on read. Self-set to 1 on internal event.
5:15 MISPKTCNT	Missed Packet Counter This field indicates the number of packets missed by the Ethernet module because the application asserted ari_pkt_flush_i[] for this queue. This counter is incremented each time the application issues ari_pkt_flush_i[] for this queue. This counter is reset when this register is read with mci_be_i[0] at 1b1. This counter is incremented by 1 when the DMA discards the packet because of buffer unavailability. Access restriction applies. Clears on read. Self-set to 1 on internal event.
20 OVFCNTOVF	Overflow Counter Overflow Bit When set, this bit indicates that the Rx Queue Overflow Packet Counter field crossed the maximum limit. Access restriction applies. Clears on read. Self-set to 1 on internal event.
21:31 OVFPKTCNT	Overflow Packet Counter This field indicates the number of packets discarded by the Ethernet module because of Receive queue overflow. This counter is incremented each time the Ethernet module discards an incoming packet because of overflow. This counter is reset when this register is read with mci_be_i[0] at 1'b1. Access restriction applies. Clears on read. Self-set to 1 on internal event.

56.2.191 Receive Q*n* Debug register (MTL_RXQ*n*_DEBUG)

The Queue 1 Receive Debug register gives the debug status of various blocks related to the Receive queue.

Offset 0x0D38 + *n**0x40 (n=1) Access: User Read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	PRXQ													
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	RXQSTS		0	RRCSTS		RWCSTS
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1170. Receive Q*n* Debug register (MTL_RXQ*n*_DEBUG)

Table 1222. MTL_RXQ*n*_DEBUG field descriptions

Field	Description
2:15 PRXQ	Number of Packets in Receive Queue This field indicates the current number of packets in the Rx Queue. The theoretical maximum value for this field is 256KB/16B = 16K Packets, that is, Max_Queue_Size/Min_Packet_Size.
26:27 RXQSTS	MTL Rx Queue Fill-Level Status This field gives the status of the fill-level of the Rx Queue: 00 Rx Queue empty 01 Rx Queue fill-level below flow-control deactivate threshold 10 Rx Queue fill-level above flow-control activate threshold 11 Rx Queue full
29:30 RRCSTS	MTL Rx Queue Read Controller State This field gives the state of the Rx queue Read controller: 00 Idle state 01 Reading packet data 10 Reading packet status (or timestamp) 11 Flushing the packet data and status
31 RWCSTS	MTL Rx Queue Write Controller Active Status When high, this bit indicates that the MTL Rx queue Write controller is active, and it is transferring a received packet to the Rx Queue.

56.2.192 Receive Q*n* Control register (MTL_RXQ*n*_CONTROL)

The Queue Receive Control register controls the receive arbitration and passing of received packets to the application.

Offset 0x0D3C + *n**0x40 (*n*=1) Access: User Read/Write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	RXQ_FRM_ARBIT	RXQ_WEGT		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1171. Receive Q*n* Control register (MTL_RXQ*n*_CONTROL)

Table 1223. MTL_RXQ*n*_CONTROL field descriptions

Field	Description
28 RXQ_FRM_ARBIT	Receive Queue Packet Arbitration When this bit is set, the Ethernet module drives the packet data to the ARI interface such that the entire packet data of currently-selected queue is transmitted before switching to other queue. When this bit is reset, the Ethernet module drives the packet data to the ARI interface such that the following amount of data of currently-selected queue is transmitted before switching to other queue: PBL amount of data (indicated by ari_qN_pbl_i[]) or complete data of a packet The status and the timestamp are not a part of the PBL data. Therefore, the Ethernet module drives the complete status (including timestamp status) during first PBL request for the packet (in store-and-forward mode) or the last PBL request for the packet (in Threshold mode).
29:31 RXQ_WEGT	Receive Queue Weight This field indicates the weight assigned to the Rx Queue 0. The weight is used as the number of continuous PBL requests or contiguous packets (depending on the RXQ_PKT_ARBIT) allocated to the queue in one arbitration cycle.

56.2.193 DMA_Mode Register (DMA_MODE)

The Bus Mode register establishes the bus operating modes for the DMA.

Offset 0x1000 Access: User Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	INTM	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	PR			TXPR	0	0	0	0	0	0	TAA			DA	SWR
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1172. DMA_Mode Register (DMA_MODE)

Table 1224. DMA_MODE field descriptions

Field	Description
14:15 INTM	Interrupt Mode This field defines the interrupt mode of the Ethernet module. The behavior of the following outputs changes depending on the following settings: sbd_perch_tx_intr_o[] (Transmit Per Channel Interrupt) sbd_perch_rx_intr_o[] (Receive Per Channel Interrupt) interrupt (Common Interrupt) It also changes the behavior of the RI/TI bits in the DMA_CH0_Status.
17:19 PR	Priority Ratio These bits control the priority ratio in weighted round-robin arbitration between the Rx DMA and Tx DMA. These bits are valid only when the DA bit is reset. The priority ratio is Rx:Tx or Tx:Rx depending on whether the TXPR bit is reset or set. 000 The priority ratio is 1:1 001 The priority ratio is 2:1 010 The priority ratio is 3:1 011 The priority ratio is 4:1 100 The priority ratio is 5:1 101 The priority ratio is 6:1 110 The priority ratio is 7:1 111 The priority ratio is 8:1
20 TXPR	Transmit Priority When set, this bit indicates that the Tx DMA has higher priority than the Rx DMA during arbitration for the system-side bus.
27:29 TAA	Transmit Arbitration Algorithm This field is used to select the arbitration algorithm for the Transmit side when multiple Tx DMAs are selected. 000 Fixed priority In fixed priority, Channel 0 has the lowest priority and the last channel has the highest priority. 001 Weighted Strict Priority (WSP) 010 Weighted Round-Robin (WRR) 011 Reserved ... 111 Reserved

Table 1224. DMA_MODE field descriptions (continued)

Field	Description
30 DA	DMA Tx or Rx Arbitration Scheme This bit specifies the arbitration scheme between the Transmit and Receive paths of all channels: 0 Weighted Round-Robin with Rx:Tx or Tx:Rx The priority between the paths is according to the priority specified in Bits[17:19] and the priority weight is specified in the TXPR bit. 1 Fixed Priority The Tx path has priority over the Rx path when the TXPR bit is set. Otherwise, the Rx path has priority over the Tx path.
31 SWR	Software Reset When this bit is set, the MAC and the DMA controller reset the logic and all internal registers of the DMA, MTL, and MAC. This bit is automatically cleared after the reset operation is complete in all Ethernet module clock domains. Before reprogramming any Ethernet module register, a value of zero should be read in this bit. **Note:** The reset operation is complete only when all resets in all active clock domains are de-asserted. Therefore, it is essential that all PHY inputs clocks (applicable for the selected PHY interface) are present for software reset completion. The time to complete the software reset operation depends on the frequency of the slowest active clock. Access restriction applies. Setting 1 sets. Self-cleared. Setting 0 has no effect.

56.2.194 System Bus Mode Register (DMA_SYSBUS_MODE)

The System Bus mode register controls the behavior of the AHB master. It mainly controls burst splitting and number of outstanding requests.

Offset 0x1004 Access: User Read/Write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	RB	MB	0	AAL	0	0	0	0	0	0	0	0	0	0	0	FB
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1173. System Bus Mode Register (DMA_SYSBUS_MODE)

Table 1225. DMA_SYSBUS_MODE field descriptions

Field	Description
16 RB	Rebuild INCRx Burst When this bit is set high and the AHB master gets SPLIT, RETRY, or Early Burst Termination (EBT) response, the AHB master interface rebuilds the pending beats of any initiated burst transfer with INCRx and SINGLE transfers. By default, the AHB master interface rebuilds pending beats of an EBT with an unspecified (INCR) burst.
17 MB	Mixed Burst When this bit is set high and the FB bit is low, the AHB master performs undefined bursts transfers (INCR) for burst length of 16 or more. For burst length of 16 or less, the AHB master performs fixed burst transfers (INCRx and SINGLE).
19 AAL	Address-Aligned Beats When this bit is set to 1, AHB master performs address-aligned burst transfers on Read and Write channels.
31 FB	Fixed Burst Length When this bit is set to 1, the AHB master will initiate burst transfers of specified length (INCRx or SINGLE). When this bit is set to 0, the AHB master will initiate transfers of unspecified length (INCR) or SINGLE transfers.

56.2.195 Interrupt Status Register (DMA_INTERRUPT_STATUS)

The application reads this Interrupt Status register during interrupt service routine or polling to determine the interrupt status of DMA channels, MTL queues, and the MAC.

Offset 0x1008 Access: User Read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	MACIS	MTLIS
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	DC1IS	DC0IS
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1174. Interrupt Status Register (DMA_INTERRUPT_STATUS)

Table 1226. DMA_INTERRUPT_STATUS field descriptions

Field	Description
14 MACIS	MAC Interrupt Status This bit indicates an interrupt event in the MAC. To reset this bit to 1'b0, the software must read the corresponding register in the MAC to get the exact cause of the interrupt and clear its source.
15 MTLIS	MTL Interrupt Status This bit indicates an interrupt event in the MTL. To reset this bit to 1'b0, the software must read the corresponding register in the MTL to get the exact cause of the interrupt and clear its source.
30 DC1IS	DMA Channel 1 Interrupt Status This bit indicates an interrupt event in DMA Channel 1. To reset this bit to 1'b0, the software must read the corresponding register in DMA Channel 1 to get the exact cause of the interrupt and clear its source.
31 DC0IS	DMA Channel 0 Interrupt Status This bit indicates an interrupt event in DMA Channel 0. To reset this bit to 1'b0, the software must read the corresponding register in DMA Channel 0 to get the exact cause of the interrupt and clear its source.

56.2.196 Debug Status 0 Register (DMA_DEBUG_STATUS0)

The Debug Status 0 register gives the Receive and Transmit process status for DMA Channel 0-Channel 1 for debugging purpose.

Offset 0x100C Access: User Read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	TPS1				RPS1			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	TPS0				RPS0				0	0	0	0	0	0	0	AXWHSTS
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1175. Debug Status 0 Register (DMA_DEBUG_STATUS0)

Table 1227. DMA_DEBUG_STATUS0 field descriptions

Field	Description
8:11 TPS1	DMA Channel 1 Transmit Process State This field indicates the Tx DMA FSM state for Channel 1. This field is similar to the TPS0 field.
12:15 RPS1	DMA Channel 1 Receive Process State This field indicates the Rx DMA FSM state for Channel 1. This field is similar to the RPS0 field.
16:19 TPS0	DMA Channel 0 Transmit Process State This field indicates the Tx DMA FSM state for Channel 0: 000 Stopped (Reset or Stop Transmit Command issued) 001 Running (Fetching Tx Transfer Descriptor) 010 Running (Waiting for status) 011 Running (Reading Data from system memory buffer and queuing it to the Tx buffer (Tx FIFO)) 100 Timestamp write state 101 Reserved for future use 110 Suspended (Tx Descriptor Unavailable or Tx Buffer Underflow) 111 Running (Closing Tx Descriptor) This field does not generate an interrupt.
20:23 RPS0	DMA Channel 0 Receive Process State This field indicates the Rx DMA FSM state for Channel 0: 000 Stopped (Reset or Stop Receive Command issued) 001 Running (Fetching Rx Transfer Descriptor) 010 Reserved for future use 011 Running (Waiting for Rx packet) 100 Suspended (Rx Descriptor Unavailable) 101 Running (Closing the Rx Descriptor) 110 Timestamp write state 111 Running (Transferring the received packet data from the Rx buffer to the system memory) The MSB of this field always returns 0. This field does not generate an interrupt.
31 AXWHSTS	AHB Master Status When high, this bit indicates that the AHB master FSMs are in the non-idle state.

56.2.197 DMA Channel *n* Control register (DMA_CH*n*_CONTROL)

The DMA Channel*n* Control register specifies the MSS value for segmentation, length to skip between two descriptors, and also the features such as 8xPBL mode.

Offset 0x1100 + *n**0x80 (n=0 to 1) Access: User Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	DSL			0	PBLx8
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

Figure 1176. DMA Channel *n* Control register (DMA_CH*n*_CONTROL)

Table 1228. DMA_CH*n*_CONTROL field descriptions

Field	Description
11:13 DSL	Descriptor Skip Length This bit specifies the Dword number (64-bit bus) to skip between two unchained descriptors. The address skipping starts from the end of the current descriptor to the start of the next descriptor. When the DSL value is equal to zero, the DMA takes the descriptor table as contiguous.
15 PBLx8	8xPBL mode When this bit is set, the PBL value programed in Bits[10:15] in DMA_CH0_Tx_Control is multiplied eight times. Therefore, the DMA transfers the data in 8, 16, 32, 64, 128, and 256 beats depending on the PBL value.

56.2.198 DMA Channel *n* Transmit Control register (DMA_CH*n*_TX_CONTROL)

The DMA Channel Transmit Control register controls the Tx features such as PBL, and Tx Channel weights.

Offset 0x1104 + *n**0x80 (n=0 to 1) Access: User Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	TxPBL					
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	OSF	TCW			ST
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1177. DMA Channel *n* Transmit Control register (DMA_CH*n*_TX_CONTROL)

Table 1229. DMA_CH*n*_TX_CONTROL field descriptions

Field	Description
10:15 TxPBL	Transmit Programmable Burst Length These bits indicate the maximum number of beats to be transferred in one DMA data transfer. This is the maximum value that is used in a single block Read or Write. The DMA always attempts to burst as specified in PBL each time it starts a burst transfer on the application bus. You can program PBL with any of the following values: 1, 2, 4, 8, 16, or 32. Any other value results in undefined behavior. To transfer more than 32 beats, perform the following steps: – Set the PBLx8 mode in DMA_CH0_Control register. – Set the PBL.
27 OSF	Operate on Second Packet When this bit is set, it instructs the DMA to process the second packet of the Transmit data even before the status for the first packet is obtained.
28:30 TCW	Transmit Channel Weight This field indicates the weight assigned to the corresponding Transmit channel. When reset is complete, this field is set to 0 for all channels by default, resulting in equal weights to all channels.
31 ST	Start or Stop Transmission Command When this bit is set, transmission is placed in the Running state. The DMA checks the Transmit list at the current position for a packet to be transmitted. The DMA tries to acquire descriptor from either of the following positions: – The current position in the list This is the base address of the Transmit list set by the DMA_CH0_TxDesc_List_Address register. – The position at which the transmission was previously stopped If the DMA does not own the current descriptor, the transmission enters the Suspended state and the TBU bit of the DMA_CH0_Status register is set. The Start Transmission command is effective only when the transmission is stopped. If the command is issued before setting the DMA_CH0_TxDesc_List_Address register, the DMA behavior is unpredictable. When this bit is reset, the transmission process is placed in the Stopped state after completing the transmission of the current packet. The Next Descriptor position in the Transmit list is saved, and it becomes the current position when the transmission is restarted. To change the list address, you need to program DMA_CH0_TxDesc_List_Address register with a new value when this bit is reset. The new value is considered when this bit is set again. The stop transmission command is effective only when the transmission of the current packet is complete or the transmission is in the Suspended state.

56.2.199 DMA Channel *n* Receive Control register (DMA_CH*n*_RX_CONTROL)

The DMA Channel *n* Receive Control register controls the Rx features such as PBL, buffer size, and extended status.

Offset 0x1108 + *n**0x80 (n=0 to 1) Access: User Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	RPF	0	0	0	0	0	0	0	0	0	RxPBL					
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	RBSZ														SR
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1178. DMA Channel *n* Receive Control register (DMA_CH*n*_RX_CONTROL)

Table 1230. DMA_CH*n*_RX_CONTROL field descriptions

Field	Description
0 RPF	DMA Rx Channel0 Packet Flush When this bit is set to 1, the DMA will automatically flush the packet from the Rx Queues destined to DMA Rx Channel 0 when the DMA Rx Channel 0 is stopped after a system bus error has occurred. The flushing happens on the Read side of the Rx Queue. When this bit is set to 0 the EQOS will not flush the packet in the Rx Queue destined to DMA Rx Channel 0 after the DMA is stopped due to a system bus error.
10:15 RxPBL	Receive Programmable Burst Length These bits indicate the maximum number of beats to be transferred in one DMA data transfer. This is the maximum value that is used in a single block Read or Write. The DMA always attempts to burst as specified in PBL each time it starts a burst transfer on the application bus. You can program PBL with any of the following values: 1, 2, 4, 8, 16, or 32. Any other value results in undefined behavior. To transfer more than 32 beats, perform the following steps: – Set the PBLx8 mode in the DMA_CH0_Control register. – Set the PBL.

Table 1230. DMA_CH*n*_RX_CONTROL field descriptions (continued)

Field	Description
17:30 RBSZ	Receive Buffer size This field indicates the size of the Rx buffers specified in bytes. The maximum buffer size is limited to 16K bytes. **Note:** The buffer size must be a multiple of 4, 8, or 16 depending on the bus widths (32, 64, or 128 respectively). This is required even if the value of buffer address pointer is not aligned to bus width. If the buffer size is not a multiple of 4, 8, or 16, it may result into undefined behavior. The 3 LSB bits for 64-bit bus width are ignored and the DMA internally takes the LSB bits as all-zero. Therefore, these LSB bits are read-only (RO).
31 SR	Start or Stop Receive When this bit is set, the DMA tries to acquire the descriptor from the Receive list and processes the incoming packets. The DMA tries to acquire descriptor from either of the following positions: The current position in the list This is the address set by the DMA_CH0_RxDesc_List_Address register. The position at which the Rx process was previously stopped If the DMA does not own the current descriptor, the reception is suspended and the RBU bit of the DMA_CH0_Status register is set. The Start Receive command is effective only when the reception is stopped. If the command is issued before setting the DMA_CH0_RxDesc_List_Address register, the DMA behavior is unpredictable. When this bit is reset, the Rx DMA operation is stopped after the transfer of the current packet. The next descriptor position in the Receive list is saved, and it becomes the current position after the Rx process is restarted. The Stop Receive command is effective only when the Rx process is in the Running (waiting for Rx packet) or Suspended state.

56.2.200 DMA Channel *n* Transmit Descriptor List Address register (DMA_CH*n*_TXDESC_LIST_ADDRESS)

The Channel *n* Tx Descriptor List Address register points the DMA to the start of Transmit descriptor list. The descriptor lists reside in the physical memory space of the application and must be Dword aligned (for 64-bit data bus). The DMA internally converts it to bus width aligned address by making the corresponding LSB to low.

You can write to this register only when the Tx DMA has stopped, that is, the ST bit is set to zero in DMA_CH0_Tx_Control register. When stopped, this register can be written with a new descriptor list address. When you set the ST bit to 1, the DMA takes the newly-programed descriptor base address. If this register is not changed when the ST bit is set to 0, the DMA takes the descriptor address where it was stopped earlier.

Offset 0x1114 + *n**0x80 (*n*=0 to 1) Access: User Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R/W	TDESLA															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R/W	TDESLA															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1179. DMA Channel *n* Transmit Descriptor List Address register (DMA_CH*n*_TXDESC_LIST_ADDRESS)

Table 1231. DMA_CH*n*_TXDESC_LIST_ADDRESS field descriptions

Field	Description
0:31 TDESLA	Start of Transmit List This field contains the base address of the first descriptor in the Transmit descriptor list. The DMA ignores the 3 LSB bits for 64-bit data bus width and internally takes these bits as all-zero. Therefore, these LSB bits are read-only (RO).

56.2.201 DMA Channel *n* Receive Descriptor List Address register (DMA_CH*n*_RXDESC_LIST_ADDRESS)

The Channel*n* Rx Descriptor List Address register points the DMA to the start of Receive descriptor list.

This register points to the start of the Receive Descriptor List. The descriptor lists reside in the physical memory space of the application and must be Dword aligned (for 64-bit data bus). The DMA internally converts it to bus width aligned address by making the corresponding LS bits low. Writing to this register is permitted only when reception is stopped. When stopped, this register must be written to before the receive Start command is given. You can write to this register only when Rx DMA has stopped, that is, SR bit is set to zero in DMA_CH0_Rx_Control register. When stopped, this register can be written with a new descriptor list address.

When you set the SR bit to 1, the DMA takes the newly programed descriptor base address.

Offset 0x111C + *n**0x80 (*n*=0 to 1) Access: User Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R/W	RDESLA															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R/W	RDESLA															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1180. DMA Channel *n* Receive Descriptor List Address register (DMA_CH*n*_RXDESC_LIST_ADDRESS)

Table 1232. DMA_CH*n*_RXDESC_LIST_ADDRESS field descriptions

Field	Description
0:31 RDESLA	Start of Receive List This field contains the base address of the first descriptor in the Rx Descriptor list. The DMA ignores the 3 LSB bits for 64-bit data bus width and internally takes these bits as all-zero. Therefore, these LSB bits are read-only (RO).

56.2.202 DMA Channel *n* Transmit Descriptor Tail Pointer register (DMA_CH*n*_TXDESC_TAIL_POINTER)

The Channel*n* Tx Descriptor Tail Pointer register points to an offset from the base and indicates the location of the last valid descriptor.

Offset 0x1120 + *n**0x80 (*n*=0 to 1) Access: User Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R/W	TDTP															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R/W	TDTP															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1181. DMA Channel *n* Transmit Descriptor Tail Pointer register (DMA_CH*n*_TXDESC_TAIL_POINTER)

Table 1233. DMA_CH*n*_TXDESC_TAIL_POINTER field descriptions

Field	Description
0:31 TDTP	Transmit Descriptor Tail Pointer This field contains the tail pointer for the Tx descriptor ring. The software writes the tail pointer to add more descriptors to the Tx channel. The hardware tries to transmit all packets referenced by the descriptors between the head and the tail pointer registers.

56.2.203 DMA Channel *n* Receive Descriptor Tail Pointer (DMA_CH*n*_RXDESC_TAIL_POINTER)

The Channel*n* Rx Descriptor Tail Pointer Points to an offset from the base and indicates the location of the last valid descriptor.

Offset 0x1128 + *n**0x80 (n=0 to 1) Access: User Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	RDTP															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	RDTP															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1182. DMA Channel *n* Receive Descriptor Tail Pointer (DMA_CH*n*_RXDESC_TAIL_POINTER)

Table 1234. DMA_CH*n*_RXDESC_TAIL_POINTER field descriptions

Field	Description
0:31 RDTP	Receive Descriptor Tail Pointer This field contains the tail pointer for the Rx descriptor ring. The software writes the tail pointer to add more descriptors to the Rx channel. The hardware tries to write all received packets to the descriptors referenced between the head and the tail pointer registers.

56.2.204 DMA Channel *n* Transmit Descriptor Ring Length register (DMA_CH*n*_TXDESC_RING_LENGTH)

The Tx Descriptor Ring Length register contains the length of the Transmit descriptor ring.

Offset 0x112C + *n**0x80 (n=0 to 1) Access: User Read/Write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	TDRL									
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1183. DMA Channel *n* Transmit Descriptor Ring Length register (DMA_CH*n*_TXDESC_RING_LENGTH)

Table 1235. DMA_CH*n*_TXDESC_RING_LENGTH field descriptions

Field	Description
22:31 TDRL	Transmit Descriptor Ring Length This field sets the maximum number of Tx descriptors in the circular descriptor ring. The maximum number of descriptors is limited to 1K descriptors. It's recommended to have a minimum ring descriptor length of 4.

56.2.205 DMA Channel *n* Receive Descriptor Ring Length register (DMA_CH*n*_RXDESC_RING_LENGTH)

The Channel*n* Rx Descriptor Ring Length register contains the length of the Receive descriptor circular ring.

Offset 0x1130 + *n**0x80 (n=0 to 1) Access: User Read/Write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	RDRL									
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1184. DMA Channel *n* Receive Descriptor Ring Length register (DMA_CH*n*_RXDESC_RING_LENGTH)

Table 1236. DMA_CH*n*_RXDESC_RING_LENGTH field description

Field	Description
22:31 RDRL	Receive Descriptor Ring Length This register sets the maximum number of Rx descriptors in the circular descriptor ring. The maximum number of descriptors is limited to 1K descriptors.

56.2.206 DMA Channel *n* Interrupt Enable register (DMA_CH*n*_INTERRUPT_ENABLE)

The Channel*n* Interrupt Enable register enables the interrupts reported by the Status register.

Offset 0x1134 + *n**0x80 (n=0 to 1) Access: User Read/Write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	NIE	AIE	CDEE	FBEE	ERIE	ETIE	RWTE	RSE	RBUE	RIE	0	0	0	TBUE	TXSE	TIE
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1185. DMA Channel *n* Interrupt Enable register (DMA_CH*n*_INTERRUPT_ENABLE)

Table 1237. DMA_CH*n*_INTERRUPT_ENABLE field descriptions

Field	Description
16 NIE	Normal Interrupt Summary Enable When this bit is set, the normal interrupt summary is enabled. This bit enables the following interrupts in the DMA_CH0_Status register: – Bit 31: Transmit Interrupt – Bit 29: Transmit Buffer Unavailable – Bit 25: Receive Interrupt – Bit 20: Early Receive Interrupt When this bit is reset, the normal interrupt summary is disabled.
17 AIE	Abnormal Interrupt Summary Enable When this bit is set, the abnormal interrupt summary is enabled. This bit enables the following interrupts in the DMA_CH0_Status register: – Bit 30: Transmit Process Stopped – Bit 24: Rx Buffer Unavailable – Bit 23: Receive Process Stopped – Bit 22: Receive Watchdog Timeout – Bit 21: Early Transmit Interrupt – Bit 19: Fatal Bus Error When this bit is reset, the abnormal interrupt summary is disabled.
18 CDEE	Context Descriptor Error Enable When this bit is set along with the AIE bit, the Context Descriptor error interrupt is enabled. When this bit is reset, the Context Descriptor error interrupt is disabled.
19 FBEE	Fatal Bus Error Enable When this bit is set along with the AIE bit, the Fatal Bus error interrupt is enabled. When this bit is reset, the Fatal Bus Error error interrupt is disabled.
20 ERIE	Early Receive Interrupt Enable When this bit is set along with the NIE bit, the Early Receive interrupt is enabled. When this bit is reset, the Early Receive interrupt is disabled.
21 ETIE	Early Transmit Interrupt Enable When this bit is set along with the AIE bit, the Early Transmit interrupt is enabled. When this bit is reset, the Early Transmit interrupt is disabled.
22 RWTE	Receive Watchdog Timeout Enable When this bit is set along with the AIE bit, the Receive Watchdog Timeout interrupt is enabled. When this bit is reset, the Receive Watchdog Timeout interrupt is disabled.
23 RSE	Receive Stopped Enable When this bit is set along with the AIE bit, the Receive Stopped Interrupt is enabled. When this bit is reset, the Receive Stopped interrupt is disabled.
24 RBUE	Receive Buffer Unavailable Enable When this bit is set along with the AIE bit, the Receive Buffer Unavailable interrupt is enabled. When this bit is reset, the Receive Buffer Unavailable interrupt is disabled.
25 RIE	Receive Interrupt Enable When this bit is set along with the NIE bit, the Receive Interrupt is enabled. When this bit is reset, the Receive Interrupt is disabled.

Table 1237. DMA_CH*n*_INTERRUPT_ENABLE field descriptions (continued)

Field	Description
29 TBUE	Transmit Buffer Unavailable Enable When this bit is set along with the NIE bit, the Transmit Buffer Unavailable interrupt is enabled. When this bit is reset, the Transmit Buffer Unavailable interrupt is disabled.
30 TXSE	Transmit Stopped Enable When this bit is set along with the AIE bit, the Transmission Stopped interrupt is enabled. When this bit is reset, the Transmission Stopped interrupt is disabled.
31 TIE	Transmit Interrupt Enable When this bit is set along with the NIE bit, the Transmit Interrupt is enabled. When this bit is reset, the Transmit Interrupt is disabled.

56.2.207 DMA Channel *n* Receive Interrupt Watchdog Timer register (DMA_CH*n*_Rx_INTERRUPT_WATCHDOG_TIMER)

The Receive Interrupt Watchdog Timer register indicates the watchdog timeout for Receive Interrupt (RI) from the DMA. When this register is written with a non-zero value, it enables the watchdog timer for the RI bit of the DMA_CH*n*_Status register.

Offset 0x1138 + *n**0x80 (n=0 to 1) Access: User Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	RWTU	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	RWT							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1186. DMA Channel *n* Receive Interrupt Watchdog Timer register (DMA_CH*n*_Rx_INTERRUPT_WATCHDOG_TIMER)

Table 1238. DMA_CH*n*_Rx_INTERRUPT_WATCHDOG_TIMER field descriptions

Field	Description
14:15 RWTU	Receive Interrupt Watchdog Timer Count Units This field indicates the number of system clock cycles corresponding to one unit in RWT field. 00 256 01 512 10 1024 11 2048 For example, when RWT=2 and RWTU=1, the watchdog timer is set for 2*512=1024 system clock cycles.
24:31 RWT	Receive Interrupt Watchdog Timer Count This field indicates the number of system clock cycles, multiplied by factor indicated in RWTU field, for which the watchdog timer is set. The watchdog timer is triggered with the programed value after the Rx DMA completes the transfer of a packet for which the RI bit is not set in the DMA_CH0_Status register, because of the setting of Interrupt Enable bit in the corresponding descriptor RDES3[1]. When the watchdog timer runs out, the RI bit is set and the timer is stopped. The watchdog timer is reset when the RI bit is set high because of automatic setting of RI as per the Interrupt Enable bit RDES3[1] of any received packet.

56.2.208 DMA Channel *n* Slot Function Control Status register (DMA_CH*n*_SLOT_FUNCTION_CONTROL_STATUS)

The Slot Function Control and Status register contains the control bits for slot function and the status for Transmit path.

Offset 0x113C + *n**0x80 (n=0 to 1) Access: User Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	RSN			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	ASC	ESC
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1187. DMA Channel *n* Slot Function Control Status register (DMA_CH*n*_SLOT_FUNCTION_CONTROL_STATUS)

Table 1239. DMA_CH*n*_SLOT_FUNCTION_CONTROL_STATUS field descriptions

Field	Description
12:15 RSN	Reference Slot Number This field gives the current value of the reference slot number in the DMA. It is used for slot comparison.
30 ASC	Advance Slot Check When set, this bit enables the DMA to fetch the data from the buffer when the slot number (SLOTNUM) programed in the Tx descriptor is – equal to the reference slot number given in the RSN field or – ahead of the reference slot number by up to two slots This bit is applicable only when the ESC bit is set.
31 ESC	Enable Slot Comparison When set, this bit enables the checking of the slot numbers programed in the Tx descriptor with the current reference given in the RSN field. The DMA fetches the data from the corresponding buffer only when the slot number is – equal to the reference slot number or – ahead of the reference slot number by one slot When reset, this bit disables the checking of the slot numbers. The DMA fetches the data immediately after the descriptor is processed.

56.2.209 DMA Channel *n* Current Application Transmit Descriptor register (DMA_CH*n*_CURRENT_APP_TXDESC)

The Channel*n* Current Application Transmit Descriptor register points to the current Transmit descriptor read by the DMA.

Offset 0x1144 + *n**0x80 (n=0 to 1) Access: User Read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	CURTDESAPTR															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	CURTDESAPTR															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1188. DMA Channel *n* Current Application Transmit Descriptor register (DMA_CH*n*_CURRENT_APP_TXDESC)

Table 1240. DMA_CHn_CURRENT_APP_TXDESC field descriptions

Field	Description
0:31 CURTDESAPTR	Application Transmit Descriptor Address Pointer The DMA updates this pointer during Tx operation. This pointer is cleared on reset.

56.2.210 DMA Channel *n* Current Application Transmit Descriptor register (DMA_CH*n*_CURRENT_APP_RXDESC)

The Channel*n* Current Application Receive Descriptor register points to the current Receive descriptor read by the DMA.

Offset 0x114C + *n**0x80 (n=0 to 1) Access: User Read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	CURRDESAPTR															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	CURRDESAPTR															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1189. DMA Channel *n* Current Application Transmit Descriptor register (DMA_CH*n*_CURRENT_APP_RXDESC)

Table 1241. DMA_CH*n*_CURRENT_APP_RXDESC field descriptions

Field	Description
0:31 CURRDESAPTR	Application Receive Descriptor Address Pointer The DMA updates this pointer during Rx operation. This pointer is cleared on reset.

56.2.211 DMA Channel *n* Current Application Transmit Buffer register (DMA_CH*n*_CURRENT_APP_TXBUFFER)

The Channel*n* Current Application Transmit Buffer Address register points to the current Tx buffer address read by the DMA.

Offset 0x1154 + *n**0x80 (n=0 to 1) Access: User Read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	CURTBUFAPTR															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	CURTBUFAPTR															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1190. DMA Channel *n* Current Application Transmit Buffer register (DMA_CH*n*_CURRENT_APP_TXBUFFER)

Table 1242. DMA_CH*n*_CURRENT_APP_TXBUFFER field descriptions

Field	Description
0:31 CURTBUFAPTR	Application Transmit Buffer Address Pointer The DMA updates this pointer during Tx operation. This pointer is cleared on reset.

56.2.212 DMA Channel *n* Current Application Receive Buffer register (DMA_CH*n*_CURRENT_APP_RXBUFFER)

The Channel 0 Current Application Receive Buffer Address register points to the current Rx buffer address read by the DMA.

Offset 0x115C + *n**0x80 (n=0 to 1) Access: User Read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	CURRBUFAPTR															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	CURRBUFAPTR															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1191. DMA Channel *n* Current Application Receive Buffer register (DMA_CH*n*_CURRENT_APP_RXBUFFER)

Table 1243. DMA_CH*n*_CURRENT_APP_RXBUFFER field descriptions

Field	Description
0:31 CURRBUFAPTR	Application Receive Buffer Address Pointer The DMA updates this pointer during Rx operation. This pointer is cleared on reset.

56.2.213 DMA Channel *n* Status register (DMA_CH*n*_STATUS)

The software driver (application) reads the Status register during interrupt service routine or polling to determine the status of the DMA.

Offset 0x1160 + *n**0x80 (n=0 to 1) Access: User Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	REB			TEB		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	NIS	AIS	CDE	FBE	ERI	ETI	RWT	RPS	RBU	RI	0	0	0	TBU	TPS	TI
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1192. DMA Channel *n* Status register (DMA_CH*n*_STATUS)

Table 1244. DMA_CH*n*_STATUS field descriptions

Field	Description
10:12 REB	Rx DMA Error Bits This field indicates the type of error that caused a Bus Error. For example, error response on the AHB interface. Bit 10 1'b1: Error during data transfer by Rx DMA 1'b0: No Error during data transfer by Rx DMA Bit 11 1'b1: Error during descriptor access 1'b0: Error during data buffer access Bit 12 1'b1: Error during read transfer 1'b0: Error during write transfer This field is valid only when the FBE bit is set. This field does not generate an interrupt.
13:15 TEB	Tx DMA Error Bits This field indicates the type of error that caused a Bus Error. For example, error response on the AHB interface. Bit 13 1'b1: Error during data transfer by Tx DMA 1'b0: No Error during data transfer by Tx DMA Bit 14 1'b1: Error during descriptor access 1'b0: Error during data buffer access Bit 15 1'b1: Error during read transfer 1'b0: Error during write transfer This field is valid only when the FBE bit is set. This field does not generate an interrupt.

Table 1244. DMA_CH*n*_STATUS field descriptions (continued)

Field	Description
16 NIS	Normal Interrupt Summary Normal Interrupt Summary bit value is the logical OR of the following bits when the corresponding interrupt bits are enabled in the DMA_CH0_Interrupt_Enable register: Bit 31: Transmit Interrupt Bit 29: Transmit Buffer Unavailable Bit 25: Receive Interrupt Bit 20: Early Receive Interrupt Only unmasked bits (interrupts for which interrupt enable is set in DMA_CH0_Interrupt_Enable register) affect the Normal Interrupt Summary bit. This is a sticky bit. You must clear this bit (by writing 1 to this bit) each time a corresponding bit which causes NIS to be set is cleared. Access restriction applies. Self-set to 1 on internal event. Setting 1 clears. Setting 0 has no effect.
17 AIS	Abnormal Interrupt Summary Abnormal Interrupt Summary bit value is the logical OR of the following when the corresponding interrupt bits are enabled in the DMA_CH0_Interrupt_Enable register: Bit 30: Transmit Process Stopped Bit 24: Receive Buffer Unavailable Bit 23: Receive Process Stopped Bit 21: Early Transmit Interrupt Bit 19: Fatal Bus Error Bit 18: Context Descriptor Error Only unmasked bits affect the Abnormal Interrupt Summary bit. This is a sticky bit. You must clear this bit (by writing 1 to this bit) each time a corresponding bit, which causes AIS to be set, is cleared. Access restriction applies. Self-set to 1 on internal event. Setting 1 clears. Setting 0 has no effect.
18 CDE	Context Descriptor Error This bit indicates that the DMA Tx engine received a context descriptor in the middle of a packet (in an intermediate descriptor), and the DMA Tx engine ignored it. Access restriction applies. Self-set to 1 on internal event. Setting 1 clears. Setting 0 has no effect.
19 FBE	Fatal Bus Error This bit indicates that a bus error occurred (as described in the EB field). When this bit is set, the corresponding DMA channel engine disables all bus accesses. Access restriction applies. Self-set to 1 on internal event. Setting 1 clears. Setting 0 has no effect.
20 ERI	Early Receive Interrupt This bit indicates that the DMA filled the first data buffer of the packet. The RI bit of this register automatically clears this bit. Access restriction applies. Self-set to 1 on internal event. Setting 1 clears. Setting 0 has no effect.

Table 1244. DMA_CH*n*_STATUS field descriptions (continued)

Field	Description
21 ETI	Early Transmit Interrupt This bit indicates that the packet to be transmitted is fully transferred to the MTL Tx FIFO. Access restriction applies. Self-set to 1 on internal event. Setting 1 clears. Setting 0 has no effect.
22 RWT	Receive Watchdog Timeout This bit is asserted when a packet with length greater than 2,048 bytes (10,240 bytes when Jumbo Packet mode is enabled) is received.
23 RPS	Receive Process Stopped This bit is asserted when the Rx process enters the Stopped state. Access restriction applies. Self-set to 1 on internal event. Setting 1 clears. Setting 0 has no effect.
24 RBU	Receive Buffer Unavailable This bit indicates that the application owns the next descriptor in the Receive list, and the DMA cannot acquire it. The Rx process is suspended. To resume processing Rx descriptors, the application should change the ownership of the descriptor and issue a Receive Poll Demand command. If this command is not issued, the Rx process resumes when the next recognized incoming packet is received. In ring mode, the application should advance the Receive Descriptor Tail Pointer register of a channel. This bit is set only when the DMA owns the previous Rx descriptor. Access restriction applies. Self-set to 1 on internal event. Setting 1 clears. Setting 0 has no effect.
25 RI	Receive Interrupt This bit indicates that the packet reception is complete. When packet reception is complete, Bit 0 of RDES1 is reset in the last descriptor, and the specific packet status information is updated in the descriptor. The reception remains in the Running state. Access restriction applies. Self-set to 1 on internal event. Setting 1 clears. Setting 0 has no effect.
29 TBU	Transmit Buffer Unavailable This bit indicates that the application owns the next descriptor in the Transmit list, and the DMA cannot acquire it. Transmission is suspended. The TPS0 field of the DMA_Debug_Status0 register explains the Transmit Process state transitions. To resume processing the Transmit descriptors, the application should do the following: 1. Change the ownership of the descriptor by setting Bit 0 of TDES0. 2. Issue a Transmit Poll Demand command. For ring mode, the application should advance the Transmit Descriptor Tail Pointer register of a channel. Access restriction applies. Self-set to 1 on internal event. Setting 1 clears. Setting 0 has no effect.

Table 1244. DMA_CH*n*_STATUS field descriptions (continued)

Field	Description
30 TPS	Transmit Process Stopped This bit is set when the transmission is stopped. Access restriction applies. Self-set to 1 on internal event. Setting 1 clears. Setting 0 has no effect.
31 TI	Transmit Interrupt This bit indicates that the packet transmission is complete. When transmission is complete, Bit 0 of TDES3 is reset in the last descriptor, and the specific packet status information is updated in the descriptor. Access restriction applies. Self-set to 1 on internal event. Setting 1 clears. Setting 0 has no effect.

56.2.214 DMA Channel *n* Miss Frame Counter register (DMA_CH*n*_MISS_FRAME_CNT)

This register has the number of packet counter that got dropped by the DMA either due to Bus Error or due to programing RPF field in DMA_CH0_Rx_Control register.

Offset 0x116C + *n**0x80 (n=0 to 1) Access: User Read

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	MFCO	0	0	0	0	MFC										
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1193. DMA Channel *n* Miss Frame Counter register (DMA_CH*n*_MISS_FRAME_CNT)

Table 1245. DMA_CH*n*_MISS_FRAME_CNT field descriptions

Field	Description
16 MFCO	Overflow status of the MFC Counter When this bit is set then the MFC counter does not get incremented further. The bit gets cleared when this register is read. Access restriction applies. Clears on read. Self-set to 1 on internal event.
21:31 MFC	This counter indicates the number of packet counters that are dropped by the DMA either because of bus error or because of programing RPF field in DMA_CH0_Rx_Control register. The counter gets cleared when this register is read. Access restriction applies. Clears on read. Self-set to 1 on internal event.

56.3 Functional Description

56.3.1 AHB Master Interface

The DMA controller interfaces with the application through the AMBA AHB interface. The AHB master interface converts the internal DMA request cycles into AHB cycles. The AHB master interface has the following features:

- The AHB master interface is AMBA 2.0-compliant AHB master with no restrictions
- The AHB master interface supports the following burst length modes: fixed, unspecified, and mixed. These modes are explained in *Section 56.3.1.1: AHB Burst Length Modes*
- The AHB master interface can handle the AHB SPLIT, RETRY, and ERROR conditions. Any ERROR response halts all further transactions for that DMA and indicates the error as fatal through CSR and interrupt. The application can choose to give a hard or soft reset to the module to restart the operation
- The AHB master interface can handle the AHB 1 K boundary breaking
- The AHB master interface can handle all 64-bit data transfers except for Descriptor Status Write accesses (which may be 32-bit). In any burst data transfer, the address bus value is always aligned to the data bus width and need not be aligned to the beat size
- The AHB master interface can align all AHB burst transfers to an address value. For more information, refer to *Section 56.3.1.2: Aligning AHB Burst Transfers to an Address Value*
- The DMA requests an AHB Burst Read transfer only when it can completely accept the received burst data. The data read from the AHB is always pushed into the DMA without any delay or Busy cycles. However, when multiple channels are selected, there may be some delay and busy cycles may be inserted because of the shared-memory architecture for the channels
- The DMA requests an AHB Burst Write transfer only when it has the sufficient data to completely transfer the burst. The AHB interface always assumes that it has the data available to push into the AHB bus. However, the DMA can prematurely indicate end-of-valid data (because of the transfer of EOP of an Ethernet packet) during the burst
- In Fixed Burst Length mode, the AHB master interface continues the burst with dummy data until the specified length is completed. In INCR mode, it ends the burst transfer prematurely

56.3.1.1 AHB Burst Length Modes

The AHB master interface supports the following burst length modes:

- Fixed Burst Length: In fixed burst length mode, the AHB master always initiates a burst with SINGLE, INCR4, INCR8, or INCR16 type. However, when such a burst is responded with SPLIT, RETRY, or Early Burst Termination (EBT), by default, the AHB master again initiates the pending transfers of the burst with INCR or SINGLE burst length type. When only one beat is remaining, the AHB master performs a SINGLE transfer. If more than one beat is remaining, the AHB master performs an INCR transfer. The AHB master terminates such INCR bursts when remaining beats of the original requested fixed burst are transferred.
  To ensure that the AHB master always performs the INCRx and SINGLE transfers even when a part of the burst is initiated again after a SPLIT, RETRY, or EBT response, use the RIB bit of the register. However, when you enable this mode, the pending burst may

further get split into multiple transfers as per the following rules before any new transfer is initiated:

  – If the DMA requests a burst transfer that is not equal to INCR4, INCR8, or INCR16, the AHB application interface splits the transfer into multiple burst transactions. For example, if the DMA requests a 15-beat burst transfer, the AHB interface splits it into multiple transfers of INCR8 and INCR4, and three SINGLE transactions
  – If the burst length is more than 16, the AHB master interface splits it into multiple burst transfers. For example, if the burst length is 32 and the DMA starts a 32-beat transfer, the AHB master splits the burst into two INCR16

- Unspecified Burst Length: In unspecified burst mode, the AHB master always initiates a transfer with INCR and completes the DMA requested burst in one transfer
- Mixed Burst Mode: In mixed burst mode, the AHB master always initiates the bursts with fixed-size (INCRx) when the DMA requests transfers of size less than or equal to 16 beats. When the DMA requests bursts of length more than 16, the AHB master initiates such transfers with INCR and completes the request in one transfer

To ensure that the pending burst of a transfer is rebuilt only with INCRx transfers when an INCRx transfer is interrupted by RETRY, SPLIT, or EBT response, use the RB bit of the DMA_SysBus_Mode register. However, if an INCR burst initiated in this mode is interrupted, the AHB master rebuilds the burst only with INCR transfers.

You can choose the burst modes by programming the FB bit in the DMA_SysBus_Mode register.

Note: The AHB master does not generate the WRAP bursts.

The AHB master does not generate the locked or protected transfers. Therefore, the Ethernet IP does not have the HLOCK or HPROT output signals.

56.3.1.2 Aligning AHB Burst Transfers to an Address Value

The AHB master interface can align all AHB burst transfers to an address value. You can enable this by using the AAL bit in the DMA_SysBus_Mode register. If Bit 31 and Bit 19 of DMA_SysBus_Mode register are set to 1, the AHB interface and the DMA ensure that all initiated beats are aligned to the address, completing the packet transfer in the minimum number of required beats.

The AHB interface and the DMA split unaligned address beats for a 64-bit data bus and initiate INCR16 beats only when the seven least-significant bits of the address have a zero value.

56.3.2 DMA Controller

The DMA has independent Transmit (Tx) and Receive (Rx) engines, and a CSR space. The Tx engine transfers data from the system memory to the device port (MTL), whereas the Rx engine transfers data from the device port to the system memory. The controller uses descriptors to efficiently move data from source to destination with minimal application CPU intervention. The DMA is designed for packet-oriented data transfers such as packets in Ethernet. The controller can be programed to interrupt the application CPU for situations such as Packet Transmit and Receive Transfer completion, and other normal or error conditions.

The DMA and the application communicate through the following two data structures:

- Control and Status registers (CSR)
- Descriptor lists and data buffers

The DMA transfers the data packets received by the MAC to the Rx Buffer in system memory and Tx data packets from the Tx Buffer in the system memory. The descriptors that reside in the system memory contain the pointers to these buffers.

The DMA supports up to 2 Tx and 2 Rx Descriptor lists (or DMA channels). The base address of each list is written to the respective Tx Descriptor List Address register and Rx Descriptor List Address register. The descriptor list is forward linked and the next descriptor is always considered at a fixed offset to the current one. You can control the offset by the DSL field of DMA_CH*n*_Control register. The number of descriptors in the list is programed in the respective Tx (or Rx) Descriptor Ring Length register. Once the DMA processes the last descriptor in the list, it automatically jumps back to the descriptor in the List Address register to create a descriptor ring.

The descriptor lists reside in the physical memory address space of the application. Each descriptor can point to a maximum of two buffers. This enables two buffers to be used, physically addressed, rather than contiguous buffers in memory.

A data buffer resides in the application physical memory space and consists of an entire packet or part of a packet but cannot exceed a single packet. Buffers contain only data. Buffer status is maintained in the descriptor. Data chaining refers to packets that span multiple data buffers. However, a single descriptor cannot span multiple packets. The DMA skips to the data buffer of next packet when EOP is detected.

The Ethernet IP supports the ring structure for the DMA descriptor. For more information on the descriptor, refer to *Section 56.3: Functional Description* that describes the descriptor structure and how the DMA accesses the descriptors.

56.3.2.1 Application Bus Burst Access

When configured through Bit 31 of the DMA_SysBus_Mode register, the DMA attempts to execute the fixed-length burst transfers on the AHB master interface. The PBL field of Transmit Control and Receive Control registers of respective DMA channel indicates and limits the maximum burst length. The Rx and Tx descriptors are always accessed in the maximum possible (limited by PBL or 16 * 8/bus width) burst length for 16 bytes to be read.

The Tx DMA initiates a data transfer only when sufficient space is available in the MTL Tx Queue to accommodate either of the following:

- Bytes corresponding to the configured burst (PBL * bus_width/8)
- Remaining bytes in Tx Buffer without EOP
- Number of bytes till EOP

The Rx DMA initiates a data transfer in the following conditions:

- Sufficient data is available in MTL Rx Queue to accommodate the configured burst
- EOP (when it is less than the configured burst length) is detected in the Rx Queue

The DMA indicates the start address and the number of transfers required to the AHB master interface. When the AHB Interface is configured for fixed-length burst, it transfers the data by using the best combination of INCR4, INCR8, or INCR16 and SINGLE transactions. If EOP is reached before the fixed-burst ends on the AHB interface, dummy transfers are performed in-order to complete the fixed-burst. Otherwise, (Bit 31 of DMA_SysBus_Mode

register is reset), the DMA transfers the data using INCR (undefined length) and SINGLE transactions.

When the AHB interface is configured for address-aligned beats, both DMA engines ensure that the first burst transfer initiated by the AHB is less than or equal to the size of the configured PBL. Therefore, all subsequent beats start at an address that is aligned to the configured PBL. The DMA can only align the address for beats up to size 16 (for PBL > 16) for AHB interface because it does not support more than INCR16.

56.3.2.1.1 Application Data Buffer Alignment

The Tx and Rx data buffers do not have any restrictions on start address alignment. For example, in systems with 32-bit memory, the start address for buffers can be aligned to any of the four bytes. However, the DMA always initiates write transfers with address aligned to the bus width and dummy data (old data) in the invalid byte lanes. This typically happens during the transfer of the beginning or end of an Ethernet packet. The software driver should discard the dummy bytes based on the start address of the buffer and size of the packet.

Table 1246. Application Data Buffer Alignment Examples

Data Buffer	Examples
Buffer Read	If the Tx buffer address is 32'h00000FF2 (for 32-bit data bus), and 15 bytes need to be transferred, the DMA reads five full words from address 32'h00000FF0, but when transferring data to the MTL Tx queue, the extra bytes (the first two bytes) are dropped or ignored. Similarly, the last 3 bytes of the last transfer are also ignored. The DMA always ensures that it transfers a full 32-bit data to the MTL Tx queue, unless it is the end of packet.
Buffer Write	If the Rx buffer address is 32'h0000FF2 (for 64-bit data bus) and 16 bytes of a received packet need to be transferred, the DMA writes 3 full words from address 32'h00000FF0. However, the first 2 bytes of the first transfer and the last 6 bytes of the third transfer have dummy data. The DMA considers the offset address only if it is the first Rx buffer of the packet. The DMA ignores the offset address and performs full word writes for the middle and the last Rx buffer of the packet.

56.3.2.1.2 Buffer Size Calculations

The DMA does not update the size fields in the Tx and Rx descriptors. The DMA updates only the status fields (RDES and TDES) of the descriptors. The driver has to perform the size calculations.

The Tx DMA transfers the exact number of bytes (indicated by buffer size field of TDES2) towards the MAC. If a descriptor is marked as first (FS bit of TDES2 is set), the DMA marks the first transfer from the buffer as SOP. If a descriptor is marked as last (LS bit of TDES2), the DMA marks the last transfer from that data buffer as EOP to the MTL.

The Rx DMA transfers data to a buffer until the buffer is full or the end of packet is received from the MTL. When the FS bit of a descriptor is set, the amount of valid data in a buffer is accurately indicated by the buffer size field (programed in DMA Channel Receive Control register) minus the data buffer pointer offset. The offset is zero when the data buffer pointer is aligned to the data bus width. If a descriptor is marked as last, the buffer may not be full (as indicated by the buffer size in Bits[17:30] of Receive Control register). To compute the amount of valid data in this final buffer, the driver must read the packet length (FL bits of RDES3[17:31]) and subtract the sum of the buffer sizes of the preceding buffers in this packet. The Rx DMA always transfers the start of next packet with a new descriptor.

Note: Even when the start address of a Rx buffer is not aligned to the data width of system bus, the system should allocate a Rx buffer of a size aligned to the system bus width. For example, if the system allocates a Rx buffer of 1024 bytes (1 KB) starting from address 0x1000, the software can program the buffer start address in the Rx descriptor to have a 0x1002 offset. The Rx DMA writes the packet to this buffer with dummy data in the first two locations (0x1000 and 0x1001). The actual packet is written from location 0x1002. Therefore, the actual useful space in this buffer is 1022 bytes, even though the buffer size is programed as 1024 bytes, because of the start address offset.

56.3.2.1.3 DMA Arbiter

The arbiter inside the DMA module performs the arbitration between the Tx and Rx channel accesses to the AHB master interface. The following two types of arbitrations are supported:

- Round-Robin Arbitration: When Bit 30 of the DMA_Mode register is reset and both Tx and Rx DMAs are simultaneously requesting for access, the arbiter allocates the data bus in ratio set by Bits[17:19] of DMA_Mode register
- Fixed-Priority Arbitration: When Bit 30 of the DMA_Mode register is set, the Rx DMA always gets priority over the Tx DMA for data access by default. When Bit 20 of DMA_Mode register is also set, the Tx DMA gets priority over the Rx DMA

56.3.2.2 DMA Transmission

56.3.2.2.1 TX DMA Operation: Default (Non-OSP) Mode

The TX DMA engine in default mode proceeds as follows:

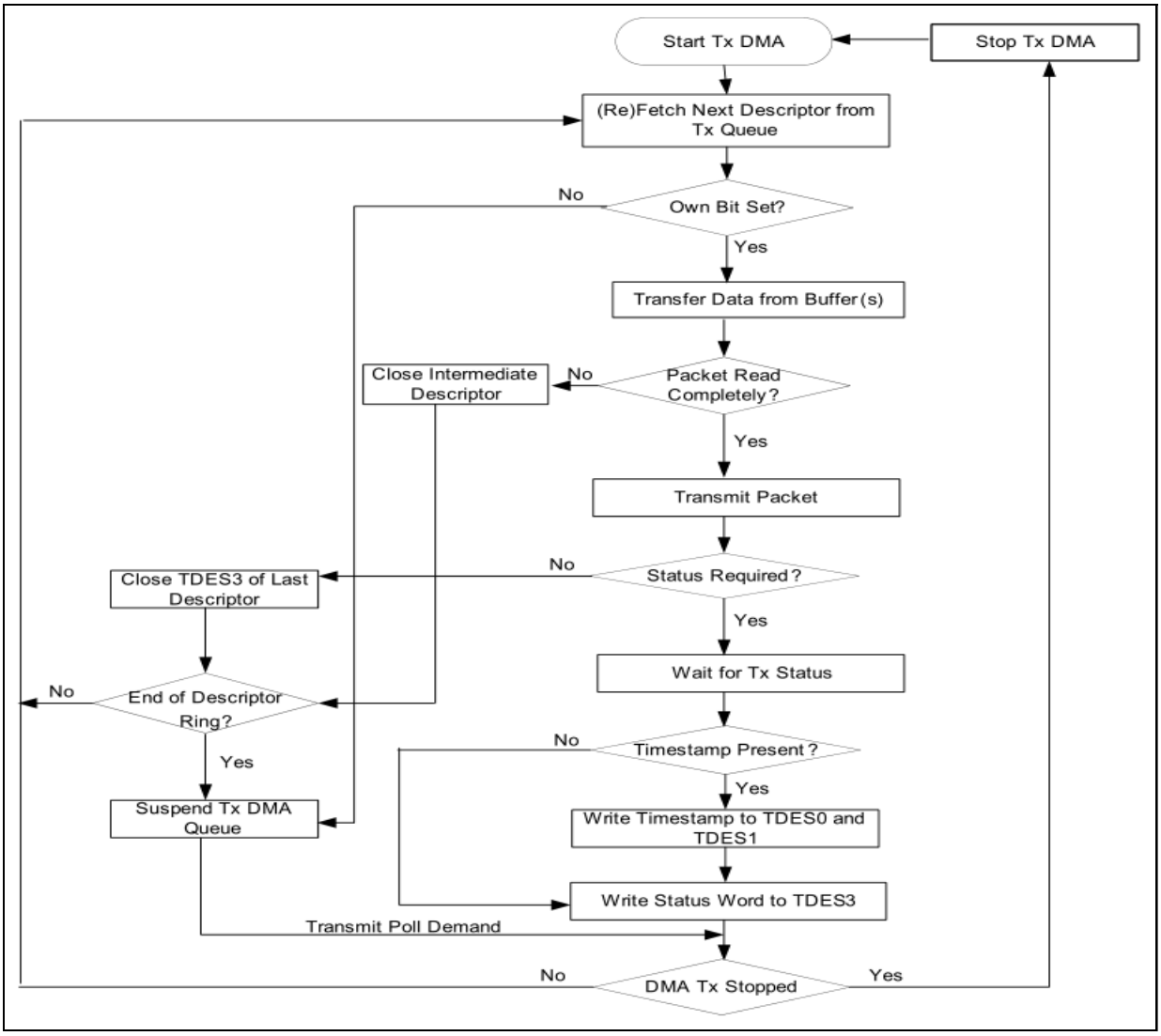
1. The application sets up the Transmit descriptor (TDES0–TDES3) and sets the Own bit (TDES0[0]) after setting up the corresponding data buffer(s) with Ethernet Packet data
2. The application advances the Descriptor Tail pointer offset value of the Transmit Channel
3. While in the Run state, the DMA runs an Arbitration cycle to select the next Tx DMA channel from which the packets requiring transmission should be processed
4. The DMA fetches the descriptor from the application memory.
5. If the DMA detects one of the following conditions, the transmission from that channel is suspended and Bit 29 and Bit 15 of Status Register of corresponding DMA channel are set and the Tx Engine proceeds to Step *11*:
   a) The descriptor is flagged as owned by the application (TDES3 [0] = 1'b0)
   b) The Descriptor Tail pointer is equal to the Current Descriptor pointer in Ring Descriptor list Mode
   c) An error condition occurs
6. If the acquired descriptor is flagged as owned by the DMA (TDES3[0] = 1), the DMA decodes the Transmit Data Buffer address from the acquired descriptor
7. The DMA fetches the Transmit data from the system memory and transfers the data to the MTL for transmission
8. If an Ethernet packet is stored over data buffers in multiple descriptors, the DMA closes the intermediate descriptor and fetches the next descriptor. Step *3* through Step *7* are repeated until the end-of-Ethernet-packet data is transferred to the MTL
9. When packet transmission is complete, if IEEE 1588 Timestamp feature was enabled for the packet (as indicated in the Tx status), the timestamp value obtained from MTL is written to the Tx descriptor (TDES0 and TDES1) that contains the EOP buffer. The

status information is written to this Tx descriptor (TDES3). The application now owns this descriptor because the Own bit is cleared during this step.
If timestamp feature is not enabled for this packet, the DMA does not alter the contents of TDES0 and TDES1

10. Bit 31 of Status Register of corresponding channel is set after completing transmission of a packet that has Interrupt on Completion (TDES2[0]) set in its Last Descriptor. The DMA engine returns to Step *3*
11. In the Suspend state, the DMA tries to acquire the descriptor again (and thereby return to Step *3*). A poll demand command is triggered by writing any value to the DMA_CH0_TxDesc_Tail_Pointer when it receives a Transmit Poll demand and the Underflow Interrupt Status bit is cleared. If the application stopped the DMA by clearing Bit 31 of Transmit Control Register of corresponding DMA channel, the DMA enters the Stop state

The Tx DMA transmission flow in default mode is shown in *Figure 1194*.

Figure 1194. TX DMA Operation in Default Mode



56.3.2.2.2 TX DMA Operation: OSP Mode

In the Run state, if Bit 27 is set in the Transmit Control Register of corresponding DMA channel, the Transmit process can simultaneously acquire two packets without closing the Status descriptor of the first packet. As the Transmit process finishes transferring the first packet, it immediately polls the Transmit Descriptor list for the second packet. If the second packet is valid, the Transmit process transfers this packet before writing the status information of the first packet.

In OSP mode, the Run state Tx DMA operates in the following sequence:

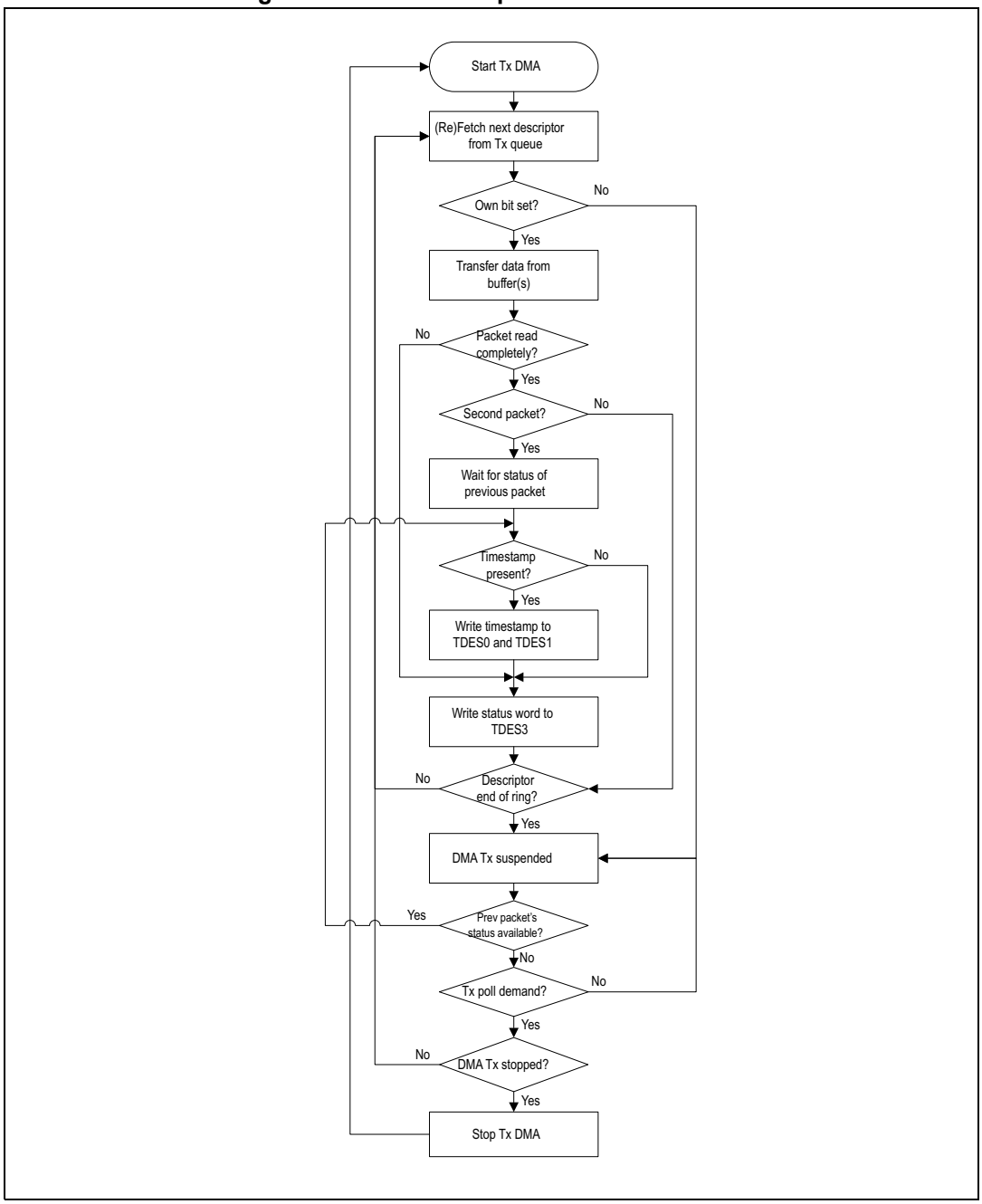
1. The DMA operates as described in Step *1* – Step *7* of the Tx DMA (default mode)
2. The DMA fetches the next descriptor without closing the last descriptor of previous packet
3. If the DMA owns the acquired descriptor, the DMA decodes the transmit buffer address in this descriptor. If the DMA does not own the descriptor, the DMA goes into Suspend mode and skips to Step *7*
4. The DMA fetches the Transmit packet from the system memory and transfers the packet to the MTL until the EOP data is transferred, closing the intermediate descriptors if this packet is split across multiple descriptors
5. The DMA waits for the packet transmission status and timestamp of previous packet. When the status is available, the DMA writes the timestamp to TDES0 and TDES1 if such timestamp was captured (as indicated by a status bit). The DMA writes the status, with a cleared Own bit, to the corresponding TDES3, thus closing the descriptor.
   If Timestamp feature is not enabled for the previous packet, the DMA does not alter the contents of TDES0 and TDES1
6. The Transmit interrupt is set (if enabled). The DMA fetches the next descriptor and proceeds to Step *3* (when Status is normal). If the previous transmission status shows an underflow error, the DMA goes into Suspend mode (Step *7*)
7. In Suspend mode, if a pending status and timestamp are received from the MTL, the DMA does the following:
   a) Writes the timestamp (if enabled for the current packet) to TDES0 and TDES1
   b) Writes the status to the corresponding TDES3
   c) Sets relevant interrupts and returns to Suspend mode

   If no status is pending and the application stopped the DMA by clearing Bit 31 of Transmit Control Register of corresponding DMA channel, the DMA enters the Stop state
8. The DMA can exit Suspend mode and enter the Run state (goes to Step *1* or Step *2* depending on pending status) only after receiving a Transmit Poll demand in Transmit Descriptor Tail Pointer register of corresponding channel

Note: The DMA fetches the next descriptor before closing the current descriptor. Therefore, the descriptor ring length must be more than two. It is recommended to have a minimum descriptor length of four.

The basic flow is described in *Figure 1195*.

Figure 1195. Tx DMA Operation in OSP Mode



56.3.2.2.3 Transmit Packet Processing

The Tx DMA expects that the data buffers contain complete Ethernet packets, excluding preamble, pad bytes, and FCS fields. The DA, SA, and Type/Length fields contain valid data. If the Tx Descriptor indicates that the MAC must disable CRC or PAD insertion, the buffer must have complete Ethernet packets (excluding preamble), including the CRC bytes.

Packets can be data-chained and can span several buffers. Packets must be delimited by the First Descriptor (TDES3[2]) and the Last Descriptor (TDES3[3]). As transmission starts, the First Descriptor must have TDES3[2] set. When this occurs, the packet data is

transferred from the application buffer to the MTL Tx Queue. Concurrently, if the current packet has the Last Descriptor (TDES3[3]) clear, the Tx Process attempts to acquire the Next Descriptor. The Tx Process expects this descriptor to have TDES3[2] clear. If TDES3[3] is clear, it indicates an intermediary buffer. If TDES3[3] is set, it indicates the last buffer of the packet.

After the last buffer of the packet has been transmitted, the DMA writes back the final status information to the Transmit Descriptor 3 (TDES3) word of the descriptor that has the Last Descriptor Bit set in Transmit Descriptor 3 (TDES3[3]). At this time, if Interrupt on Completion (TDES2[0]) is set, Bit 31 of Status Register of corresponding DMA channel is set, the Next Descriptor is fetched, and the process repeats. The actual packet transmission begins after either of the following:

- The MTL Tx Queue has reached a programmable Transmit threshold (Bits[25:27] of Transmit Operation Mode register of corresponding MTL Transmit Queue)
- A full packet is contained in the FIFO

You can also use the store-and-forward mode (Bit 30 of MTL Transmit Operation Mode Register of a queue). In this mode, descriptors are released (Own bit TDES0[0] clears) when the DMA finishes transferring the packet.

Note: To ensure proper transmission of a packet and the next packet, you must specify a non-zero buffer size for the Transmit descriptor that has the Last Descriptor (TDES3[3]) set.

56.3.2.2.4 Transmit Polling Suspended

Transmit polling can be suspended by any of the following conditions:

- The DMA detects a descriptor owned by the application (TDES3[0]=0).
  To resume, the driver must give descriptor ownership to the DMA and then issue a Poll Demand command by writing the Tail Pointer register. If the DMA goes into SUSPEND state because of this condition, Bit 16 and Bit 29 of Status Register of corresponding DMA channel are set
- A packet transmission is aborted when a Transmit error is detected because of underflow.
  The appropriate Transmit Descriptor 3 (TDES3) bit is set. When this condition occurs, the following bits are set and the information is written to Transmit Descriptor 0, causing the suspension:
  - Bit 17 of Status Register of corresponding DMA channel
  - Transmit Underflow bit of corresponding queue in MTL_Interrupt_Status
- The DMA detects that the Tail Pointer is equal to the Current descriptor closed by it.
  To resume, the software driver must modify the Tail Pointer register

In all conditions, the position in the Transmit List is retained. The retained position is that of the descriptor following the Last Descriptor closed by the DMA. The driver must explicitly issue a Transmit Poll Demand command after rectifying the suspension cause.

56.3.2.2.5 Transmit Channel Arbitration

An arbiter provides access to multiple DMAs trying to access the Bus Interface Unit (BIU). *Figure 1196* shows the DMA Tx Channel Arbitration Process.

When there is any request in the Tx queue, the DMA arbiter checks the type of the request: packet buffer fetch or descriptor fetch request. The descriptor fetch requests have higher priority than the buffer requests. Therefore, when there is a descriptor fetch request, the

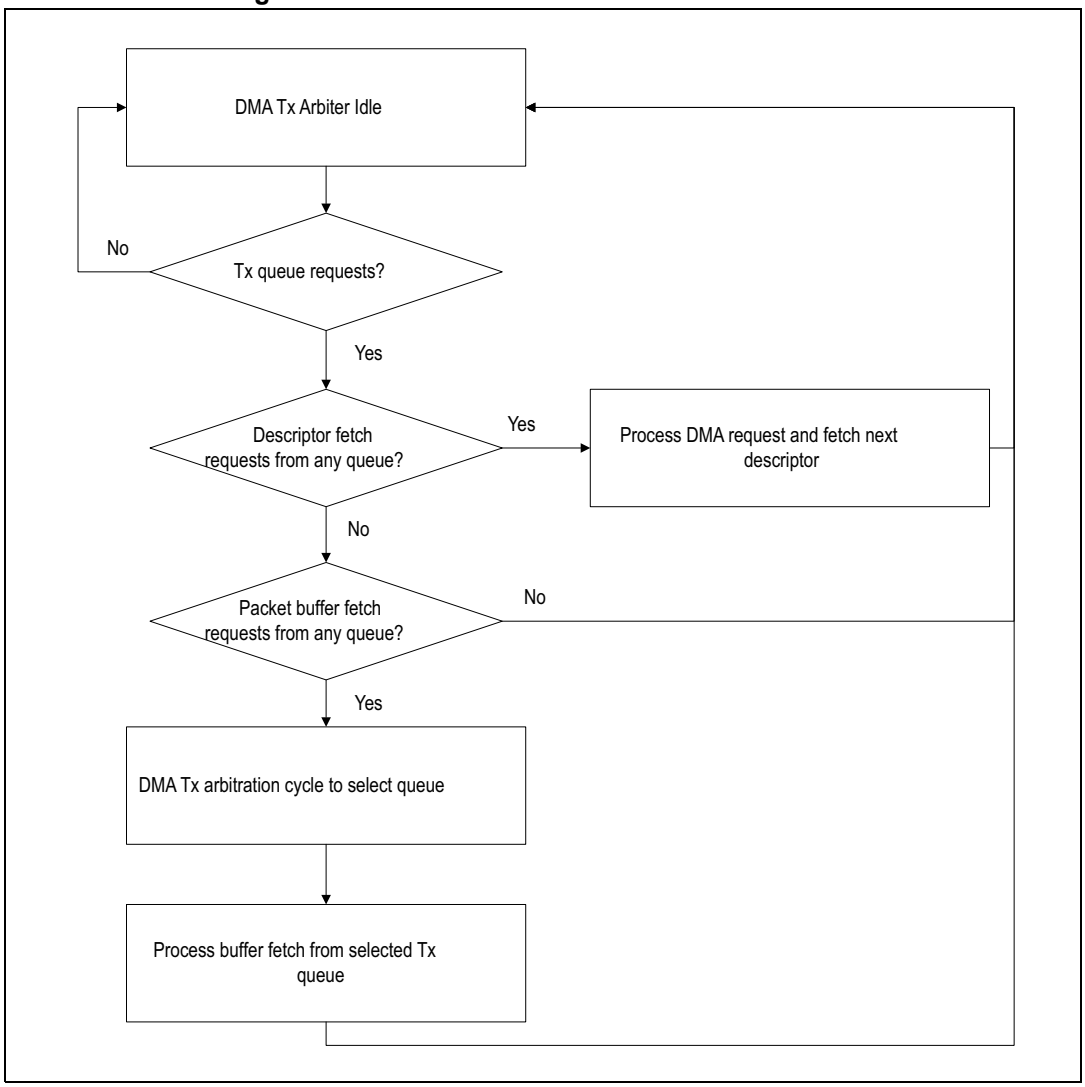
DMA arbiter acknowledges the DMA channel that is requesting for a descriptor fetch. If there is no descriptor fetch request, the arbiter looks for packet buffer fetch requests.

The DMA arbiter acknowledges the descriptor fetch request of one DMA channel at a time. Descriptor fetch requests are granted using a fixed priority with the higher channel having higher priority (Channel 1 having priority over Channel 0). For packet buffer fetches, the DMA arbiter uses the programed channel weight and priority to decide which channel to acknowledge. The DMA arbiter performs a burst-by-burst arbitration based on one of the following algorithms:

- Weighted Strict Priority (WSP): In WSP arbitration mode, the arbiter first processes Channel 1 (or the last enabled channel) and then Channel 0. If any channel does not have a frame to transmit, the weight of that channel gets reassigned to Channel1 (or last enabled channel). If Channel 1 has no frames to transmit, the remaining weight is assigned to Channel 0
- Weighted Round Robin (WRR): In WRR arbitration mode, the arbiter first selects the channel with the highest weight programed, and then the channel with next highest weight, and so on. If any channel does not have a frame to transmit, the weight of that channel gets equally distributed to all channels that have frames to transmit
- Fixed priority (FP): In Fixed priority mode, Channel 0 has the lowest priority and the last selected channel has the highest priority. The weight programed in the Transmit Control register of a channel is ignored

In WSP or WRR arbitration, the channel weight corresponds to the number of DMA burst transfers for which the DMA arbiter grants the bus to a channel. When a channel completes all the DMA burst transfers, the arbiter grants the bus to the next channel.

Figure 1196. DMA Tx Channel Arbitration Process



56.3.2.3 DMA Reception

In the Receive path, the DMA reads a packet from the MTL receive queue and writes it to the packet data buffers of the corresponding DMA channel. The ARI data at the start of the frame indicates the channel number to which the current frame must be written. If only one DMA channel is selected, this information is not provided.

Figure 1197 shows the reception sequence for Rx DMA engine. The following list describes this sequence:

1. The application sets up the Rx descriptors (RDES0-RDES3) and the Own bit (RDES3[0]).
   The application must set the correct value in the Receive Descriptor Tail Pointer register of corresponding DMA channel
2. When Bit 31 of Receive Control register of corresponding DMA channel is set, the DMA enters the Run state.
   The DMA looks for free descriptors based on the Rx Current Descriptor and Descriptor

Tail Pointer register values. If there are no free descriptors, the DMA Channel enters the suspend state and goes to Step *11*

3. The DMA fetches the next available descriptor in the ring and decodes the receive data buffer address from acquired descriptors
4. If IEEE 1588 timestamping is enabled and the timestamp is available for the previous packet, the DMA writes the timestamp (if available) to the RDES0 and RDES1 of current descriptor and sets the RDES3[CTXT] field
5. The DMA processes the incoming packets and places these in the data buffers of acquired descriptor
6. If the current packet transfer is not complete, the DMA closes the current descriptor as intermediate and goes to Step *10*
7. The DMA takes the status of the Receive frame from the MTL and writes the status word to current descriptor with the Own bit cleared and the Last Descriptor bit set
8. The DMA writes the Frame Length to RDES3 and VLAN Tag to RDES0. The DMA also writes the MAC control frame opcode, OAM control frame code, and extended status information (if available) to RDES1 of the last descriptor
9. If IEEE 1588 Timestamp feature is enabled, the DMA stores the timestamp (if available). The DMA writes the context descriptor after the last descriptor for the current packet (in the next available descriptor)
10. If more descriptors are available in the Rx DMA Descriptor Ring, go to step *3*; otherwise, go to the Suspend state (Step *11*)
11. The Receive DMA exits the Suspend state when a Receive Poll demand is given and the application advances the Receive Tail Pointer register of a channel

The engine proceeds to Step *2* and re-fetches the next descriptor.

Figure 1197. RX DMA Operation



56.3.2.3.1 Receive Descriptor Acquisition

The Receive Engine always attempts to acquire an extra descriptor in anticipation of an incoming packet. Descriptor acquisition is attempted if any of the following conditions is satisfied:

- Bit 31 of Receive Control Register of corresponding DMA channel is set immediately after being placed in the Run state
- The Descriptor Tail pointer register value is ahead of the Current Descriptor acquired by the Rx DMA
- The controller has completed packet reception, but the current Receive Descriptor is not yet closed
- A Receive poll demand is issued (update of the Tail Pointer register)

56.3.2.3.2 Receive Packet Processing

The sequence for processing a Receive packet is as follows:

1. The MAC transfers the received packets to the MTL memory only if the packet passes the address filter. If the packet fails the address filtering, it is dropped in the MAC block (unless Bit 0 of MAC_Packet_Filter register is set)
2. If packet size is greater than or equal to configurable threshold bytes set for Rx Queue of MTL, or when the complete packet is written to the queue in the store-and-forward mode, the MTL block requests the DMA block to begin transferring the packet data to the Receive Buffer pointed by the current descriptor.
   Packets smaller than 64 bytes, because of collision or premature termination, are removed from the MTL Receive Queue
3. When the DMA application Interface (AHB) becomes ready, it transfers the data and sets the following:
   a) If the packet fits in a single descriptor, the DMA sets both Last Descriptor (RDES3[3]) and First Descriptor (RDES3[2])
   b) If the packets fits into more than one descriptor, the DMA sets the First Descriptor (RDES3[2]) to delimit the packet
4. The DMA releases the descriptors by resetting the Own (RDES3[0]) bit to 0, either because the Receive buffer filled up or the last segment of the packet is transferred to the Receive buffer. The received packets status is updated in the last descriptor
5. If Interrupt Enabled on Completion (RDES3[1]) bit is set in any of the Descriptors between the First and Last Descriptor of the Packet and Bit 25 of Interrupt Enable Register of corresponding DMA channel is set, the DMA sets Bit 25 of Status register of corresponding DMA channel.
   The same process repeats unless the DMA encounters a descriptor flagged as being owned by the application or when there are no more descriptors in the ring. When the DMA finds a descriptor owned by the application and if Bit 24 of Interrupt Enable Register of corresponding DMA channel is set, the Receive Process sets Bit 24 of Status register of corresponding DMA channel and then enters the Suspend state. The position in the receive list is retained

56.3.2.4 DMA Error Response

For any data transfer initiated by a DMA channel, if the slave replies with an error response, the DMA stops all operations and updates the error bits and the Fatal Bus Error bit in the Status Register of corresponding DMA channel. The application can either perform a reset to Ethernet IP or re-initialize the DMA descriptor list and start again. The rest of the DMA

channels are not affected by such errors. In AHB the DMA receives the error response through hresp_i signal.

56.3.3 MAC Transaction Layer (MTL)

The MAC Transaction Layer (MTL) provides the FIFO memory Interface to buffer and regulate the packets between the application system memory and the MAC. It also enables the data to be transferred between the application clock and MAC clock domains. The MTL layer has two data paths: Transmit path and Receive Path. The data path for both directions is 64-bit wide.

56.3.3.1 MTL Transmit Path

The DMA pushes the Ethernet packets read from the application or system memory into the corresponding queue. The packet is then popped out and transferred to the MAC when the queue threshold is reached (threshold mode) or complete packet is in the queue (store-and-forward mode). When EOP is transferred, the status of the transmission is taken from the MAC and transferred back to the application. The fill level of the queue is indicated to the internal DMA (using PBL and watermark) so that it can initiate a data fetch in required bursts from the application or system memory. The following two modes of operation trigger reading of the data towards the MAC:

- Threshold mode: In Threshold (or cut-through) mode, as soon as the number of bytes in the Queue cross the configured threshold level (or when the end of packet is written before the threshold is crossed), the data is ready to be popped out and forwarded to the MAC. The threshold level is configured by using the TTC bits of MTL_TxQ0_Operation_Mode register corresponding to an MTL queue
- Store-and-forward mode: In store-and-forward mode, the MTL pops out the packet towards the MAC only when one or more of the following conditions are true:
  - A complete packet is stored in the Queue
  - The Tx FIFO becomes almost full
  - The ATI watermark becomes low

The watermark becomes low when the requested Queue does not have space to accommodate the requested burst length on the ATI. Therefore, the MTL when operating in the store-and-forward mode allows packet transmission even if the packet length is bigger than the Tx Queue size.

The application can flush complete content of the Tx Queue by setting Bit 31 (FTQ) of Transmit Operation Mode register corresponding to an MTL queue. This bit is self-clearing and initializes the Queue pointers to the default state. If the FTQ bit is set during a packet transfer from the MTL to the MAC, the MTL stops further transfer because the queue is considered to be empty. Therefore, an underflow event occurs at the MAC transmitter.

56.3.3.1.1 Initialization Flow

Upon reset, the MTL is ready to manage the flow of data to and from the application or DMA, and MAC.

First you need to initialize the Queue size for each of the queues by programming the TQS bits of MTL_TxQ0_Operation_Mode register corresponding to a Transmit queue. You also need to initialize the MAC block. The internal DMA controllers must be individually enabled through their respective Cars.

56.3.3.1.2 Data Flow between the DMA and the MTL

This section describes the interaction process between the MTL and the DMA.

56.3.3.1.2.1 Single-Packet Transmit Operation

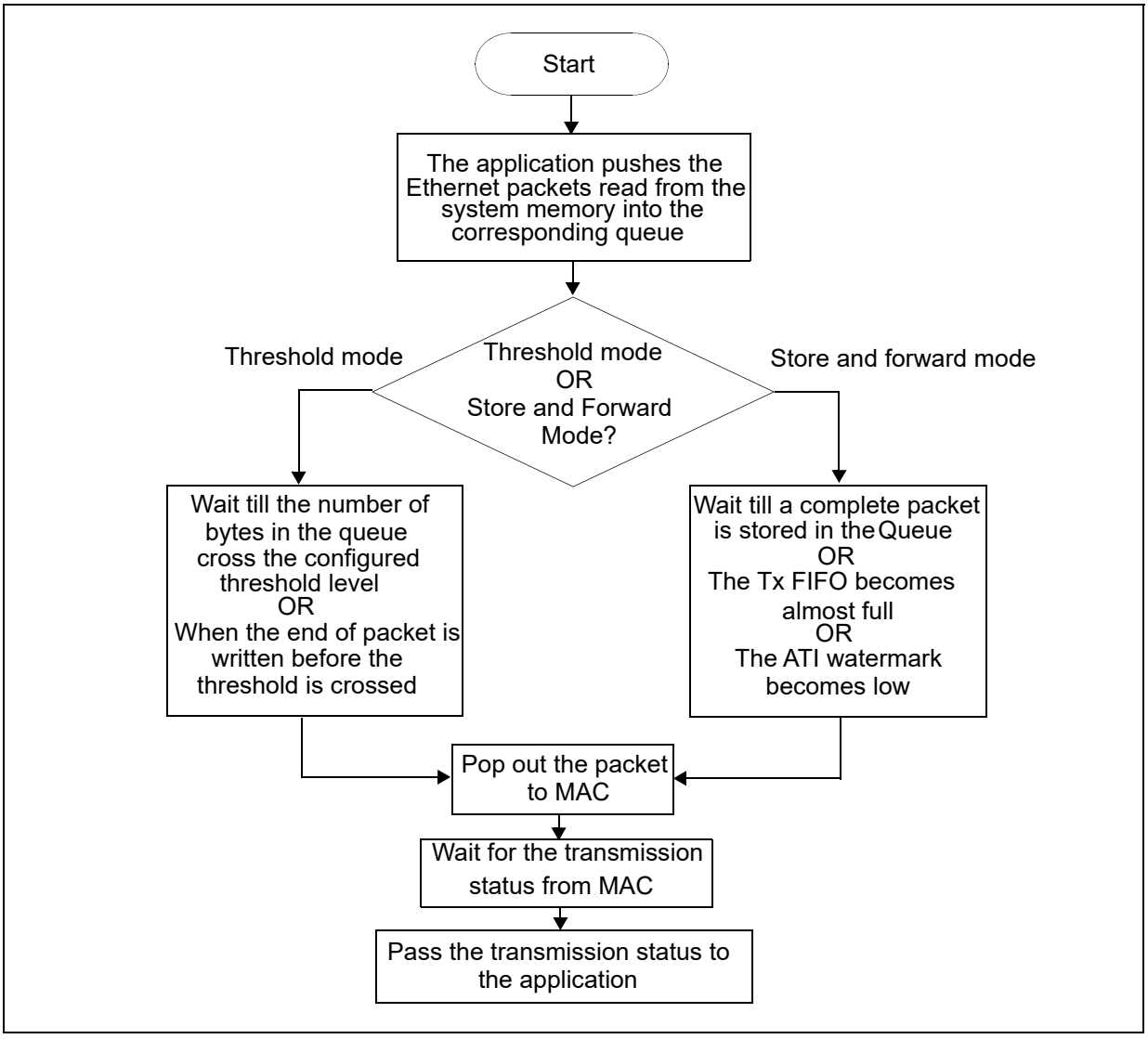
The following flow is valid when the core is operating in non-OSP mode, that is, when bit 27 (OSP) of the DMA_CH*n*_TX_Control register of the DMA channel is set to 0.

During a Transmit operation, the MTL block is a slave for the DMA controller. The general sequence of events for a Transmit operation is as follows:

1. If the system has data to be transferred, the DMA controller (if enabled) fetches the data from the application through the AHB master interface and starts forwarding it to the MTL. The DMA descriptor memory contains the information related to packet control which is used to drive the MTL Control Word. The DMA data buffer memory contains the packet data
2. The MTL pushes the data received from the DMA into the corresponding queue. This process continues until the EOP is transferred
3. When the threshold level is crossed or a full packet of data is received into the queue, the MTL reads the packet data and drives it to the MAC
4. The queue controller continues to transfer data from the queue until a complete packet is transferred to the MAC
5. When the packet transfer is complete, the MTL receives the status from the MAC and notifies the DMA controller

Figure 1198 illustrates the MTL single-packet transmit operation.

Figure 1198. MTL Single Packet Transmit Flow



Note: When Bit 30 (DTXSTS) of Section 56.2.163 register is set, the MTL does not provide any status to the DMA.

56.3.3.1.2.2 Transmit Operation: Two Packets in the Buffer

The following flow is valid when the core is operating in OSP mode, that is, when bit 27 (OSP) of the DMA_CH*n*_TX_Control register of the DMA channel is set to 1.

The flow is similar to the flow described in Single-Packet Transmit Operation. However after fetching a packet, the DMA instead of waiting for the status, it continues to fetch another packet if available in the system memory. So, MTL can receive the second packet while it is processing the first packet. This flow improves the performance because the DMA can process two packets back to back before waiting for the status of the first packet.

56.3.3.1.2.3 Relationship between the Number of TX DMA Channels and Tx Queues

The number of Tx DMA channels is always equal to the Tx queues. This is because the DMA is designed to arbitrate among multiple channels (to fetch the descriptor and packet data from the system memory) in terms of PBLs (Programmable Bursts Lengths). However, the Tx queues are packet-based storage memory. Therefore, one to one mapping is required between the DMA channels and Tx queues to maintain the packet-level coherency.

56.3.3.1.2.4 Retransmission during Collision

While a packet is being transferred from the MTL to the MAC, a collision event can occur on the MAC line interface in half-duplex mode. The MAC indicates a Retry attempt to the MTL by giving the status even before the EOP is transferred from the MTL. The MTL then enables the retransmission by popping out the packet again from the queue.

After more than 96 bytes are read out towards the MAC, the Queue Controller frees up that space and makes it available to the application or the DMA to push in more data. This means that the retransmission is not possible after this threshold is crossed or when the MAC indicates a late-collision event.

When a packet transmission is aborted because of underflow and a collision event immediately follows (initiating a retry), retry has higher priority than abort.

Note: Before issuing the Transmit Queue Flush command, make sure that the Tx Queue is not empty, that is, Bit 27 (TXQSTS) of Transmit Debug Register of corresponding queue is reset.

56.3.3.1.2.5 Transmit Status Word

At the end of transfer of the Ethernet packet to the MAC and after the MAC completes the transmission of the packet, the MTL provides the Transmit status on respective queues. The detailed description of the Transmit Status is the same as for Bits[17:31] of TDES3 normal descriptor in write-back format. If IEEE 1588 Timestamp feature is enabled, the MTL returns 64-bit timestamp of the packet, along with the Transmit status.

If Bit 30 (DTXSTS) of MTL_Operation_Mode register is set then it disables the dependency of the DMA to read the status words from the Status FIFO thus the DMA can push as many packets as possible into the Tx queues because the application or DMA is not required to read the status words from the Status FIFO.

56.3.3.2 Receive Path

The MTL Rx module receives the packets from the MAC and pushes them into the Rx Queue. The status (fill level) of the queue is indicated to the DMA when it crosses the configured Receive threshold (RTC bits[30:31] of MTL_RxQ0_Operation_Mode register of corresponding MTL queue), or the complete packet is received. The MTL also indicates the fill level of the queue so that the DMA can initiate pre-configured burst transfers towards the AHB interface.

56.3.3.2.1 Receive Operation

During a Receive operation, the MTL is a slave for the MAC. The general sequence of events is as follows:

1. When the MAC receives a packet, it indicates the availability of receive data
2. The MAC indicates the SOP and EOP delimiters
3. The MTL accepts the data and pushes it into corresponding Rx queue.
4. After the EOP is transferred, the MAC drives the status word which is also pushed into the corresponding Rx queue by the MTL
5. If IEEE 1588 timestamp feature is enabled and the 64-bit timestamp is available along with the packet status, it is pushed into the Rx queue as a part of the status word. For the 64-bit bus, one additional location is taken
6. The MTL takes the data out of the queue and sends it to the DMA depending on the below two modes
7. Threshold mode: In the (default) Threshold mode, the MTL reads the data and indicates its availability to DMA when one of the following happens:
   a) data bytes equal to the threshold amount are written to the Rx queue (RTC bits[30:31] of MTL_RxQ0_Operation_Mode register of corresponding MTL queue)
   b) a full packet of data is received into the queue
8. Store-and-forward mode: In the store-and-forward mode (when Bit 26 of MTL_RxQ0_Operation_Mode register of a queue is set to 1), the initial Rx queue locations are reserved for the status words before writing the SOP. A packet is read out only after it is completely written into the Rx queue. In this mode, all error packets are dropped (if configured through Bit 27 of MTL_RxQ0_Operation_Mode register of a queue) such that only valid packets are read and forwarded to the application

Note: In Threshold (cut-through) mode, the status words are stored after the packet EOP. In store-and-forward mode, the location for the maximum status words are reserved before writing the SOP and the status is written to reserved locations after writing the EOP.

56.3.3.2.2 Receive Operation Multi-Packet Handling

In Threshold mode, the packet status is available immediately after the packet data. In store-and-forward mode, the packet data is available after the packet status. The MTL is capable of storing any number of packets into the queue as long as it is not full.

If the MAC receives a packet when the corresponding Rx queue is full, the MTL ignores that packet. In addition, the MTL increments the overflow counter in the MTL_RxQ0_Missed_Packet_Overflow_Cnt of corresponding queue.

56.3.3.2.3 Error Handling

If the MTL Rx queue is full before it receives the EOP data from the MAC, the following happens:

1. An overflow is declared
2. The whole packet (including the status word) is dropped
3. The overflow counter in the DMA (Overflow Counter register of corresponding MTL queue) is incremented

This is true even if Bit 27 (FEP) of MTL_RxQ0_Operation_Mode register of corresponding MTL queue is set.

If the start address of such a packet has already been transferred to the Read Controller, the rest of the packet is dropped and a dummy EOP is written to the queue along with the status word with overflow status. The status indicates a partial packet because of overflow. In such packets, the Packet Length field is invalid. If the MTL Receive Queue is configured to

operate in the store-and-forward mode and the length of the received packet is more than the queue size, overflow occurs and all such packets are dropped.

The MTL Rx Control logic can filter error and undersized packets, if enabled by using the FEP and FUP bits of MTL_RxQ0_Operation_Mode register of corresponding MTL queue. If the start address of such a packet has already been transferred to the Rx Queue Read Controller, that packet is not filtered. The start address of the packet is transferred to the Read Controller after the packet crosses the receive threshold set by Bits[30:31] of MTL_RxQ0_Operation_Mode register of corresponding MTL queue.

If the MTL Receive Queue is configured to operate in the store-and-forward mode, all error packets can be filtered and dropped. The MTL then starts the transfer of next packet (if available).

56.3.3.2.4 Receive Status Word Format

The Rx packet status words are sent to the application after the packet data in Threshold mode and before the packet data in the Store and Forward Mode.

56.3.4 MAC

The MAC supports MII and RMII interfaces towards the PHY chip. The PHY interface can be selected only once after reset.

56.3.4.1 MAC Transmission

The transmission process is as follows:

1. Transmission is initiated when the MTL pushes in data with the SOP
2. When the SOP is detected, the MAC accepts the data and begins transmitting to the MII. The time required to transmit the packet data to the MII after the application initiates the transmission depends on delay factors such as IPG delay, time to transmit preamble or SFD, and any back-off delays for half-duplex mode. While the packets data is being transmitted, the MAC can throttle the MTL
3. After the EOP is transferred to the MAC, the MAC does one of the following:
   a) The MAC completes the normal transmission and gives the transmission status to the MTL
   b) If a normal collision (in half-duplex mode) occurs during transmission, the MAC gives the Transmit Status with retry bit set to the MTL. The MAC gives the Retry request till either the packet is successfully transmitted or maximum retry requests expire
4. When maximum retry requests expire, the MAC aborts the packet transmission with Excessive Collision Transmit Status. The MAC accepts and drops all further data until

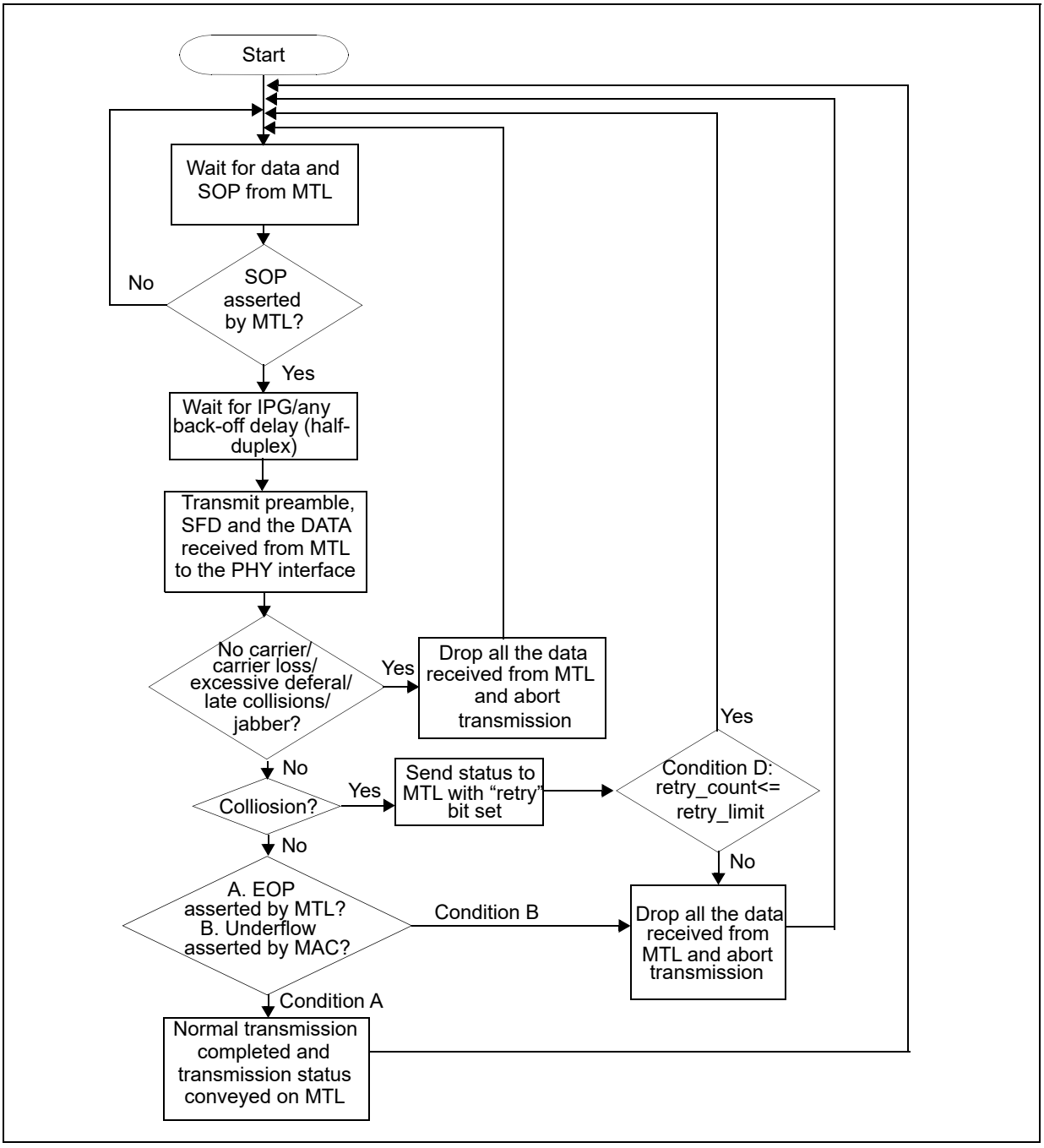
the next SOP is received. The MTL block should retransmit the same packet from SOP on observing a Retry request (in the Status) from the MAC

5. The MAC aborts the packet transmission under the following conditions:
   a) No carrier (half-duplex mode)
   b) Loss of carrier (half-duplex mode)
   c) Excessive deferral (half-duplex mode)
   d) Late collisions (half-duplex mode)
   e) Jabber

   The MAC accepts and drops all further data until the next SOP is received
6. The MAC issues an underflow status if the MTL is not able to provide the data continuously during the transmission. The MAC accepts and drops all further data until the next SOP is received
7. During the normal transfer of a packet from MTL, if the MAC receives a SOP without getting an EOP for the previous packet, it ignores the SOP and considers the new packet as continuation of the previous packet

Figure 1199 illustrates the MAC transmission process flow.

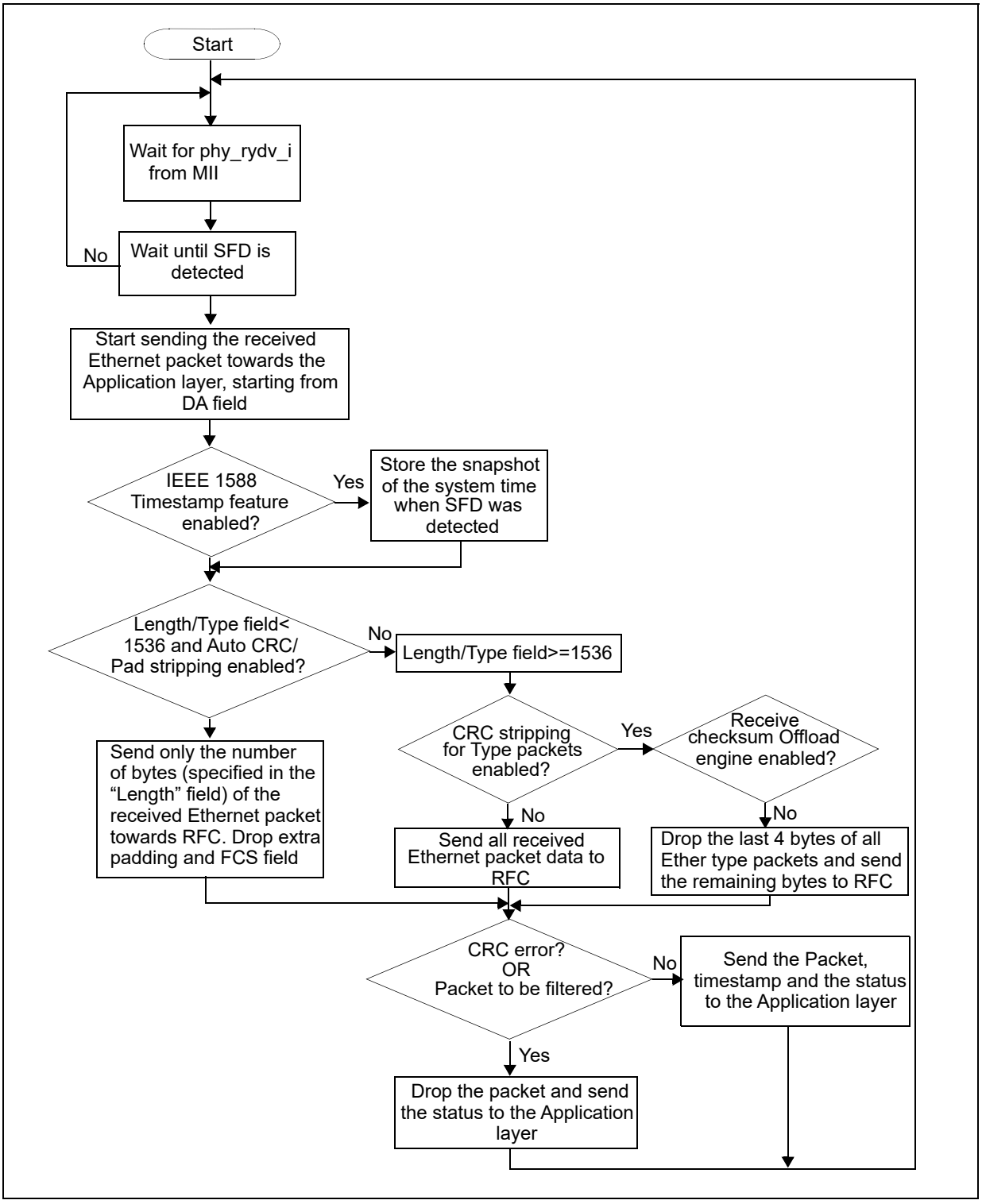
Figure 1199. Overview of MAC Transmission Process Flow



56.3.4.2 MAC Reception

A receive operation is initiated when the MAC detects an SFD on MII. The MAC strips the preamble and SFD before proceeding to process the packet. The header fields are checked for filtering and the FCS field used to verify the CRC for the packet. The received packet is stored in a shallow buffer until the address filtering is performed. The packet is dropped in the MAC if it fails the address filter.

Figure 1200. Overview of MAC RX Operation



56.3.5 Multiple Channels and Queues Support

The Ethernet module supports 2- queues and 2-channels on Tx and Rx paths. *Figure 972* shows the architecture of the Ethernet module with multiple queues and channels. This section describes how the Ethernet module supports multiple queues and channels. It contains the following sub-sections:

- DMA with Multiple Channels
- MTL with Multiple Queues

56.3.5.1 DMA with Multiple Channels

This section describes how multiple queues are supported in the Transmit and Receive DMA paths.

56.3.5.1.1 Multi-Channel DMA Transmit Path

The Ethernet module supports 2- Tx channels with the following priority schemes:

- Fixed Priority Scheme
- Weighted Strict Priority
- Weighted Round Robin

56.3.5.1.1.1 Fixed Priority Scheme

The fixed priority scheme is the default priority scheme for the DMA channels. In fixed priority scheme, the channel with highest priority always wins the arbitration when it requests the bus. Fixed Priority Scheme for DMA Channels provides information about the priority levels of DMA channels.

Table 1247. Fixed Priority Scheme for DMA Channels

Priority Level	Channel
0 (low)	Channel 0
1 (high)	Channel 1

56.3.5.1.1.2 Weighted Strict Priority

In Weighted Strict Priority (WSP), the weight corresponds to the number of burst transfers given to a channel in one arbitration cycle. The unused burst transfers of one or more channels are reallocated based on the priority. The channel with highest priority gets the unused burst transfer time before it is allocated to a channel with next highest priority.

The channel priorities are fixed. Channel 1 has the highest priority and Channel 0 has the lowest priority. When a channel uses the allocated burst transfers, the channel with next lower priority is processed. After processing the allocated bandwidth of all channels that had packets to transmit, any unused burst transfer time is allocated to the channel of the highest priority (if required), and then next highest priority (if required), and so on.

56.3.5.1.1.3 Weighted Round Robin

In Weighted Round Robin (WRR), all channels are serviced in round-robin order according to the weights settings. The TCW field of the DMA_CHIn Weighted Round Robin (WRR), all

channels are _Tx_Control register provides the weight for each Transmit channel as shown in Weights for DMA Channels.

Table 1248. Weights for DMA Channels

TCW Field	Transmit Channel Weight
000	1
001	2
010	3
011	4
100	5
101	6
110	7
111	8

The configured weights correspond to the number of burst transfers given to a channel in one arbitration cycle. The unused or excess burst transfers are distributed equally to all channels.

56.3.5.1.2 Multi-Channel DMA Receive Path

The Ethernet module supports 2- Rx channels. In the Rx direction, the MTL Rx Controller selects the Rx DMA for which it is transferring or reading the data from the Rx FIFO memory. This scheduling is based on the programming done in the respective MTL_RxQ[x]_Control register.

Each Rx DMA indicates when it is ready to transfer data and the size of the burst-length (number of beats) that it has to transfer. The scheduler checks whether sufficient data (of requested burst length) is available to be transferred to these DMAs and then selects the Rx DMA that gets serviced using the programed priorities.

56.3.5.1.3 Priority Scheme for Tx DMA and Rx DMA

The DMA arbiter performs the arbitration between the Tx and Rx paths of DMA channels for accessing descriptors and data buffers. The DMA arbiter supports two types of arbitration: fixed priority and weighted round-robin. The DA bit of the register specifies the arbitration scheme (fixed or weighted round-robin) between the Tx and Rx DMA of a channel.

If you have enabled the Tx DMA and Rx DMA of a channel, you can specify which DMA gets the bus when the channel gets the control of the bus. You can set the priority between the corresponding Tx DMA and Rx DMA by using the TXPR field of the register. For round-robin arbitration, you can use the PR field of the register to specify the weighted priority between the Tx DMA and Rx DMA. Table 5-1 Priority Scheme for Tx DMA and Rx DMA5-1 provides information about the priority scheme between Tx DMA and Rx DMA.

Table 1249. Priority Scheme for Tx DMA and Rx DMA(1)

Bit 17	Bit 18	Bit 19	Bit 20	Bit 30	Priority Scheme
x	x	x	0	1	Rx always has priority over Tx
0	0	0	0	0	Tx and Rx have equal priority. Rx gets the access first on simultaneous requests
0	0	1	0	0	Rx has priority over Tx in ratio 2:1
0	1	0	0	0	Rx has priority over Tx in ratio 3:1
0	1	1	0	0	Rx has priority over Tx in ratio 4:1
1	0	0	0	0	Rx has priority over Tx in ratio 5:1
1	0	1	0	0	Rx has priority over Tx in ratio 6:1
1	1	0	0	0	Rx has priority over Tx in ratio 7:1
1	1	1	0	0	Rx has priority over Tx in ratio 8:1
x	x	x	1	1	Tx always has priority over Rx
0	0	0	1	0	Tx and Rx have equal priority. Tx gets the access first on simultaneous requests
0	0	1	1	0	Tx has priority over Rx in ratio 2:1
0	1	0	1	0	Tx has priority over Rx in ratio 3:1
0	1	1	1	0	Tx has priority over Rx in ratio 4:1
1	0	0	1	0	Tx has priority over Rx in ratio 5:1
1	0	1	1	0	Tx has priority over Rx in ratio 6:1
1	1	0	1	0	Tx has priority over Rx in ratio 7:1
1	1	1	1	0	Tx has priority over Rx in ratio 8:1

1. Bits 17, 18, 19, 20 and 30 are not valid if one of the paths (Transmit or Receive) is not present in a channel. The Transmit and Receive paths are always present in Channel 0.

56.3.5.1.4 Slot Number Function

When the AV feature is enabled, you can use the slot number function to schedule the data fetching from the system memory by the DMA. This feature is useful when the source AV data needs to be transmitted at specific intervals. You can program the slot number at which the DMA should fetch the data from system memory in the Transmit Descriptor Word 3 (TDES3). This 4-bit field allows the application to schedule data up to 16 slots of 125 µs each. This field is applicable only for the AV channels.

When the DMA fetches a Tx descriptor, it compares the slot number of the Tx descriptor with the internally generated reference slot interval. The slot interval is a counter that is updated every 125 µs of the IEEE 1588 system time. In addition, the slot interval counter is initialized to zero when the second's field of the system time is incremented, that is, the sub-second counter rolls over. The DMA fetches the data only if it matches the current slot or the next slot. The DMA remains in the descriptor fetch state till there is a match.

To enable the DMA to fetch the data only if it matches the current slot or the next two slots, you can program Bit 31 of the Slot Function Control and Status register of the corresponding DMA channel.

You can enable the check for slot number by setting Bit 31 of the Slot Function Control and Status register of corresponding DMA channel. When this check is not enabled, the packets are fetched immediately after the descriptor is read. In addition, Bits [12:15] indicate the value of the reference slot number in DMA.

Note: If the slot number in the descriptor is less than the reference slot number, the DMA takes it as a future slot.

56.3.5.1.5 Rx Queue to DMA RX Channel Mapping

The packets in the MTL Rx Queues can be routed to any one of the 2-RX DMA channels by programming the MTL_RxQ_DMA_Map0 Register. The following types of Rx Queue to DMA mapping is possible through programming:

- Static Mapping
- Dynamic (Per Packet) Mapping

56.3.5.1.5.1 Static Mapping

In this mode, all the packets of an Rx Queue is connected to a specific DMA channel. For example, all the packets from Rx Queue 0 can be routed to a DMA channel by programming Q0MDMACH (bit[28:31]) and Q0DDMACH (bit[24] = 0) of the MTL_RxQ_DMA_Map0 Register.

Similarly, packets from other Rx Queues can be routed to any DMA channel by programming register fields corresponding to each Queue.

56.3.5.1.5.2 Dynamic (Per Packet) Mapping

In this mode, the destination DMA channel of a packet being read from an Rx Queue is not constant but decided independently for each packet. For example, if you set the Q1DDMACH bit of the MTL_RxQ_DMA_Map0 register, the static mapping is disabled for Rx Queue 1 and the value in Q1MDMACH is ignored. The destination DMA channel is decided by the MAC Core receiver for each packet, depending on the following in decreasing order of priority:

1. L3-L4 filter Based DMA Selection: When the Layer 3 and Layer 4 Packet Filter is enabled, the TCP/UDP and IP header fields of the received packet are matched against the corresponding values programed and enabled for comparison in the MAC_L3_L4_Address_Control register. If the match is successful, the DMA channel number programed in the DMCHN field of the MAC_L3_L4_Address_Control register is selected as the destination DMA channel number provided DMCHEN bit of the same register is set. If the L3-L4 Registers do not give a comparison match, then the Ethernet module proceeds to the next step below
2. Ethernet DA-Based DMA Selection: The DA address of the received packet is compared against the programed DA values in MAC Address Registers. If the address matches any of the programed values, the corresponding DCS field (when enabled) determines the destination DMA channel number

If none of the above operations is able to make a successful match/decision, then the packet is routed to DMA Channel 0 by default

56.3.5.1.6 Selection of Tag Priorities assigned to Tx and Rx Queues

The VLAN Tag priorities can be assigned to Tx Queues by programming the corresponding PSTQIn field in the *Section 56.2.14: Transmit Queue Priority Mapping 0 Register (MAC_TXQ_PRTY_MAP0)* (for Tx Queues 0 and 1).

The bit corresponding to the VLAN Tag priority can be set in the PSTQIn field for assigning that priority to the Tx Queue. For example, if you want to assign VLAN Tag priority of 3 to Tx Queue 0, set bit [3] in PSTQ0 field.

The same VLAN Tag priority can be set for multiple Tx queues. The Tx packet association to particular Tx Queue is governed by the application, so the MAC does not route the packet based on Tx Queue priority mapping; it is only used for Pause Flow Control (PFC). The settings in the PSTQ*n* field determines the Tx Queues blocked for transmission when the PFC packet is received with corresponding VLAN Tag priorities enabled.

The VLAN Tag priorities can be assigned to Rx Queues by programming the PSRQ field in the corresponding MAC_RxQ_Ctrl2 register. The bit corresponding to the VLAN Tag priority can be set in the PSRQ field for assigning that priority to the Rx Queue. For example, if you want to assign VLAN Tag priority of 3 to Rx Queue 0, set bit [3] in PSRQ field of MAC_RxQ_Ctrl2 register. The VLAN Tag priority assigned to particular Rx Queue must be unique, that is, more than one Rx Queue cannot be assigned the same VLAN Tag priority. However, more than one VLAN Tag priorities can be assigned to same Rx Queue.

The settings in the PSRQ field are used for VLAN Tagged Rx packet routing to Rx Queues as well as for PFC based Tx flow control. The received VLAN Tagged Rx packet is routed to Rx Queue that has the VLAN Tag priority match. In PFC based Tx flow control, PSRQ field corresponding to a particular Rx Queue is used for enabling VLAN Tag priorities in the PFC packet transmitted when corresponding Rx Queue threshold levels are reached.

56.3.5.1.7 Rx Side Routing from MAC to Queues

The MAC routes the Rx packets to the Rx Queues based on following packet types:

- VLAN Tag Priority field in VLAN Tagged AV data packets
- AV Control packets
- VLAN Tag Priority field in VLAN Tagged packets
- Untagged IEEE1588 PTP over Ethernet packets

The outer C-VLAN Tagged AV data Rx packets can be routed based on the priorities assigned to Rx Queues through PSRQ field in corresponding MAC_RxQ_Ctrl2 register and the corresponding Rx Queue is enabled for AV through RXQ*n*EN field in MAC_RxQ_Ctrl0 register. These packets may be single VLAN Tagged with C-VLAN type or Double VLAN Tagged with outer VLAN Tag of C-VLAN type when Double VLAN feature is enabled (EDVLP bit in MAC_VLAN_Tag register is set) with inner C-VLAN Tagged or inner S-VLAN Tagged when SVLAN processing is enabled (ESVL bit in MAC_VLAN_Tag register is set). This type of Rx packet routing is available when AV feature and Multiple Rx Queues are selected in the configuration.

The AV Control Rx packets can be routed based on the Rx Queue number specified in the AVCPQ field in MAC_RxQ_Ctrl1 register and corresponding Rx Queue is enabled for AV through RXQ*n*EN field in MAC_RxQ_Ctrl0 register. These packets may be single VLAN Tagged with C-VLAN type or Double VLAN Tagged with outer VLAN Tag of C-VLAN type when Double VLAN feature is enabled (EDVLP bit in MAC_VLAN_Tag register is set) with inner C-VLAN Tagged or inner S-VLAN Tagged when SVLAN processing is enabled (ESVL

bit in MAC_VLAN_Tag register is set). This type of Rx packet routing is available when AV feature and Multiple Rx Queues are selected in the configuration.

The VLAN Tagged Rx packets can be routed based on the priorities assigned to Rx Queues through PSRQ field in corresponding MAC_RxQ_Ctrl2 register and the corresponding Rx Queue is enabled for generic through RXQ*n*EN field in MAC_RxQ_Ctrl0 register. This type of Rx packet routing is available when Multiple Rx Queues are selected in the configuration.

The untagged IEEE1588 PTP over Ethernet Rx packets can be routed based on the Rx Queue number specified in the AVPTPQ field in MAC_RxQ_Ctrl1 register and the corresponding Rx Queue is enabled for AV through RXQ*n*EN field in MAC_RxQ_Ctrl0 register. This type of Rx packet routing is available when AV feature and Multiple Rx Queues are selected in the configuration.

If the Rx packet cannot be classified in any of the defined packet type for routing, it will be routed through Rx Queue 0.

56.3.5.1.8 Rx Side Arbitration between DMA and MTL

After completing the current packet processing, the DMA Channel Controller fetches the next descriptor. After the descriptor fetching is complete, the DMA Channel Controller evaluates the amount of data to be to be transferred to the Rx Buffer based on the programed PBL and Rx Buffer Length. Accordingly, it requests the MTL to transfer the data.

After servicing the current request, the MTL Rx Queue arbitration scheme selects Rx Queue based on the arbitration scheme (RAA field of the MTL_Operation_Mode register) and the weights programed in the Queue <n> Receive Control Register. The arbitration is done among Queues for which DMA is ready to service. After the Rx Queue is selected, PBL (Programmable Burst Length) amount of data is read out from that Queue and is routed to the Rx DMA Channel based on the Rx Channel selection criteria.

The arbitration takes place when every PBL data transfer is completed and descriptors are ready for the processing from at least one DMA Channel.

56.3.5.2 MTL with Multiple Queues

The Ethernet module supports 2xTx and 2xRx queues. The number of Tx queues is equal to the number of DMA channels. You can control a Tx queue through *Section 56.2.169: Queue 0 Transmit Operation Mode Register (MTL_TXQ0_OPERATION_MODE)*. You can control an Rx queue through *Section 56.2.175: Queue 0 Receive Operation Mode Register (MTL_RXQ0_OPERATION_MODE)*.

56.3.5.2.1 Queue Memory

All MTL Tx queues share one Tx memory and all Rx queues share one Rx memory with programmable individual queue sizes in blocks of 256 bytes. The application can program the Tx queue size in the TQS field of *Section 56.2.169: Queue 0 Transmit Operation Mode Register (MTL_TXQ0_OPERATION_MODE)* and the Rx queue size in the RQS field of the *Section 56.2.175: Queue 0 Receive Operation Mode Register (MTL_RXQ0_OPERATION_MODE)*.

56.3.5.2.2 Queue Modes

You can enable a Tx queue for generic, AV, or all types of traffic. Generic Tx queues use WRR or WSP queuing algorithms. AV Tx queue uses CBS or SP queuing algorithms.

You can enable an Rx queue for generic, AV, or all types of traffic. A particular queue enabled for generic or AV based routing is determined by the RXQ*n*EN field of corresponding queue. The following list explains how Rx queues are enabled based on the selected features:

- Multiple Generic Rx queues: All queues are enabled for generic queuing based on the VLAN Tag priority. The VLAN Tag priority should match the PSRQ field of the MAC_RxQ_Ctrl2 register. Untagged packets are routed through Queue 0. The Rx packets can also be routed to a particular DMA channel based on the DCS field of perfectly-matched MAC Address register
- Multiple Rx queues with AV feature: When you select the *Enable Audio Video Bridging* option, AV is enabled for all selected Rx queues. The queuing is done based on the VLAN Tag priority. The VLAN Tag priority should match the PSRQ field of the MAC_RxQ_Ctrl2 register. The Rx packets can also be routed to a particular DMA channel based on the DCS field of perfectly-matched MAC Address register. The AV control packets (tagged or untagged) are routed based on the AVCPQ field of the MAC_RxQ_Ctrl1 register. The PTP over Ethernet packets are routed based on the AVPTPQ field of MAC_RxQ_Ctrl1 register

56.3.5.2.3 Queue Priorities

You can program the priority of a Tx queue in the MAC_TxQ_Prty_Map0 register. You can program the priority of an Rx queue in Bits[16:23] of corresponding Rx Flow Control Register. The priority should be assigned in the following order:

a) AV queue (high priority)

b) Best-effort queue (low priority)

The software should put packets with correct priorities in the respective programed queue on Tx side. The MAC uses the programed priorities for blocking the Tx queues when a PFC packet is received. If a single queue is selected for multiple priorities and PFC is enabled, the entire queue is paused if one or more priorities in the queue are paused.

56.3.5.2.4 AV Feature

The Ethernet module supports the AV data transfer in 100 Mbps mode. The AV feature enables transmission of time-sensitive traffic over bridged local area networks (LANs). The following standards define various aspects of the AV feature implementation:

- IEEE 802.1Qav-2009: Allows the bridges to provide time-sensitive and loss-sensitive real-time audio video data transmission (AV traffic). It specifies the priority regeneration and controlled bandwidth queue draining algorithms that are used in bridges and AV traffic sources
- IEEE 802.1Qat-2009: Allows the network resources to be reserved for specific traffic streams traversing a bridged local area network
- IEEE 802.1AS-2011: Specifies the protocol and procedures used to ensure that the synchronization requirements are met for time-sensitive applications such as audio and video across bridged and virtual-bridged LANs consisting of LAN media where the transmission delays are fixed and symmetrical. For example, IEEE 802.3 full-duplex links include the maintenance of synchronized time during normal operation followed by addition, removal, or failure of network components and network reconfiguration

56.3.5.2.4.1 Transmit Path Functions

When you select the AV feature, the Transmit paths of Queue 0 and the highest enabled queue are enabled by default.

The Transmit path of Queue 0 supports the strict priority algorithm, and it is used for best-effort traffic. For a queue, the strict priority algorithm determines that a packet is available for transmission if the queue contains one or more packets. When the threshold mode for MTL Tx FIFO is enabled, the strict priority algorithm determines that a packet is available for transmission if the queue contains a partial packet of size equal to the programed threshold limit.

The Transmit paths of additional queues support traffic management by using the credit-based shaper algorithm. For a queue, the credit-based shaper algorithm determines that a queue is available for transmission if the following conditions are true:

- The queue contains one or more packets
- The credit for the queue is positive as per the algorithm

You can disable the credit-based shaper algorithm for all queues or lower-priority queues. When you disable the credit-based shaper algorithm for a queue, the channel uses the default strict priority algorithm.

Each Transmit DMA has a separate descriptor chain for fetching the transmit data. The Transmit channel that gets the access to the system bus depends on the DMA arbiter.

The Transmit path has a shared FIFO (MTL layer) for each queue, as shown in *Figure 972*. The data fetched by the DMA is put in the respective part of the FIFO. The MTL Tx Queue Scheduler controls which part of the FIFO data is transmitted by the MAC. If the credit-based shaper algorithm is enabled for a queue, the corresponding queue is selected for transmission if the following conditions are true:

- If the packet is available in the channel and has a positive or zero credit
- If the higher priority queue has no packet waiting in the FIFO

If the credit-based shaper algorithm is disabled for all queues, the packet to be transmitted from a queue is selected based on the priority scheme described in Fixed Priority Scheme for DMA Channels.

56.3.5.2.4.2 Receive Path Functions

When you select the AV feature, the Receive path of Queue 0 is enabled by default. All traffic is received on this channel. You can enable the Receive paths of additional queues. By enabling the Receive paths of multiple channels, you can demultiplex the received data to send the packets into separate receive channels.

To differentiate between the AV and non-AV traffic, the MAC provides a status that indicates if it is an AV packet and its corresponding VLAN Priority tag value. This status is updated in the Extended Status field of the Receive descriptor as explained in *Section 56.4.1.4: Receive Descriptor*. All received packets with the EtherType field of 0x22F0 are detected as AV packets. The AV packets can be of the following two types:

- AV data packets: The AV data packets are always tagged. The tagged AV data packets are received based on the programed priority value. To specify the channel to which an AV packet with a given priority must be sent, program Bits[16:23] in the Transmit Flow Control Register of the corresponding queue
- AV control packets: The AV control packets can be either tagged or untagged. The untagged AV control packets are received on Queue 0 by default. To receive these

packets on any other queue, you can program Bits[29:31] of the MAC_RxQ_Ctrl1 register. Similar to the AV data packets, the tagged AV control packets are received based on the programed priority value

In addition to the AV packets, you can receive the untagged PTP packets on any queue. By default, the PTP packets (tagged or untagged) are received on Queue 0. To receive these packets on any other queue, you need to program Bits[25:27] of the MAC_RxQ_Ctrl1 register.

56.3.5.2.4.3 Credit-Based Shaper Algorithm

The MTL Queue Scheduler uses the credit-based shaper algorithm to arbitrate the AV traffic in all queues and the legacy Ethernet traffic in Queue 0. You can program the additional queues to use the credit-based shaper algorithm.

The following sections provide information about how you can implement the credit-based shaper algorithm:

Credit Value

The credit value is accumulated every transmit clock cycle, that is, 40 ns for 100 Mbps and 8 ns for 1000 Mbps. The credit to be added or subtracted per cycle can be fractional based on the required idleSlope and sendSlope values, as described in Credit Value Per Transmit Cycle Example.

Table 1250. Credit Value Per Transmit Cycle Example

Mode	Values	Description
100 Mbps	– portTransmitRate = 100 Mbps – idleSlope = 70 Mbps (assuming 70% bandwidth reserved for a higher priority traffic class) – sendSlope = 30 Mbps	– credit = 2.8 bits accumulates per cycle (40 ns) for the higher priority traffic class when best-effort packet is being transmitted. – credit = 1.2 bits drains per cycle (40 ns) when higher priority traffic class packet is being transmitted.

The DMA stores the queue traffic in the respective part of the Tx FIFO based on the slot number in the transmit descriptor (if enabled) or the bandwidth availability on the AMBA AHB application bus.

The credit for a queue builds up only when the packet is available but it cannot be transmitted because the MAC is sending a packet from another queue. The Ethernet module supports another mode in which the credit can build up in advance for a queue in which no packet is available in respective part of the FIFO. This enables sending a burst of high priority traffic in a queue as soon as the data is available. You can enable this mode with Bit 30 of the CBS Control registers of corresponding queues.

When reset, the accumulated credit parameter in the credit-based shaper algorithm is set to zero if there is positive credit, and there is no packet to transmit in a queue. The credit does not accumulate when there is no packet waiting in a queue and other queues are transmitting. When set, the accumulated credit parameter in the credit-based shaper algorithm is not reset to zero if there is positive credit and no packet to transmit in a queue. The credit accumulates even when there is no packet waiting in a queue and other queues are transmitting.

idleSlopeCredit and sendSlopeCredit Values

The software must program the idleSlopeCredit and sendSlopeCredit values. The programed values should be the credit accumulated or drained per clock cycle scaled by 1024, such as, 2.8 x 1024 = 2867 and 1.2 x 1024 = 1229. In addition, the software must program the hiCredit and loCredit values, scaled by 1024, to adjust for scaling of the idleSlopeCredit and sendSlopeCredit values. This means that if computed hiCredit and loCredit values are 12,000 bits and 3,036 bits respectively, the values to be programed in the hiCredit and loCredit registers of the corresponding channel are 12000 x 1024 bits and two's complement of 3036 x 1024, respectively.

Bandwidth Status

The hardware maintains the status of the actual bandwidth consumed by each higher priority queue in the CBS status registers. This enables the software to estimate the average bandwidth consumed by numerically higher traffic classes as compared to the reserved bandwidth.

The CBS status register gives the average number of bits transmitted during the previous programed slot interval (1, 2, 4, 8, or 16 slots of 125 µs) in a queue. The status register is updated even if the credit-based shaper algorithm is not enabled for a queue. The number of slots over which the average bits transmitted per slot are computed is programed in Bits[25:27] of the CBS control register of the respective queue. For example, if you have programed two slots, the average bits are computed over slot numbers 0–1, 2–3, 4–5, and so on.

The value programed in the idleSlopeCredit register of a queue is proportional to the bandwidth reserved for the queue. The software can allocate any bandwidth that is not used by the higher priority queue to the reserved bandwidth of the lower priority queue.

A lower priority queue, which is using the credit-based shaper algorithm, cannot use the unused reserved bandwidth of any higher priority queue that is using the credit-based shaper algorithm. However, a lower priority queue, which is using the strict-priority algorithm, can use the unused reserved bandwidth of any higher priority queue that uses the credit-based shaper algorithm. For example, Queue 1 and Queue 2 use the credit-based shaper algorithm (with reserved bandwidth of 50% and 25%, respectively) and Queue 0 uses the strict-priority algorithm. If Queue 1 uses only 40% of reserved bandwidth, the remaining 10% is used by Queue 0. Queue 2 cannot exceed the reserved bandwidth of 25%.

56.3.5.2.5 Tx Side Arbitration between DMA and MTL

Because the number of Tx DMA channels is always equal to the number of Tx Queues in the MTL, they are always mapped directly. For instance, each Tx DMA pushes data into its respective Tx Queue assigned to it. The main reason for this strategy is to optimize and increase the efficiency of the DMA data transfer, as well as to simplify the TX Checksum Engines in the MTL.

The data inside each Tx Queue is stored in packets. Hence, if two DMAs are allowed to transfer data into the same queue, when a Tx DMA starts a packet transfer, the other DMA cannot transfer data unless the previous packet is completely pushed-in. This means that the second DMA remains idle until the first packet is transferred. Hence, each DMA is always connected directly to its corresponding Tx Queue.

56.3.6 Double VLAN Processing

The Ethernet module supports the double VLAN tagging feature in which the MAC can process up to two VLAN tags (inner and outer). The Ethernet module supports the following:

- Insertion, replacement, or deletion of up to two VLAN tags in the Transmit path
- Packet filtering and stripping based on any one of the two VLAN Tags in the Receive path. Stripping and providing up to two VLAN Tags in the Receive path as a part of the Receive status

56.3.6.1 VLAN Processing in Transmit Path

Table 1251 describes the double VLAN processing features supported by the MAC on the Transmit side.

Table 1251. Double VLAN Processing Features in Tx Path

Feature	Description
Support for C-VLAN and S-VLAN Tag types	The inner or outer VLAN tag can be of C-VLAN and S-VLAN type. The VLAN type is specified through the CSVL bit of MAC_VLAN_Incl and MAC_Inner_VLAN_Incl registers. The Ethernet module supports processing of any sequence of outer and inner VLAN tags. **Note:** The Ethernet module does not support the C-VLAN S-VLAN sequence. The MAC does not check whether the packet provided by the application has a valid sequence of the VLAN Tag types or the insertion or replacement operation results in invalid sequence of VLAN Tag type. Therefore, the application must provide correct sequence of VLAN Tag types and program the MAC in such a way that it results in correct sequence of VLAN Tag types in the transmitted packet. The application must ensure the following: – The inner tag should not be S-VLAN when outer C-VLAN Tag insertion is enabled. – The outer tag should not be C-VLAN when inner S-VLAN Tag insertion is enabled. – The inner tag should not be S-VLAN when outer tag should be replaced with C-VLAN. – The outer tag should not be C-VLAN when inner tag should be replaced with S-VLAN.
VLAN Tag deletion	You can enable the VLAN tag deletion for outer or inner tag through VLC field in the MAC_VLAN_Incl or MAC_Inner_VLAN_Incl register, respectively. When VLAN deletion is enabled, the MAC deletes the tag present at the corresponding position. When a packet has only one tag, it is considered as the outer tag. If inner tag deletion is enabled and the packet has only one tag, the MAC does not delete the tag.
VLAN Tag Insertion or Replacement	You can enable the VLAN tag insertion or replacement for outer or inner tag through VLC field in the MAC_VLAN_Incl or MAC_Inner_VLAN_Incl register, respectively. When VLAN tag insertion or replacement is enabled, the VLTI bit in the MAC_VLAN_Incl or MAC_Inner_VLAN_Incl register is used to determine whether the VLAN tag should be taken from the register or the Control Word.

The software can use the VLAN insertion, replacement, or deletion feature to instruct the MAC to do the following for Tx packets:

- Delete the VLAN Type and VLAN Tag fields
- Insert or replace the VLAN Type and VLAN Tag fields. Insertion or replacement is done based on the setting of VLTI bit in the MAC_VLAN_Incl register as described in *Table 1252*

Table 1252. VLAN Insertion or Replacement Based on VLTI Bit

Condition	Description
VLTI bit is set	The MAC inserts or replaces the following: – VLAN Type field (C-VLAN or S-VLAN as indicated by the CSVL bit of MAC_VLAN_Incl register) – VLAN Tag field with the Content of the VT field of Transmit context descriptor of the packet.
VLTI bit is reset	The MAC inserts or replaces the following: – VLAN Type field (C-VLAN or S-VLAN as indicated by the CSVL bit of MAC_VLAN_Incl register) – VLAN Tag field with the VLT field of MAC_VLAN_Incl register

When VLAN replacement or deletion is enabled, the MAC checks if the VLAN Type field (0x8100 or 0x88a8) is present after the DA and SA fields in the Transmit packet. The replace or delete operation does not occur if the VLAN Type field is not detected in two bytes following the DA and SA fields. However, when VLAN insertion is enabled, the MAC does not check the presence of VLAN Type field in the Transmit packet and just inserts the VLAN Type and VLAN Tag fields.

You can enable the VLAN insertion, replacement, or deletion feature for all Tx packets or selective packets:

- Enabling VLAN insertion, replacement, or deletion for all packets. To enable this feature for all packets, program the VLC and VLP fields of MAC_VLAN_Incl register
- Enabling VLAN insertion, replacement, or deletion for selective packets. To enable this feature for selective packets, Program the VTIR field of TDES2 Normal Descriptor

56.3.6.2 VLAN Processing in Receive Path

Table 1253 describes the features supported by the MAC on the Receive side and the corresponding bits in the MAC_VLAN_Tag register.

Table 1253. Double VLAN Processing in Rx Path

Feature	Description
Outer or inner VLAN tag-based filtering	The MAC can filter packets based on the outer or inner VLAN tag through the ERIVLT bit.
C-VLAN or S-VLAN tag-based filtering	The MAC can filter packets based on the C-VLAN or S-VLAN type based on the ERSVLM bit.
Outer and Inner VLAN Tag stripping	The MAC can strip the outer and inner VLAN Tags from received frame based on the EVLS and EIVLS bits.

Table 1253. Double VLAN Processing in Rx Path (continued)

Feature	Description
16-bit outer and inner VLAN Tag and Type in Rx status	The MAC can provide the 16-bit outer and inner VLAN Tag and Type in the Rx status based on the EVLRXS and EIVLRXS bits, respectively.
Disabling or skipping checking of outer VLAN Tag type	The MAC can disable or skip checking of outer VLAN Tag type to match C-VLAN or S-VLAN based on the DOVLTC bit.

56.3.7 Source Address Insertion, Replacement, or Deletion

Software can use the SA insertion or replacement feature to instruct the MAC to do the following for Tx packets:

- Insert the content of the MAC Address Registers in the SA field
- Replace the content of the SA field with the content of the MAC Address Registers

When SA insertion is enabled, the application must ensure that the packets sent to the MAC do not have the SA field. The MAC does not check whether the SA field is present in the Transmit packet and it inserts the content of MAC Address Registers in the SA field. Similarly, when SA replacement is enabled, the application must ensure that the SA field is present in the packets sent to the MAC. The MAC replaces the six bytes following the Destination Address field in the Transmit packet with the content of the MAC Address Registers.

You can enable the SA insertion or replacement feature for all Transmit packets or selective packets:

- To enable this feature for all packets, program the SARC field of the MAC_Configuration register
- To enable this feature for selective packets, use the following method:
  Program the SA Insertion Control field (Bits[6:8] of TDES3 Normal Descriptor (Read Format)) in the first Transmit descriptor of the packet. When Bit 6 of TDES3 is set, the SA Insertion Control field indicates insertion or replacement by MAC Address1 registers. When Bit 6 of TDES3 is reset, it indicates insertion or replacement by MAC Address 0 registers
- If both MAC_Configuration Register and TDES3 are programed for SA insertion, then setting of MAC_Configuration register takes precedence

If MAC Address1 Registers are not enabled, the MAC Address0 registers are used for insertion or replacement irrespective of the value of the most-significant bit of the SA Insertion Control field.

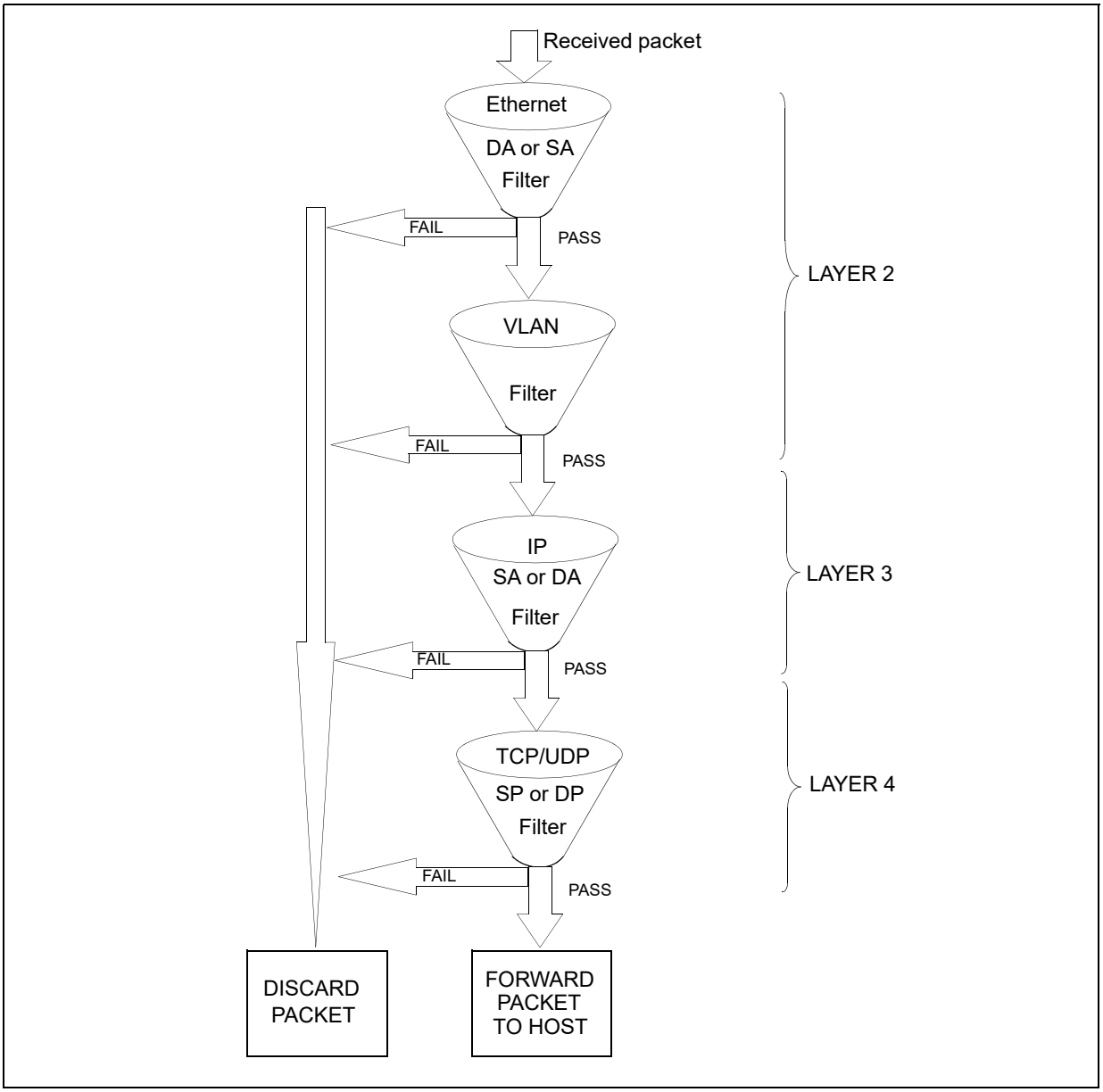
56.3.8 Packet Filtering

The Ethernet module supports the following types of filtering for Rx packets:

- Source Address or Destination Address Filtering: The Address Filtering Module (AFM) checks the source address and destination address fields of each incoming packet
- VLAN Filtering: The Ethernet module supports the VLAN tag-based and VLAN hash filtering
- Layer 3 and Layer 4 Filtering: Layer 3 filtering refers to source address and destination address filtering. Layer 4 filtering refers to source port and destination port filtering

Figure 1201 shows the filtering sequence for Rx packets.

Figure 1201. Packet Filtering Sequence



56.3.8.1 Source Address or Destination Address Filtering

The Address Filtering Module of the MAC checks the source address (SA) and destination address (DA) fields of each incoming packet.

56.3.8.1.1 Unicast Destination Address Filtering

The MAC supports up to 128 MAC addresses for unicast perfect filtering. If perfect filtering is selected (HUC bit of MAC_Packet_Filter register is reset), the MAC compares all 48 bits of received unicast address with the programed MAC address for any match. The default MacAddr0 is always enabled.

The MacAddr1 to MacAddr127 addresses are selected with an individual enable bit. For MacAddr1 to MacAddr31 addresses, you can mask each byte during comparison with corresponding received DA byte by setting the corresponding Mask Byte Control bit in the register. This enables group address filtering for the DA. The MacAddr32 to MacAddr127 addresses do not have mask control and all 6-bytes of the MAC address are compared with the received 6-bytes of DA.

In hash filtering mode (when HUC of MAC_Packet_Filter register bit is set), the MAC performs imperfect filtering for unicast addresses using a 64-bit Hash table. For hash filtering, the MAC uses the upper 6 bits CRC of the received destination address to index the content of the Hash table. A value of 00000 selects Bit 63 of selected register, and a value of 11111 selects Bit 0 of Hash Table register. If the corresponding bit (indicated by the 6-bit CRC) is set to 1, the unicast packet is considered to have passed the Hash filter; otherwise, the packet is considered to have failed the Hash filter.

56.3.8.1.2 Multicast Destination Address Filtering

To program the MAC to pass all multicast packets, set the PM bit in *Section 56.2.3: MAC Packet Filter Register (MAC_PACKET_FILTER)*. If the PM bit is reset, the MAC performs the filtering for multicast addresses based on the HMC bit of the MAC_Packet_Filter register.

In Perfect filtering mode, the multicast address is compared with the programed MAC Destination Address registers (1–31). Group address filtering is also supported.

In Hash filtering mode, the MAC performs imperfect filtering using a 64-bit Hash table. The MAC uses the upper 6-bits CRC of received multicast address to index the content of the Hash table. A value of 000000 selects Bit 63 of selected register and a value of 111111 selects Bit 0 of the Hash Table register. If the corresponding bit is set to 1, the multicast packet is considered to have passed the Hash filter. Otherwise, the packet is considered to have failed the Hash filter.

56.3.8.1.3 Hash or Perfect Address Filtering

To configure the DA filter to pass a packet when its DA matches either the Hash filter or the Perfect filter, set the HPF bit and the corresponding HUC or HMC bits in *Section 56.2.3: MAC Packet Filter Register (MAC_PACKET_FILTER)*. This is applicable to both unicast and multicast packets. If the HPF bit is reset, only one of the filters (Hash or Perfect) is applied to receive packet.

56.3.8.1.4 Broadcast Address Filtering

The MAC does not filter any broadcast packets by default. To program the MAC to reject all broadcast packets, set the DBF bit in MAC_Packet_Filter register.

56.3.8.1.5 Unicast Source Address Filtering

The MAC can perform perfect filtering based on the source address field of received packets. By default, the MAC compares the SA field with the values programed in the SA registers. You can configure the MAC Address registers[1–31] to use SA instead of DA for comparison by setting Bit 1 of corresponding register.

The MAC also supports group filtering with SA. You can filter a group of addresses by masking one or more bytes of the address. The MAC drops the packets that fail the SA filter if the SAF bit is set in MAC_Packet_Filter register. Otherwise, the result of the SA filter is given as a status bit in the Receive Status word. When the SAF bit is set, the SA filter and

DA filter result is ANDed to decide whether the packet needs to be forwarded. This means that the packet is dropped if either filter fails. The packet is forwarded to the application only if the packet passes both filters in-order.

56.3.8.1.6 Inverse Filtering

For DA and SA filtering, you can invert the filter-match result at the final output by setting the DAIF and SAIF bits of MAC_Packet_Filter register. The DAIF bit is applicable for both Unicast and Multicast DA packets. The result of the unicast or multicast destination address filter is inverted in this mode. Similarly, when the SAIF bit is set, the result of unicast SA filter is reversed.

Table 1254 and *Table 1255* summarize the DA and SA filtering based on the type of packets received.

Table 1254. Destination Address Filtering

Packet Type	PR	HPF	HUC	DAIF	HMC	PM	DBF	DA Filter Operation
Broadcast	1	X	X	X	X	X	X	Pass
	0	X	X	X	X	X	0	Pass
	0	X	X	X	X	X	1	Fail
Unicast	1	X	X	X	X	X	X	Pass all packets
	0	X	0	0	X	X	X	Pass on Perfect/Group filter match
	0	X	0	1	X	X	X	Fail on Perfect/Group filter match
	0	0	1	0	X	X	X	Pass on Hash filter match
	0	0	1	1	X	X	X	Fail on Hash filter match
	0	1	1	0	X	X	X	Pass on Hash or Perfect/Group filter match
	0	1	1	1	X	X	X	Fail on Hash or Perfect/Group filter match
Multicast	1	X	X	X	X	X	X	Pass all packets
	X	X	X	X	X	1	X	Pass all packets
	0	X	X	0	0	0	X	Pass on Perfect/Group filter match and drop Pause packets if PCF = 0x
	0	0	X	0	1	0	X	Pass on Hash filter match and drop Pause packets if PCF = 0x
	0	1	X	0	1	0	X	Pass on Hash or Perfect/Group filter match and drop Pause packets if PCF = 0x
	0	X	X	1	0	0	X	Fail on Perfect/Group filter match and drop Pause packets if PCF = 0x
	0	0	X	1	1	0	X	Fail on Hash filter match and drop Pause packets if PCF = 0x
	0	1	X	1	1	0	X	Fail on Hash or Perfect/Group filter match and drop Pause packets if PCF = 0x

Table 1255. Source Address Filtering

Packet Type	PR	SAIF	SAF	SA Filter Operation
Unicast	1	X	X	Pass all packets.
	0	0	0	Pass status on Perfect or Group filter match but do not drop packets that fail
	0	1	0	Fail status on Perfect or Group filter match but do not drop packet
	0	0	1	Pass on Perfect or Group filter match and drop packets that fail
	0	1	1	Fail on Perfect or Group filter match and drop packets that fail

Note: When the RA bit of MAC_Packet_Filter register is set, all packets are forwarded to the system along with the correct result of the address filtering in the Rx Status.

56.3.8.2 VLAN Filtering

The Ethernet module supports the following types of VLAN filtering:

- VLAN Tag Perfect Filtering
- VLAN Tag Hash Filtering

56.3.8.2.1 VLAN Tag Perfect Filtering

In VLAN tag perfect filtering, the MAC compares the VLAN tag of received packet and provides the VLAN packet status to the application. Based on the programed mode, the MAC compares the lower 12 bits or all 16 bits of received VLAN tag to determine the perfect match.

If VLAN tag perfect filtering is enabled, the MAC forwards the VLAN-tagged packets along with VLAN tag match status and drops the VLAN packets that do not match. You can also enable the inverse matching for VLAN packets by setting the VTIM bit of MAC_VLAN_Tag register. In addition, you can enable matching of S-VLAN tagged packets along with the default C-VLAN tagged packets by setting the ESVL bit of MAC_VLAN_Tag register. The VLAN packet status bit (Bit 21 of RDES0) indicates the VLAN tag match status for the matched packets.

Note: The source or destination address (if enabled) has precedence over the VLAN tag filters. This means that a packet that fails the source or destination address filter is dropped irrespective of the VLAN tag filter results.

56.3.8.2.2 VLAN Tag Hash Filtering

The MAC provides VLAN tag hash filtering with a 16-bit Hash table. The MAC performs the VLAN hash matching based on the VTHM of the MAC_VLAN_Tag register. If the VTHM bit is set, the most significant four bits of CRC-32 of VLAN tag are used to index the content of the VLAN Hash Table register. A value of 1 in the VLAN Hash Table register, corresponding to the index, indicates that the VLAN tag of the packet matched and the packet should be forwarded. A value of 0 indicates that VLAN-tagged packet should be dropped.

The MAC also supports the inverse matching for VLAN packets. In the inverse matching mode, when the VLAN tag of a packet matches the perfect or hash filter, the packet should be dropped. If the VLAN perfect and VLAN hash match are enabled, a packet is considered

as matched if either the VLAN hash or the VLAN perfect filter matches. When inverse match is set, a packet is forwarded only when both perfect and hash filters indicate mismatch.

Table 1256 shows the different possibilities for VLAN matching and the final VLAN match status. When the RA bit of the MAC_Packet_Filter register is set, all packets are received and the VLAN match status is indicated in the VF bit of RDES2 Normal Descriptor (Write-Back Format). When the RA bit is not set and the VTFE bit is set in the MAC_Packet_Filter register, the packet is dropped if the final VLAN match status is Fail. In *Table 1256*, value *X* means that this column can have any value.

When VLAN VID is programed to 0 in the VL field of MAC_VLAN_Tag register, all VLAN-tagged packets are considered as perfect matched but the status of the VLAN hash match depends on the VTHM and VTIM bits in MAC_VLAN_Tag register.

Table 1256. VLAN Match Status(1)

VID	VLAN Perfect Filter Match Result	VTHM Bit	VLAN Hash Filter Match Result	VTIM Bit	Final VLAN Match Status
VID = 0	Pass	0	X	X	Pass
	Pass	1	X	0	Pass
	Pass	1	Fail	1	Pass
	Pass	1	Pass	1	Fail
VID!= 0	Pass	X	X	0	Pass
	Fail	0	X	0	Fail
	Fail	1	Fail	0	Fail
	Fail	1	Pass	0	Pass
	Fail	0	X	1	Pass
	Pass	X	X	1	Fail
	Fail	1	Pass	1	Fail
	Fail	1	Fail	1	Pass

1. In this table, X represents any value.

Note: The 16 or 12 bits of VLAN Tag are considered for CRC-32 computation based on ETV bit in MAC_VLAN_Tag register. When ETV bit is reset, most significant four bits of CRC-32 of VLAN Tag are inverted and used to index the content of MAC_VLAN_Hash_Table register. When ETV bit is set, most significant four bits of CRC-32 of VLAN Tag are directly used to index the content of MAC_VLAN_Hash_Table register.

56.3.8.3 Layer 3 and Layer 4 Filtering

The Ethernet module supports Layer 3 and Layer 4 based packet filtering. The Layer 3 filtering refers to the IP Source or Destination Address filtering in the IPv4 or IPv6 packets whereas Layer 4 filtering refers to the Source or Destination Port number filtering in TCP or UDP.

The Layer 3 and Layer 4 packet filtering feature automatically enables the IPC Full Checksum Offload Engine on the Receive side. For Layer 3 or Layer 4 filtering operation, you must set the IPC bit of the MAC_Configuration register to enable the Rx Checksum Offload Engine.

When Layer 3 and Layer 4 filtering is enabled, the packets are filtered in the following way:

- Matched Packets: The MAC forwards the packets that match all enabled fields to the application along with the status. The MAC gives the matched field status only if the IPC bit of MAC_Configuration register is set and one of the following conditions is true:
  - All enabled Layer 3 and Layer 4 fields match
  - At least one of the enabled field matches and other fields are bypassed or disabled
- Unmatched Packets: The MAC drops the packets that do not match any of the enabled fields. You can use the inverse match feature to block or drop a packet with specific TCP or UDP over IP fields and forward all other packets. The aborted or partial packets can be dropped in the MTL Rx FIFO. If the Rx FIFO operates in the Threshold (cut-through) mode and the threshold is programed to a small value, such that packet transfer to application starts before the failed Layer 3 and Layer 4 filter results are available, the application may receive a partial packet with appropriate abort status.
- Non-TCP or UDP IP Packets: By default, all non-TCP or UDP IP packets are bypassed from the Layer 3 and Layer 4 filters. You can optionally program the MAC to drop all non-TCP or UDP over IP packets.

Note: The source or destination address and VLAN tag filters (if enabled) have precedence over Layer 3 and Layer 4 filter. This means that a packet which fails the source or destination address or VLAN tag filter is dropped irrespective of the Layer 3 and Layer 4 filter results.

56.3.8.3.1 Layer 3 and Layer 4 Filters Register Set

The MAC implements a set of registers for Layer 3 and Layer 4 based packet filtering. In a register set, there is a control register, such as MAC_L3_L4_Control0, to control the packet filtering. In addition, there are five address registers to program the Layer 3 and Layer 4 fields to be matched, such as:

- MAC_Layer4_Address0
- MAC_Layer3_Address0_Reg0
- MAC_Layer3_Address1_Reg0
- MAC_Layer3_Address2_Reg0
- MAC_Layer3_Address3_Reg0

56.3.8.3.2 Layer 3 Filtering

The Ethernet module supports perfect matching or inverse matching for IP Source Address and Destination Address. In addition, you can match the complete IP address or mask the lower bits matching, that is, compare all bits of the address except the specified lower mask bits.

For IPv6 packets filtering, you can enable the last four data registers to contain the 128-bit IP Source Address or IP Destination Address. The IP Source Address or Destination Address should be programed in the order defined in the IPv6 specification, that is, the first byte of the IP Source Address or Destination Address in the received packet is in the higher byte of the register and the subsequent registers follow the same order.

For IPv4 packet filtering, you can enable the second and third data registers to contain the 32-bit IP Source Address and IP Destination Address. The remaining two data registers are reserved. The IP Source Address or Destination Address should be programed in the order defined in the IPv4 specification, that is, the first byte of IP Source Address and Destination Address in the received packet in the higher byte of the respective register.

56.3.8.3.3 Layer 4 Filtering

The Ethernet module supports perfect matching or inverse matching for TCP or UDP Source and Destination Port numbers. However, you can program only one type (TCP or UDP) at a time. The first data register contains the 16-bit Source and Destination Port numbers of TCP or UDP, that is, the lower 16 bits for Source Port number and higher 16 bits for Destination Port number.

The TCP or UDP Source and Destination Port numbers should be programed in the order defined in the TCP or UDP specification, that is, the first byte of TCP or UDP Source and Destination Port number in the received packet is in the higher byte of the register.

56.3.9 IEEE 1588 Timestamps

The IEEE 1588 defines a Precision Time Protocol (PTP) which enables precise synchronization of clocks in measurement and control systems implemented with technologies such as network communication, local computing, and distributed objects. The PTP applies to systems communicating by local area networks supporting multicast messaging, including (but not limited to) Ethernet. This protocol enables heterogeneous systems that include clocks of varying inherent precision, resolution, and stability to synchronize. The protocol supports system-wide synchronization accuracy in the sub-microsecond range with minimal network and local clock computing resources.

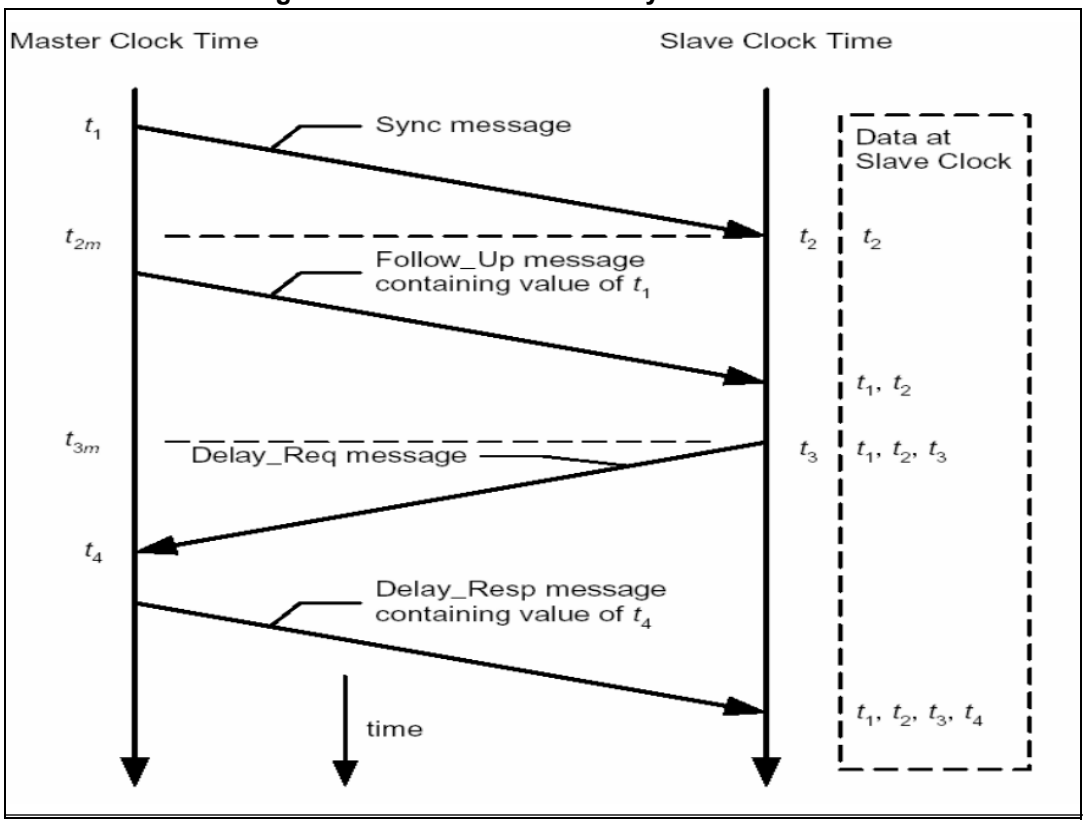
The Ethernet module supports the IEEE 1588-2002 (version 1) and IEEE 1588-2008 (version 2). The IEEE 1588-2002 supports PTP transported over UDP/IP. The IEEE 1588-2008 supports PTP transported over Ethernet. The Ethernet module provides programmable support for both standards. It supports the following features:

- Supports both timestamp formats
- Provides an option to take snapshot of all packets or only PTP type packets
- Provides an option to take snapshot of only event messages
- Provides an option to take the snapshot based on the clock type: ordinary, boundary, end-to-end transparent, and peer-to-peer transparent
- Provides an option to select the node to be a master or slave for ordinary and boundary clock
- Identifies the PTP message type, version, and PTP payload in packets sent directly over Ethernet and sends the status
- Provides an option to measure sub-second time in digital or binary format

56.3.9.1 Delay Request-Response Mechanism

The system or network is classified into the master and slave nodes for distributing the timing and clock information. *Figure 1202*, shows the process that PTP uses for synchronizing a slave node to a master node by exchanging PTP messages.

Figure 1202. Networked Time Synchronization



As shown, the PTP uses the following process:

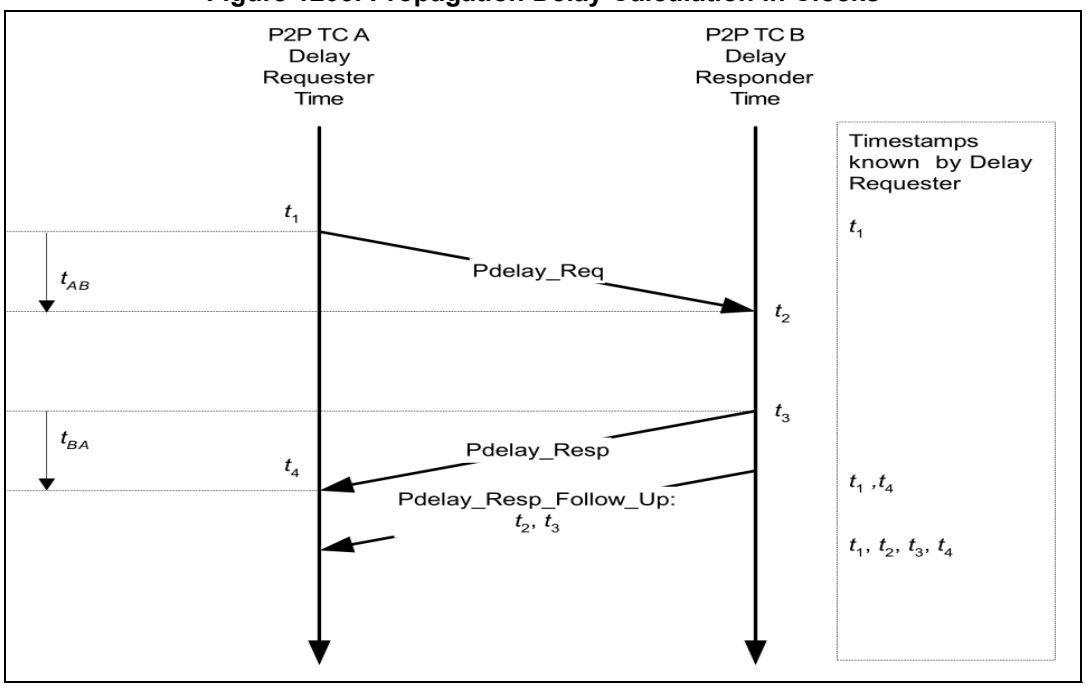
1. The master broadcasts the PTP Sync messages to all its nodes. The Sync message contains the reference time information of the master. This message leaves the system of the master at t1. This time must be captured for Ethernet ports at MII
2. The slave receives the Sync message and also captures the exact time, t2, using its timing reference
3. The master sends a Follow_up message to the slave, which contains t1 information for later use
4. The slave sends a Delay_Req message to the master and notes the exact time, t3, at which this packet leaves the MII interface
5. The master receives the message, capturing the exact time t4, at which the message enters its system
6. The master sends the t4 information to the slave in the Delay_Resp message.
7. The slave uses the four values of t1, t2, t3, and t4 to synchronize its local timing reference to the timing reference of the master

Most of the PTP implementation is done in the software above the UDP layer. However, the hardware support is required to capture the exact time when specific PTP packets enter or leave the Ethernet port at the MII interface. This timing information must be captured and returned to the software for proper implementation of PTP with high accuracy.

56.3.9.2 Peer-to-Peer PTP Transparent Clock (P2P TC) Message Support

The IEEE 1588-2008 supports peer-to-peer PTP (Pdelay) message in addition to the SYNC, Delay Request, Follow-up, and Delay Response messages. *Figure 1203* shows the method to calculate the propagation delay in clocks supporting peer-to-peer path correction.

Figure 1203. Propagation Delay Calculation in Clocks



As shown, the propagation delay is calculated in the following way:

1. Port 1 issues a Pdelay_Req message and generates a timestamp (t1) for the Pdelay_Req message
2. Port 2 receives the Pdelay_Req message and generates a timestamp (t2) for this message
3. Port 2 returns a Pdelay_Resp message and generates a timestamp (t3) for this message

To minimize errors because of any frequency offset between the two ports, Port 2 returns the Pdelay_Resp message as quickly as possible after the receipt of the Pdelay_Req message. Port 2 returns any one of the following:

   a) Difference between the timestamps t2 and t3 in the Pdelay_Resp message
   b) Difference between the timestamps t2 and t3 in the Pdelay_Resp_Follow_Up message
   c) Timestamps t2 and t3 in the Pdelay_Resp and Pdelay_Resp_Follow_Up messages, respectively

4. Port 1 generates a timestamp (t4) on receiving the Pdelay_Resp message
5. Port 1 uses all four timestamps to compute the mean link delay

56.3.9.3 Clock Types

The Ethernet module supports the following clock types defined in the IEEE 1588-2008:

- Ordinary Clock
- Boundary Clock
- End-to-End Transparent Clock
- Peer-to-Peer Transparent Clock

56.3.9.3.1 Ordinary Clock

The ordinary clock in a domain supports a single copy of the protocol. The ordinary clock has a single PTP state and a single physical port. In typical industrial automation applications, an ordinary clock is associated with an application device such as a sensor or an actuator. In telecom applications, the ordinary clock can be associated with a timing demarcation device.

The ordinary clock can be a grandmaster or a slave clock. It supports the following features:

- Sends and receives PTP messages. The timestamp snapshot can be controlled
- Maintains the data sets such as timestamp values

Table 1257 shows the messages for which you can take the timestamp snapshot on the receive side for master and slave nodes.

Table 1257. Ordinary Clock: PTP Messages for Snapshot

Master	Slave
Delay_Req	SYNC

For an ordinary clock, you can take the snapshot of either of the following PTP message types: version 1 or version 2. You cannot take the snapshots for both PTP message types. You can take the snapshot by setting the TSVER2ENA bit and selecting the snapshot mode in MAC_Timestamp_Control Register.

56.3.9.3.2 Boundary Clock

The boundary clock typically has several physical ports communicating with the network. The messages related to synchronization, master-slave hierarchy, and signaling terminate in the protocol engine of the boundary clock and such messages are not forwarded. The PTP message type status given by the MAC helps to identify the type of message and take appropriate action.

The boundary clock is similar to the ordinary clock except for the following features:

- The clock data sets are common to all ports of the boundary clock
- The local clock is common to all ports of the boundary clock

56.3.9.3.3 End-to-End Transparent Clock

The end-to-end transparent clock supports the end-to-end delay measurement mechanism between the slave clocks and the master clock. The end-to-end transparent clock forwards all messages like normal bridge, router, or repeater. The residence time of a PTP packet is the time taken by the PTP packet from the Ingress port to the Egress port.

The residence time of a SYNC packet inside the end-to-end transparent clock is updated in the correction field of the associated Follow_Up PTP packet before it is transmitted. Similarly, the residence time of a Delay_Req packet, inside the end-to-end transparent clock, is updated in the correction field of the associated Delay_Resp PTP packet before it is transmitted. Therefore, the snapshot needs to be taken at both Ingress and Egress ports only for the messages mentioned in the bullet points below. The snapshot is taken by setting the SNAPTYPSEL bits to '10' in the MAC_Timestamp_Control register.
End to End Transparent Clock: PTP Messages for Snapshot:

- SYNC
- Delay_Req

56.3.9.3.4 Peer-to-Peer Transparent Clock

The peer-to-peer transparent clock differs from the end-to-end transparent clock in the way it corrects and handles the PTP timing messages. In all other aspects, it is identical to the end-to-end transparent clock.

In the peer-to-peer transparent clock, the computation of the link delay is based on an exchange of Pdelay_Req, Pdelay_Resp, and Pdelay_Resp_Follow_Up messages with the link peer. The residence time of the Pdelay_Req and the associated Pdelay_Resp packets is added and inserted into the correction field of the associated Pdely_Resp_Followup packet. Therefore, support for taking snapshot for the event messages related to Pdelay is added as mentioned in the below bullet points.
Peer-to-Peer Transparent Clock: PTP Messages for Snapshot:

- SYNC
- Pdelay_Req
- Pdelay_Resp

The snapshot is taken by setting the SNAPTYPESEL bit to '11' in MAC_Timestamp_Control register.

56.3.9.4 Reference Timing Source

To get a snapshot of the time, the MAC requires a reference time in 64-bit format as defined in the IEEE 1588-2002 (80-bit format as defined in the IEEE 1588-2008).

The Ethernet module provides an Internal Reference Time (80-bit). It takes the reference clock input and uses it to internally generate the Reference time (also called the system time) and capture timestamps. The timestamp has the following fields:

- UInteger48 secondsField: The seconds field is the integer portion of the timestamp in units of seconds. It is 48-bits wide. For example, 2.000000001 seconds are represented as secondsField = 0x0000_0000_0002
- UInteger32 nanosecondsField: The nanoseconds field is the fractional portion of the timestamp in units of nanoseconds. For example, 2.000000001 nanoseconds are represented as nanoSeconds = 0x0000_0001

The nanoseconds field supports the following two modes:

- Digital rollover mode: In this mode, the maximum value in the nanoseconds field is 0x3B9A_C9FF, that is, (10e9-1) nanoseconds
- Binary rollover mode: In this mode, the nanoseconds field rolls over and increments the seconds field after value 0x7FFF_FFFF. Accuracy is ~0.466 ns per bit

These modes are set through TSCTRLSSR bit in MAC_Timestamp_Control register.

The timestamp maintained in the MAC is 64-bit wide. The overflow to upper 16-bits of seconds register happens once in 130 years. You can read the values of the upper 16-bits of the seconds field from the CSR register.

The Ethernet module supports the following reference timing source features:

- Fixed Pulse-Per-Second OutputPulse-Per-Second Output
- Flexible Pulse-Per-Second OutputPulse-Per-Second Output
- Auxiliary Snapshots with External Events

56.3.9.4.1 Fixed Pulse-Per-Second Output

The Ethernet module supports the pulse-per-second (PPS) output that is given to indicate 1 second interval (default). You can change the frequency of the PPS output by setting the PPSCTRL0 field in the MAC_PPS_Control register.

56.3.9.4.2 Flexible Pulse-Per-Second Output

The Ethernet module also provides the flexibility to program the start or stop time, width, and interval of the pulse generated on the ptp_pps_o output. The Ethernet module provides the following features with the flexible PPS output:

- Supports programming the start or stop time in terms of system time
- Supports programming the start point of the single pulse and start and stop points of the pulse train in terms of 64-bit system time. The Target Time registers are used to program the start and stop time
- Supports programming the stop time in advance, that is, you can program the stop time before the actual start time has elapsed
- Supports programming the width between the rising edge and corresponding falling edge of PPS signal output in terms of number of units of sub-second increment value programed in the MAC_Sub_Second_Increment register. You can program the width of pulse from 1 to 2^{32}-1 units of sub-second increment value
- Supports programming the interval, between the rising edges of PPS signal, in terms of number of units of sub-second increment value. You can program the interval between pulses from 1 to 2^{32}-1 units of sub-second increment value
- Provides the option to cancel the programed PPS start or stop request
- Indicates error if the start or stop time being programed has already elapsed

Note: The PTP Reference clock mentioned in the following sections is the clock at which the system time gets updated. When the TSCFUPDT bit of MAC_Timestamp_Control register is set to 0, this clock is similar to the clk_ptp_ref_i clock. In the Fine Correction mode, this is the clock tick at which the system time gets updated (using incr_sub_sec_reg shown in Figure 1204).

56.3.9.4.2.1 PPS Start or Stop Time

You can initially program the start time in the Target Time register. If required, you can again program the start or stop time but you can do it only after the earlier programed value is synchronized to the PTP clock domain. Bit 0 of MAC_PPS*n*_Target_Time_Nanoseconds register indicates that the synchronization is complete. This enables you to program the start or stop time in advance even before the earlier stop or start time has elapsed.

To ensure proper PPS signal output, you should program advanced system time for the start or stop time. If the application programs a start or stop time that has already elapsed, the

MAC sets an error status bit indicating the programming error. If enabled, the MAC also sets the Target Time Reached interrupt event. The application can cancel the start or stop request only if the corresponding start or stop time has not elapsed. If the time has elapsed, the cancel command has no effect.

56.3.9.4.2.2 PPS Width and Interval

The PPS width and interval are programed in terms of granularity of system time, that is, number of the units of sub-second increment value. For example, to have a PPS pulse width of 40 ns and interval of 100 ns with PTP reference clock of 50 MHz, you should program the width and interval to values 2 and 5, respectively. You can achieve smaller granularity by using a faster PTP reference clock.

Before giving the command to trigger a pulse or pulse train on the PPS output, you should program or update the interval and width of the PPS signal output.

56.3.9.4.3 Auxiliary Snapshots with External Events

The auxiliary snapshot feature allows you to store a snapshot of the system time based on an external event. The event is considered to be the rising edge of the ptp_aux_ts_trig_i sideband signal.

The Ethernet module has two auxiliary snapshot inputs and depth 4 of a single common auxiliary snapshot FIFO. The snapshots taken for any input are stored in a common FIFO. The application can read the MAC_Timestamp_Status register to know the timestamp of which input is available for reading at the top of this FIFO.

The MAC stores these snapshots in a FIFO. Only 64-bits of the timestamp are stored in the FIFO. You can read the upper 16-bits of seconds from the MAC_System_Time_Higher_Word_Seconds register. When a snapshot is stored, the MAC indicates this to the application with an interrupt. The value of the snapshot is read through a FIFO register access. If the FIFO becomes full and an external trigger to take the snapshot is asserted, a snapshot trigger-missed status (ATSSTM) is set in the MAC_Timestamp_Status register. This indicates that the latest auxiliary snapshot of the timestamp is not stored in the FIFO. The latest snapshot is not written to the FIFO when it is full.

When an application reads the 64-bit timestamp from the FIFO, the space becomes available to store the next snapshot. You can clear a FIFO by setting the ATSFC bit in MAC_Auxiliary_Control register. When multiple snapshots are present in the FIFO, the count is indicated in Bits[4:6] of MAC_Timestamp_Status register.

56.3.9.5 System Time Register Module

The 64-bit time is maintained in this module and updated using the input reference clock (clk_ptp_ref_i). This time is the source for taking snapshots (timestamps) of Ethernet packets being transmitted or received at the MII interface.

The system time counter can be initialized or corrected using the coarse correction method. In this method, the initial value or the offset value is written to the Timestamp Update register. For initialization, the system time counter is written with the value in the Timestamp Update register. For system time correction, the offset value is added to or subtracted from the system time.

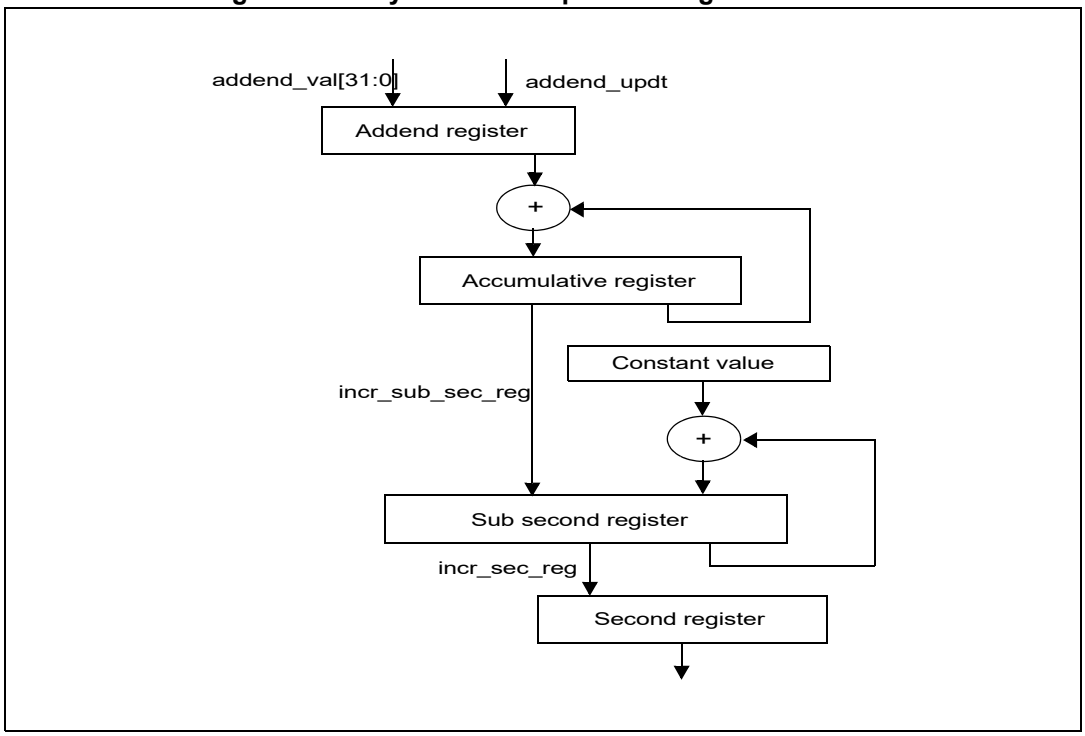
In the fine correction method, the frequency drift of a slave clock (clk_ptp_ref_i) with respect to the master clock (as defined in IEEE 1588-2002) is corrected over a period of time

instead of in one clock, as in coarse correction. This helps maintain linear time and does not introduce drastic changes (or a large jitter) in the reference time between PTP Sync message intervals. In this method, an accumulator sums up the contents of the Addend register, as shown in *Figure 1204*. The arithmetic carry that the accumulator generates is used as a pulse to increment the system time counter. The accumulator and the addend are 32-bit registers. The accumulator acts as a high-precision frequency multiplier or divider.

Note: You must connect a PTP clock with a frequency higher than the frequency required for the specified accuracy.

This algorithm is shown in *Figure 1204*.

Figure 1204. System Time Update Using Fine Method



The System Time Update logic requires a 50-MHz clock frequency to achieve 20-ns accuracy. The frequency division is the ratio of the reference clock frequency to the required clock frequency. For example, if the reference clock (clk_ptp_ref_i) is 66 MHz, this ratio is calculated as 66 MHz / 50 MHz = 1.32. Therefore, the default addend value to be set in the register is 2^{32} / 1.32, 0xC1F07C1F.

If the reference clock drifts lower, for example, to 65 MHz, the ratio is 65 / 50, or 1.3 and the value to set in the addend register is 2^{32} / 1.30, or 0xC4EC4EC4. If the clock drifts higher, for example, to 67 MHz, the addend register must be set to 0xBF0B7672. When the clock drift is nil, the default addend value of 0xC1F07C1F (2^{32} / 1.32) must be programed.

In 5-12, the constant value used to accumulate the sub-second register is decimal 43, which achieves an accuracy of 20 ns in the system time (in other words, it is incremented in 20 ns steps). Two different methods are used to update the System Time register depending on the configuration.

The software must calculate the drift in frequency based on the Sync messages and accordingly update the Addend register.

Initially, the slave clock is set with FreqCompensationValue0 in the Addend register. This value is as follows:

$FreqCompensationValue_0 = 2^{32} / FreqDivisionRatio$

If MasterToSlaveDelay is initially assumed to be the same for consecutive Sync messages, the algorithm given in this section must be applied. After a few Sync cycles, frequency lock occurs. The slave clock can then determine a precise MasterToSlaveDelay value and re-synchronize with the master using the new value.

The algorithm is as follows:

- At time $MasterSyncTime_n$ the master sends the slave clock a Sync message. The slave receives this message when its local clock is $SlaveClockTime_n$ and computes $MasterClockTime_n$ as
  $MasterClockTime_n = MasterSyncTime_n + MasterToSlaveDelay_n$
- The master clock count for current Sync cycle, $MasterClockCount_n$ is
  $MasterClockCount_n = MasterClockTime_n – MasterClockTime_{n-1}$ (assuming that MasterToSlaveDelay is the same for Sync cycles n and n – 1)
- The slave clock count for current Sync cycle, $SlaveClockCount_n$ is
  $SlaveClockCount_n = SlaveClockTime_n – SlaveClockTime_{n-1}$
- The difference between master and slave clock counts for current Sync cycle, $ClockDiffCount_n$ is
  $ClockDiffCount_n = MasterClockTime_n – SlaveClockTime_n$
- The frequency-scaling factor for slave clock, $FreqScaleFactor_n$ is
  $FreqScaleFactor_n = (MasterClockCount_n + ClockDiffCount_n) / SlaveClockCount_n$
- The frequency compensation value for Addend register, $FreqCompensationValue_n$ is
  $FreqCompensationValue_n = FreqScaleFactor_n * FreqCompensationValue_{n-1}$

In theory, this algorithm achieves lock in one Sync cycle. However, it may take several cycles, because of changing network propagation delays and operating conditions. This algorithm is self-correcting. If the slave clock is initially set to an incorrect value from the master, the algorithm corrects it at the cost of more Sync cycles.

56.3.9.6 Transmit Path Functions

The MAC captures a timestamp when the Start Packet Delimiter (SFD) of a packet is sent on the MII interface. The packets, for which you want to capture timestamps, can be controlled on per-packet basis. Each Transmit packet can be marked to indicate whether a timestamp should be captured for it.

The MAC does not process the transmitted packets to identify the PTP packets. You need to specify the packets for which you want to capture timestamps. You can do this by using the control bits in the Transmit descriptor. The MAC returns the timestamp to the software inside the corresponding Transmit descriptor, thus connecting the timestamp automatically to the specific PTP packet. The 64-bit timestamp information is written to the TDES0 and TDES1 fields. The TDES0 field holds the 32 least significant bits of the timestamp.

56.3.9.7 Receive Path Functions

The MAC can be programed to capture the timestamp of all packets received on the MII interface or to process packets to identify the valid PTP messages. You can control the

snapshot of the time to be sent to the application by using the following options of the MAC_Timestamp_Control register.

- Enable snapshot for all packets
- Enable snapshot for IEEE 1588 version 1 or version 2 timestamp
- Enable snapshot for PTP packets transmitted directly over Ethernet or UDP-IP-Ethernet
- Enable timestamp snapshot for the received packet for IPv4 or IPv6
- Enable timestamp snapshot only for EVENT messages (SYNC, DELAY_REQ, PDELAY_REQ, or PDELAY_RESP)
- Enable the node to be a master or slave and select the snapshot type

This feature controls the type of messages for which snapshots are taken.

The DMA returns the timestamp to the software inside the corresponding Receive Descriptor. The extended status, containing the timestamp message status and the IPC status, is written in normal descriptor RDES1 and the snapshot of the timestamp is written in RDES0 and RDES1 fields of context descriptor. The RDES0 field holds the 32 least significant bits of the timestamp.

Note: The Ethernet module also supports the PTP messages over VLAN packets.

56.3.9.8 Timestamp Correction

According to the IEEE 1588 specification, a timestamp must be captured when the message timestamp point (leading edge of the first bit of the octet immediately following the Start Frame Delimiter octet) crosses the boundary between the node and the network. Because the reference timing source (the PTP clock ***clk_ptp_ref_i***) is different from the MAC Tx or Rx clock, the captured timestamp must be corrected for latency issues because of synchronization. In addition, latency issues between the internal snapshot point and the recommended capture point (the boundary between the node and the network), must also be corrected.

56.3.9.8.1 Ingress Correction

In the Receive side the timestamp captured at the internal snapshot point is corrected for latency and synchronization by adding the correction value (Ingress Correction Value) programed in the Ingress Correction register. The value that needs to be programed in the ingress correction register is calculated as mentioned below:

The timestamp correction because of synchronization is compensated by adding INGRESS_SYNC_CORR to the synchronized timestamp value.

```
INGRESS_SYNC_CORR = -(2*PTP_CLK_PER)
```

The latency correction between the message timestamp point and the internal timestamp snapshot point is done by subtracting the latency value (INGRESS_LATENCY) with the captured timestamp.

```
Ingress Correction = INGRESS_SYNC_CORR - INGRESS_LATENCY
```

Ingress correction is performed by programming the TSIC field in the MAC Timestamp Ingress correction register. The ingress correction is always negative and has a unit of nanoseconds. The value is represented in complement form.

When TSCTRLSSR bit in MAC_Timestamp_Control register is set, this has an accuracy of 1 ns.

It is represented by setting bit 0 to 30 and bits 1:31 containing 10^9 - <ingress_correction_value> represented in binary. For example, if the required correction value is -5 ns, then the programed value is 0xBB9A_C9FB.

When TSCTRLSSR bit in MAC_Timestamp_Control register is reset, this has an accuracy of ~0.466 ns. It is represented by setting bit 0 to 30 and bits 1:31 containing 2^31 - <ingress_correction_value> represented in binary.

56.3.9.8.2 Egress Correction

In the Transmit side the timestamp captured at the internal snapshot point is corrected for latency and synchronization by adding the correction value (Egress Correction Value) programed in the Egress Correction register. The value that needs to be programed in the egress correction register as mentioned below:

The timestamp correction because of synchronization is compensated by adding EGRESS_SYNC_CORR to the synchronized timestamp value.

When Enable one step timestamp feature is selected,

```
EGRESS_SYNC_CORR = (1*PTP_CLK_PER + 4*TX_CLK_PER)
```

Otherwise,

```
EGRESS_SYNC_CORR = -(2*PTP_CLK_PER)
```

The egress latency correction between the recommended capture point and the internal timestamp snapshot point is done by adding the latency value (EGRESS_LATENCY) with the captured timestamp.

```
Egress Correction = EGRESS_SYNC_CORR + EGRESS_LATENCY
```

Egress correction is performed by programming the TSEC field in the MAC Timestamp Egress correction register. The egress correction can be positive or negative and has a unit of nanoseconds. Negative values are represented in complement form.

When TSCTRLSSR bit in MAC_Timestamp_Control register is set, this has an accuracy of 1 ns.

If the correction is positive, it is represented by setting bit 0 to 31 and bits 1:31 containing <egress_correction_value> represented in binary. The value must not exceed 0x3B9A_C9FF. If the correction is negative, it is represented by setting bit 0 to 30 and bits 1:31 containing 10^9 - <egress_correction_value> represented in binary.

For example, if the required correction value is -5 ns, then the programed value should be 0xBB9A_C9FB

When TSCTRLSSR bit in MAC_Timestamp_Control register is reset, this has an accuracy of ~0.466 ns. If the correction is positive, it is represented by setting bit 0 to 31 and bits 1:31 containing <egress_correction_value> represented in binary. The maximum value is 0x7FFF_FFFF. If the correction is negative, it is represented by setting bit 0 to 30 and bits 1:31 containing 2^31 - <egress_correction_value> represented in binary.

56.3.9.9 Frequency Range of Reference Timing Clock

The timestamp information is transferred across asynchronous clock domains, that is, from the MAC clock domain to the application clock domain. Therefore, a minimum delay is required between two consecutive timestamp captures. This delay is 4 clock cycles of MII and 3 clock cycles of PTP clocks. If the delay between two timestamp captures is less than this delay, the MAC does not take a timestamp snapshot for the second packet.

56.3.9.9.1 Maximum PTP Clock Frequency

The maximum PTP clock frequency is limited by the maximum resolution of the reference time (1 ns resulting in 1 GHz) and the timing constraints achievable for logic operating on the PTP clock. In addition, the resolution or granularity of the reference time source determines the accuracy of the synchronization. Therefore, a higher PTP clock frequency gives better system performance.

56.3.9.9.2 Minimum PTP Clock Frequency

The minimum PTP clock frequency depends on the time required between two consecutive SFD bytes and the time taken for synchronizing the time to the MII clock domain. This relationship is given in the following equation:

Equation 33

3 * PTP clock period + 4 * MII clock period <= Minimum gap between two SFDs

The MII clock frequency is fixed by IEEE specification. Therefore, the minimum PTP clock frequency required for proper operation depends on the operating mode and operating speed of the MAC as shown in *Table 1258*.

Note: It is recommended that you use a clock that has constant frequency, preferably a divided application clock.
When IEEE 1588 timestamp feature is enabled with internal timestamp, use a PTP clock frequency which is greater than 5 MHz. This is because the 8-bit MAC_Sub_Second_Increment register limits the minimum PTP frequency that can be used to ~4 MHz.

Table 1258. Minimum PTP Clock Frequency Example

Mode	Minimum Gap Between Two SFDs	Minimum PTP Frequency with Internal Timestamp
10 Mbps full duplex	168 MII clocks (128 clocks for a 64-byte packet + 24 clocks of min IFG + 16 clocks of preamble)	5 MHz
10 Mbps half duplex	48 MII clocks (8 clocks for a JAM pattern sent just after SFD because of collision + 24 IFG + 16 preamble)	5 MHz
100 Mbps full duplex	168 MII clocks (128 clocks for a 64-byte packet + 24 clocks of min IFG + 16 clocks of preamble)	5 MHz
100 Mbps half duplex	48 MII clocks (8 clocks for a JAM pattern sent just after SFD because of collision + 24 IFG + 16 preamble)	5 MHz

56.3.9.10 PTP Processing and Control

Table 1259 shows the common message header for the PTP messages. This format is taken from the IEEE 1588-2008.

Table 1259. Message Format Defined in IEEE 1588-2008

Bits								Octets	Offset
0	1	2	3	4	5	6	7		
transportSpecific				messageType				1	0
Reserved				versionPTP				1	1
messageLength								2	2
domainNumber								1	4
Reserved								1	5
flagField								2	6
correctionField								8	8
Reserved								4	16
sourcePortIdentity								10	20
sequenceId								2	30
controlField(1)								1	32
logMessageInterval								1	33

1. ControlField is used in version 1. In version 2, messageType field is used for detecting different message types.

There are some fields in the Ethernet payload that you can use to detect the PTP packet type and control the snapshot to be taken. These fields are different for the following PTP packets:

- PTP Packets over IPv4Packets over IPv4
- PTP Frames over IPv6Frames over IPv6
- PTP Packets over EthernetPackets over Ethernet

56.3.9.10.1 PTP Packets over IPv4

Table 1260 provides information about the fields that are matched to control snapshot for the PTP packets sent over UDP over IPv4 for IEEE 1588 version 1 and 2. The octet positions for the tagged packets are offset by 4. This is based on the IEEE 1588-2008, *Annex D* and the message format defined in *Table 1260*.

Table 1260. IPv4-UDP PTP Packet Fields Required for Control and Status

Field Matched	Octet Position	Matched Value	Description
MAC Packet Type	12, 13	0x0800	IPv4 datagram
IP version and Header Length	14	0x45	IP version is IPv4
Layer 4 Protocol	23	0x11	UDP
IP Multicast Address (IEEE 1588 version 1)	30, 31, 32, 33	0xE0, 0x00, 0x01, 0x81 (or 0x82 or 0x83 or 0x84)	Multicast IPv4 addresses allowed: 224.0.1.129 224.0.1.130 224.0.1.131 224.0.1.132
IP Multicast Address (IEEE 1588 version 2)	30, 31, 32, 33	0xE0, 0x00, 0x01, 0x81 (Hex) 0xE0, 0x00, 0x00, 0x6B (Hex)	PTP Primary multicast address: 224.0.1.129 PTP Pdelay multicast address: 224.0.0.107
UDP Destination Port	36, 37	0x013F, 0x0140	0x013F: PTP event message(1) 0x0140: PTP general messages
PTP Control Field (IEEE 1588 version 1)	74	0x00, 0x01, 0x02, 0x03, 0x04	0x00: SYNC 0x01: Delay_Req 0x02: Follow_Up 0x03: Delay_Resp 0x04: Management
PTP Message Type Field (IEEE 1588 version 2)	42 (nibble)	0x0, 0x1, 0x2, 0x3, 0x8, 0x9, 0xB, 0xC, 0xD	0x0: SYNC 0x1: Delay_Req 0x2: Pdelay_Req 0x3: Pdelay_Resp 0x8: Follow_Up 0x9: Delay_Resp 0xA: Pdelay_Resp_Follow_Up 0xB: Announce 0xC: Signaling 0xD: Management
PTP Version	43 (nibble)	0x1 or 0x2	0x1: Supports PTP version 1 0x2: Supports PTP version 2

1. PTP event messages are SYNC, Delay_Req (IEEE 1588 version 1 and 2) or Pdelay_Req, Pdelay_Resp (IEEE 1588 version 2 only)

56.3.9.10.2 PTP Frames over IPv6

Table 1261 provides information about the fields that are matched to control the snapshots for the PTP packets sent over UDP over IPv6 for IEEE 1588 version 1 and 2. The octet positions for the tagged packets are offset by 4. This is based on the IEEE 1588-2008, *Annex D* and the message format defined in *Table 1261*.

56.3.9.10.3 PTP Packets over Ethernet

Table 1261. IPv6-UDP PTP Packet Fields Required for Control and Status

Field Matched	Octet Position	Matched Value	Description
MAC Packet Type	12, 13	0x86DD	IP datagram
IP Version	14 (Bits [0:3])	0x6	IP version is IPv6
Layer 4 Protocol	20(1)	0x11	UDP
PTP Multicast Address	38 – 53	FF0x:0:0:0:0:0:0:181 (Hex) FF02:0:0:0:0:0:0:6B (Hex)	PTP Primary multicast address: FF0x:0:0:0:0:0:0:0:181 (Hex) PTP Pdelay multicast address: FF02:0:0:0:0:0:0:0:6B (Hex)
UDP Destination Port	56, 57(1)	0x013F, 0x140	0x013F: PTP event message 0x0140: PTP general messages
PTP Control Field (IEEE 1588 version 1)	94(1)	0x00, 0x01, 0x02, 0x03, or 0x04	0x00: SYNC 0x01: Delay_Req 0x02: Follow_Up 0x03: Delay_Resp 0x04: Management (version1)
PTP Message Type Field (IEEE 1588 version 2)	62(1) (nibble)	0x0, 0x1, 0x2, 0x3, 0x8, 0x9, 0xB, 0xC, or 0xD	0x0: SYNC 0x1: Delay_Req 0x2: Pdelay_Req 0x3: Pdelay_Resp 0x8: Follow_Up 0x9: Delay_Resp 0xA: Pdelay_Resp_Follow_Up 0xB: Announce 0xC: Signaling 0xD: Management
PTP Version	63 (nibble)	0x1 or 0x2	0x1: Supports PTP version 1 0x2: Supports PTP version 2

1. The Extension Header is not defined for PTP packets.

Table 1262 provides information about the fields that are matched to control the snapshots for the PTP packets sent over Ethernet for IEEE 1588 version 1 and 2. The octet positions for the tagged packets are offset by 4. This is based on the IEEE 1588-2008, *Annex D* and the message format defined in Table 5-12 Message Format Defined in IEEE 1588-2008n 5-12.

56.3.9.11 One-Step Timestamp

Table 1262. Ethernet PTP Packet Fields Required for Control and Status

Field Matched	Octet Position	Matched Value	Description
MAC Destination Multicast Address(1)	0–5	01-1B-19-00-00-00 01-80-C2-00-00-0E	All PTP messages can use any of the following multicast addresses(2): 01-1B-19-00-00-00 01-80-C2-00-00-0E(3)
MAC Packet Type	12, 13	0x88F7	PTP Ethernet packet
PTP Control Field (IEEE 1588 version 1)	46	0x00, 0x01, 0x02, 0x03, or 0x04	0x00: SYNC 0x01: Delay_Req 0x02: Follow_Up 0x03: Delay_Resp 0x04: Management
PTP Message Type Field (IEEE 1588 version 2)	14 (nibble)	0x0, 0x1, 0x2, 0x3, 0x8, 0x9, 0xB, 0xC, or 0xD	0x0: SYNC 0x1: Delay_Req 0x2: Pdelay_Req 0x3: Pdelay_Resp 0x8: Follow_Up 0x9: Delay_Resp 0xA: Pdelay_Resp_Follow_Up 0xB: Announce 0xC: Signaling 0xD: Management
PTP Version	15 (nibble)	0x1 or 0x2	0x1: Supports PTP version 1 0x2: Supports PTP version 2

1. The unicast address match of destination addresses (DA), programed in MAC address 0 to 31, is used if the TSENMACADDR bit of MAC_Timestamp_Control register is set.
2. IEEE 1588-2008, *Annex F*
3. The MAC does not consider the PTP version 1 messages with Peer delay multicast address (01-80-C2-00-00-0E) as valid PTP messages.

The Ethernet module supports the one-step timestamp feature. When this feature is enabled, the MAC identifies the offset in the packet and inserts the timestamp received from the application at that offset.

You can enable the one-step timestamp feature for a packet by setting Bit 11 (OSTC) in ATI Control Word. The inserted timestamp consists of 80-bits, the 64-bit TSSL and TSSH received from the application and the content of higher 16 bits of Higher Timestamp Word Register.

Note: The one-step timestamp feature is supported only for the PTP over Ethernet packets. It is not supported for PTP over IPv4/IPv6 packets because the timestamp insertion happens in the MAC and the IP payload checksum needs to be updated. However, the checksum computation happens in the MTL.

56.3.9.11.1 MAC Transmit PTP Mode

Depending upon the type of message and its mode, the MAC updates the following fields of Transmit PTP packets:

- Correction Field in the PTP header of messages
- OriginTimestamp in SYNC, Delay_Req, and Pdelay_Req messages

Table 1263 shows how the PTP mode is selected based on the settings of SNAPTYPSEL, TSMSTRENA, and TSEVNTENA bits of the *Section 56.2.140: Timestamp Control Register (MAC_TIMESTAMP_CONTROL)* and the fields that are updated for the incoming PTP packets based on the message type in that mode, during the one-step timestamping operation.

Table 1263. MAC Transmit PTP Mode and One-Step Timestamping Operation

Programming			Mode	Per Packet Control(1)			Messages Processed on Tx
SNAPTYPSEL	TSMSTRENA	TSEVNTENA		TTSE(2)	OSTC(3)	TTS(4)	
X	X	X	N/A	1	X	X	Timestamp is captured and returned to application
X	X	X	N/A	X	0	X	OST operation is not performed (PTP packet is not modified)
2'b00	X	0	End-to-end transparent	0	1	Ingress TS	Sync (correction field for residence time and Ingress Asym cor)
							Delay_Req (correction field for residence time and Egress Asym Cor)
2'b00	0	1	Ordinary or Boundary Slave	1	1	X	Delay_Req (originTimestamp field) Delay_Req (correction field for Egress Asym cor)
2'b00	1	1	Ordinary or Boundary Master	0	1	X	Sync (originTimestamp field) Sync (correction field for sub-nanosecond cor)
2'b01	X	0	End-to-End Transparent with support for peer delay mechanism	0	1	Ingress TS	Sync (correction field for residence time and Ingress Asym cor)
						Ingress TS	Pdelay_Req (correction field for residence time and Egress Asym Cor)
						Ingress TS	Pdelay_Resp (correction field for residence time and Ingress Asym Cor)

Table 1263. MAC Transmit PTP Mode and One-Step Timestamping Operation (continued)

Programming			Mode	Per Packet Control[1]			Messages Processed on Tx
SNAPTYPSEL	TSMSTRENA	TSEVNTENA		TTSE[2]	OSTC[3]	TTS[4]	
2'b01	0	1	Ordinary or Boundary Slave with support for peer delay mechanism or Peer to Peer Transparent	0	1	Ingress TS	Sync (correction field for residence time and Ingress Asym cor) (applicable only for Peer to Peer transparent clock operation)
				1	1	X	Delay_Req (originTimestamp field) Delay_Req (correction field for Egress Asym cor)
				1	1	X	Pdelay_Req (originTimestamp field) Pdelay_Req (correction field for Egress Asym Cor)
				0	1	Ingress TS for Pdelay_Req	Pdelay_Resp (correction field for turnaround time and Ingress Asym Cor)
2'b01	1	1	Ordinary or Boundary Master with support for peer delay mechanism	0	1	X	Sync (originTimestamp field) Sync (correction field for sub-nanosecond cor)
				1	1	X	Pdelay_Req (originTimestamp field) Pdelay_Req (correction field for Egress Asym Cor)
				0	1	Ingress TS for Pdelay_Req	Pdelay_Resp (correction field for turnaround time and Ingress Asym Cor)
2'b10	X	X	End-to-End Transparent	0	1	Ingress TS	Sync (correction field for residence time and Ingress Asym cor)
						Ingress TS	Delay_Req (correction field for residence time and Egress Asym Cor)
2'b11	X	X	Peer-to-Peer Transparent	0	1	Ingress TS	Sync (correction field for residence time and Ingress Asym cor)
				1	1	X	Pdelay_Req (originTimestamp field) Pdelay_Req (correction field for Egress Asym Cor)
				0	1	Ingress TS for Pdelay_Req	Pdelay_Resp (correction field for turnaround time and Ingress Asym Cor)

1. The per packet control values provided here are the recommended settings used by devices in typical PTP operation, for the programed mode.
2. TTSE represents TTSE bit of transmit descriptor in EQOS-AXI configuration, TTSE bit in TX control word in EQOS-MTL configuration, mti_ena_timestamp_i signal in EQOS-CORE configuration. The TTSE function is independent of the OST function and the programed operation mode for OST. The MAC captures and returns the timestamp when the TTSE bit is set.
3. OSTC represents OSTC bit of transmit descriptor in EQOS-AXI configuration, OSTC bit in TX control word in EQOS-MTL configuration, mti_ost_en_i signal in EQOS-CORE configuration.

4. TTS represents the timestamp value provided in the TTSH, TTSL fields of transmit descriptor in EQOS-AXI configuration, timestamp provided in the TTSH, TTSL fields in the TX control word in EQOS-MTL configuration, mti_ost_i signal in EQOS-CORE configuration.
Residence time/ turnaround time is calculated as the difference between the captured timestamp (egress timestamp) and the ingress timestamp.
When sub-nanosecond feature is enabled, residence time calculation includes sub-nanosecond accuracy.
Clocks supporting peer delay mechanism do not use delay request or response, but it is included in OST for flexibility.

56.3.10 IPv4 ARP Offload

The Ethernet module supports the Address Recognition Protocol (ARP) Offload for IPv4 packets. The Ethernet module generates the ARP reply packets for appropriate ARP request packets**.** The ARP packet for IPv4 is L2 layer packet with Length/Type of 0x0806.

The ARP offloading process is as follows:

1. The MAC receiver gets an ARP request if the Target Protocol Address of request matches the IPv4 address programed in the L3 register of the MAC
2. The MAC generates an ARP reply packet
3. The MAC copies the Sender Hardware Address field in the ARP request to the following fields:
   – DA field of the Ethernet packet header
   – Target Hardware Address field of the ARP reply packet
4. The MAC copies the Sender Protocol Address field in the ARP request to the Target Protocol Address field in the ARP reply packet
5. The MAC places its MAC address in the following fields:
   – SA field of the Ethernet packet header
   – Sender Hardware Address field of the ARP reply packet
6. The MAC copies the Target Protocol Address field in the ARP request to the Sender Protocol Address field in the ARP reply packet
7. The MAC sets the opcode field in ARP reply packet to 2 indicating ARP reply
8. The MAC recalculates the CRC and performs padding for generated ARP reply packet
9. The MAC transmitter sends the ARP reply

The MAC processes only one ARP request at a time. It does not store the fields of multiple ARP requests. If the MAC receives an ARP request when it is already processing an earlier ARP request, the MAC does not generate the ARP reply for new ARP request. The MAC forwards the new ARP request packet to application with ARP Reply Not Generated (Bit 21) status bit set. However, in power-down mode, if the MAC receives an ARP request when it is already processing an earlier ARP request, the MAC drops the new ARP request.

If the Disable CRC check bit of the MAC Extension Configuration bit is set, then the MAC does not check for valid CRC of an ARP request Packet. The ARP request Packet must always have a valid CRC. It can generate an ARP response packet if the other conditions are valid.

56.3.11 Memory Access in Debug Mode

The Ethernet module supports the debug access to the FIFO memory. When you enable this feature, you must program the entire selected Tx memory to Tx Queue 0 and Rx memory to Rx Queue 0. In addition, the offload features such as Checksum Offload Engine and IEEE timestamp should not be enabled with this feature.

When this feature is enabled, the MAC insert the pad (if required) and computed CRC in the Transmit packet. The FIFOs operate in the store-and-forward mode.

56.3.11.1 Slave Mode Access

You can use this mode to send or receive a packet through a slave interface. The following sections describe how to transmit or receive a packet through a slave interface.

56.3.11.1.1 Transmitting Packets through a Slave Interface

To write a packet to the Tx memory through a slave interface, complete the steps given in *Table 1264*.

Table 1264. Steps to Write a Packet to Tx memory through a Slave Interface

Register Name	Step Number and Description
MTL_DBG_CTL	Select the FIFO for slave access by using the DBGMOD and FDBGEN fields.
MTL_FIFO_DEBUG_DATA	Write the Packet control word.
MTL_DBG_CTL	Complete the following: a. Set the PKTSTATE field to Control field. b. Set the FIFOSEL field to Tx FIFO. c. Set the FIFOWREN bit. d. Reset the FIFORDEN bit.
MTL_DBG_STS	Read the FIFOBUSY bit and wait till this bit is low. When this bit is low, read the LOCR field to know the number of free locations.
MTL_FIFO_DEBUG_DATA	Write the SOP, the first word of the packet data.
MTL_DBG_CTL	Complete the following steps: a. Set the PKTSTATE field to SOP. b. Set the FIFOSEL field to Tx FIFO. c. Set the FIFOWREN bit.
MTL_DBG_STS	Read the FIFOBUSY bit and wait till this bit is low. When this bit is low, read the LOCR field to know the number of free locations.
MTL_FIFO_DEBUG_DATA	Write subsequent data words.
MTL_DBG_CTL	Complete the following steps: a. Set the PKTSTATE field to Packet Data. b. Set the FIFOSEL field to Tx FIFO. c. Set the FIFOWREN bit.
MTL_DBG_STS	Read the FIFOBUSY bit and wait till this bit is low. When this bit is low, read the LOCR field (if required).
MTL_DBG_CTL and MTL_DBG_STS	Repeat Write subsequent data words. and Read the FIFOBUSY bit and wait till this bit is low. When this bit is low, read the LOCR field (if required). till the penultimate word of the packet data.

Table 1264. Steps to Write a Packet to Tx memory through a Slave Interface (continued)

Register Name	Step Number and Description
MTL_FIFO_DEBUG_DATA	Write the last data word in the FDBGDATA field.
MTL_DBG_CTL	Complete the following steps: a. Set the PKTSTATE field to EOP. b. Set the FIFOSEL field to Tx FIFO. c. Set the FIFOWREN bit. d. Set the number of valid bytes in the field.
MTL_DBG_STS	Read the FIFOBUSY bit and wait till this bit is low. When this bit is low, read the LOCR field to know the number of free locations.

After Read the FIFOBUSY bit and wait till this bit is low. When this bit is low, read the LOCR field to know the number of free locations. in *Table 1264*, the core automatically initiates the packet transmission. The status of the packet transmission can be obtained as explained in *Table 1264*. To get the status of a Tx packet, complete the steps given in *Table 1265*.

Table 1265. Steps to Get the Status of a Tx Packet

Register Name	Step Number and Description
MTL_DBG_STS	Wait for the STSI interrupt bit to set.
MTL_DBG_CTL	Complete the following steps: a. Set the FIFOSEL field to Tx Status FIFO. b. Reset the FIFWREN bit. c. Set the FIFORDEN bit.
MTL_DBG_CTL	Read the FIFOBUSY bit and wait till this bit is low.
MTL_FIFO_DEBUG_DATA	Read the FDBGDATA field.

56.3.11.1.2 Receiving Packets through a Slave Interface

To read a packet from the Rx memory through a slave interface, complete the steps given in *Table 1266*.

Table 1266. Steps to Read a Packet from Rx memory through a Slave Interface

Register Name	Step Number and Description
MTL_DBG_CTL	Select the FIFO for slave access by using the DBGMOD and FDBGEN fields.
MTL_DBG_STS	Wait for the PKTI interrupt.

Table 1266. Steps to Read a Packet from Rx memory through a Slave Interface (continued)

Register Name	Step Number and Description
MTL_DBG_CTL	Complete the following: a. Set the FIFOSEL field to Rx FIFO. b. Reset the FIFOWREN bit. c. Set the FIFORDEN bit.
MTL_DBG_STS	Read the FIFOBUSY bit and wait till this bit is low.
MTL_DBG_CTL	Read the PKTSTATE and fields.
MTL_FIFO_DEBUG_DATA	Read the data in the FDBGDATA field.
MTL_DBG_STS	Read the LOCR field to know the number of available locations.
MTL_DBG_STS, MTL_DBG_CTL, MTL_FIFO_DEBUG_DATA	Repeat Read the FIFOBUSY bit and wait till this bit is low. through Read the LOCR field to know the number of available locations. till the complete packet, along with the Rx status, is received. **Note:** The format of the received packet is as follows First (One or multiple words) Last Status Word (One Word. Acts as delimiter between the status and data) Frame Data Word (multiple) Last Frame Data Word (One Word. Acts as a delimited among subsequent packets on the received Queue)

56.3.11.2 Debug Mode Access

Debug Mode enables you to do both the following operations:

- Write and Read back the Tx Queue
- Read and Write to the Tx Queue

This mode is useful for Memory BIST applications.

56.3.11.2.1 Writing to FIFO Memory in Debug Mode

To write to FIFO memory in the debug mode, complete the steps given in *Table 1267*.

Table 1267. Steps to Write to FIFO memory in Debug Mode

Register Name	Step Number and Description
MTL_DBG_CTL	Select the FIFO for debug access by using the DBGMOD and FDBGEN fields.
MTL_FIFO_DEBUG_DATA	Write the data in the FDBGDATA field.
MTL_DBG_CTL	Complete the following: – Set the FIFOSEL field to Tx FIFO, Rx FIFO, or TSO FIFO. – Reset the FIFORDEN bit. – Set the FIFOWREN bit.

Table 1267. Steps to Write to FIFO memory in Debug Mode (continued)

Register Name	Step Number and Description
MTL_DBG_STS	Read the FIFOBUSY bit and wait till this bit is low. When this bit is low, read the LOCR field (if required).
MTL_DBG_CTL - MTL_DBG_STS	Repeat Complete the following: and Read the FIFOBUSY bit and wait till this bit is low. When this bit is low, read the LOCR field (if required). twice to write one complete data word in 64-bit mode.

56.3.11.2.2 Reading from FIFO Memory in Debug Mode

To read from FIFO memory in the debug mode, complete the steps given in *Table 1268*.

Table 1268. Steps to Read from FIFO Memory in Debug Mode

Register Name	Step Number and Description
MTL_DBG_CTL	Select the FIFO for debug access by using the DBGMOD and FDBGEN fields. Complete the following: a. Set the FIFOSEL field to Tx FIFO, Rx FIFO, or TSO FIFO. b. Reset the FIFOWEN bit. c. Set the FIFORDEN bit.
MTL_DBG_STS	Read the FIFOBUSY bit and wait till this bit is low. When this bit is low, read the LOCR field (if required).
MTL_FIFO_DEBUG_DATA	Read the data in the FDBGDATA field.
MTL_DBG_STS	Read the LOCR field (if required).
MTL_DBG_CTL, MTL_DBG_STS, MTL_FIFO_DEBUG_DATA	Repeat Complete the following: and Read the LOCR field (if required). twice to read one complete data word in 64-bit mode.

56.3.12 Loopback

The MAC supports Loopback of transmitted packets to its receiver. By default, the MAC Loopback function is disabled. You can enable this feature by programming the LM bit of the MAC_Configuration register.

You can enable loopback for all PHY interfaces. The data is always looped back on the MII interface irrespective of which PHY interface is selected. The loopback data is also passed through the corresponding interface block. The following are some guidelines for using the loopback mode:

- Enable loopback only with the full-duplex mode. In half-duplex mode, the carrier sense signal (crs) or collision (col) signal inputs get sampled which may result into issues such as packet dropping
- If the loopback mode is enabled without connecting a PHY chip, you should externally generate the Tx and Rx clocks and provide these clocks to the MAC
- Do not loop back big packets. Big packets may get corrupted in the loopback FIFO
- The MAC does not process ARP or PMT packets that are looped back

56.3.13 MAC Flow Control

This section describes the flow control for Transmit and Receive paths.

The Transmit Flow Control involves transmitting Pause packets in full-duplex mode and backpressure in half-duplex mode to control the flow of packets from the remote end.

In the Receive path, the Flow Control is functional only in the full-duplex mode. The MAC receiver detects or processes the Pause packets and responds to such packets by stopping the MAC transmitter.

56.3.13.1 Transmit Flow Control

The Transmit Flow Control involves transmitting Pause packets in full-duplex mode and backpressure in half-duplex mode to control the flow of packets from the remote end.

The Transmit Flow Control is enabled independently for each Rx queue when TFE bit is set in MAC_Q0_Tx_Flow_Ctrl /MAC_Q1_TX_Flow_Ctrl register.

SW can request the MAC to send a Pause packet or initiate backpressure by setting the FCB_BPA bit in the corresponding MAC_Q0_Tx_Flow_Ctrl /MAC_Q1_TX_Flow_Ctrl register.

HW can automatically trigger TX flow control based on the Rx Queue Threshold**.** The HW flow control operation of the MAC is enabled when the EHFC bit of the corresponding MTL_RxQ*n*_Operation_Mode register is set. The flow control signal to the MAC is asserted when the fill level of the Rx queue crosses the threshold configured in the RFA field of the MTL_RxQ*n*_Operation_Mode register. This flow control signal is de-asserted when the fill-level of the queue falls below the threshold configured in the RFD field.

Table 1269, describes the flow control in the Tx path for Queue 0 based on the setting of the following bits:

- EHFC bit of the MTL_RxQ0_Operation_Mode register
- TFE bit of the MAC_Q0_Tx_Flow_Ctrl register
- DM bit of the MAC_Configuration register
- RFA field of the MTL_RxQ0_Operation_Mode register
- RFD field of the MTL_RxQ0_Operation_Mode register
- FCB_BPA bit of the MAC_Q0_Tx_Flow_Ctrl

Flow control is similar for Queue1.

Table 1269. TX MAC Flow Control

EHFC	TFE	DM	Description
x	0	x	The MAC transmitter does not perform the flow control or backpressure operation.
0	1	0	The MAC transmitter performs backpressure when FCB_BPA of the MAC_Q0_Tx_Flow_Ctrl register is set.
1	1	0	The MAC transmitter performs backpressure when FCB_BPA of the MAC_Q0_Tx_Flow_Ctrl register is set. In addition, the MAC TX performs backpressure when the Rx Queue level crosses the threshold set by the RFA field of the MTL_RxQ0_Operation_Mode register.

Table 1269. TX MAC Flow Control (continued)

EHFC	TFE	DM	Description
0	1	1	The MAC transmitter sends the pause packet when FCB_BPA of the MAC_Q0_Tx_Flow_Ctrl register is set.
1	1	1	The MAC transmitter sends the pause packet when FCB_BPA of MAC_Q0_Tx_Flow_Ctrl register is set. In addition, the MAC TX sends a pause packet when the Rx Queue level crosses the threshold set by the RFA field of the MTL_RxQ0_Operation_Mode register.

56.3.13.1.1 Flow Control in Full-Duplex Mode

In full-duplex mode, the Ethernet module uses IEEE 802.3x Pause packets for flow control. *Table 1270*, describes the fields of a Pause packet. When the FCB_BPA bit is set, the MAC generates and transmits a single Pause packet. If the FCB bit is set again after the Pause packet transmission is complete, the MAC sends another Pause packet irrespective of whether the pause time is complete or not. To extend the pause or terminate the pause prior to the time specified in the previously-transmitted Pause packet, the application should program the Pause Time register with appropriate value and then again set the FCB bit.

Table 1270. Pause Packet Fields

Field	Description
DA	Contains the special multicast address
SA	Contains the MAC address 0
Type	Contains 8808
MAC Control opcode	Contains 0001 for IEEE 802.3x Pause Control packets
PT	Contains Pause time specified in the PT field of the MAC_Q0_Tx_Flow_Ctrl /MAC_Q1_TX_Flow_Ctrl register

56.3.13.1.2 Flow Control in Half-Duplex Mode

In half-duplex mode, the MAC uses the deferral mechanism for the flow control (backpressure). When the application requests to stop receiving packets, the MAC sends a JAM pattern of 32 bytes when it senses a packet reception, provided the transmit flow control is enabled. This results in a collision and the remote station backs off. If the application requests a packet to be transmitted, it is scheduled and transmitted even when the backpressure is activated. If the backpressure is kept activated for a long time (and more than 16 consecutive collision events occur), the remote stations abort the transmission because of excessive collisions.

56.3.13.2 Receive Flow Control

In the Receive path, the Flow Control is functional only in the full-duplex mode. If any Pause packet is received in the half-duplex mode, the packet is considered as a normal control packet. You can enable the Pause flow control by setting the RFE bit in the MAC_Rx_Flow_Ctrl register. *Table 1271*, describes the flow control in the Rx path based on the setting of the following bits:

- RFE bit of MAC_Rx_Flow_Ctrl register
- DM bit of MAC_Configuration register

Table 1271. RX MAC Flow Control

RFE	DM	Description
0	x	The MAC receiver does not detect the received Pause packets.
1	0	The MAC receiver does not detect the received Pause packets but recognizes such packets as Control packets.
1	1	The MAC receiver detects or processes the Pause packets and responds to such packets by stopping the MAC transmitter.

When the RFE bit is enabled, the MAC transmitter is blocked when the 802.3x Pause packet is received.

The following list describes the Rx flow control:

1. The MAC checks the destination address of the received Pause packet for either of the following:
   – Multicast destination address: The DA matches the unique multicast address specified for the control packet (48'h0180C2000001)
   – Unicast destination address: The DA matches the content of the MAC Address Register 0 and the UP bit of MAC_Rx_Flow_Ctrl register is set.
   If the UP bit is set and the MAC processes Pause packets with unicast destination address in addition to the unique multicast address
2. The MAC decodes the following fields of the received packet:
   – Type field: This field is checked for 16'h8808
   – Opcode field: This field is checked for 16'h0001 (Pause packet)
   – Pause Time: The Pause time (for Pause packet) is captured to determine the time for which transmitter needs to be blocked. If the byte count of the status indicates 64 bytes and there is no CRC error, the MAC pauses the transmission of any data packet for the duration of the decoded Pause Time value multiplied by the slot time (64 byte times)
3. The MAC transfers the received control packet to the application based on the setting of the PCF field in MAC_Packet_Filter register
4. If subsequent Pause packets are received before the earlier Pause Time expires, the MAC updates the Pause Timer with new value

56.3.14 Checksum Offload Engine

Communication protocols such as TCP and UDP implement checksum fields, which help determine the integrity of data transmitted over a network. The most widespread use of Ethernet is to encapsulate TCP and UDP over IP datagrams. Therefore, the MAC has a Checksum Offload Engine (COE) to support checksum calculation and insertion in the Transmit path, and error detection in the Receive path. On the transmit path the Checksum Offload Engine is present in Queue0 only.

56.3.14.1 Transmit Checksum Offload Engine

The COE module supports two types of checksum calculation and insertion. The checksum engine can be controlled for each packet by setting the CIC bits (TDES3 Bits[14:15]).

Note: The checksum for TCP, UDP, or ICMP is calculated over a complete packet, and then inserted into its corresponding header field. Because of this requirement, when this function is enabled, the Tx FIFO automatically operates in the store-and-forward mode even if the Ethernet module is configured for Threshold (cut-through) mode.
You must make sure that the Tx FIFO is deep enough to store a complete packet before that packet is transferred to the MAC transmitter. The reason being that when space is not available to accept the programed burst length of data, then the MTL Tx FIFO starts reading to avoid dead-lock. When reading starts, the COE fails and consequently all succeeding packets may get corrupted because of improper recovery. Therefore, you must enable the checksum insertion only in the packets that are less than the following number of bytes in size: TXFIFO_SIZE – ((PBL + N)(DATAWIDTH/8)), Where N= 5*

56.3.14.1.1 IP Header Checksum Engine

In IPv4 datagrams, the integrity of the header fields is indicated by the 16-bit Header Checksum field (the eleventh and twelfth bytes of the IPv4 datagram). The COE detects an IPv4 datagram when the Type field of Ethernet packet has the value 0x0800 and the Version field of IP datagram has the value 0x4. The checksum field of the input packet is ignored during calculation and replaced with the calculated value.

IPv6 headers do not have a checksum field. Therefore, the COE does not modify the IPv6 header fields.

The result of this IP header checksum calculation is indicated by the IP Header Error status bit in the Transmit status (Bit 31). This status bit is set whenever the values of the Ethernet Type field and the Version field of IP header are not consistent, or when the Ethernet packet does not have enough data, as indicated by the IP header Length field. In other words, this bit is set when an IP header error is asserted under the following circumstances:

- For IPv4 datagrams:
  - The received Ethernet type is 0x0800, but the Version field of IP header is not equal to 0x4
  - The IPv4 Header Length field indicates a value less than 0x5 (20 bytes)
  - The total packet length is less than the value given in the IPv4 Header Length field
- For IPv6 datagrams:
  - The Ethernet type is 0x86dd but the IP header Version field is not equal to 0x6
  - The packet ends before the IPv6 header (40 bytes) or extension header (as given in the corresponding Header Length field in an extension header) is completely received

56.3.14.1.2 TCP/UDP/ICMP Checksum Engine

The TCP/UDP/ICMP Checksum Engine processes the IPv4 or IPv6 header (including extension headers) and determines whether the encapsulated payload is TCP, UDP, or ICMP. The checksum is calculated for the TCP, UDP, or ICMP payload and inserted into its corresponding field in the header. The Tx COE can work in the following two modes:

- The TCP, UDP, or ICMPv6 pseudo-header is not included in the checksum calculation and is assumed to be present in the Checksum field of the input packet. This engine includes the Checksum field in the checksum calculation, and then replaces the Checksum field with the final calculated checksum
- The engine ignores the Checksum field, includes the TCP, UDP, or ICMPv6 pseudo-header data into the checksum calculation, and overwrites the checksum field with the final calculated value

Note: For ICMP-over-IPv4 packets, the Checksum field in the ICMP packet must always be 16'h0000 in both modes, because pseudo-headers are not defined for such packets. If it does not equal 16'h0000, an incorrect checksum may be inserted into the packet.

The result of this operation is indicated by the Payload Checksum Error status bit in the Transmit Status vector (Bit 19). This engine sets the Payload Checksum Error status bit when it detects that the packet has been forwarded to the MAC Transmitter engine in the store-and-forward mode without the end of packet (EOP) being written to the FIFO, or when the packet ends before the number of bytes indicated by the Payload Length field in the IP Header is received. When the packet is longer than the indicated payload length, the COE ignores them as stuff bytes, and no error is reported. When this engine detects the first type of error, it does not modify the TCP, UDP, or ICMP header. For the second error type, it still inserts the calculated checksum into the corresponding header field.

Table 1272 describes the functions supported by Transmit Checksum Offload engine based on the packet type. When the MAC does not insert the checksum, it is indicated as “No” in the table.

Note: You should not enable checksum insertion for IPv4 or IPv6 packets that are greater than the frame size constraint specified in Transmit Checksum Offload Engine because it may result in incorrect checksum insertion or unexpected behavior.

Table 1272. Transmit Checksum Offload Engine Functions for Different Packet Types

Packet type	Hardware IP header checksum insertion	Hardware TCP/UDP checksum insertion
Non-IPv4 or IPv6 packet	No	No
IPv4 packet is greater than 1,522 bytes (2,000 bytes when IEEE 802.3ad Support for 2K Packets is enabled in MAC) but less than or equal to the frame size constraint specified in Transmit Checksum Offload Engine.	Yes	Yes
IPv6 packet is greater than 1,522 bytes (2,000 bytes when IEEE 802.3ad Support for 2 K Packets is enabled in MAC) but less than or equal to the frame size constraint specified in Transmit Checksum Offload Engine.	Not applicable	Yes
IPv4 with TCP, UDP, or ICMP	Yes	Yes
IPv4 packet has IP options (IP header is longer than 20 bytes)	Yes	Yes
Packet is an IPv4 fragment	Yes	No
IPv4 packet has TCP header with Options fields	Yes	Yes
IPv6 packet with the following next header fields in main or extension headers		
Hop-by-hop options (in IPv6 main header)	Not applicable	Yes
Hop-by-hop options (in IPv6 extension header)	Not applicable	No
Destinations options	Not applicable	Yes
Routing (with segment left 0)	Not applicable	No
Routing (with segment left > 0)	Not applicable	No
TCP, UDP, or ICMP	Not applicable	Yes
Authentication	Not applicable	Yes

Table 1272. Transmit Checksum Offload Engine Functions for Different Packet Types (continued)

Packet type	Hardware IP header checksum insertion	Hardware TCP/UDP checksum insertion
Any other next header field in main or extension headers	Not applicable	No
IPv4 Tunnels		
IPv4 packet in an IPv4 tunnel	Yes (IPv4 tunnel header)	No
IPv6 packet in an IPv4 tunnel	Yes (IPv4 tunnel header)	No
IPv6 Tunnels		
IPv4 packet in an IPv6 tunnel	Not applicable	No
IPv6 packet in an IPv6 tunnel	Not applicable	No
IPv4 packet has 802.3ac tag (with C-VLAN Tag or S-VLAN Tag when enabled).	Yes	Yes
IPv6 packet has 802.3ac tag (with C-VLAN Tag or S-VLAN Tag when enabled).	Not applicable	Yes
IPv4 frames with security features (such as encapsulated security payload)	Yes	No
IPv6 frames with security features (such as encapsulated security payload)	Not applicable	No

56.3.14.2 Receive Checksum Offload Engine

You can enable the Receive Checksum Offload Engine (Rx COE) by setting the IPC bit of MAC_Configuration register. When enabled, both IPv4 and IPv6 packet in the received Ethernet packets are detected and processed for data integrity. The MAC receiver identifies IPv4 or IPv6 packets by checking for value 0x0800 or 0x86DD, respectively, in the Type field of the received Ethernet packet. This identification is applicable to single VLAN-tagged packets. It is also applicable to double VLAN-tagged packets when the EDVLP bit of the MAC_VLAN_Tag register is set.

The Rx COE calculates the IPv4 header checksums and checks that they match the received IPv4 header checksums. The result of this operation (pass or fail) is sent in the receive status word. The IP Header Error bit is set for any mismatch between the indicated payload type (Ethernet Type field) and the IP header version, or when the received packet does not have enough bytes, as indicated by the Length field of the IPv4 header (or when fewer than 20 bytes are available in an IPv4 or IPv6 header).

This engine also identifies a TCP, UDP, or ICMP payload in the received IP datagrams (IPv4 or IPv6) and calculates the checksum of such payloads properly, as defined in the TCP, UDP, or ICMP specifications. This engine includes the TCP, UDP, or ICMPv6 pseudo-header bytes for checksum calculation and checks whether the received checksum field matches the calculated value. The result of this operation is given as a Payload Checksum Error bit in the receive status word. This status bit is also set if the length of the TCP, UDP, or ICMP payload does not match the expected payload length given in the IP header.

Table 1273, describes the functions supported by the Rx COE based on the packet type. When the payload of an IP packet is not processed (indicated as "No" in the table), the information (whether the checksum engine is bypassed or not) is given in the receive status*Table 1274*.

Note: The MAC does not append any payload checksum bytes to the received Ethernet packets.

Table 1273. Receive Checksum Offload Engine Functions for Different Packet Types

Packet type	Hardware IP header checksum checking	Hardware TCP/UDP/ICMP checksum checking
Non-IPv4 or IPv6	No	No
IPv4 packet is greater than 1,522 bytes (2,000 bytes when IEEE 802.3ad Support for 2K Packets is enabled in the MAC)	Yes	Yes
IPv6 packet is greater than 1,522 bytes (2,000 bytes when IEEE 802.3ad Support for 2K Packets is enabled in the MAC)	Not applicable	Yes
IPv4 with TCP, UDP, or ICMP	Yes	Yes
IPv4 header's protocol field contains a protocol other than TCP, UDP, or ICMP	Yes	No
IPv4 packet has IP options (IP header is longer than 20 bytes)	Yes	Yes
Packet is an IPv4 fragment	Yes	No
IPv6 packet with the following next header fields in main or extension headers		
Hop-by-hop options (in IPv6 main header)	Not applicable	Yes
Hop-by-hop options (in IPv6 extension header)	Not applicable	No
Destinations options	Not applicable	Yes
Routing (with segment left 0)	Not applicable	Yes
Routing (with segment left > 0)	Not applicable	No
TCP, UDP, or ICMP	Not applicable	Yes
Any other next header field in main or extension headers	Not applicable	No
IPv4 packet has TCP header with Options fields	Yes	Yes
IPv4 Tunnels		
IPv4 packet in an IPv4 tunnel	Yes (IPv4 tunnel header)	No
IPv6 packet in an IPv4 tunnel	Yes (IPv4 tunnel header)	No
IPv6 Tunnels		
IPv4 packet in an IPv6 tunnel	Not applicable	No
IPv6 packet in an IPv6 tunnel	Not applicable	No
IPv4 packet has 802.3ac tag (with C-VLAN Tag or S-VLAN Tag when enabled).	Yes	Yes
IPv6 packet has 802.3ac tag (with C-VLAN Tag or S-VLAN Tag when enabled).	Not applicable	Yes

Table 1273. Receive Checksum Offload Engine Functions for Different Packet Types (continued)

Packet type	Hardware IP header checksum checking	Hardware TCP/UDP/ICMP checksum checking
IPv4 frames with security features (such as encapsulated security payload)	Yes	No
IPv6 frames with security features (such as encapsulated security payload)	Not applicable	No

56.3.15 MAC Management Counters

The counters in the MAC Management Counters (MMC) module can be viewed as an extension of the register address space of the CSR module. The MMC module maintains a set of registers for gathering statistics on the received and transmitted packets. The register set includes a control register for controlling the behavior of the registers, two 32-bit registers containing interrupts generated (receive and transmit), and two 32-bit registers containing masks for the Interrupt register (receive and transmit). These registers are accessible from the Application through the CSR. Each register is 32-bits wide. The width of each MMC counter is 32-bit.

The MMC counters are free running. There is no separate enable for the counters to start. If a particular MMC counter is present, it starts counting when corresponding packet is received or transmitted. The Receive MMC counters are updated for packets that are passed by the Address Filter (AFM) block. The statistics of packets, dropped by the AFM module, are not updated unless they are runt packets of less than 6 bytes (DA bytes are not received fully). To get statistics of all packets, you should set Bit 31 in the MAC_Packet_Filter register.

The MMC module gathers statistics on encapsulated IPv4, IPv6, TCP, UDP, or ICMP payloads in received Ethernet packets.

The following definitions define the terminology.

- Transmitted packets are considered “good” if transmitted successfully. In other words, a transmitted packet is good if the packets transmission is not aborted because of any of the following errors:
  - Jabber Timeout
  - No Carrier or Loss of Carrier
  - Late Collision
  - Packet Underflow
  - Excessive Deferral
  - Excessive Collision
- Received packets are considered “good” if none of the following errors exists:
  - CRC error
  - Runt packet (shorter than 64 bytes)
  - Alignment error (in 10/100 Mbps only)
  - Length error (non-Type packet only)
  - Out of Range (non-Type packet only, longer than 1518 bytes)
  - MII_RXER Input error
- The maximum transmit frame size depends on the frame type, as follows:
  - Untagged frame maxsize = 1,518
  - VLAN Frame maxsize = 1,522
  - Jumbo Frame maxsize = 9,018
  - JumboVLAN Frame maxsize = 9,022
- The maximum receive packet size depends on the packet type and control bits (JE, S2KP, GPSLCE and EDVLP), as shown in *Table 1274*.

Table 1274. Size of the Maximum Receive Packet Size

JE	S2KP	GPSLCE	EDVLP	Untagged Frame maxsize in bytes	Single VLAN Frame maxsize in bytes	Double VLAN Frame maxsize in bytes
1	X	X	1	9018	9022	9026
0	1	X	X	2000	2000	2000
0	0	1	1	GPSL	GPSL+4	GPSL+8
0	0	0	1	1518	1522	1526
1	X	X	0	9018	9022	9022
0	0	1	0	GPSL	GPSL+4	GPSL+4
0	0	0	0	1518	1522	1522

56.3.16 Low Power Modes

This chapter describes the low power modes supported by the Ethernet module. It contains the following sections:

- Energy Efficient Ethernet
- Power Management

56.3.16.1 Energy Efficient Ethernet

Energy Efficient Ethernet (EEE) is an operational mode that enables the IEEE 802.3 Media Access Control (MAC) sub layer along with a family of Physical layers to operate in the Low-Power Idle (LPI) mode. The EEE operational mode supports the IEEE 802.3 MAC operation at 100 Mbps. The Ethernet module supports the IEEE 802.3az-2010 for EEE.

The LPI mode allows power saving by switching off the parts of the communication device functionality when there is no data to be transmitted and received. The systems on both sides of the link can disable some functionalities to save power during the periods of low-link utilization. The MAC controls whether the system should enter or exit the LPI mode and communicates this to the PHY.

The EEE specifies the capabilities, negotiation methods that the link partners can use to determine whether EEE is supported, and then select the set of parameters that are common to both devices.

Note: The EEE feature is not supported when the MAC is configured to use the RMII PHY interface. Even if the MAC supports multiple PHY interfaces, you should activate the EEE mode only when the MAC is operating with the MII interface.
According to the Energy Efficient Ethernet standard (IEEE 802.3az-2010), the LPI mode is supported only in the full-duplex mode. Therefore, you should not enable the LPI mode when the MAC Transmitter is configured for the half-duplex mode.

56.3.16.1.1 Transmit Path Functions

In the Transmit path, the software must set the LPIEN bit of the MAC_LPI_Control_Status register to indicate to the MAC to stop transmission and initiate the LPI protocol. The MAC completes the transmission in progress, generates its transmission status, and starts transmitting the LPI pattern instead of the IDLE pattern if the link status has been up continuously for a period specified in the LPI LS TIMER field of MAC_LPI_Timers_Control register. The PHY Link Status bit of the LPI Control and Status Register indicates the link status of the PHY.

Note: According to the Energy Efficient Ethernet standard (IEEE 802.3az-2010), the PHY must not stop the TxCLK clock during the LPI state in the MII (10 or 100) mode.

To make the PHY enter the LPI state, the MAC performs the following tasks:

1. De-asserts TX_EN
2. Asserts TX_ER
3. Sets TXD[3:0] to 0x1 (for 100 Mbps)
4. Updates the status (TLPIEN bit of MAC_LPI_Control_Status register) and generates an interrupt

To bring the PHY out of the LPI state, that is, when the software resets the LPIEN bit, the MAC performs the following tasks:

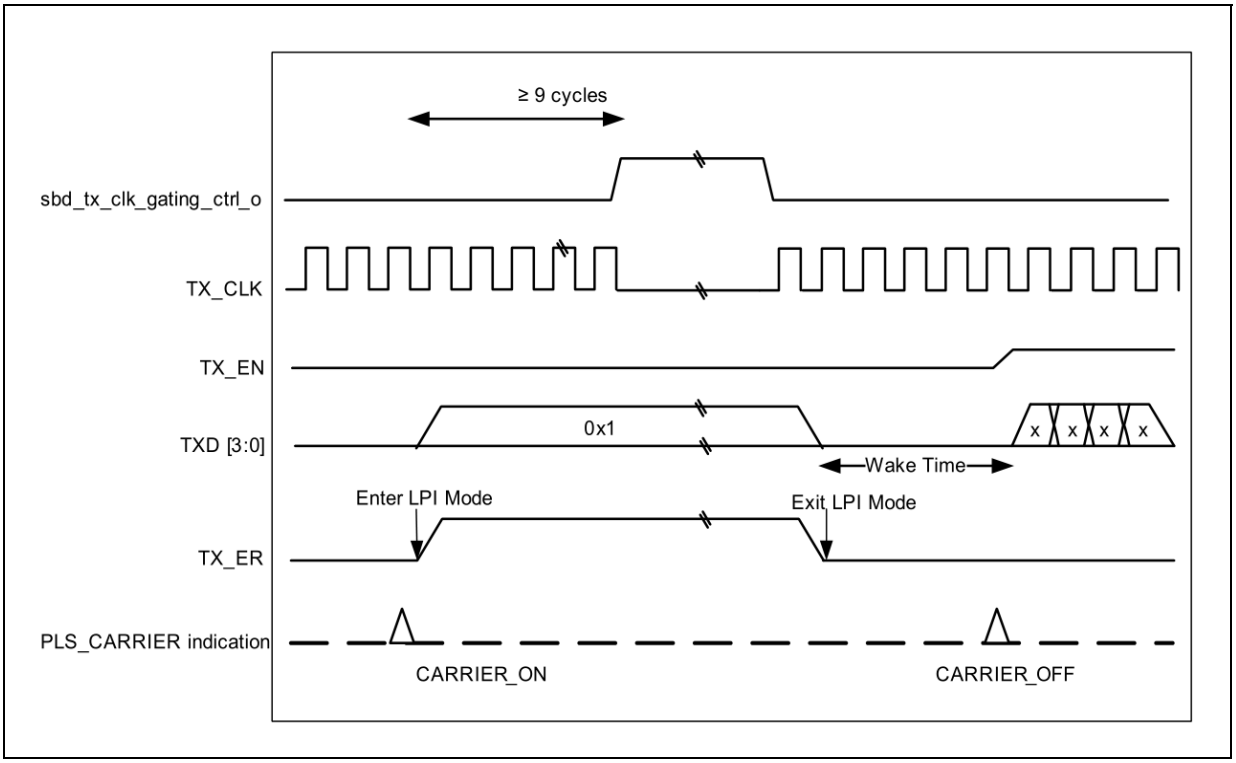
1. Stops transmitting the LPI pattern and starts transmitting the IDLE pattern
2. Starts the LPI TW TIMER
   The MAC cannot start the transmission until the wake-up time specified for the PHY expires. The auto-negotiated wake-up interval is programed in the TWT field of the MAC_LPI_Timers_Control register
3. Updates the LPI exit status (TLPIEX bit of the MAC_LPI_Control_Status register) and generates an interrupt

Figure 1205 shows the behavior of TX_EN, TX_ER, and TXD[3:0] signals during the LPI mode transitions.

Note: The MAC maintains the same state of the TX_EN, TX_ER, and TXD signals for the entire duration during which the PHY remains in the LPI state.

The MAC does not stop the TX_CLK clock. You can stop this clock (as shown in Figure 1205) if your PHY supports it and when the MAC sets the sbd_tx_clk_gating_ctrl_o signal to 1. The sbd_tx_clk_gating_ctrl_o signal is asserted after 9 Tx Clock Cycles, one Pulse Synchronizer delay (BCM22), and one CSR clock cycle. The assertion of the sbd_tx_clk_gating_ctrl_o signal is dependent on the LPITCSE bit of the MAC_LPI_Control_Status Register.
If the MAC is in the Tx LPI mode and the Tx clock is stopped, the application should not write to CSR registers that are synchronized to Tx clock domain.
If the MAC is in the LPI mode and the application issues a soft reset or hard reset, the MAC transmitter comes out of the LPI mode.

Figure 1205. LPI Transitions (Transmit)



56.3.16.1.1.1 Automated Entry/Exit of LPI mode in TX Path

The MAC transmitter can be programed to enter and exit LPI IDLE mode automatically based on whether it is IDLE for a specific period of time or has a packet to transfer. These modes are enabled and controlled by MAC_LPI_Control_Status register.

When LPITXA (Bit[12]) and LPITXEN (Bit[15]) of MAC_LPI_Control_Status register are set, the MAC transmitter enters LPI IDLE state when the MAC transmit path (including the MTL layers and DMA layers) are idle. The MAC transmitter will exit the LPI IDLE state and clear the LPITXEN bit as soon as any of functions in the TX path (DMA, MTL or MAC) becomes non-idle due to initiation of a packet transfer.

In addition to the above, when Bit[11] (LPIATE) is also set, the MAC transmitter will enter LPI IDLE state only if the Transmit path remains in idle state (no activity) for the time period indicated by the value in MAC_LPI_Entry_Timer. In this mode also, the MAC transmitter will exit the LPI IDLE state as soon as any of the functions becomes non-idle. However, the LPITXEN bit is not cleared but remains active so that re-entry to LPI IDLE state is possible without any software intervention when the MAC becomes idle again.

When both LPIATE and LPITXA bits are cleared, you can directly control the entry and exit of LPI IDLE state by programming the LPITXEN bit.

56.3.16.1.2 Receive Path Functions

In the receive path, when the PHY receives the signals from the link partner to enter into the LPI state, the PHY and MAC perform the following tasks:

1. The PHY asserts RX_ER
2. The PHY sets RXD[7:0] to 0x01
3. The PHY de-asserts RX_DV
4. The MAC updates the RLPIEN bit of the MAC_LPI_Control_Status register and immediately generates an interrupt

Note: If the LPI pattern is detected for a very short duration (that is, less than 2 cycles of Rx clock), the MAC does not enter the Rx LPI mode.
If the duration between end of the current Rx LPI pattern and start of the next Rx LPI pattern, is very short (that is, less than 2 cycles of Rx clock), then the MAC exits and again enters the Rx LPI mode. The MAC does not give the Rx LPI Exit and Entry interrupts.

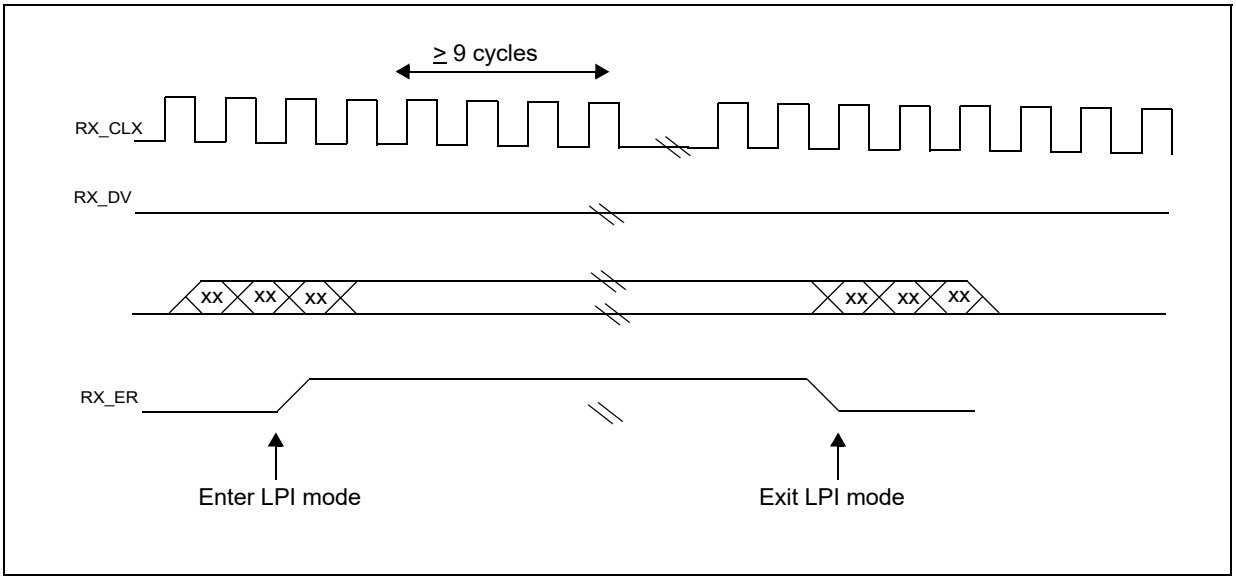
The PHY maintains the same state of the RX_ER, RXD, and RX_DV signals for the entire duration during which it remains in the LPI state.

When the PHY receives signals from the link partner to exit the LPI state, the PHY and MAC perform the following tasks:

1. The PHY de-asserts RX_ER and returns to a normal inter-packet state.
2. The MAC updates the RLPIEX bit of the MAC_LPI_Control_Status register and generates an interrupt immediately. The sideband signal lpi_intr_o (synchronous to Rx clock) is also asserted.

Figure 1206 shows the behavior of RX_ER, RX_DV, and RXD[3:0] signals during the LPI mode transitions.

Figure 1206. LPI Transitions (Receive)



Note: If the RX_CLK_stoppable bit (in the PHY register written through MDIO) is asserted when the PHY is indicating LPI to the MAC, the PHY may halt the RX_CLK at any time more than 9 clock cycles after the start of the LPI state as shown in Figure 1206.
If the MAC is in the LPI mode and the application issues a soft reset or hard reset, the MAC receiver comes out of the LPI mode during reset. If the LPI pattern is still received after the reset is de-asserted, the MAC receiver again enters the LPI state.
If the RX clock is stopped in the RX LPI mode, the application should not write to the CSR registers that are being synchronized to the RX clock domain.
When the PHY sends the LPI pattern, if EEE feature is enabled, the MAC automatically enters the LPI state. There is no software control to prevent the MAC from entering the LPI state.

56.3.16.1.3 LPI Timers

The transmitter maintains the following two timers that are loaded with the respective values from the MAC_LPI_Timers_Control and MAC_LPI_Entry_Timer registers:

- LPI LS TIMER: The LPI LS TIMER counts, in milliseconds, the time expired since the link status is up. Software should get the PHY link status by reading the PHY register and accordingly update the PLS bit.
  This timer is cleared every time the link goes down. It starts to increment when the link is up again and continues to increment until the value of the timer becomes equal to the terminal count. Once the terminal count is reached, the timer remains at the same value as long as the link is up. The terminal count is the value programed in Bits[6:15] of the MAC_LPI_Timers_Control register. The MII interface does not assert the LPI pattern unless the terminal count is reached. This ensures a minimum time for which no LPI pattern is asserted after a link is established with the remote station. This period is defined as 1 second in the IEEE 802.3-az-2010. The LPI LS TIMER is 10-bit wide. Therefore, the software can program up to 1023 milliseconds.
- LPI TW TIMER: The LPI TW TIMER counts, in microseconds, the time expired since the de-assertion of LPI. The terminal count should be programed in Bit[16:31] of MAC_LPI_Timers_Control register. The terminal count of the timer is the value of resolved Transmit TW that is the auto-negotiated time after which the MAC can resume the normal transmit operation. After exiting the LPI mode, the MAC resumes its normal

operation after the TW timer reaches the terminal count.
The MAC supports the LPI TW TIMER in units of microsecond. The LPI TW TIMER is 16-bit wide. Therefore, the software can program up to 65535 µs.

- LPI AUTO ENTRY TIMER: This timer counts in steps of eight microseconds, the time for which the MAC transmit path has to remain in idle state (no activity), before the MAC Transmitter enters the LPI IDLE state and starts transmitting the LPI pattern. This timer is enabled when LPITE bit in MAC_LPI_Control_Status register is set.

56.3.16.2 Power Management

The power management (PMT) block supports the reception of network (remote) wake-up packets and magic packets. The PMT block does not perform the clock gate function, but generates interrupts for remote wake-up packets and magic packets that the MAC receives.

When you enable the power-down mode in the PMT block, the MAC drops all received packets and does not forward any packet to the MTL RxFIFO or the application. The MAC comes out of the power-down mode only when a magic packet or a remote wake-up packet is received and the corresponding detection is enabled.

The PMT block is available in the receive path of MAC. You can select both types of power management packet (remote wake-up packet and magic packet). You can use the RWKPKTEN and MGKPKTEN bits of the MAC_PMT_Control_Status register to generate power management events. The application should program these bits. You can access the PMT registers in the similar manner as you access the MAC CSR registers.

Note: The magic packet feature is implemented based on the Magic Packet Technology white paper. The remote wake-up packet feature is implemented based on the Device Class Power Management Reference Specification and various implementation-specific white papers.

56.3.16.2.1 PMT Block Registers

The following are the PMT block registers:

- MAC_PMT_Control_Status
- MAC_RWK_Packet_Filter

56.3.16.2.2 Remote Wake-Up Packet Detection

When the MAC is in sleep mode and the remote wake-up bit is enabled in the MAC_PMT_Control_Status, the normal operation is resumed after a remote wake-up packet is received. The application writes all eight wake-up filter registers, by performing a sequential Write to address (00C4H). The application enables remote wake-up by writing 1 to the RWKPKTEN bit of the MAC_PMT_Control_Status register.

The PMT block supports 4 programmable filters that allow support of different receive packet patterns. If the incoming packet passes the address filtering of Filter Command, and if Filter CRC-16 matches the CRC of the incoming pattern, the MAC identifies the packet as a wake-up packet.

The Filter Offset determines the offset from which the packet is to be examined. The Filter Byte Mask determines which bytes of the packet must be examined. The bit 0 of Byte Mask must be set to zero.

The remote wake-up CRC block determines the CRC value that is compared with Filter CRC-16. The remote wake-up packet is checked only for length error, FCS error, dribble bit

error, MII error, and collision. In addition, the remote wake-up packet is checked to ensure that it is not a runt packet. Even if the remote wake-up packet is more than 512 bytes long, if the packet has a valid CRC value, it is considered valid. The remote wake-up packet detection is updated in the PMT Control and Status register for every remote wake-up packet received. A PMT interrupt to the application triggers a Read to the PMT Control and Status register to determine reception of a remote wake-up packet.

Note: The watchdog timeout limit for a remote wake-up frame is 2,048 bytes irrespective of the value programed in WD bit of MAC_Configuration register and PWE bit in MAC_Watchdog_Timeout register.
The value programed in DCRCC bit of MAC_Ext_Configuration register is applicable to a remote wake-up frame only when Enable IPv4 ARP Offload is selected in the configuration.

56.3.16.2.3 Magic Packet Detection

The magic packet is based on a method that uses the magic packet technology from Advanced Micro Device to power up the sleeping device on the network. The MAC receives a specific packet of information, called a magic packet, addressed to the node on the network.

The MAC checks only those magic packets that are addressed to the MAC or a multicast address (including broadcast address) to determine whether these packets meet the wake-up requirements. The magic packets that pass the address filtering (unicast or multicast (including broadcast) address) are checked to determine whether they meet the remote wake-up packet data format of 6 bytes of all ones followed by a Unicast MAC Address (that matches the value in MAC Address 0) appearing 16 times.

The application enables the magic packet wake-up by writing 1 to the MGKPKTEN bit of the MAC_PMT_Control_Status register. The PMT block constantly monitors each packet addressed to the node for a specific magic packet pattern. Each packet received is checked for a 48’hFF_FF_FF_FF_FF_FF pattern following the destination and source address field. The PMT block then checks the packet for 16 repetitions of the MAC address without any breaks or interruptions. In case of a break in the 16 repetitions of the address, the PMT block again scans the 48’hFF_FF_FF_FF_FF_FF pattern in the incoming packet. The 16 repetitions can be anywhere in the packet, but must be preceded by the synchronization stream (48’hFF_FF_FF_FF_FF_FF). The device can also accept a multicast packet, as long as the 16 duplications of the MAC address are detected. If the number of repetitions of 8'hFF are more than 6, the PMT block checks for 16 repetitions of the MAC address without any breaks or interruptions, after the last 6 repetitions of 8'hFF.

If the MAC address of a node is 48'h00_11_22_33_44_55, the MAC scans for the following data sequence:

Destination Address Source Address …….. FF FF FF FF FF FF

00 11 22 33 44 55 00 11 22 33 44 55 00 11 22 33 44 55 00 11 22 33 44 55

00 11 22 33 44 55 00 11 22 33 44 55 00 11 22 33 44 55 00 11 22 33 44 55

00 11 22 33 44 55 00 11 22 33 44 55 00 11 22 33 44 55 00 11 22 33 44 55

00 11 22 33 44 55 00 11 22 33 44 55 00 11 22 33 44 55 00 11 22 33 44 55

…CRC

The Ethernet module checks the remote wake-up packet only for length error, FCS error, dribble bit error, MII error, collision. In addition, the remote wake-up packet is checked to

ensure that it is not a runt packet. Even if the remote wake-up packet is more than 512 bytes long, if the packet has a valid CRC value, the Ethernet module considers it a valid packet.

The magic packet detection is updated in the MAC_PMT_Control_Status register for the received magic packet. A PMT interrupt to the Application triggers a read to the MAC_PMT_Control_Status register to determine whether a magic packet has been received.

Note: The watchdog timeout limit for a magic packet is 2,048 bytes irrespective of the value programed in WD bit of MAC_Configuration register and PWE bit in MAC_Watchdog_Timeout register.
The value programed in DCRCC bit of MAC_Ext_Configuration register is applicable to a magic packet only when Enable IPv4 ARP Offload is selected in the configuration.

56.3.16.2.4 System Considerations during Power Down

The Ethernet module neither gates nor stops clocks when the power-down mode is enabled. Power saving by clock gating must be done outside the core by the application. The receive data path must be clocked with clk_rx_i during the power-down mode, because it is involved in the magic packet or remote wake-up packet detection. However, the application path clock can be gated during the power-down mode. The transmit Path clock must be gated during the power-down mode only if ARP Offload is not enabled.

The recommended power-down and wake-up sequence is as follows.

1. Disable the Transmit DMA (if applicable) and wait for any previous packet transmissions to complete. These transmissions can be detected when Transmit Interrupt [Bit 31 of the register] is received
2. Disable the MAC transmitter and MAC receiver by clearing the appropriate bits in the_ MAC_Configuration register
3. Wait until the Receive DMA empties all the packets from the Rx FIFO to system memory. You can do this by reading the appropriate bits of Debug registers in the DMA and MTL CSR space
4. Enable Power-Down mode by appropriately configuring the PMT registers
5. Enable the MAC Receiver and enter the Power-Down mode
6. Gate the application and transmit clock (transmit clock must be gated only if ARP Offload is not enabled) inputs to the core (and other relevant clocks in the system) to reduce power and enter Sleep mode
7. On receiving a valid remote wake-up packet, the Ethernet module asserts the pmt_intr_o signal and exits the Power-Down mode
8. On receiving the interrupt, the system must enable the application and transmit clock inputs to the core
9. Read the PMT Status register to clear the interrupt, then enable the other modules in the system and resume normal operation

56.3.17 Interrupts

Interrupt can be generated as a result of various events in the Ethernet IP core. These events are captured in status registers and interrupt enables are provided for each source of interrupt such that the interrupt signal (interrupt) is asserted for an event only when the corresponding interrupt enable is set.

The interrupt status and corresponding enable registers are organized in a hierarchical manner so that it is easier for software to traverse and identify the source of interrupt event quickly. When interrupt is asserted, the DMA_Interrupt_Status register is first level that indicates the major blocks for the interrupt event source. This register is read-only, and it contains bits corresponding to each DMA channel (TX & RX pair), the MTL, and the MAC. The software application must then read one (or more) of the following registers corresponding to the bits that are set:

- DMA_CH*n*_Status register
- MTL_Interrupt_Status register
- MAC_Interrupt_Status register

The interrupt is a level signal and will get de-asserted only when all the enabled interrupt events are cleared in their respective status registers and correspondingly all the bits in the DMA_Interrupt_Status register are cleared.

56.3.17.1 DMA Interrupts

The DMA_CH*n*_Status register captures all the interrupt events of that TxDMA and RxDMA channel pair. The DMA_CH*n*_Interrupt_Enable register contains the corresponding enable bits for each of the interrupt event. There are two groups of interrupts in the DMA channel namely Normal and Abnormal interrupts. They are indicated by Bits[16:17] of DMA_CH*n*_Status register respectively. The normal group is for events that happen during the normal transfer of packets (TI, RI, TBU) while the abnormal interrupt events are for error events. Interrupt events are cleared by writing 1 to the corresponding bit position. When all the enabled interrupt events are cleared (including the NIS and AIS), the interrupt source from the DMA Channel is cleared and the corresponding bit in register is also cleared.

Interrupts are not queued. If the same interrupt event occurs again before the driver responds to the previous one, no additional interrupts are generated. For example, Receive Interrupt Bit[25] of DMA_CH*n*_Status register indicates that one or more packets were transferred to the application buffer. The driver must scan all descriptors, from the last recorded position to the first one, owned by the DMA to determine how many packets are received.

An interrupt is generated only once for multiple events. The driver must scan the register for the cause of the interrupt and clear the source in the respective Status register. The interrupt is cleared only when all the bits of the DMA_Interrupt_Status register are cleared.

56.3.17.1.1 Periodic scheduling of DMA Transmit and Receive Interrupt

It is not preferable to generate interrupts for every packet transferred by DMA (RI and TI) for system throughput performance reasons. The Ethernet IP gives the flexibility to schedule the interrupt at regular intervals using two methods:

1. Set Interrupt on Completion bit in Transmit descriptor (TDES2[0]) once for every “required” number of packets to be transmitted
2. Similarly, set the IOC (RDES3[1]) bit only at some specific intervals of Receive descriptors. This way, whenever a received packet transfer to system memory is complete and any of the descriptors used for that packet transfer has the IOC bit set, only then the RI event is generated

In addition to above, an interrupt timer (DMA_CH*n*_Rx_Interrupt_Watchdog_Timer) is given for flexible control and periodic scheduling of Receive Interrupt. When this interrupt timer is programed with a non-zero value, it gets activated as soon as the Rx DMA completes a transfer of a received packet to system memory without asserting the Receive Interrupt

because the corresponding interrupt of completion IOC bit (RDES3[1]) is not set. When this timer runs out as per the programed value, RI bit is set and the interrupt is asserted if the corresponding RIE is enabled in DMA_CH*n*_Interrupt_Enable register. The timer is stopped and cleared before it expires, if the RI is set for a packet transfer whose descriptor's IOC was set. The timer is reactivated automatically after the next packet transfer is complete without the RI event being generated.

56.3.17.1.2 Per DMA channel Transfer Complete Interrupt

The Transmit Transfer complete interrupt (TI) and Receive Transfer complete interrupt (RI) is reflected in DMA_CH*n*_Status register. The TI bit is set whenever the Tx DMA channel closes the a descriptor in which the IOC (Interrupt On Completion - TDES2[0]) bit is set. Similarly, the RI bit is set whenever the Rx DMA channel closes the descriptor with LD bit set and in any of the descriptors used for transferring that packet, IOC (Interrupt Enable on completion - RDES3[1]) bit is set.

The common interrupt output signal is asserted for the Transfer complete interrupts only when the corresponding interrupts are enabled in DMA_CH*n*_Interrupt_Enable register.

56.3.17.2 MTL Interrupts

In MTL, the interrupts are mainly related to exception events in the TxQ or RxQ in the Transmit and Receive paths respectively. The interrupt status are captured and organized in a hierarchical manner in order to identify the root cause quickly.

The MTL_Interrupt_Status register identifies the top level modules that can cause the interrupt to be asserted.

- Bits[30:31] identify 2 Queues (Tx or Rx). They are read-only bits and the application should read the corresponding MTL_Q*n*_Interrupt_Status register to identify the exact cause and set the corresponding bits to 1 to clear the interrupt event. The assertion of interrupt due to these events is enabled by the corresponding enable bits in MTL_Q*n*_Interrupt_Enable register
- Bit[15] is made high in case the event is present in the MAC core. The application should in turn read the MTL_Interrupt_Status register to identify the cause
- Bit[14] is related to indirect access completion events to the TxFIFO and RxFIFO memory in Debug Mode

56.3.17.3 MAC Interrupts

Interrupts can be generated from the MAC as a result of various events. These interrupt events are combined with the events in the DMA. The MAC interrupts are of level type, that is, the interrupt remains asserted (high) until it is cleared by the application or software.

The MAC_Interrupt_Status register describes the events that can cause an interrupt from the MAC. The MAC interrupts are enabled by default. Each event can be prevented from asserting the interrupt by setting the corresponding mask bits in the MAC_Interrupt_Enable register.

The interrupt register bits only indicate the block from which the event is reported. You must read the corresponding status registers and other registers to clear the interrupt. For example, when set high, Bit 26 of the MAC_Interrupt_Status register indicates that the LPI interrupt status has changed. You must read the TLPIEN bit of MAC_LPI_Control_Status register to clear this interrupt event.

The interrupts from the MAC are combined (OR'ed) and given as the MACIS bit in the DMA_Interrupt_Status register.

56.3.17.4 LPI Interrupt

The MAC generates the LPI interrupt when the Tx or Rx side enters or exits the LPI state. The interrupt mci_intr_o (interrupt in EQOS-DMA configurations) is asserted when the LPI interrupt status is set. The LPI interrupt can be cleared by reading the MAC_LPI_Control_Status register.

When the MAC exits the Rx LPI state, then in addition to the mci_intr_o (interrupt in EQOS-DMA configurations), the sideband signal lpi_intr_o (synchronous to Rx clock) is asserted. You can use the lpi_intr_o signal to trigger the external clock-gating circuitry to restore the application clock to the MAC. The lpi_intr_o signal, synchronous to the Rx clock domain, is provided so that you can stop the application clock when the MAC is in the LPI state. If you do not want to gate-off the application clock during the Rx LPI state, you can leave the lpi_intr_o signal unconnected and use the mci_intr_o (interrupt in EQOS-DMA configurations) signal to detect Rx LPI exit.

The lpi_intr_o signal is generated in the Rx clock domain. It may not be cleared immediately after the MAC_LPI_Control_Status register is read. This is because the clear signal, generated in CSR clock domain, has to cross the Rx clock domain, and then clear the interrupt source. This delay is at least four clock cycles of Rx clock and can be significant when the Ethernet module is operating in the 10 Mbps mode.

56.3.17.5 PMT Interrupts

The PMT interrupt signal is asserted when a valid remote wake-up packet is received. In addition to the interrupt, the pmt_intr_o (synchronous to Rx clock) signal is asserted. The pmt_intr_o signal, synchronous to the Rx clock domain, is provided so that you can stop the application clock when the MAC is in the power-down mode.

You can use the pmt_intr_o interrupt to trigger the external clock-gating circuitry to restore the application clock and Tx clock to the MAC. If you do not want to gate-off these clocks during the power-down mode, you can leave the pmt_intr_o signal unconnected and use the interrupt to detect the MAC wake-up operation.

The pmt_intr_o signal is generated in the Rx clock domain. It is not cleared immediately when the PMT Control and Status register is read. This is because the resultant clear signal has to cross to the Rx clock domain, and then clear the interrupt source. This delay is at least four clock cycles of Rx clock and can be significant when the Ethernet module is operating in the 10 Mbps mode.

When software resets the PWRDWN bit in Remote Wake-Up Packet Detection register, the MAC comes out of the power-down mode, but this event does not generate the PMT interrupt.

56.4 Descriptors

56.4.1 Overview

The DMA in the Ethernet subsystem transfers data based on a linked list of descriptors. The application creates the descriptors in the system memory. The Ethernet module supports the following two types of descriptors:

- Normal Descriptor: Normal descriptors are used for packet data and to provide control information applicable to the packets to be transmitted
- Context Descriptor: Context descriptors are used to provide control information applicable to the packet to be transmitted

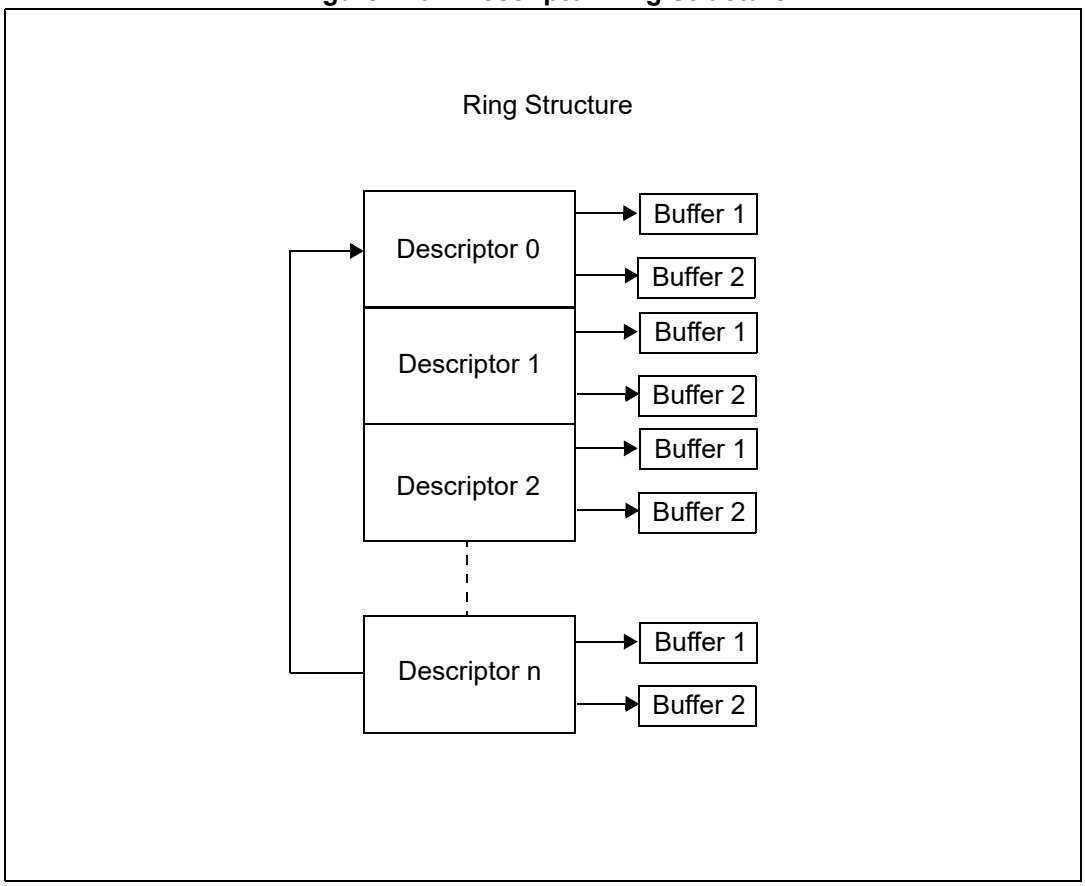
Each normal descriptor contains two buffers and two address pointers. These buffers enable the adapter port to be compatible with various types of memory management schemes.

Note: There is no limit for number of descriptors that can be used for a single packet.

56.4.1.1 Descriptor Structure

The Ethernet module supports the ring structure for DMA descriptor.

Figure 1207. Descriptor Ring Structure



In Ring structure, descriptors are separated by the DWord number programed in the DSL field of the DMA_CH*n*_Control register. The application needs to program the total ring

length, that is, the total number of descriptors in ring span in the following registers of a DMA channel:

- Transmit Descriptor Ring Length Register (DMA_CH*n*_TxDesc_Ring_Length)
- Receive Descriptor Ring Length Register (DMA_CH*n*_RxDesc_Ring_Length)

The Descriptor Tail Pointer Register contains the pointer to the descriptor address (*N*). The base address and the current descriptor pointer decide the address of the current descriptor that the DMA can process. The descriptors up to one location less than the one indicated by the descriptor tail pointer (*N* – 1) are owned by the DMA. The DMA continues to process the descriptors until the following condition occurs:

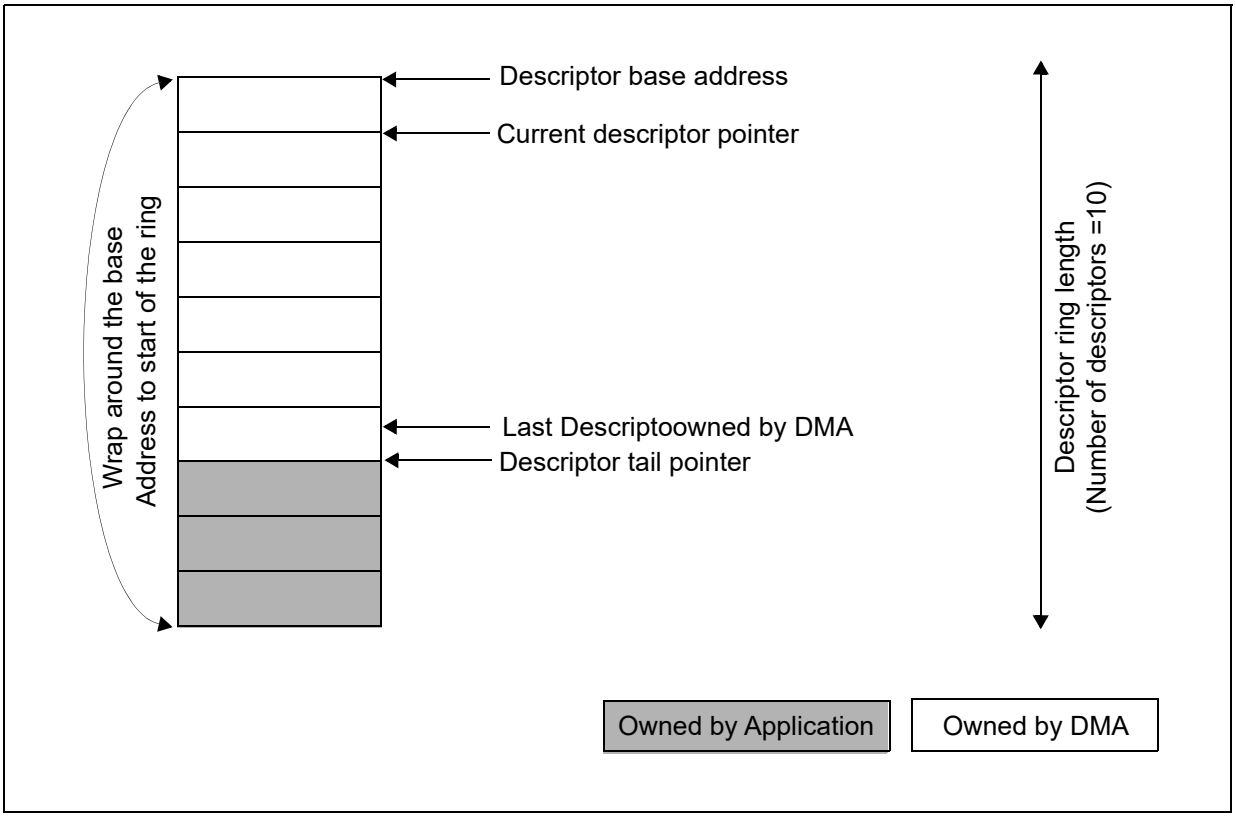
```
Current Descriptor Pointer == Descriptor Tail Pointer;
```

The DMA goes into the Suspend mode when this condition occurs. The application must perform a write to the Descriptor Tail pointer register and update the tail pointer so that the following condition is true:

```
Current Descriptor Pointer < Descriptor Tail Pointer;
```

The DMA automatically wraps around the base address when the end of ring is reached, as shown in *Figure 1208*.

Figure 1208. DMA Descriptor Ring



For descriptors owned by the application, the OWN bit of DES3 is reset to 0. For descriptors owned by the DMA, the OWN bit is set to 1. If the application has only one descriptor in the beginning, the application sets the last descriptor address (tail pointer) to Descriptor Base Address + 1. The DMA processes the first descriptor and then waits for the application to advance the tail pointer.

56.4.1.2 Descriptor Endianness

The descriptor addresses must be aligned to configured bus width (DWord for 64-bit bus). The data bus can be configured for either little-endian or big-endian format.

Figure 1209 shows the normal Receive and Transmit descriptors for 64-bit data bus when the endian mode of data bus is little-endian or big-endian and the endian mode of descriptors is reverse-endian.

Figure 1209. Rx/Tx Descriptors in Same-Endian Mode for 64-Bit, Little-Endian Data Bus

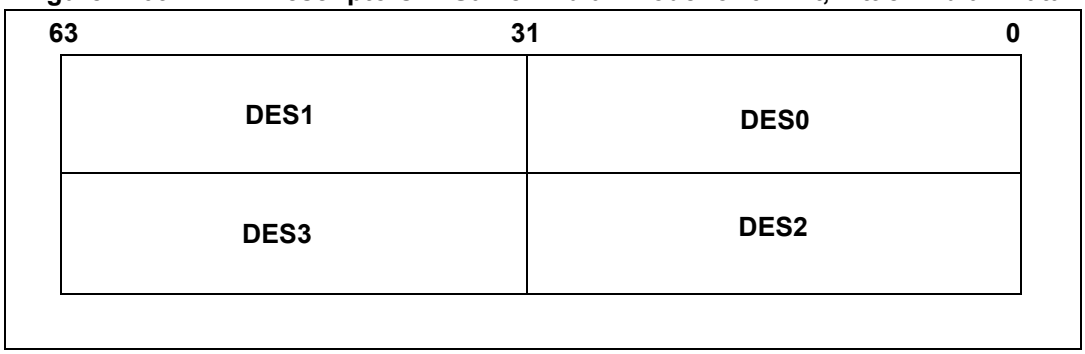
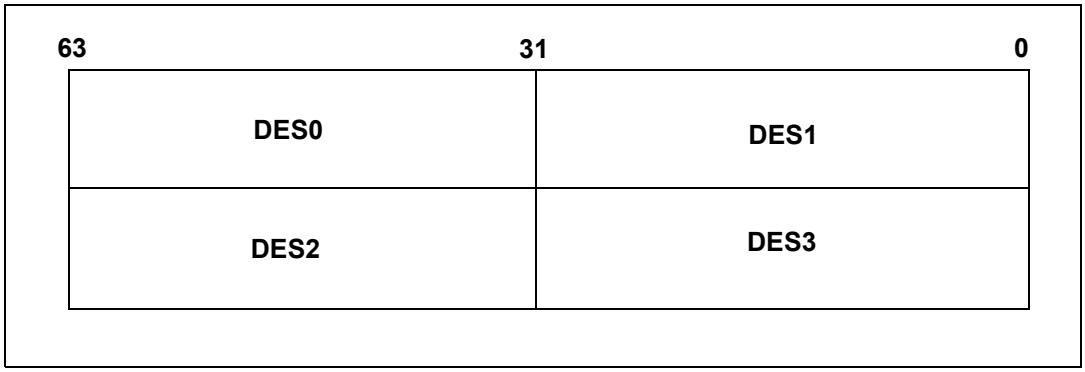


Figure 1209 and *Figure 1210* show the normal Receive and Transmit descriptors for 64-bit data bus when the endian mode of data bus is big-endian and the endian mode of descriptors is reverse-endian. In 64-bit big-endian data bus systems, when the descriptors are configured for reverse endianness, the order of the descriptor subset is reversed from the default (7-3 Rx/Tx Descriptors in Same-Endian Mode for 64-Bit, Little-Endian Data Bus). For example, Bits[63:32] are DES0 while Bits[31:0] are DES1.

Figure 1210. Rx/Tx Descriptors in Reverse-Endian Mode for 64-Bit, Big-Endian Data Bus



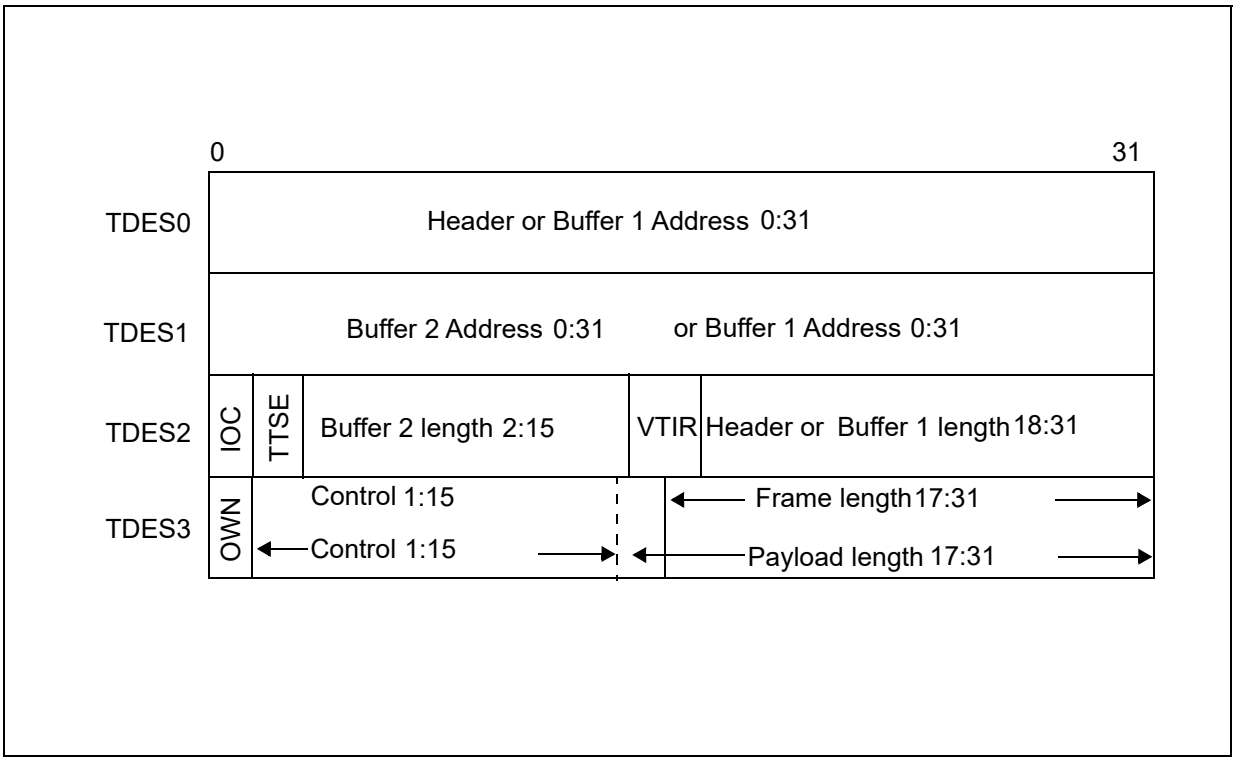
56.4.1.3 Transmit Descriptor

The DMA in the Ethernet module requires at least one descriptor for a transmit packet. In addition to two buffers, two byte-count buffers, and two address pointers, the transmit descriptor has control fields which can be used to control the MAC operation on per-transmit packet basis. The Transmit Normal descriptor has the following two formats: Read format and Write-Back format

56.4.1.3.1 Transmit Normal Descriptor (Read Format)

Figure 1211 shows the Read Format for a Transmit normal descriptor.

Figure 1211. Transmit Descriptor Read Format



56.4.1.3.1.1 TDES0 Normal Descriptor (Read Format)

Table 1275. TDES0 Normal Descriptor (Read Format)

Field	Description
0:31 BUF1AP	Buffer 1 Address Pointer These bits indicate the physical address of Buffer 1.

56.4.1.3.1.2 TDES1 Normal Descriptor (Read Format)

Table 1276. TDES1 Normal Descriptor (Read Format)

Field	Description
0:31 BUF2AP	Buffer 2 Address Pointer This bit indicates the physical address of Buffer 2 when a descriptor ring structure is used. There is no limitation for the buffer address alignment.

56.4.1.3.1.3 TDES2 Normal Descriptor (Read Format)

Figure 1212. TDES2 Normal Descriptor (Read Format)

0	1	2:15	16:17	18:31
IOC	TTSE	B2L	VTIR	B1L

Table 1277. TDES2 Normal Descriptor (Read Format)

Field	Description
0 IOC	Interrupt on Completion This bit sets the TI bit in the DMA_CH*n*_Status register after the present packet has been transmitted.
1 TTSE	Transmit Timestamp Enable This bit enables the IEEE1588 time stamping for Transmit packet referenced by the descriptor.
2:15 B2L	Buffer 2 Length The driver sets this field. When set, this field indicates Buffer 2 length.
16:17 VTIR	VLAN Tag Insertion or Replacement These bits request the MAC to perform VLAN tagging or untagging before transmitting the packets. The application must set the CRC Pad Control bits appropriately when VLAN Tag Insertion, Replacement, or Deletion is enabled for the packet. The following list describes the values of these bits: 00 Do not add a VLAN tag. 01 Remove the VLAN tag from the packets before transmission. This option should be used only with the VLAN packets. 10 Insert a VLAN tag with the tag value programed in the MAC_VLAN_Incl register or context descriptor. 11 Replace the VLAN tag in packets with the tag value programed in the MAC_VLAN_Incl register or context descriptor. This option should be used only with the VLAN packets.
18:31 B1L	Buffer 1 Length

56.4.1.3.1.4 TDES3 Normal Descriptor (Read Format)

Figure 1213. TDES3 Normal Descriptor (Read Format)

0	1	2	3	4:5	6:8	9:12	13	14:15	16	17:31
OWN	CTXT	FD	LD	CPC	SAIC	SLOTNUM	Rsvd	CIC	Reserved	FL

Table 1278. TDES3 Normal Descriptor (Read Format)

Field	Description
0 OWN	Own Bit When this bit is set, it indicates that the DMA owns the descriptor. When this bit is reset, it indicates that the application owns the descriptor. The DMA clears this bit after it completes the transfer of data given in the associated buffer(s).
1 CTXT	Context Type This bit should be set to 1'b0 for normal descriptor.
2 FD	First Descriptor When this bit is set, it indicates that the buffer contains the first segment of a packet.
3 LD	Last Descriptor When this bit is set, it indicates that the buffer contains the last segment of the packet. When this bit is set, the B1L or B2L field should have a non-zero value.
4:5 CPC	CRC Pad Control This field controls the CRC and Pad Insertion for Tx packet. This field is valid only when the first descriptor bit (TDES3[2]) is set. The following list describes the values of Bits[4:5]: 00 CRC and Pad Insertion - The MAC appends the cyclic redundancy check (CRC) at the end of the transmitted packet of length greater than or equal to 60 bytes. The MAC automatically appends padding and CRC to a packet with length less than 60 bytes. 01 CRC Insertion (Disable Pad Insertion) - The MAC appends the CRC at the end of the transmitted packet but it does not append padding. The application should ensure that the padding bytes are present in the packet being transferred from the Transmit Buffer, that is, the packet being transferred from the Transmit Buffer is of length greater than or equal to 60 bytes. 10 Disable CRC Insertion - The MAC does not append the CRC at the end of the transmitted packet. The application should ensure that the padding and CRC bytes are present in the packet being transferred from the Transmit Buffer. 11 CRC Replacement - The MAC replaces the last four bytes of the transmitted packet with recalculated CRC bytes. The application should ensure that the padding and CRC bytes are present in the packet being transferred from the Transmit Buffer.
6:8 SAIC	SA Insertion Control These bits request the MAC to add or replace the Source Address field in the Ethernet packet with the value given in the MAC Address 0 register. The application must set the CRC Pad Control bits appropriately when SA Insertion Control is enabled for the packet. Bit 6 specifies the MAC Address Register (1 or 0) value that is used for Source Address insertion or replacement. The following list describes the values of Bits[7:8]: 00 Do not include the source address 01 Include or insert the source address. For reliable transmission, the application must provide frames without source addresses. 10 Replace the source address. For reliable transmission, the application must provide frames with source addresses. 11 Reserved

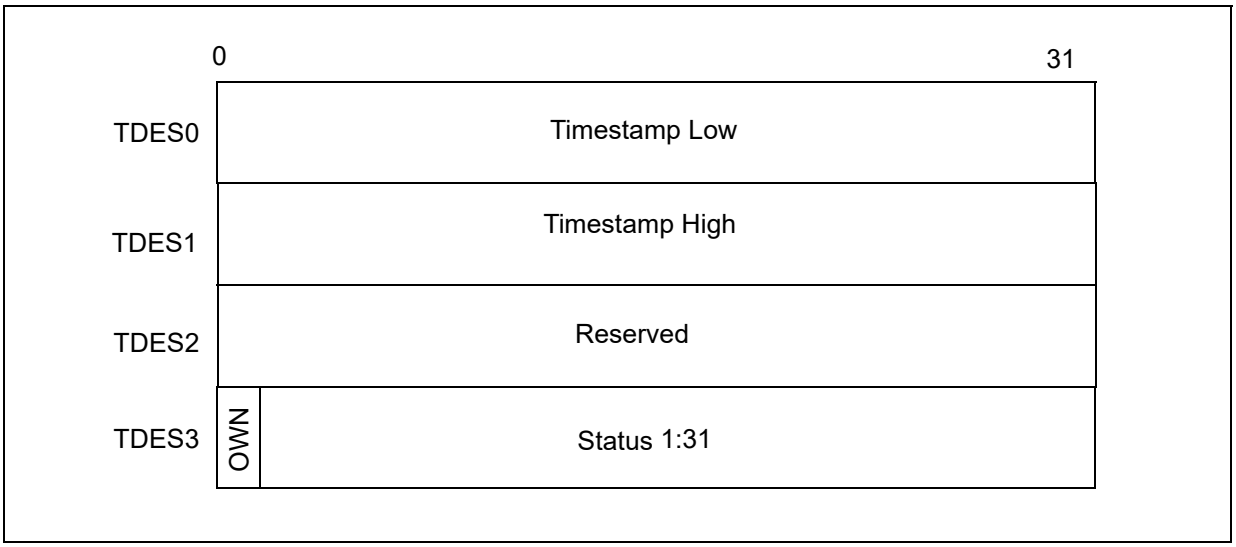
Table 1278. TDES3 Normal Descriptor (Read Format) (continued)

Field	Description
9:12 SLOTNUM	SLOTNUM: Slot Number Control Bits in AV Mode These bits indicate the slot interval in which the data should be fetched from the corresponding buffers addressed by TDES0 or TDES1. When the Transmit descriptor is fetched, the DMA compares the slot number value in this field with the slot interval maintained in the RSN field DMA_CH*n*_Slot_Function_Control_Status. It fetches the data from the buffers only if a value matches. These bits are valid only for the AV channels.
13	Reserved
14:15 CIC	Checksum Insertion Control These bits control the checksum calculation and insertion. The following list describes the bit encoding: 00 Checksum Insertion Disabled. 01 Only IP header checksum calculation and insertion are enabled. 10 IP header checksum and payload checksum calculation and insertion are enabled, but pseudo-header checksum is not calculated in hardware. 11 IP Header checksum and payload checksum calculation and insertion are enabled, and pseudo-header checksum is calculated in hardware.
16	Reserved
17:31 FL	Packet Length This field is equal to the length of the packet to be transmitted in bytes. When the TSE bit is not set, this field is equal to the total length of the packet to be transmitted: Ethernet Header Length + TCP /IP Header Length – Preamble Length – SFD Length + Ethernet Payload Length

56.4.1.3.1.5 Transmit Normal Descriptor (Write-Back Format)

The write-back format is applicable only for the last descriptor of the corresponding packet. The LD bit (TDES3[3]) is set in the descriptor where the DMA writes back the status and timestamp information for the corresponding Transmit packet.

Figure 1214. Transmit Descriptor Write-Back Format



56.4.1.3.1.6 TDES0 Normal Descriptor (Write-Back Format)

This format is only applicable to the last descriptor of a packet.

Table 1279. TDES0 Normal Descriptor (Write-Back Format)

Field	Description
0:31 TTSL	Transmit Packet Timestamp Low The DMA updates this field with least significant 32 bits of the timestamp captured for the corresponding Transmit packet. The DMA writes the timestamp only if TTSE bit of TDES2 is set in the first descriptor of the packet. This field has the timestamp only if the Last Segment bit (LS) in the descriptor is set and the Timestamp status (TTSS) bit is set.

56.4.1.3.1.7 TDES1 Normal Descriptor (Write-Back Format)

This format is only applicable to the last descriptor of a packet.

Table 1280. TDES1 Normal Descriptor (Write-Back Format)

Field	Description
0:31 TTSH	Transmit Packet Timestamp High The DMA updates this field with the most significant 32 bits of the timestamp captured for corresponding Receive packet. The DMA writes the timestamp only if the TTSE bit of TDES2 is set in the first descriptor of the packet. This field has the timestamp only if the Last Segment bit (LS) in the descriptor is set and Timestamp status (TTSS) bit is set.

56.4.1.3.1.8 TDES2 Normal Descriptor (Write-Back Format)

This format is applicable only to the last descriptor of a packet.

Table 1281. TDES2 Normal Descriptor (Write-Back Format)

Field	Description
0:31	Reserved

56.4.1.3.1.9 TDES3 Normal Descriptor (Write-Back Format)

This format is applicable only to the last descriptor of a packet.

Figure 1215. TDES3 Normal Descriptor (Write-Back Format)

0	1	2	3	4:13	14	15	16	17	18	19	20	21	22	23	24:27	28	29	30	31
OWN	CTXT	FD	LD	Rsvd	TTSS	Rsvd	ES	JT	FF	PCE	LoC	NC	LC	EC	CC	ED	UF	DB	IHE

Table 1282. TDES3 Normal Descriptor (Write-Back Format)

Field	Description
0 OWN	Own Bit When this bit is set, it indicates that the Ethernet module DMA owns the descriptor. The DMA clears this bit when it completes the packet transmission. After the write-back is complete, this bit is set to 'b0.
1 CTXT	Context Type This bit should be set to 'b0 for Normal descriptor.
2 FD	First Descriptor This bit indicates that the buffer contains the first segment of a packet.
3 LD	Last Descriptor This bit is set 'b1 for last descriptor of a packet. The DMA writes the status fields only in the last descriptor of the packet.
4:13	Reserved
14 TTSS	Tx Timestamp Status This status bit indicates that a timestamp has been captured for the corresponding transmit packet. When this bit is set, TDES2 and TDES3 have timestamp values that were captured for the Transmit packet. This field is valid only when the Last Segment control bit (TDES3 [3]) in a descriptor is set. This bit is valid only when IEEE1588 timestamping feature is enabled; otherwise, it is reserved.
15	Reserved
16 ES	Error Summary This bit indicates the logical OR of the following bits: TDES3[31]: IP Header Error TDES3[17]: Jabber Timeout TDES3[18]: Packet Flush TDES3[19]: Payload Checksum Error TDES3[20]: Loss of Carrier TDES3[21]: No Carrier TDES3[22]: Late Collision TDES3[23]: Excessive Collision TDES3[28]: Excessive Deferral TDES3[29]: Underflow Error
17 JT	Jabber Timeout This bit indicates that the MAC transmitter has experienced a jabber time-out. This bit is set only when the JD bit of the MAC_Configuration register is not set.
18 FF	Packet Flushed This bit indicates that the DMA or MTL flushed the packet because of a software flush command given by the CPU.

Table 1282. TDES3 Normal Descriptor (Write-Back Format) (continued)

Field	Description
19 PCE	Payload Checksum Error This bit indicates that the Checksum Offload engine had a failure and did not insert any checksum into the encapsulated TCP, UDP, or ICMP payload. This failure can be either because of insufficient bytes, as indicated by the Payload Length field of the IP Header or the MTL starting to forward the packet to the MAC transmitter in Store-and-Forward mode without the checksum having been calculated yet. This second error condition only occurs when the Transmit FIFO depth is less than the length of the Ethernet packet being transmitted to avoid deadlock, the MTL starts forwarding the packet when the FIFO is full, even in the store-and-forward mode.
20 LoC	Loss of Carrier This bit indicates that Loss of Carrier occurred during packet transmission (that is, the MII_CRS signal was inactive for one or more transmit clock periods during packet transmission). This is valid only for the packets transmitted without collision and when the MAC operates in the half-duplex mode.
21 NC	No Carrier This bit indicates that the carrier sense signal form the PHY was not asserted during transmission.
22 LC	Late Collision This bit indicates that packet transmission was aborted because a collision occurred after the collision window (64 byte times including Preamble in MII mode). This bit is not valid if Underflow Error is set.
23 EC	Excessive Collision This bit indicates that the transmission was aborted after 16 successive collisions while attempting to transmit the current packet. If the DR bit is set in the MAC_Configuration register, this bit is set after first collision and the transmission of the packet is aborted.
24:27 CC	Collision Count This 4-bit counter value indicates the number of collisions occurred before the packet was transmitted. The count is not valid when the EC bit is set.
28 ED	Excessive Deferral This bit indicates that the transmission ended because of excessive deferral of over 24,288 bit times (155,680 bits times in 1000 Mbps mode or Jumbo Packet enabled mode) if DC bit is set in the MAC_Configuration register.
29 UF	Underflow Error This bit indicates that the MAC aborted the packet because the data arrived late from the system memory. The underflow error can occur because of either of the following conditions: The DMA encountered an empty Transmit Buffer while transmitting the packet The application filled the MTL Tx FIFO slower than the MAC transmit rate The transmission process enters the suspended state and sets the underflow bit corresponding to a queue in the MTL_Interrupt_Status register.

Table 1282. TDES3 Normal Descriptor (Write-Back Format) (continued)

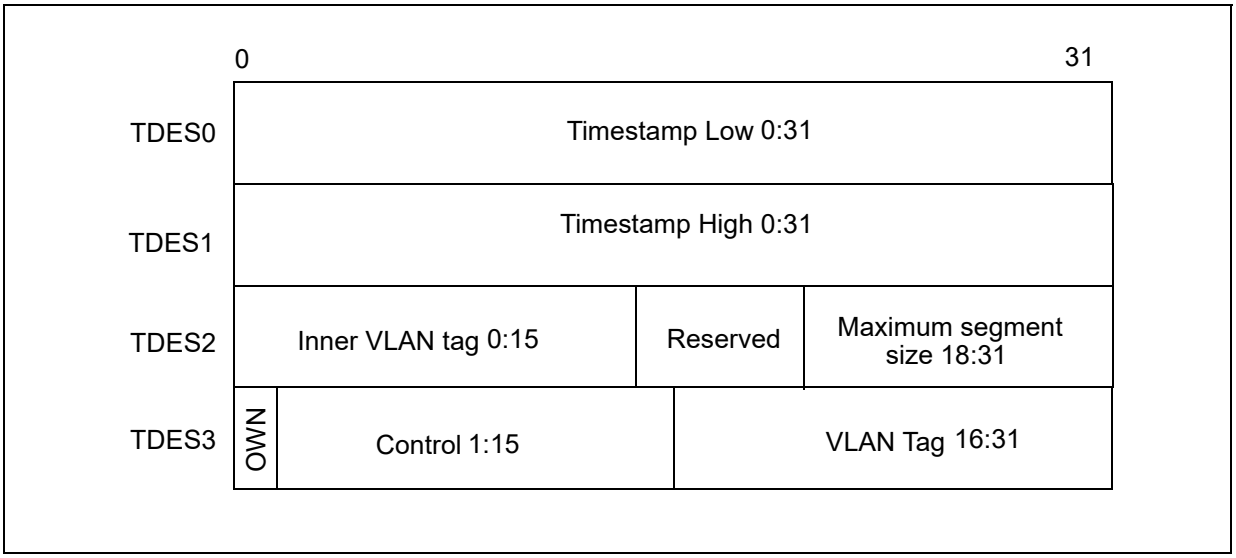
Field	Description
30 DB	Deferred Bit This bit indicates that the MAC deferred before transmitting because of presence of carrier. This bit is valid only in the half-duplex mode.
31 IHE	IP Header Error When IP Header Error is set, this bit indicates that the Checksum Offload engine detected an IP header error. This bit is valid only when Tx Checksum Offload is enabled. Otherwise, it is reserved. If COE detects an IP header error, it still inserts an IPv4 header checksum if the Ethernet Type field indicates an IPv4 payload.

56.4.1.3.2 Transmit Context Descriptor

The Transmit Context descriptor can be provided any time before a packet descriptor. The context is valid for the current packet and subsequent packets. The context descriptor is used to provide the timestamps for one-step timestamp correction, VLAN Tag ID for VLAN insertion feature, and SA insertion bit for SA insertion. Write back is done on a context descriptor only to reset the OWN bit.

Note: The VLAN Tag IDs and MSS values, provided by the application in a context descriptor with their corresponding Valid bits set, are stored internally by the DMA.
When the outer or inner VLAN tag is provided with the Valid bit set, the DMA always passes the last valid VLAN tag to the MTL. The application cannot invalidate the valid VLAN tag stored by the DMA. The VLAN tag is inserted or replaced based on the control inputs provided for the packet.
The Inner VLAN Tag Control input is used only for the next packet that immediately follows the context descriptor. The application must provide a context descriptor before the normal descriptor of each packet for which the DMA should use the inner VLAN Tag control input.

Figure 1216. Transmit Context Descriptor Format



56.4.1.3.2.1 TDES0 Context Descriptor

Table 1283. TDES0 Context Descriptor

Field	Description
0:31 TTSL	Transmit Packet Timestamp Low For one-step correction, the driver can provide the lower 32 bits of timestamp in this descriptor word. The DMA uses this value as the low word for doing one-step timestamp correction. This field is valid only if the OSTC and TCMSSV bits of TDES3 context descriptor are set.

56.4.1.3.2.2 TDES1 Context Descriptor

Table 1284. TDES1 Context Descriptor

Field	Description
0:31 TTSH	Transmit Packet Timestamp High For one-step correction, the driver can provide the upper 32 bits of timestamp in this descriptor. The DMA uses this value as the high word for doing one-step timestamp correction. This field is valid only if the OSTC and TCMSSV bits of TDES3 context descriptor are set.

56.4.1.3.2.3 TDES2 Context Descriptor

Table 1285. TDES2 Context Descriptor

Field	Description
0:15 IVT	Inner VLAN Tag When the IVLTV bit of TDES3 context descriptor is set and the TCMSSV and OSTC bits of TDES3 context descriptor are reset, TDES2[0:15] contains the inner VLAN Tag to be inserted in the subsequent Transmit packets.
16:31	Reserved

56.4.1.3.2.4 TDES3 Context Descriptor

Figure 1217. TDES3 Context Descriptor

0	1	2:3	4	5	6:7	8	9:13	14	15	16:31
OWN	CTXT	Rsvd	OSTC	TCMSSV	Rsvd	CDE	Rsvd	IVLTV	VLTV	VT

Table 1286. TDES3 Context Descriptor

Field	Description
0 OWN	Own Bit When this bit is set, it indicates that the Ethernet module DMA owns the descriptor. When this bit is reset, it indicates that the application owns the descriptor. The DMA clears this bit when either of the following conditions is true: The DMA completes the packet reception The buffers associated with the descriptor are full
1 CTXT	Context Type This bit should be set to 1'b1 for Context descriptor.
2:3	Reserved
4 OSTC	One-Step Timestamp Correction Enable When this bit is set, the DMA performs a one-step timestamp correction with reference to the timestamp values provided in TDES0 and TDES1.
5 TCMSSV	One-Step Timestamp Correction Input When this bit and the OSTC bit are set, it indicates that the Timestamp Correction input provided in TDES0 and TDES1 is valid.
6:7	Reserved
8 CDE	Context Descriptor Error When this bit is set, it indicates that the context descriptor was provided in the incorrect sequence and the DMA ignored it. The DMA sets this bit during write-back while closing the context descriptor.
9:11	Reserved
12:13 IVTIR	Inner VLAN Tag Insert or Replace When this bit is set, these bits request the MAC to perform Inner VLAN tagging or untagging before transmitting the packets. If the packet is modified for VLAN tags, the MAC automatically recalculates and replaces the CRC bytes. The following list describes the values of these bits: 00 Do not add the inner VLAN tag. 01 Remove the inner VLAN tag from the packets before transmission. This option should be used only with the VLAN frames. 10 Insert an inner VLAN tag with the tag value programed in the MAC_VLAN_Incl register or context descriptor. 11 Replace the inner VLAN tag in packets with the tag value programed in the MAC_Inner_VLAN_Incl register or context descriptor. This option should be used only with the VLAN frames.
14 IVLTV	Inner VLAN Tag Valid When this bit is set, it indicates that the IVT field of TDES2 is valid.
15 VLTV	VLAN Tag Valid When this bit is set, it indicates that the VT field of TDES3 is valid.
16:13 VT	VLAN Tag This field contains the VLAN Tag to be inserted or replaced in the packet. This field is used as VLAN Tag only when the VLTI bit of the MAC_VLAN_Incl register is reset.

56.4.1.4 Receive Descriptor

The DMA in the Ethernet module attempts to read a descriptor only if the Tail Pointer is different from the Base Pointer or current pointer. It is recommended to have a descriptor ring with a length that can accommodate at least two complete packets received by the MAC. Otherwise, the performance of the DMA is impacted greatly because of the unavailability of the descriptors. In such situations, the RxFIFO in MTL becomes full and starts dropping packets.

The following Receive Descriptors are present:

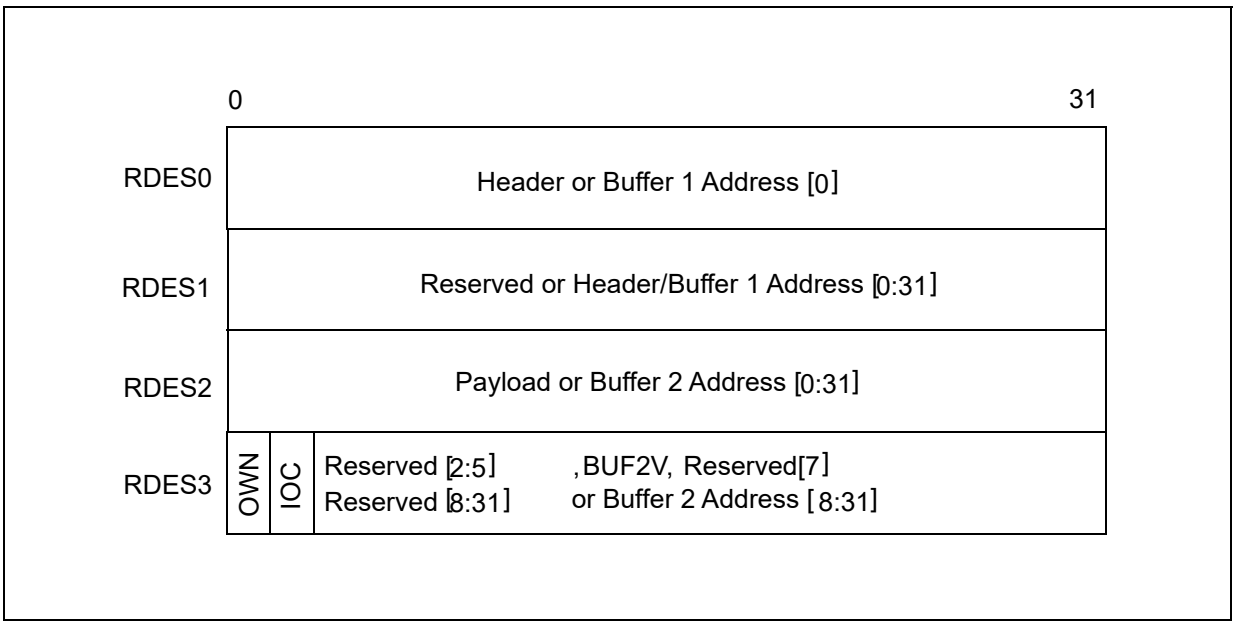
- Normal and
- Context descriptors

All RX descriptors are prepared by the software and given to the DMA as “Normal” Descriptors with the content as shown in *Section 56.4.1.4.1: Receive Normal Descriptor (Read Format)*. The DMA reads this descriptor and after transferring a received packet (or part of) to the buffers indicated by the descriptor, the Rx DMA will close the descriptor with the corresponding packet status. The format of this status is given in the *Section 56.4.1.4.2: Receive Normal Descriptor (Write-Back Format)*.

For some packets, the normal descriptor bits are not enough to write the complete status. For such packets, the RX DMA will write the extended status to the next descriptor (without processing or using the Buffers Pointers embedded in that descriptor). The format and content of this descriptor write back is described in *Section 56.4.1.4.3: Receive Context Descriptor*.

56.4.1.4.1 Receive Normal Descriptor (Read Format)

Figure 1218. Receive Normal Descriptor (Read Format)



Note: In the Receive Descriptor (Read Format), if the Buffer Address field is all 0s, the Ethernet module does not transfer data to that buffer and skips to the next buffer or next descriptor.

56.4.1.4.1.1 RDES0 Normal Descriptor (Read Format)

Table 1287. RDES0 Normal Descriptor (Read Format)

Field	Description
0:31 BUF1AP	Buffer 1 Address Pointer These bits indicate the physical address of Buffer 1. The application can program a byte-aligned address for this buffer which means that the LS bits of this field can be non-zero. However, while transferring the start of packet, the DMA performs a Write operation with RDES0[29:31] as zero. However, the packet data is shifted as per actual offset as given by buffer address pointer. If the address pointer points to a buffer where the middle or last part of the packet is stored, the DMA ignores the offset address and writes to the full location as indicated by the data-width.

56.4.1.4.1.2 RDES1 Normal Descriptor (Read Format)

Table 1288. RDES1 Normal Descriptor (Read Format)

Field	Description
0:31	This field is reserved.

56.4.1.4.1.3 RDES2 Normal Descriptor (Read Format)

Table 1289. RDES2 Normal Descriptor (Read Format)

Field	Description
0:31 BUF2AP	Buffer 2 Address Pointer These bits indicate the physical address of Buffer 2. RxDMA uses the LS Bits of the pointer address only while transferring the start bytes of a packet. If the BUF2AP is giving the address of a buffer in which the middle or last part of a packet is stored, the DMA ignores BUF2AP[29:31] and writes to the complete location.

56.4.1.4.1.4 RDES3 Normal Descriptor (Read Format)

Figure 1219. RDES3 Normal Descriptor (Read Format)

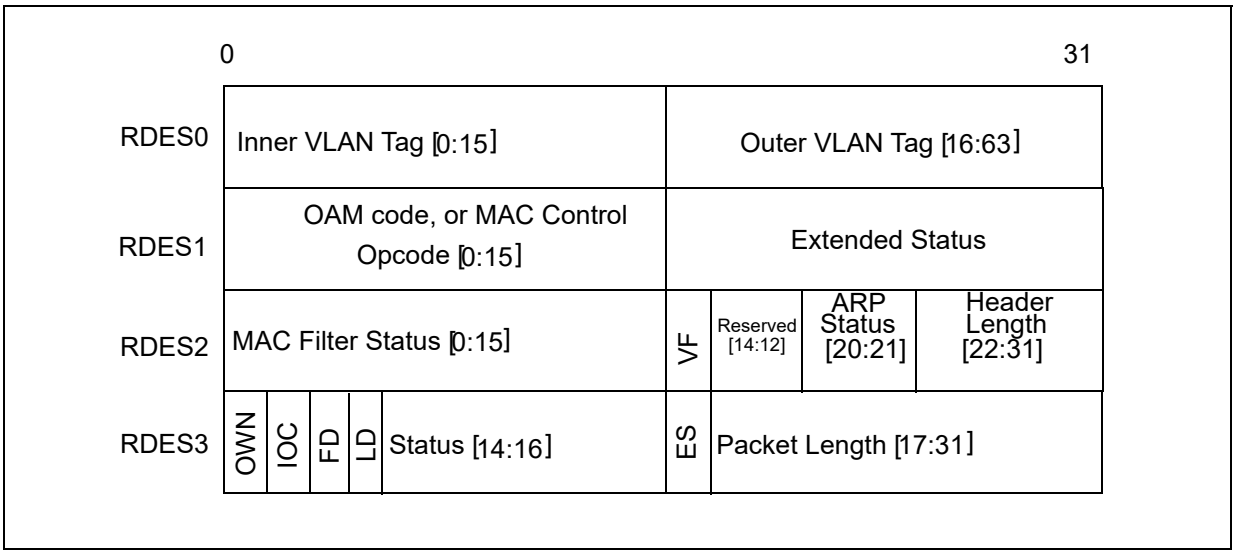
0	1	2:5	6	7	8:31
OWN	IOC	Rsvd	BUF2V	BUF1V	Rsvd

Table 1290. RDES3 Normal Descriptor (Read Format)

Field	Description
0 OWN	Own Bit When this bit is set, it indicates that the Ethernet module DMA owns the descriptor. When this bit is reset, it indicates that the application owns the descriptor. The DMA clears this bit when either of the following conditions is true: The DMA completes the packet reception The buffers associated with the descriptor are full
1 IOC	Interrupt Enabled on Completion When this bit is set, an interrupt is issued to the application when the DMA closes this descriptor.
2:5	Reserved
6 BUF2V	Buffer 2 Address Valid When this bit is set, it indicates to the DMA that the buffer 2 address specified in RDES2 is Valid. The application must set this bit so that the DMA can use the address, to which the Buffer 2 address in RDES2 is pointing, to write received packet data.
7 BUF1V	Buffer 1 Address Valid When set, this indicates to the DMA that the buffer 1 address specified in RDES1 is valid. The application must set this value if the address pointed to by Buffer 1 address in RDES1 can be used by the DMA to write received packet data.
8:31	Reserved

56.4.1.4.2 Receive Normal Descriptor (Write-Back Format)

Figure 1220. Receive Normal Descriptor (Write-Back Format)



56.4.1.4.2.1 RDES0 Normal Descriptor (Write-Back Format)

Figure 1221. RDES0 Normal Descriptor (Write-Back Format)

0:15	16:31
IVT	OVT

Table 1291. RDES0 Normal Descriptor (Write-Back Format)

Field	Description
0:15 IVT	Inner VLAN Tag This field contains the Inner VLAN tag of the received packet if the RS0V bit of RDES3 is set. This is valid only when Double VLAN tag processing and VLAN tag stripping are enabled.
16:31 OVT	Outer VLAN Tag This field contains the Outer VLAN tag of the received packet if the RS0V bit of RDES3 is set.

56.4.1.4.2.2 RDES1 Normal Descriptor (Write-Back Format)

The status fields in write-back format are valid only for the last descriptor (RDES3[28] is set).

Figure 1222. RDES1 Normal Descriptor (Write-Back Format)

0:15	16	17	18	19	20:23	24	25	26	27	28	29:31
OPC	TD	TSA	PV	PFT	PMT	IPCE	IPCB	IPV6	IPV4	IPHE	PT

Table 1292. RDES1 Normal Descriptor (Write-Back Format)

Field	Description
0:15 OPC	OAM Sub-Type Code, or MAC Control Packet opcode OAM Sub-Type Code If bits[13:15] of RDES3 are set to 111, this field contains the OAM sub-type and code fields. MAC Control Packet opcode If bits[13:15] of RDES3 are set to 110, this field contains the MAC Control packet opcode field.
16 TD	Timestamp Dropped This bit indicates that the timestamp was captured for this packet but it got dropped in the MTL Rx FIFO because of overflow. This bit is available only when you select the Timestamp feature. Otherwise, this bit is reserved.

Table 1292. RDES1 Normal Descriptor (Write-Back Format) (continued)

Field	Description
17 TSA	Timestamp Available When Timestamp is present, this bit indicates that the timestamp value is available in a context descriptor word 2 (RDES2) and word 1(RDES1). This is valid only when the Last Descriptor bit (RDES3 [3]) is set. The context descriptor is written in the next descriptor just after the last normal descriptor for a packet.
18 PV	PTP Version This bit indicates that the received PTP message has the IEEE 1588 version 2 format. When this bit is reset, it indicates the IEEE 1588 version 1 format. This bit is available only when you select the Timestamp feature. Otherwise, this bit is reserved.
19 PFT	PTP Packet Type This bit indicates that the PTP message is sent directly over Ethernet. This bit is available only when you select the Timestamp feature. Otherwise, this bit is reserved.
20:23 PMT	PTP Message Type These bits are encoded to give the type of the message received: 0000 No PTP message received 0001 SYNC (all clock types) 0010 Follow_Up (all clock types) 0011 Delay_Req (all clock types) 0100 Delay_Resp (all clock types) 0101 Pdelay_Req (in peer-to-peer transparent clock) 0110 Pdelay_Resp (in peer-to-peer transparent clock) 0111 Pdelay_Resp_Follow_Up (in peer-to-peer transparent clock) 1000 Announce 1001 Management 1010 Signaling 1011–1110 Reserved 1111 PTP packet with Reserved message type These bits are available only when you select the Timestamp feature.
24 IPCE	IP Payload Error When this bit is set, it indicates either of the following: The 16-bit IP payload checksum (that is, the TCP, UDP, or ICMP checksum) calculated by the MAC does not match the corresponding checksum field in the received segment. The TCP, UDP, or ICMP segment length does not match the payload length value in the IP Header field. The TCP, UDP, or ICMP segment length is less than minimum allowed segment length for TCP, UDP, or ICMP. Bit 16 (ES) of RDES3 is not set when this bit is set.
25 IPCB	IP Checksum Bypassed This bit indicates that the checksum offload engine is bypassed.
26 IPV6	IPv6 header Present This bit indicates that an IPV6 header is detected.
27 IPV4	IPV4 Header Present This bit indicates that an IPV4 header is detected.

Table 1292. RDES1 Normal Descriptor (Write-Back Format) (continued)

Field	Description
28 IPHE	IP Header Error When this bit is set, it indicates either of the following: The 16-bit IPv4 header checksum calculated by the MAC does not match the received checksum bytes. The IP datagram version is not consistent with the Ethernet Type value. Ethernet packet does not have the expected number of IP header bytes. This bit is valid when either bit 26 or bit 27 is set.
29:31 PT	Payload Type These bits indicate the type of payload encapsulated in the IP datagram processed by the Receive Checksum Offload Engine (COE): 000 Unknown type or IP/AV payload not processed 001 UDP 010 TCP 011 ICMP 110 AV Tagged Data Packet 111 AV Tagged Control Packet 101 AV Untagged Control Packet 100 IGMP if IPV4 Header Present bit is set else data center bridging (DCB) link layer discovery protocol (LLDP) Control Packet If the COE does not process the payload of an IP datagram because there is an IP header error or fragmented IP, it sets these bits to 000.

56.4.1.4.2.3 RDES2 Normal Descriptor (Write-Back Format)

Figure 1223. RDES2 Normal Descriptor (Write-Back Format)

0:2	3	4	5:12	13	14	15	16	17:20	21	22:31
L3L4FM	L4FM	L3FM	MADRM	HF	DAF	SAF	VF	Rsvd	ARPNR	Rsvd

Table 1293. RDES2 Normal Descriptor (Write-Back Format)

Field	Description
0:2 L3L4FM	Layer 3 and Layer 4 Filter Number Matched These bits indicate the number of the Layer 3 and Layer 4 Filter that matched the received packet: 000 Filter 0 001 Filter 1 010 Filter 2 011 Filter 3 100 Filter 4 101 Filter 5 110 Filter 6 111 Filter 7 This field is valid only when Bit 3 or Bit 4 is set high. When more than one filter matches, these bits give the number of lowest filter.
3 L4FM	Layer 4 Filter Match When this bit is set, it indicates that the received packet matches one of the enabled Layer 4 Port Number fields. This status is given only when one of the following conditions is true: Layer 3 fields are not enabled and all enabled Layer 4 fields match All enabled Layer 3 and Layer 4 filter fields match When more than one filter matches, this bit gives the layer 4 filter status of filter indicated by Bits[0:2].
4 L3FM	Layer 3 Filter Match When this bit is set, it indicates that the received packet matches one of the enabled Layer 3 IP Address fields. This status is given only when one of the following conditions is true: All enabled Layer 3 fields match and all enabled Layer 4 fields are bypassed All enabled filter fields match When more than one filter matches, this bit gives the layer 3 filter status of filter indicated by Bits[0:2].
5:12 MADRM	MAC Address Match or Hash Value When the HF bit is reset, this field contains the MAC address register number that matched the Destination address of the received packet. This field is valid only if the DAF bit is reset. When the HF bit is set, this field contains the hash value computed by the MAC. A packet passes the hash filter when the bit corresponding to the hash value is set in the hash filter register.
13 HF	Hash Filter Status When this bit is set, it indicates that the packet passed the MAC address hash filter. Bits[5:12] indicate the hash value.
14 DAF	Destination Address Filter Fail When this bit is set, it indicates that the packet failed the DA Filter in the MAC.
15 SAF	SA Address Filter Fail When this bit is set, it indicates that the packet failed the SA Filter in the MAC.
16 VF	VLAN Filter Status When this bit is set, it indicates that the VLAN Tag of received packet passed the VLAN filter.

Table 1293. RDES2 Normal Descriptor (Write-Back Format) (continued)

Field	Description
17:20	Reserved
21 ARPNR	ARP Reply Not Generated When this bit is set, it indicates that the MAC did not generate the ARP Reply for received ARP Request packet. This bit is set when the MAC is busy transmitting ARP reply to earlier ARP request (only one ARP request is processed at a time).
22:31	Reserved

56.4.1.4.2.4 RDES3 Normal Descriptor (Write-Back Format)

Figure 1224. RDES3 Normal Descriptor (Write-Back Format)

0	1	2	3	4	5	6	7	8	9	10	11	12	13:15	16	17:31
OWN	CTXT	FD	LD	RS2V	RS1V	RS0V	CE	GP	RWT	OE	RE	DE	LT	ES	PL

Table 1294. RDES3 Normal Descriptor (Write-Back Format)

Field	Description
0 OWN	Own Bit When this bit is set, it indicates that the Ethernet module DMA owns the descriptor. When this bit is reset, it indicates that the application owns the descriptor. The DMA clears this bit when either of the following conditions is true: The DMA completes the packet reception The buffers associated with the descriptor are full
1 CTXT	Receive Context Descriptor When this bit is set, it indicates that the current descriptor is a context type descriptor. The DMA writes 1'b0 to this bit for normal receive descriptor.
2 FD	First Descriptor When this bit is set, it indicates that this descriptor contains the first buffer of the packet. If the size of the first buffer is 0, the second buffer contains the beginning of the packet. If the size of the second buffer is also 0, the next descriptor contains the beginning of the packet.
3 LD	Last Descriptor When this bit is set, it indicates that the buffers to which this descriptor is pointing are the last buffers of the packet.
4 RS2V	Receive Status RDES2 Valid When this bit is set, it indicates that the status in RDES2 is valid and it is written by the DMA. This bit is valid only when the LD bit of RDES3 is set.
5 RS1V	Receive Status RDES1 Valid When this bit is set, it indicates that the status in RDES1 is valid and it is written by the DMA. This bit is valid only when the LD bit of RDES3 is set.
6 RS0V	Receive Status RDES0 Valid When this bit is set, it indicates that the status in RDES0 is valid and it is written by the DMA. This bit is valid only when the LD bit of RDES3 is set.

Table 1294. RDES3 Normal Descriptor (Write-Back Format) (continued)

Field	Description
7 CE	CRC Error When this bit is set, it indicates that a Cyclic Redundancy Check (CRC) Error occurred on the received packet. This field is valid only when the LD bit of RDES3 is set.
8 GP	Giant Packet When this bit is set, it indicates that the packet length exceeds the specified maximum Ethernet size of 1518, 1522, or 2000 bytes (9018 or 9022 bytes if jumbo packet enable is set). **Note:** Giant packet indicates only the packet length. It does not cause any packet truncation.
9 RWT	Receive Watchdog Timeout When this bit is set, it indicates that the Receive Watchdog Timer has expired while receiving the current packet. The current packet is truncated after watchdog timeout.
10 OE	Overflow Error When this bit is set, it indicates that the received packet is damaged because of buffer overflow in Rx FIFO. **Note:** This bit is set only when the DMA transfers a partial packet to the application. This happens only when the Rx FIFO is operating in the threshold mode. In the store-and-forward mode, all partial packets are dropped completely in Rx FIFO.
11 RE	Receive Error When this bit is set, it indicates that the MII_RXER signal is asserted while the MII_RX_DV signal is asserted during packet reception. This error also includes carrier extension error in the MII and half-duplex mode. Error can be of less or no extension, or error (rxd!= 0f) during extension.
12 DE	Dribble Bit Error When this bit is set, it indicates that the received packet has a non-integer multiple of bytes (odd nibbles). This bit is valid only in the MII Mode.
13:15 LT	Length/Type Field This field indicates if the packet received is a length packet or a type packet. The encoding of the 3 bits is as follows: 000 The packet is a length packet 001 The packet is a type packet. 011 The packet is a ARP Request packet type 100 The packet is a type packet with VLAN Tag 101 The packet is a type packet with Double VLAN Tag 110 The packet is a MAC Control packet type 111 The packet is a OAM packet type 010 Reserved

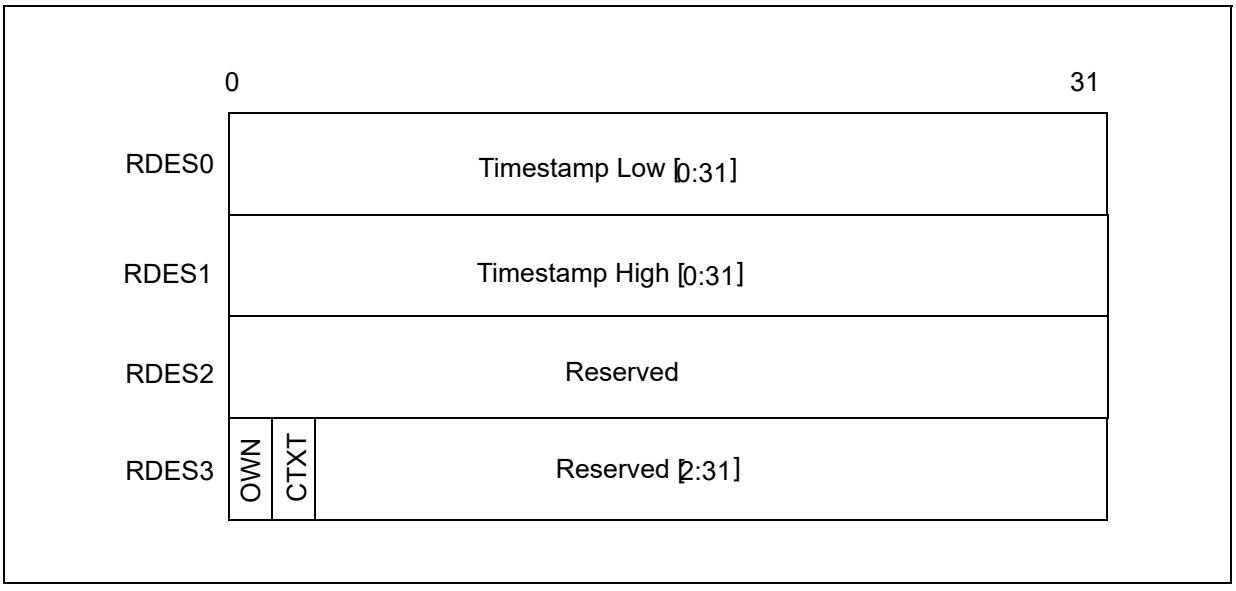
Table 1294. RDES3 Normal Descriptor (Write-Back Format) (continued)

Field	Description
16 ES	Error Summary When this bit is set, it indicates the logical OR of the following bits: RDES3[7]: CRC Error RDES3[12]: Dribble Error RDES3[11]: Receive Error RDES3[9]: Watchdog Timeout RDES3[10]: Overflow Error RDES3[8]: Giant Packet This field is valid only when the LD bit of RDES3 is set.
17:31 PL	Packet Length These bits indicate the byte length of the received packet that was transferred to system memory (including CRC). This field is valid when the LD bit of RDES3 is set and either the Descriptor Error (RDES3[18]) or Overflow Error bits are reset. The packet length also includes the two bytes appended to the Ethernet packet when IP checksum calculation is enabled and the received packet is not a MAC control packet. This field is valid when the LD bit of RDES3 is set. When the Last Descriptor and Error Summary bits are not set, this field indicates the accumulated number of bytes that have been transferred for the current packet.

56.4.1.4.3 Receive Context Descriptor

This descriptor is read-only for the application. Only DMA can write to this descriptor. The context descriptor provides information about the extended status related to the last received packet. The Bit 1 of RDES3 indicates the context type descriptor.

Figure 1225. Receive Context Descriptor



56.4.1.4.3.1 RDES0 Context Descriptor

Table 1295. RDES0 Context Descriptor

Field	Description
0:31 RTSL	Receive Packet Timestamp Low The DMA updates this field with least significant 32 bits of the timestamp captured for corresponding Receive packet. When this field and the RTSH field of RDES1 show all-ones value, the timestamp must be considered as corrupt.

56.4.1.4.3.2 RDES1 Context Descriptor

Table 1296. RDES1 Context Descriptor

Field	Description
0:31 RTSH	Receive Packet Timestamp High The DMA updates this field with most significant 32 bits of the timestamp captured for corresponding receive packet. When this field and the RTSL field of RDES0 show all-ones value, the timestamp must be considered as corrupt.

56.4.1.4.3.3 RDES2 Context Descriptor

Table 1297. RDES2 Context Descriptor

Field	Description
0:31	Reserved

56.4.1.4.3.4 RDES3 Context Descriptor

Figure 1226. RDES3 Context Descriptor

0	1	2:31
OWN	CTXT	Rsvd

Table 1298. RDES3 Context Descriptor

Field	Description
0 OWN	Own Bit When this bit is set, it indicates that the DMA owns the descriptor. When this bit is reset, it indicates that the application owns the descriptor. The DMA clears this bit when either of the following conditions is true: The DMA completes the packet reception The buffers associated with the descriptor are full
1 CTXT	Receive Context Descriptor When this bit is set, it indicates that the current descriptor is a context descriptor. The DMA writes 1'b1 to this bit for context descriptor.
2:31	Reserved

56.5 Programming Guidelines

This section provides the instructions for initializing the DMA or MAC registers in the proper sequence.

56.5.1 Initializing DMA

Complete the following steps to initialize the DMA:

1. Provide a software reset. This resets all of the MAC internal registers and logic (bit 31 of *DMA_Mode Register (DMA_MODE)*)
2. Wait for the completion of the reset process (poll bit 31 of the *DMA_Mode Register (DMA_MODE)*, which is only cleared after the reset operation is completed)
3. Program the following fields to initialize the *System Bus Mode Register (DMA_SYSBUS_MODE)*:
   a) AAL
   b) Fixed burst or undefined burst
   c) Burst mode values for AHB bus interface
4. Create a descriptor list for transmit and receive. In addition, ensure that the receive descriptors are owned by DMA (set bit 0 of descriptor TDES3/RDES3)

Note: Descriptor address from start to end of the ring should not cross the 4 GB boundary.

5. Program the Transmit and Receive Ring length registers_ (DMA_CH*n*_TxDesc_Ring_Length and DMA_CH*n*_RxDesc_Ring_Length). The ring length programed must be at least 4
6. Initialize receive and transmit descriptor list address with the base address of the transmit and receive descriptor (DMA_CH*n*_TxDesc_List_Address, DMA_CH*n*_RxDesc_List_Address). Also, program transmit and receive tail pointer

registers indicating to the DMA about the available descriptors (DMA_CH*n*_TxDesc_Tail_Pointer and DMA_CH*n*_RxDesc_Tail_Pointer)

7. Program the settings of the following registers for the parameters such as maximum burst-length (PBL) initiated by DMA, descriptor skip lengths, OSP in case of TxDMA, RBSZ in case of RxDMA, and so on:
   a) DMA_CH*n*_TX_Control
   b) DMA_CH*n*_TX_Control
   c) DMA_CH*n*_TX_Control
8. Enable the interrupts by programming the DMA_CH*n*_Interrupt_Enable register
9. Start the Receive and Transmit DMAs by setting SR (bit 31) of the DMA_CH*n*_RX_Control and ST (bit 31) of the DMA_CH*n*_TX_Control register
10. Repeat steps *4* to *9* for all the Tx DMA and Rx DMA channels selected in the hardware

56.5.2 Initializing MTL Registers

Complete the following steps to initialize the MTL Registers:

1. Program the Tx Scheduling (SCHALG) and Receive Arbitration Algorithm (RAA) fields in MTL_Operation_Mode to initialize the MTL operation for multiple Tx and Rx queues
2. Program the Receive Queue to DMA mapping in the MTL_RxQ_DMA_Map0 register
3. Program the following fields to initialize the mode of operation in the MTL_TxQ0_Operation_Mode register:
   a) Transmit Store And Forward (TSF) or Transmit Threshold Control (TTC) in case of threshold mode
   b) Transmit Queue Enable (TXQEN) to value 2‘b10 to enable Transmit Queue0
   c) Transmit Queue Size (TQS)
4. Program the following fields to initialize the mode of operation in the MTL_RxQ0_Operation_Mode register:
   a) Receive Store and Forward (RSF) or RTC in case of Threshold mode
   b) Flow Control Activation and De-activation thresholds for MTL Receive FIFO (RFA and RFD)
   c) Error Packet and undersized good Packet forwarding enable (FEP and FUP)
   d) Receive Queue Size (RQS)
5. Repeat previous two steps for the second MTL Tx and Rx queue

56.5.3 Initializing MAC

The following MAC Initialization operations can be performed after DMA initialization. If the MAC initialization is completed before the DMA is configured, enable the MAC receiver (last step in the following sequence) only after the DMA is active. Otherwise, received frames fill the Rx FIFO and overflow.

1. Provide the MAC address registers: MAC_Address0_High and MAC_Address0_Low. For more than one MAC address program the MAC addresses appropriately
2. Program the following fields to set the appropriate filters for the incoming frames in the MAC_Packet_Filter register:
   a) Receive All
   b) Promiscuous mode
   c) Hash or Perfect Filter
   d) Unicast, multicast, broadcast, and control frames filter settings
3. Program the following fields for proper flow control in the MAC_Q0_Tx_Flow_Ctrl register:
   a) Pause time and other Pause frame control bits
   b) Transmit Flow control bits
   c) Flow Control Busy
4. Program the MAC_Interrupt_Enable register, as required
5. Program the appropriate fields in the MAC_Configuration register. For ex: Inter-packet gap while transmission and jabber disable.
6. Set bit 31 and 30 in MAC_Configuration registers to start the MAC transmitter and receiver

For normal operation, complete the following steps:

1. For normal transmit and receive interrupts, read the interrupt status. Then, poll the descriptors, reading the status of the descriptor owned by the Host (either transmit or receive)
2. Set appropriate values for the descriptors, ensuring that transmit and receive descriptors are owned by the DMA to resume the transmission and reception of data
3. If the descriptors are not owned by the DMA (or no descriptor is available), the DMA goes into SUSPEND state. The transmission or reception can be resumed by freeing the descriptors and writing the descriptor tail pointer to Tx/Rx tail pointer register
   a) DMA_CH*n*_TxDesc_Tail_Pointer
   b) DMA_CH*n*_RxDesc_Tail_Pointer
4. The values of the current host transmitter or receiver descriptor address pointer can be read for the debug process
   a) DMA_CH*n*_Current_App_TxDesc
   b) DMA_CH*n*_Current_App_RxDesc
5. The values of the current host transmit buffer address pointer and receive buffer address pointer can be read for the debug process
   a) DMA_CH*n*_Current_App_TxBuffer
   b) DMA_CH*n*_Current_App_RxBuffer

56.5.3.1 For Stopping and Starting Transmission

Complete the following steps to pause the transmission for some time.

Note: The steps are provided for Channel 0.

1. Disable the transmit DMA (if applicable) by clearing Bit 31 (ST) of the DMA_CH*n*_TX_Control register
2. Wait for any previous frame transmissions to complete. You can check this by reading the appropriate bits of the MTL_TxQ0_Debug register (TRCSTS is not 01 and TXQSTS=0)
3. Disable the MAC transmitter and MAC receiver by clearing Bit 31 (RE) and Bit 30(TE) of the MAC_Configuration register
4. Disable the receive DMA (if applicable), after making sure that the data in the Rx FIFO is transferred to the system memory (by reading the appropriate bits of the MTL_RxQ0_Debug register, PRXQ=0 and RXQSTS=00)
5. Make sure that both Tx Queue and Rx Queue are empty (TXQSTS is 0 in the MTL_TxQ0_Debug register and RXQSTS is 0 in the MTL_RxQ0_Debug register)
6. To restart the operation, first start the DMAs, and then enable the MAC transmitter and receiver

Note: Do not change the configuration (such as duplex mode, speed, port, or loop back) when the MAC is actively transmitting or receiving. The Software should change these parameters only when the MAC transmitter and receiver are not active.
Similarly the DMA related configuration should not be changed when Transmit and Receive DMA are in active.

56.5.4 Programming Guidelines for Multi-Channel Multi-Queuing

Following are the steps for Transmit:

1. Program the Transmit queue size in the TQS field of MTL_TxQ*n*_Operation_Mode register. Based on the value programed in TQS field, the size of the queue is determined. In the Transmit operation, the number of channels is equal to the number of the queues. Due to this reason, the Channel to Queue mapping is fixed
2. Enable ST bit of DMA_CH*n*_TX_Control register and corresponding TXQEN in MTL_TxQ*n*_Operation_Mode register
3. The scheduling method need to be programed in SCHALG of the MTL_Operation_Mode register
4. In case of CBS algorithm in AVB queues, the following registers also needs to be programed as required:
   a) MTL_TxQ*n*_ETS_Control
   b) MTL_TxQ*n*_SendSlopeCredit
   c) MTL_TxQ*n*_HiCredit
   d) MTL_TxQ*n*_LoCredit

Following are the steps for Receive:

1. Program the Receive queue size in the RQS field of MTL_RxQ*n*_Operation_Mode Register. Based on the value programed in RQS field, the size of the queue is determined
2. Enable the Receive Queues 0 to 1 in the fields RXQ0EN to RXQ1EN in MAC_RxQ_Ctrl0 Register for AV. In DMA configurations, SR bit of statically or

dynamically mapped DMA_CH*n*_RX_Control Register and corresponding RXQ*n*_EN in MAC_RxQ_Ctrl0 Register needs to be enabled

3. The MAC routes the Rx packets to the Rx Queues based on following packet types:
   a) AV PTP Packets: Based on the programming of AVPTPQ in MAC_RxQ_Ctrl1 Register
   b) AV Untagged Control packets: Based on the programming of AVCPQ in MAC_RxQ_Ctrl1 register
   c) VLAN Tag Priority field in VLAN Tagged packets: Program PSRQ1-0 of the MAC_RxQ_Ctrl2 register for the routing of tagged packets based on the USP (user Priority) field of the received packets to the Rx Queues 0 to 1
   d) The AV tagged control and data packets are also routed based on PSRQ field of the MAC_RxQ_Ctrl2 register
4. If multiple RX DMA channels are enabled, the following programming should be done for proper arbitration and mapping:
   a) Program the RAA field of MTL_Operation_Mode register to select the arbitration algorithm to decide which RxQ is read out from the RxFIFO memory
   b) Program the MTL_RxQ*n*_Control to decide the weights and the packet arbitration for each RxQ
   c) If static mapping is programed in MTL_RxQ_DMA_Map0 register (RXQ*n*DADMACH is reset to 0), bits RXQx2DMA and others need to be programed to select the channel for which each queue is mapped
   d) Set RXQ*n*DADMACH bit in MTL_RxQ_DMA_Map0 Register to select dynamic mapping of packets in each RxQueue
   e) In dynamic channel mapping, the routing of a packet to a specific RxDMA channel is decided by the value of DCS field in the lowest MAC Address Register

Note: The priorities set in PSRQ1-0 should be unique.

56.5.5 Programming Guidelines for MII Link State Transitions

The following steps are to be performed when the link is down but the Transmit and Receive clocks are running:

Note: The steps are provided for Channel 0.

1. Disable the Transmit DMA (if applicable) by clearing Bit 31 (ST) of DMA_CH*n*_TX_Control Register
2. Disable the MAC receiver by clearing Bit 29 (RE) of the MAC_Configuration register
3. Wait for any previous frame transmissions to complete. You can check this by reading the appropriate bits of MTL_TxQ0_Debug register (TRCSTS is not 01).
   -or-
   Flush the Tx FIFO for faster empty operation
4. Disable the MAC transmitter by clearing Bit 30 (TE) of the MAC_Configuration register
5. Make sure that both Tx Queue and Rx Queue are empty (TXQSTS is 0 in the MTL_TxQ0_Debug register and RXQSTS is 0 in the MTL_RxQ0_Debug register)
6. After the link is up, read the PHY registers to know the latest configuration and accordingly program the MAC registers
7. Restart the operation by starting the Tx DMA, and then enabling the MAC Transmitter and Receiver

You do not need to disable the Rx DMA. As the Receiver is disabled, the FIFO does not get any data in the Rx FIFO.

Do the following steps when the link is down and the Transmit and Receive clocks are stopped:

Note: The steps are provided for Channel 0.

1. Disable the MAC Transmitter and Receiver by clearing bits RE & TE of MAC_Configuration register. This will not take effect immediately as the clocks are absent
2. Wait till the link is up and the clocks are restored
3. Wait for the completion of the transfer of any partial frame if any at time of stopping of Transmit/Receive clock. This can be checked by reading the MAC_Debug Register (should be all-zero). Some old packets may still remain in the TXFIFO as the MAC Transmitter is stopped
4. Read the PHY registers to know the latest operating mode and accordingly program the MAC registers
5. Restart the MAC Transmitter and Receiver by setting RE & TE bits

56.5.6 Programming Guidelines for IEEE 1588 Timestamping

56.5.6.1 Initialization Guideline for System Time Generation

The timestamp feature is enabled by setting Bit 31 of the MAC_Timestamp_Control register. However, it is essential that the timestamp counter should be initialized after this bit is set. Complete the following steps during for initialization:

1. Mask the Timestamp Trigger interrupt by clearing the bit 15 of MAC_Interrupt_Enable Register
2. Set Bit 31 of MAC_Timestamp_Control register to enable timestamping
3. Program MAC_Sub_Second_Increment Register based on the PTP clock frequency
4. If you are using the Fine Correction approach, program MAC_Timestamp_Addend and set Bit 26 of MAC_Timestamp_Control register
5. Poll the MAC_Timestamp_Control register until Bit 26 is cleared
6. Program Bit 30 of MAC_Timestamp_Control register to select the Fine Update method (if required)
7. Program MAC_System_Time_Seconds_Update Register and MAC_System_Time_Nanoseconds_Update Register with the appropriate time value
8. Set Bit 29 in MAC_Timestamp_Control register.
   The timestamp counter starts operation as soon as it is initialized with the value written in the Timestamp Update registers.
   If one-step timestamping is enabled:
   a) To enable one-step timestamping, program Bit 4 of the TDES Context Descriptor
   b) Program registers MAC_Timestamp_Ingress_Asym_Corr and MAC_Timestamp_Egress_Asym_Corr to update the correction field in PDelay_Req PTP messages
9. Enable the MAC receiver and transmitter for proper timestamping

Note: If timestamp operation is disabled by clearing Bit 31 of MAC_Timestamp_Control Register, you need to repeat all these steps to restart the timestamp operation.

56.5.6.2 System Time Correction

To synchronize or update the system time in one process (coarse correction method), complete the following steps:

1. Set the offset (positive or negative) in the Timestamp Update registers (MAC_System_Time_Seconds_Update and MAC_System_Time_Nanoseconds_Update)
2. Set Bit 28 (TSUPDT) of the MAC_Timestamp_Control Register.
   The value in the Timestamp Update registers is added to or subtracted from the system time when the TSUPDT bit is cleared

To synchronize or update the system time to reduce system-time jitter (fine correction method), complete the following steps:

1. With the help of the algorithm explained in System Time Register Module, calculate the rate by which you want to make the system time increments slower or faster.
2. Update the MAC_Timestamp_Addend with the new value and set Bit 26 of the MAC_Timestamp_Control Register
3. Wait for the time for which you want the new value of the Addend register to be active. You can do this by enabling the Timestamp Trigger interrupt after the system time reaches the target value
4. Program the required target time in the MAC_PPS0_Target_Time_Seconds register and the MAC_PPS0_Target_Time_Nanoseconds register
5. Enable the Timestamp interrupt in bit 19 of MAC_Interrupt_Enable register
6. Set bit 27 in Register MAC_Timestamp_Control
7. When this trigger causes an interrupt, read MAC_Interrupt_Status Register
8. Reprogram MAC_Timestamp_Addend Register with the old value and set bit 26 again

56.5.7 Programming Guidelines for AV Feature

Complete the following steps to initialize the DMA for AV Feature:

1. Provide a software reset to reset all QOS internal registers and logic (bit 31 in DMA_Mode register)
2. Wait for the completion of the reset process. Poll bit 31 of the DMA_Mode register, which is cleared only after the reset operation is completed
3. Program the fields to initialize the DMA register by setting the values in DMA_Mode register
4. Create a proper descriptor list for transmit and receive. In addition, ensure that the DMA owns the Transmit and Receive descriptors. When OSF mode is used, at least two TX descriptors are required
5. Make sure that your software creates three or more different transmit or receive descriptors in the list before reusing any of the descriptors
6. Program the Transmit and Receive Ring length registers. The ring length programed must be at least 4. Then initialize receive and transmit descriptor list address with the

base address of transmit and receive descriptor. Finally program the Transmit and Receive tail pointer registers indicating to the DMA about the available descriptors:

a) DMA_CH*n*_TxDesc_Ring_Length
b) DMA_CH*n*_RxDesc_Ring_Length
c) DMA_CH*n*_TxDesc_List_Address
d) DMA_CH*n*_RxDesc_List_Address
e) DMA_CH*n*_TxDesc_Tail_Pointer
f) DMA_CHn_RxDesc_Tail_Pointer

7. Program the following fields to initialize the mode of operation in the MTL_TxQ1_Operation_Mode register:
   a) Transmit Store And Forward (TSF)
   b) Transmit Threshold Control (TTC)
   c) Transmit Queue Enable (TXQEN) to value 2'b10 to enable Transmit Queue0
   d) Transmit Queue Size (TQS)
8. Enable the interrupts by programming the DMA_CH*n*_Interrupt_Enable register
9. Repeat steps *4* through *9* for all additional channels of AV feature.
10. Program the CBS control register, idleSlope, sendSlope, hiCredit, and loCredit registers of the AV Queues
11. Start the Receive and Transmit DMA by setting bit 31 of the DMA_CH*n*_TX_Control register and bit 31 of the DMA_CH*n*_RX_Control register

56.5.7.1 Enabling Slot Number Checking

The slot number check feature is used to specify the intervals at which the DMA Channels mapped to AV Queues fetches the frames from the AHB system bus. This feature is useful for a uniform and periodic transfer of the AV traffic from the host memory. The feature is available only when you enable time-stamping and program the MAC_Sub_Second_Increment register. Complete the following steps to enable the slot number checking:

Note: These steps should be completed after Step 10 and before Step 11 of Section 56.5.7: Programming Guidelines for AV Feature.

1. Enable time-stamping by following the steps described in *Section 56.5.6.1: Initialization Guideline for System Time Generation*
2. Make sure that the SLOTNUM field (bits 9:12) of TDES3 Normal Descriptor (Read Format) contains a valid slot number. You can read the current reference slot number from the DMA_CH*n*_Slot_Function_Control_Status
3. Set Bit 31 (ESC) of the Slot Function Control and Status register of a channel to enable the slot number checking

56.5.7.2 Enabling Average Bits Per Slot Reporting

The CBS Status register of the additional AV channels provides information about the average bits that are transmitted in a slot. The software can asynchronously read this register to retrieve information about the average bits transmitted per slot. Complete the following steps to enable average bits per slot reporting:

1. Enable time-stamping by following the steps described in *Section 56.5.6.1: Initialization Guideline for System Time Generation*
2. Program Bits [25:27], SLC, of the MTL_TxQ*n*_ETS_Control register of a channel with number of slots over which the average transmitted bits per slot need to be computed
3. Enable Bit 22 (ABPSSIE) of the MTL_Q*n*_Interrupt_Control_Status register of a channel to generate the average bits per slot interrupt
4. Read Bits [15:31], ABS, from the MTL_TxQ*n*_ETS_Status register of a channel on each interrupt

Note: The software can read the ABS bits in polling mode even if the ABPSIE bit is not enabled. When high, bit 30 (ABPSIS) of the MTL_TxQn_ETS_Status register indicates that a new value is updated in the ABS field.

The frequency of this interrupt depends on the value programed in the Step 2. For example, when you program value 0 in the SLC field, the interrupt is generated at every 125 microsecond.
When not required, you can disable this interrupt to stop the interrupt flooding.

56.5.8 Programming Guidelines for Energy Efficient Ethernet

56.5.8.1 Entering and Exiting the Tx LPI Mode

For entering and exiting the Tx LPI Mode, complete the following steps:

1. Read the PHY register through the MDIO interface, check if the remote end has the EEE capability, and then negotiate the timer values
2. Program the PHY registers through the MDIO interface (including the RX_CLK_stoppable bit that indicates to the PHY whether to stop Rx clock in LPI mode)
3. Program Bits[6:15] and Bits[16:31] in MAC_LPI_Timers_Control register
4. Read the link status of the PHY chip by using the MDIO interface and update Bit 14 of MAC_LPI_Control_Status accordingly. This update should be done whenever the link status in the PHY chip changes
5. Program the MAC_1US_Tic_Counter as per the frequency of the clock used for accessing the CSR slave port
6. Program the MAC_LPI_Entry_Timer register (LPIET) with the IDLE time for which the MAC should wait before entering the LPI state on its own
7. Set LPIET and LPITXA (bit[11:12]) of MAC_LPI_Control_Status register to enable the auto-entry into LPI and auto-exit of MAC from LPI state
8. Set Bit 15 of MAC_LPI_Control_Status register to make the MAC Transmitter enter the LPI state.
   The MAC enters the LPI mode after completing all scheduled packets and remains IDLE for the time indicated by LPIET. It sets the TLPIEN (bit[31]) after entry to LPI state
9. When a packet is scheduled for transmission (when the TxDMA comes out of IDLE state or when a packet is presented at ATI or MTI interface), the MAC Transmitter exits LPI state automatically. It waits for TWT time before setting the TLPIEX interrupt status bit and then resume the packet transmission
10. MAC Transmitter re-enters LPI state if it remains IDLE for LPIET time and sets the TLPIEN bit and the entry-exit cycle continues
11. Reset LPITXEN in case the application wants to over-ride the auto-entry/exit modes and make the MAC Transmitter exit the LPI state directly

Note: To make the MAC enter the LPI state only after it completes the transmission of all queued frames in the Tx FIFO, you should set Bit 12 in MAC_LPI_Control_Status register.
To switch off the MII Transmit Clock during the LPI state, use the sbd_tx_clk_gating_ctrl_o signal for gating the clock input.
To switch off the CSR clock or power to the rest of the system during the LPI state, you should wait for the TLPIEN interrupt of MAC_LPI_Control_Status register to be generated. Restore the clocks before performing step 6 when you want to come out of the LPI state.

56.5.8.2 Gating Off the CSR Clock in the LPI Mode

You can gate off the CSR clock to save the power when the MAC is in the Low-Power Idle (LPI) mode.

56.5.8.3 Gating Off the CSR Clock in the Rx LPI Mode

The following operations are performed when the MAC receives the LPI pattern from the PHY.

1. The MAC RX enters the LPI mode and the Rx LPI entry interrupt status [RLPIEN interrupt of MAC_LPI_Control_Status register is set
2. The interrupt pin is asserted. The interrupt is cleared when the host reads the MAC_LPI_Control_Status register

After the interrupt is asserted and the MAC Tx is also in the LPI mode, you can gate-off the CSR clock. If the MAC TX is not in the LPI mode when you gate off the CSR clock, the events on the MAC transmitter do not get reported or updated in the CSR.

For restoring the CSR clock, wait for the LPI exit indication from the PHY after which the MAC asserts the LPI exit interrupt on lpi_intr_o (synchronous to clk_rx_i). The lpi_intr_o interrupt is cleared when MAC_LPI_Control_Status register is read.

56.5.8.4 Gating Off the CSR Clock in the Tx LPI Mode

The following operations are performed when Bit 15 (LPIEN) of MAC_LPI_Control_Status register is set:

1. The Transmit LPI Entry interrupt (TLPIEN bit of MAC_LPI_Control_Status register) is set
2. The interrupt pin is asserted. The interrupt is cleared when the host reads the MAC_LPI_Control_Status register

After the interrupt is asserted and the MAC RX is also in the LPI mode, you can gate off the CSR clock. If the MAC RX is not in the LPI mode when you gate off the CSR clock, the events on the MAC receiver do not get reported or updated in the CSR.

For restoring the CSR clock, switch on the CSR clock when the MAC has to come out of the TX LPI mode. After the CSR clock is resumed, reset Bit 15 (LPIEN) of MAC_LPI_Control_Status register to bring the MAC out of the LPI mode.

56.5.9 Programming Guidelines for Flexible Pulse-Per-Second (PPS) Output

56.5.9.1 Generating single Pulse on PPS

1. Program 11 or 10 (for interrupt) in Bits [25:26], TRGTMODSEL, of the MAC_PPS_Control register. This instructs the MAC to use the Target Time registers

(MAC_PPS0_Target_Time_Seconds and MAC_PPS0_Target_Time_Nanoseconds) for start time of PPS signal output

2. Program the start time value in the Target Time registers (MAC_PPS0_Target_Time_Seconds and MAC_PPS0_Target_Time_Nanoseconds)
3. Program the width of the PPS signal output in MAC_PPS0_Width Register
4. Program Bits [28:31], PPSCMD, of MAC_PPS_Control to 0001. This instructs the MAC to generate single pulse on the PPS signal output at the time programed in the Target Time registers

When the PPSCMD is executed (PPSCMD bits = 0), you can cancel the pulse generation by giving the Cancel Start Command (PPSCMD=0011) before the programed start time elapses. You can also program the behavior of the next pulse in advance. To program the next pulse:

1. Program the start time for the next pulse in the Target Time registers. This time should be more than the time at which the falling edge occurs for the previous pulse
2. Program the width of the next PPS signal output in MAC_PPS0_Width
3. Program Bits [28:3], PPSCMD, of MAC_PPS_Control to generate a single pulse after the time at which the previous pulse is de-asserted. This instructs the MAC to generate single pulse on the PPS signal output, at the time programed in Target Time registers

If you give this command before the previous pulse becomes low, then the new command overwrites the previous command and the QOS may generate only 1 extended pulse.

56.5.9.2 Generating a pulse train on PPS

1. Program 11 or 10 (for interrupt) in Bits [25:26], TRGTMODSEL, of MAC_PPS_Control register. This instructs the MAC to use the Target Time registers (MAC_PPS0_Target_Time_Seconds and MAC_PPS0_Target_Time_Nanoseconds) for start time of the PPS signal output
2. Program the start time value in the Target Time registers (MAC_PPS0_Target_Time_Seconds and MAC_PPS0_Target_Time_Nanoseconds)
3. Program the interval value between the train of pulses on the PPS signal output in MAC_PPS0_Interval Register
4. Program the width of the PPS signal output in MAC_PPS0_Width Register
5. Program Bits[28:3], PPSCMD, of MAC_PPS_Control Register to 0010. This instructs the MAC to generate train of pulses on the PPS signal output with start time programed in Target Time registers.
   By default, the PPS pulse train is free-running unless stopped by ‘STOP Pulse train at time’ or ‘STOP Pulse Train immediately’ commands.
6. Program the stop value in the Target Time registers. Ensure that Bit 0 (TSTRBUSY) of MAC_PPS0_Target_Time_Nanoseconds Register is reset before programming the Target Time registers again.
7. Program the PPSCMD field (bit 28:31) of MAC_PPS_Control to 0100. This stops the train of pulses on PPS signal output after the programed stop time specified in Step *6* elapses.

You can stop the pulse train at any time by programming 0101 in the PPSCMD field. Similarly, you can cancel the Stop Pulse train command (given in Step 7) by programming 0110 in the PPSCMD field before the time (programed in Step 6) elapses. You can cancel the pulse train generation by programming 0011 in the PPSCMD field before the programed start time (in Step 2) elapses.

56.5.9.3 Generating an Interrupt without Affecting the PPS

The Bits [25:26], TRGTMODSEL, of the MAC_PPS_Control Register enable you to program the Target Time registers (MAC_PPS0_Target_Time_Seconds and MAC_PPS0_Target_Time_Nanoseconds) to do any one of the following:

- Generate only interrupts
- Generate interrupts and the PPS start and stop time
- Generate only PPS start and stop time

To program the Target Time registers to generate only interrupt event:

1. Program 00 (for interrupt) in Bits [25:26], TRGTMODSEL, of MAC_PPS_Control register. This instructs the MAC to use the Target Time registers for target time interrupt
2. Program a target time value in the Target Time registers. This instructs the MAC to generate an interrupt when the target time elapses.
   If Bits [25:26], TRGTMODSEL, are changed (for example, to control the PPS), then the interrupt generation is over-written with the new mode and new programed Target Time register value

56.5.10 Programming Guidelines for VLAN filtering on Receive

1. Program MAC_VLAN_Tag Register for the following bit to select the filtering method:
   a) ETV: Enable 12-Bit VLAN Tag Comparison or 16-bit VLAN Tag comparison. VTHM: VLAN Tag Hash Table Match Enable
   b) ERIVLT: Enable inner VLAN Tag or outer VLAN Tag (to enable the inner or outer VLAN Tag filtering, Double VLAN Processing should enabled by setting EDVLP)
   c) ERSVLM: Enable Receive S-VLAN Match or C-VLAN match (for S-VLAN processing to be enabled, set ESVL)
   d) DOVLTC: Ignores VLAN Type for Tag Match
   e) VTIM: to enable VLAN Tag Inverse Match instead of the normal VLAN Tag matching
2. Program VL of MAC_VLAN_Tag register for the 12 bit or 16 bit VLAN tag.
3. If Hash filtering of VLAN tag is enabled, program MAC_VLAN_Hash_Table Register. The upper four bits of the calculated CRC are used to index the contents of the VLAN Hash table. For example, a hash value of 4b'1000 selects Bit 8 of the VLAN Hash table

57 FlexRay communication controller (FlexRay)

57.1 Introduction

57.1.1 Reference

The following documents are referenced in this chapter. They may be consulted for further information on the FlexRay communication controller's operation.

- FlexRay Communications System Protocol Specification, Version 2.1 Rev A[24]
- FlexRay Communications System Electrical Physical Layer Specification, Version 3.0

57.1.2 Color coding

Throughout this chapter, some items are displayed in an italicized, colored font.

FlexRay protocol parameters, constants, and variables are highlighted with *blue italics*. An example is the parameter *gdActionPointOffset*.

FlexRay protocol states are highlighted in *green italics*. An example is the state *POC:normal active*.

57.1.3 Overview

The Communication Controller (CC) is a FlexRay communication controller that implements the *FlexRay Communications System Protocol Specification, Version 2.1 Rev A.*

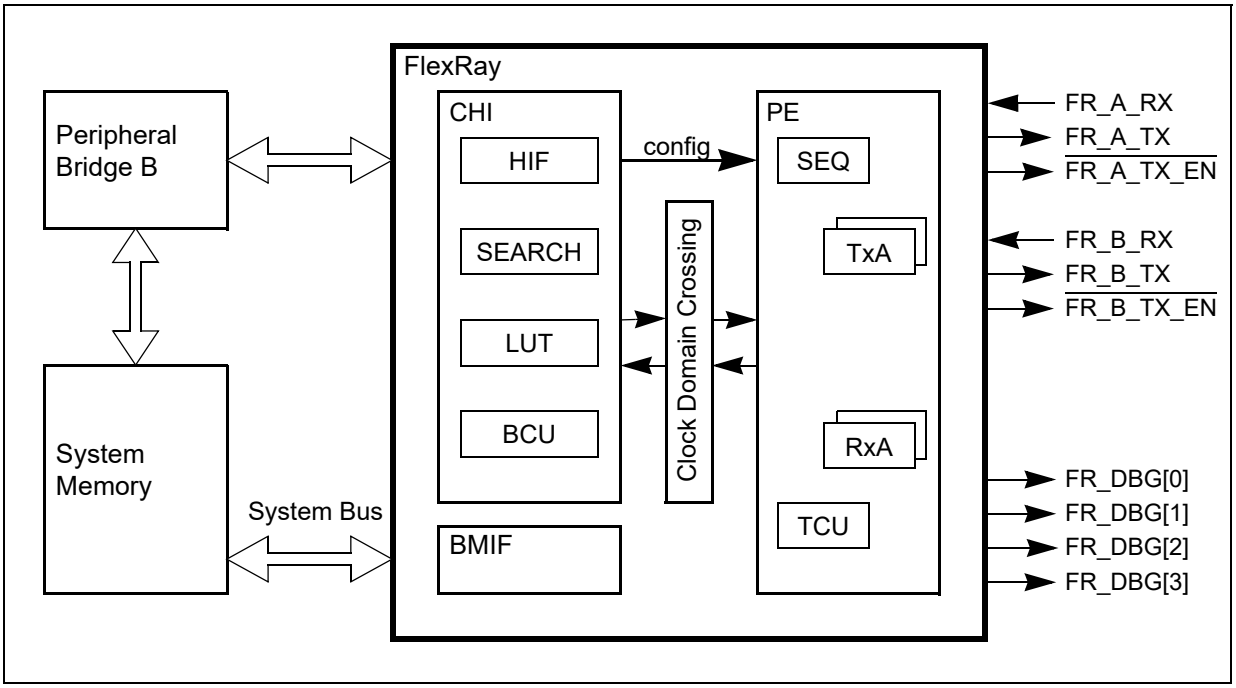
The CC has three main components:

- Controller Host Interface (CHI)
- Protocol Engine (PE)
- Clock Domain Crossing unit (CDC)

A block diagram of the CC with its surrounding modules is given in *Figure 1227*.

24. The FlexRay specifications have been developed for automotive applications. The FlexRay specifications have been neither developed nor tested for non-automotive applications.

Figure 1227. FlexRay block diagram



The protocol engine has two transmitter units, TxA and TxB, and two receiver units, RxA and RxB, for sending and receiving frames through the two FlexRay channels. The time control unit (TCU) is responsible for maintaining global clock synchronization to the FlexRay network. The overall activity of the protocol engine (PE) is controlled by the sequencer engine (SEQ).

The controller host interface provides host access to the module's configuration, control, and status registers, as well as to the message buffer configuration, control, and status registers. The message buffers themselves, which contain the frame header and payload data received or to be transmitted, as well as the slot status information, are stored in the FlexRay memory area.

The clock domain crossing unit implements signal crossing from the controller host interface (CHI) clock domain to the PE clock domain and vice versa, to allow for asynchronous PE and CHI clock domains.

The CC stores the frame header and payload data of frames received or of frames to be transmitted in the FlexRay memory area. The application accesses the FlexRay memory area to retrieve and provide the frames to be processed by the CC. In addition to the frame header and payload data, the CC stores the synchronization frame related tables in the FlexRay memory area for application processing.

The FlexRay memory area is located in the system memory of the MCU. The CC has access to the FlexRay memory area via its bus master interface (BMIF). The host provides the start address of the FlexRay memory area within the system memory by programming the *Section 57.5.2.5: System Memory Base Address Register (FR_SYMBADR)*. All FlexRay memory area related offsets are stored in offset registers. The physical address pointer into the FlexRay memory area is calculated using the offset values the FlexRay memory base address.

Note: The CC does not provide a memory protection scheme for the FlexRay memory area.

57.1.4 Features

The CC provides the following features:

- FlexRay Communications System Protocol Specification, Version 2.1 Rev A-compliant protocol implementation
- FlexRay Communications System Electrical Physical Layer Specification, Version 3.0-compliant bus driver interface
- Single channel support
  - FlexRay Port A can be configured to be connected either to physical FlexRay channel A or physical FlexRay channel B
- FlexRay bus data rates of 10 Mbit/s, 8 Mbit/s, 5 Mbit/s, and 2.5 Mbit/s supported
- 128 configurable message buffers with
  - Individual frame ID filtering
  - Individual channel ID filtering
  - Individual cycle counter filtering
- Message buffer header, status, and payload data stored in dedicated FlexRay memory area
  - Allows for flexible and efficient message buffer implementation
  - Ensures consistent data access by means of a buffer-locking scheme
  - Application can lock multiple buffers at the same time
- Size of message buffer payload data section configurable from 0–254 bytes
- Two independent message buffer segments with configurable size of payload data section
  - Each segment can contain message buffers assigned to the static segment and message buffers assigned to the dynamic segment at the same time
- Zero padding for transmit message buffers in static segment
  - Applied when the frame payload length exceeds the size of the message buffer data section
- Transmit message buffers configurable with state/event semantics
- Message buffers can be configured as
  - Receive message buffer
  - Transmit message buffer
- Individual message buffers can be reconfigured
  - Individual message buffers can be disabled safely
  - Disabled message buffers can be reconfigured
- Two independent receive FIFOs
  - One receive FIFO per channel
  - Up to 255 entries for each FIFO
  - Global frame ID filtering, based on both value/mask filters and range filters
  - Global channel ID filtering
  - Global message ID filtering for the dynamic segment
- Four configurable slot error counters
- Four dedicated slot status indicators
  - Used to observe slots without using receive message buffers

- Measured value indicators for clock synchronization
  - Internal synchronization frame ID and synchronization frame measurement tables can be copied into the FlexRay memory area
- Fractional macroticks are supported for clock correction
- Maskable interrupt sources provided via individual and combined interrupt lines
- One absolute timer
- One timer that can be configured as absolute or relative
- SECDED for protocol engine data RAM
- SEDDED for CHI lookup table RAM

57.1.5 Modes of operation

This section describes the basic operational power modes of the CC.

57.1.5.1 Disabled mode

The CC enters Disabled mode during hard reset or when the host clears the 'MEN' field in the Module Configuration Register (FR_MCR). The host can clear this field by writing '0' and only when the module is in *POC:default config* mode. The Disabled mode can be checked by reading the module enable field, MEN, in the *Module Configuration Register (FR_MCR)* .

No communication is performed on the FlexRay bus.

All registers with the write access conditions *Any Time* and *Disabled mode* can be accessed for writing as stated in *Section 57.5: Memory map and register description*.

The application configures the CC by accessing the configuration fields in the *Module Configuration Register (FR_MCR)* as described in *Section 57.7.2.1: Module initialization*.

57.1.5.1.1 Leave Disabled mode

The CC leaves Disabled mode and enters Normal mode, when the application writes 1 to FR_MCR[MEN].

Note: Once the CC is enabled, it cannot be disabled later.

57.1.5.2 Normal mode

In this mode, the CC is fully functional. The CC indicates that it is in Normal mode by asserting FR_MCR[EN].

57.1.5.2.1 Enter Normal mode

This mode is entered when the application requests that the CC leave Disabled mode. If Normal mode is entered after leaving Disabled mode, the application has to perform the protocol initialization described in *Section 57.7.2.2: Protocol initialization* to achieve full FlexRay functionality.

Depending on the values of the SCM, CHA, and CHB fields in the *Module Configuration Register (FR_MCR)*, the corresponding FlexRay bus driver ports are enabled and driven.

57.2 External signal description

This section lists and describes the CC signals connected to external pins. These signals are summarized in *Table 1299* and described in detail in *Section 57.2.1: Detailed signal description*.

Note: The off-chip FR_A_RX, FR_A_TX, and $\overline{\text{FR_A_TX_EN}}$ signals are available on each package option. The availability of the other off-chip signals depends on the package option and is chip-specific.

Table 1299. External signal properties

Name	Direction	Active	Reset	Function
FR_A_RX	Input	—	—	Receive Data Channel A
FR_A_TX	Output	—	1	Transmit Data Channel A
$\overline{\text{FR_A_TX_EN}}$	Output	Low	1	Transmit Enable Channel A
FR_B_RX	Input	—	—	Receive Data Channel B
FR_B_TX	Output	—	1	Transmit Data Channel B
$\overline{\text{FR_B_TX_EN}}$	Output	Low	1	Transmit Enable Channel B
FR_DBG[0]	Output	—	0	Debug Strobe Signal 0
FR_DBG[1]	Output	—	0	Debug Strobe Signal 1
FR_DBG[2]	Output	—	0	Debug Strobe Signal 2
FR_DBG[3]	Output	—	0	Debug Strobe Signal 3

57.2.1 Detailed signal description

This section provides a detailed description of the CC signals connected to external pins.

57.2.1.1 FR_A_RX – Receive Data Channel A

The FR_A_RX signal carries the receive data for channel A from the corresponding FlexRay bus driver.

57.2.1.2 FR_A_TX – Transmit Data Channel A

The FR_A_TX signal carries the transmit data for channel A to the corresponding FlexRay bus driver.

57.2.1.3 $\overline{\text{FR_A_TX_EN}}$ – Transmit Enable Channel A

The $\overline{\text{FR_A_TX_EN}}$ signal indicates to the FlexRay bus driver that the CC is attempting to transmit data on channel A.

57.2.1.4 FR_B_RX – Receive Data Channel B

The FR_B_RX signal carries the receive data for channel B from the corresponding FlexRay bus driver.

57.2.1.5 FR_B_TX – Transmit Data Channel B

The FR_B_TX signal carries the transmit data for channel B to the corresponding FlexRay bus driver

57.2.1.6 $\overline{\text{FR_B_TX_EN}}$ – Transmit Enable Channel B

The $\overline{\text{FR_B_TX_EN}}$ signal indicates to the FlexRay bus driver that the CC is attempting to transmit data on channel B.

57.2.1.7 FR_DBG[3], FR_DBG[2], FR_DBG[1], FR_DBG[0] – Strobe Signals

These signals provide the selected debug strobe signals. For details on the debug strobe signal selection, refer to *Section 57.6.16: Strobe signal support*.

57.3 Controller host interface clocking

The clock for the CHI is derived from the system bus clock and has the same phase and frequency as the system bus clock. There are two constraints for the minimum CHI clock frequency.

- The first constraint corresponds to the number of utilized message buffers and is specified in *Section 57.7.5: Number of usable message buffers*.
- The second constraint corresponds to the value of the TIMEOUT field in the *Section 57.5.2.33: System Memory Access Timeout Register (FR_SYMATOR)* and is specified in *Section 57.7.1.1: Configure System Memory Access Timeout Register (FR_SYMATOR)*.

57.4 Protocol engine clocking

The clock for the protocol engine can be generated by two sources. The first source is the internal crystal oscillator and the second source is an internal PLL. The clock source to be used is selected by the clock source select field, CLKSEL, in the *Section 57.5.2.4: Module Configuration Register (FR_MCR)*.

Note: Refer to the device clocking details for the exact clock sources used.

57.4.1 Oscillator clocking

If the protocol engine is clocked by the internal crystal oscillator, a 40 MHz crystal or CMOS-compatible clock must be connected to the oscillator pins. The crystal or clock must fulfill the requirements given by the *FlexRay Communications System Protocol Specification, Version 2.1 Rev A.*

57.4.2 PLL clocking

If the protocol engine is clocked by the internal PLL, the PLL clock should be divided down appropriately to generate the 40 MHz and 180 degree phase-shifted 40 MHz clocks and provide them to the PE. For more details, refer to the clocking chapter of the device reference manual.

57.5 Memory map and register description

The CC occupies 8 KB (8192 bytes) of address space starting at the CC base address defined by the memory map of the MCU.

Note: The following registers are 16-bit write accessible only:

- *Strobe Signal Control Register (FR_STBSCR)*
- *PE DRAM Access Register (FR_PEDRAR)*
- *PE DRAM Data Register (FR_PEDRDR)*
- *Sync Frame ID Rejection Filter Register (FR_SFIDRFR)*
- *Slot Status Selection Register (FR_SSSR)*
- *Slot Status Counter Condition Register (FR_SSCCR)*
- *Receive Shadow Buffer Index Register (FR_RSBIR)*
- *Receive FIFO Range Filter Configuration Register (FR_RFRFCFR)*
- *Message Buffer Cycle Counter Filter Register (FR_MBCCFRn)*
- *Message Buffer Frame ID Register (FR_MBFIDRn)*
- *Message Buffer Index Register (FR_MBIDXRn)*
- *Message Buffer Data Field Offset Register (FR_MBDORn)*
- *LRAM ECC Error Test Register (FR_LEETRn)*

57.5.1 Memory map

The complete memory map of the CC is shown in *Table 1300*. The addresses presented here are the offsets relative to the CC base address, which is defined by the MCU address map.

Table 1300. FlexRay memory map

Address offset	Register name	Location
Module configuration and control		
0x0000	Module Version Register (FR_MVR)	*Section 57.5.2.3*
0x0002	Module Configuration Register (FR_MCR)	*Section 57.5.2.4*
0x0004	System Memory Base Address High Register (FR_SYMBADHR)	*Section 57.5.2.5*
0x0006	System Memory Base Address Low Register (FR_SYMBADLR)	*Section 57.5.2.5*
0x0008	Strobe Signal Control Register (FR_STBSCR)	*Section 57.5.2.6*
0x000A	Strobe Port Control Register (FR_STBPCR)	*Section 57.5.2.7*
0x000C	Message Buffer Data Size Register (FR_MBDSR)	*Section 57.5.2.8*
0x000E	Message Buffer Segment Size and Utilization Register (FR_MBSSUTR)	*Section 57.5.2.9*
PE access registers		
0x0010	PE DRAM Access Register (FR_PEDRAR)	*Section 57.5.2.10*
0x0012	PE DRAM Data Register (FR_PEDRDR)	*Section 57.5.2.11*

Table 1300. FlexRay memory map (continued)

Address offset	Register name	Location
Interrupt and error handling		
0x0014	Protocol Operation Control Register (FR_POCR)	*Section 57.5.2.12*
0x0016	Global Interrupt Flag and Enable Register (FR_GIFER)	*Section 57.5.2.13*
0x0018	Protocol Interrupt Flag Register 0 (FR_PIFR0)	*Section 57.5.2.14*
0x001A	Protocol Interrupt Flag Register 1 (FR_PIFR1)	*Section 57.5.2.15*
0x001C	Protocol Interrupt Enable Register 0 (FR_PIER0)	*Section 57.5.2.16*
0x001E	Protocol Interrupt Enable Register 1 (FR_PIER1)	*Section 57.5.2.17*
0x0020	CHI Error Flag Register (FR_CHIERFR)	*Section 57.5.2.18*
0x0022	Message Buffer Interrupt Vector Register (FR_MBIVEC)	*Section 57.5.2.19*
0x0024	Channel A Status Error Counter Register (FR_CASERCR)	*Section 57.5.2.20*
0x0026	Channel B Status Error Counter Register (FR_CBSERCR)	*Section 57.5.2.21*
Protocol status		
0x0028	Protocol Status Register 0 (FR_PSR0)	*Section 57.5.2.22*
0x002A	Protocol Status Register 1 (FR_PSR1)	*Section 57.5.2.23*
0x002C	Protocol Status Register 2 (FR_PSR2)	*Section 57.5.2.24*
0x002E	Protocol Status Register 3 (FR_PSR3)	*Section 57.5.2.25*
0x0030	Macrotick Counter Register (FR_MTCTR)	*Section 57.5.2.26*
0x0032	Cycle Counter Register (FR_CYCTR)	*Section 57.5.2.27*
0x0034	Slot Counter Channel A Register (FR_SLTCTAR)	*Section 57.5.2.28*
0x0036	Slot Counter Channel B Register (FR_SLTCTBR)	*Section 57.5.2.29*
0x0038	Rate Correction Value Register (FR_RTCORVR)	*Section 57.5.2.30*
0x003A	Offset Correction Value Register (FR_OFCORVR)	*Section 57.5.2.31*
0x003C	Combined Interrupt Flag Register (FR_CIFR)	*Section 57.5.2.32*
0x003E	System Memory Access Time-Out Register (FR_SYMATOR)	*Section 57.5.2.33*
Sync frame counter and tables		
0x0040	Sync Frame Counter Register (FR_SFCNTR)	*Section 57.5.2.34*
0x0042	Sync Frame Table Offset Register (FR_SFTOR)	*Section 57.5.2.35*
0x0044	Sync Frame Table Configuration, Control, Status Register (FR_SFTCCSR)	*Section 57.5.2.36*
Sync frame filter		
0x0046	Sync Frame ID Rejection Filter Register (FR_SFIDRFR)	*Section 57.5.2.37*
0x0048	Sync Frame ID Acceptance Filter Value Register (FR_SFIDAFVR)	*Section 57.5.2.38*
0x004A	Sync Frame ID Acceptance Filter Mask Register (FR_SFIDAFMR)	*Section 57.5.2.39*

Table 1300. FlexRay memory map (continued)

Address offset	Register name	Location
Network management vector		
0x004C	Network Management Vector Register 0 (FR_NMVR0)	*Section 57.5.2.40*
0x004E	Network Management Vector Register 1 (FR_NMVR1)	
0x0050	Network Management Vector Register 2 (FR_NMVR2)	
0x0052	Network Management Vector Register 3 (FR_NMVR3)	
0x0054	Network Management Vector Register 4 (FR_NMVR4)	
0x0056	Network Management Vector Register 5 (FR_NMVR5)	
0x0058	Network Management Vector Length Register (FR_NMVLR)	*Section 57.5.2.41*
Timer configuration		
0x005A	Timer Configuration and Control Register (FR_TICCR)	*Section 57.5.2.42*
0x005C	Timer 1 Cycle Set Register (FR_TI1CYSR)	*Section 57.5.2.43*
0x005E	Timer 1 Macrotick Offset Register (FR_TI1MTOR)	*Section 57.5.2.44*
0x0060	Timer 2 Configuration Register 0 (FR_TI2CR0)	*Section 57.5.2.45*
0x0062	Timer 2 Configuration Register 1 (FR_TI2CR1)	*Section 57.5.2.46*
Slot status configuration		
0x0064	Slot Status Selection Register (FR_SSSR)	*Section 57.5.2.47*
0x0066	Slot Status Counter Condition Register (FR_SSCCR)	*Section 57.5.2.48*
Slot status		
0x0068	Slot Status Register 0 (FR_SSR0)	*Section 57.5.2.49*
0x006A	Slot Status Register 1 (FR_SSR1)	
0x006C	Slot Status Register 2 (FR_SSR2)	
0x006E	Slot Status Register 3 (FR_SSR3)	
0x0070	Slot Status Register 4 (FR_SSR4)	
0x0072	Slot Status Register 5 (FR_SSR5)	
0x0074	Slot Status Register 6 (FR_SSR6)	
0x0076	Slot Status Register 7 (FR_SSR7)	
0x0078	Slot Status Counter Register 0 (FR_SSCR0)	*Section 57.5.2.50*
0x007A	Slot Status Counter Register 1 (FR_SSCR1)	
0x007C	Slot Status Counter Register 2 (FR_SSCR2)	
0x007E	Slot Status Counter Register 3 (FR_SSCR3)	
MTS generation		
0x0080	MTS A Configuration Register (FR_MTSACFR)	*Section 57.5.2.51*
0x0082	MTS B Configuration Register (FR_MTSBCFR)	*Section 57.5.2.52*

Table 1300. FlexRay memory map (continued)

Address offset	Register name	Location
Shadow buffer configuration		
0x0084	Receive Shadow Buffer Index Register (FR_RSBIR)	*Section 57.5.2.53*
Receive FIFO – Configuration		
0x0086	Receive FIFO Watermark and Selection Register (FR_RFWMSR)	*Section 57.5.2.54*
0x0088	Receive FIFO Start Index Register (FR_RFSIR)	*Section 57.5.2.55*
0x008A	Receive FIFO Depth and Size Register (FR_RFDSR)	*Section 57.5.2.56*
Receive FIFO – Control		
0x008C	Receive FIFO A Read Index Register (FR_RFARIR)	*Section 57.5.2.57*
0x008E	Receive FIFO B Read Index Register (FR_RFBRIR)	*Section 57.5.2.58*
Receive FIFO – Filter		
0x0090	Receive FIFO Message ID Acceptance Filter Value Register (FR_RFMIDAFVR)	*Section 57.5.2.59*
0x0092	Receive FIFO Message ID Acceptance Filter Mask Register (FR_RFMIDAFMR)	*Section 57.5.2.60*
0x0094	Receive FIFO Frame ID Rejection Filter Value Register (FR_RFFIDRFVR)	*Section 57.5.2.61*
0x0096	Receive FIFO Frame ID Rejection Filter Mask Register (FR_RFFIDRFMR)	*Section 57.5.2.62*
0x0098	Receive FIFO Range Filter Configuration Register (FR_RFRFCFR)	*Section 57.5.2.63*
0x009A	Receive FIFO Range Filter Control Register (FR_RFRFCTR)	*Section 57.5.2.64*
Dynamic segment status		
0x009C	Last Dynamic Transmit Slot Channel A Register (FR_LDTXSLAR)	*Section 57.5.2.65*
0x009E	Last Dynamic Transmit Slot Channel B Register (FR_LDTXSLBR)	*Section 57.5.2.66*
Protocol configuration		
0x00A0 0x00A2 … 0x00DC	Protocol Configuration Register 0 (FR_PCR0) Protocol Configuration Register 1 (FR_PCR1) … Protocol Configuration Register 30 (FR_PCR30)	*Section 57.5.2.67.1* *Section 57.5.2.67.2* … *Section 57.5.2.67.31*
0x00DE	Stop Watch Count High Register (FR_STPWHR)	*Section 57.5.2.68*
0x00E0	Stop Watch Count Low Register (FR_STPWLR)	*Section 57.5.2.68*
Event output and StopWatch configuration		
0x00E2	Protocol Event Output Enable and StopWatch Control Register (FR_PEOER)	*Section 57.5.2.69*
0x00E4–0x00E5	Reserved	

Table 1300. FlexRay memory map (continued)

Address offset	Register name	Location
Receive FIFO – Configuration		
0x00E6	Receive FIFO Start Data Offset Register (FR_RFSDOR)	*Section 57.5.2.70*
0x00E8	Receive FIFO System Memory Base Address High Register (FR_RFSYMBADHR)	*Section 57.5.2.71*
0x00EA	Receive FIFO System Memory Base Address Low Register (FR_RFSYMBADLR)	*Section 57.5.2.71*
0x00EC	Receive FIFO Periodic Timer Register (FR_RFPTR)	*Section 57.5.2.72*
Receive FIFO – Control		
0x00EE	Receive FIFO Fill Level and POP Count Register (FR_RFFLPCR)	*Section 57.5.2.73*
ECC registers		
0x00F0	ECC Error Interrupt Flag and Enable Register (FR_EEIFER)	*Section 57.5.2.74*
0x00F2	ECC Error Report and Injection Control Register (FR_EERICR)	*Section 57.5.2.75*
0x00F4	ECC Error Report Address Register (FR_EERAR)	*Section 57.5.2.76*
0x00F6	ECC Error Report Data Register (FR_EERDR)	*Section 57.5.2.77*
0x00F8	ECC Error Report Code Register (FR_EERCR)	*Section 57.5.2.78*
0x00FA	ECC Error Injection Address Register (FR_EEIAR)	*Section 57.5.2.79*
0x00FC	ECC Error Injection Data Register (FR_EEIDR)	*Section 57.5.2.80*
0x00FE	ECC Error Injection Code Register (FR_EEICR)	*Section 57.5.2.81*
0x0100–0x07FF	Not Implemented	
Message buffers configuration, control and status		
0x0800	Message Buffer Configuration, Control, Status Register 0 (FR_MBCCSR0)	*Section 57.5.2.82*
0x0802	Message Buffer Cycle Counter Filter Register 0 (FR_MBCCFR0)	*Section 57.5.2.83*
0x0804	Message Buffer Frame ID Register 0 (FR_MBFIDR0)	*Section 57.5.2.84*
0x0806	Message Buffer Index Register 0 (FR_MBIDXR0)	*Section 57.5.2.85*
0x0808	Message Buffer Configuration, Control, Status Register 1 (FR_MBCCSR1)	*Section 57.5.2.82*
0x080A	Message Buffer Cycle Counter Filter Register 1 (FR_MBCCFR1)	*Section 57.5.2.83*
0x080C	Message Buffer Frame ID Register 1 (FR_MBFIDR1)	*Section 57.5.2.84*
0x080E	Message Buffer Index Register 1 (FR_MBIDXR1)	*Section 57.5.2.85*
…	…	…
0x0BF8	Message Buffer Configuration, Control, Status Register 127 (FR_MBCCSR127)	*Section 57.5.2.82*
0x0BFA	Message Buffer Cycle Counter Filter Register 127 (FR_MBCCFR127)	*Section 57.5.2.83*
0x0BFC	Message Buffer Frame ID Register 127 (FR_MBFIDR127)	*Section 57.5.2.84*
0x0BFE	Message Buffer Index Register 127 (FR_MBIDXR127)	*Section 57.5.2.85*

Table 1300. FlexRay memory map (continued)

Address offset	Register name	Location
0x0C00-0x0FFF	Not Implemented	—
0x1000 0x1002 … 0x1106	Message Buffer Data Field Offset Register 0 (FR_MBDOR0) Message Buffer Data Field Offset Register 1 (FR_MBDOR1) … Message Buffer Data Field Offset Register 131 (FR_MBDOR131)	*Section 57.5.2.86*
0x1108 0x110A … 0x1112	LRAM ECC Error Test Register 0 (FR_LEETR0) LRAM ECC Error Test Register 1 (FR_LEETR1) … LRAM ECC Error Test Register 5 (FR_LEETR5)	*Section 57.5.2.87*
0x1114–0x1FFF	Not Implemented	

57.5.2 Register descriptions

This section provides detailed descriptions of all registers in ascending address order, presented as 16-bit wide entities.

57.5.2.1 Register reset

All registers except the Message Buffer Cycle Counter Filter Registers (FR_MBCCFRn), Message Buffer Frame ID Registers (FR_MBFIDRn), and Message Buffer Index Registers (FR_MBIDXRn) are reset to their reset value on system reset. The registers mentioned above are located in physical memory blocks and, thus, they are not affected by reset. For some register fields, additional reset conditions exist. These additional reset conditions are described in the detailed description of the register. The additional reset conditions are explained in *Table 1301*.

Table 1301. Additional register reset conditions

Condition	Description
Protocol RUN Command	The register field is reset when the application writes RUN command “0101” to the POCCMD field in the *Section 57.5.2.12: Protocol Operation Control Register (FR_POCR)*.
Message Buffer Disable	The register field is reset when the application has disabled the message buffer. This happens when the application writes 1 to the message buffer disable trigger bit FR_MBCCSRn[EDT] while the message buffer is enabled (FR_MBCCSRn[EDS] = 1) and the CC grants a disable to the application by clearing the FR_MBCCSRn[EDS] bit.

57.5.2.2 Register write access

This section describes the write access restriction terms that apply to all registers.

57.5.2.2.1 Register write access restriction

For each register bit and register field, the write access conditions are specified in the detailed register description. A description of the write access conditions is given in

Table 1302. If, for a specific register field, none of the given write access conditions is fulfilled, any write attempt to this register field is ignored without any notification. The values of the fields are not changed. The condition term [A or B] indicates that the register or field can be written to if at least one of the conditions is fulfilled. The condition term [A and B] indicates that the register or field can be written to if both conditions are fulfilled.

Table 1302. Register write access restrictions

Condition	Indication	Description
Anytime	—	No write access restriction.
Disabled mode	FR_MCR[MEN] = 0	Write access only when CC is in Disabled mode.
Normal mode	FR_MCR[MEN] = 1	Write access only when CC is in Normal mode.
POC:config	FR_PSR0[PROTSTATE] = *POC:config*	Write access only when protocol is in the *POC:config* state.
MB_DIS	FR_MBCCSRn[EDS] = 0	Write access only when related Message Buffer is disabled.
MB_LCK	FR_MBCCSRn[LCKS] = 1	Write access only when related Message Buffer is locked.
IDL	FR_EEIRICR[BSY] = 0	Write access only when ECC configuration is idle.

57.5.2.2.2 Register write access requirements

All registers can be accessed with 8-bit, 16-bit, and 32-bit wide operations.

For some of the registers, at least a 16-bit wide write access is required to ensure correct operation. This write access requirement is described in the Memory map and register definition section. If an 8-bit wide write access is performed to any of these registers, this access is ignored without notification.

57.5.2.2.3 Internal register access

The following memory-mapped registers are used to access multiple internal registers.

- Strobe Signal Control Register (FR_STBSCR)
- Slot Status Selection Register (FR_SSSR)
- Slot Status Counter Condition Register (FR_SSCCR)
- Receive Shadow Buffer Configuration Data

Each of these memory-mapped registers provides a SEL field and a WMD bit. The SEL field is used to select the internal register. The WMD bit controls the write mode. If WMD is set to 0 during the write access, all fields of the internal register are updated. If WMD is set to 1, only the SEL field is changed. All other fields of the internal register remain unchanged. This allows for reading back the values of the selected internal register in a subsequent read access.

57.5.2.3 Module Version Register (FR_MVR)

Offset: 0x0000 Access: Read-only

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	CHIVER								PEVER							
W																
Reset	1	0	1	0	0	1	0	1	0	1	1	0	1	0	0	0

Figure 1228. Module Version Register (FR_MVR)

This register provides the CC version number. The module version number is derived from the CHI version number and the PE version number.

Table 1303. FR_MVR field descriptions

Field	Description
0:7 CHIVER	**CHI Version Number** This field provides the version number of the controller host interface.
8:15 PEVER	**PE Version Number** This field provides the version number of the protocol engine.

57.5.2.4 Module Configuration Register (FR_MCR)

Access: Read/Write
Write: SBFF, SCM, CHB, CHA, ECCE, FUM, FAM, CLKSEL, BITRATE: Disabled mode
Offset: 0x0002
Write: MEN: Disabled or *POC:default config*
SFFE: Disabled Mode or *POC:config*

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	MEN	SBFF	SCM	CHB	CHA	SFFE	ECCE	Reserved(1)	FUM	FAM	0	CLKSEL	BITRATE			0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. It is read as 0. Application must not write 1 to this bit.

Figure 1229. Module Configuration Register (FR_MCR)

This register defines the global configuration of the CC.

Table 1304. FR_MCR field descriptions

Field	Description
0 MEN	**Module Enable** This bit indicates whether or not the CC is in Disabled mode. The application requests that the CC leave Disabled mode by writing 1 to this bit Before leaving Disabled mode, the application must configure the SCM, SBFF, CHB, CHA, TMODE, and BITRATE values. For details, refer to *Section 57.1.5: Modes of operation*. 0 Write: only during *POC:default config*, CC disable Read: CC disabled 1 Write: enable CC Read: CC enabled **Note:** If the CC is enabled, it can only be disabled during mode: POC:default config
1 SBFF	**System Bus Failure Freeze** This bit controls the behavior of the CC in case of a system bus failure. 0 Continue normal operation 1 Transition to Freeze mode
2 SCM	**Single Channel Device Mode** This control bit defines the channel device mode of the CC as described in *Section 57.6.10: Channel device modes*. 0 CC works in Dual-Channel Device mode 1 CC works in Single-Channel Device mode
3 CHB	**Channel Enable** protocol related parameter: *pChannels* The semantic of this control bit depends on the channel device mode controlled by the SCM bit and is given *Table 1305*.
4 CHA	**Channel Enable** protocol related parameter: *pChannels* The semantic of this control bit depends on the channel device mode controlled by the SCM bit and is given *Table 1305*.
5 SFFE	**Synchronization Frame Filter Enable** This bit controls the filtering for received synchronization frames. For details, refer to *Section 57.6.15: Sync frame filtering*. 0 Synchronization frame filtering disabled 1 Synchronization frame filtering enabled
6 ECCE	**ECC Functionality Enable** This bit controls the ECC memory error detection functionality. For details, refer to *Section 57.6.24: Memory content error detection*. 0 ECC functionality (injection, detection, reporting, response) disabled 1 ECC functionality enabled
8 FUM	**FIFO Update Mode** This bit controls the FIFO update behavior when the interrupt flags FR_GIFER[FAFAIF] and FR_GIFER[FAFBIF] are written by the application (refer to *Section 57.6.9.8: FIFO update*) 0 FIFOA/FIFOB is updated on writing 1 to FR_GIFER[FAFAIF] /FR_GIFER[FAFBIF] 1 FIFOA/FIFOB) is *not* updated on writing 1 to FR_GIFER[FAFAIF]/FR_GIFER[FAFBIF]

Table 1304. FR_MCR field descriptions (continued)

Field	Description
9 FAM	**FIFO Address Mode** This bit controls the location of the system memory base address for the FIFOs (refer to *Section 57.6.9.2: FIFO configuration*). 0 FIFO Base Address located in *Section 57.5.2.5: System Memory Base Address Register (FR_SYMBADR)* 1 FIFO Base Address located in *Section 57.5.2.71: Receive FIFO System Memory Base Address Register (FR_RFSYMBADR)*
11 CLKSEL	**Protocol Engine Clock Source Select** This bit is used to select the clock source for the protocol engine. 0 PE clock source is generated by on-chip crystal oscillator. 1 PE clock source is generated by on-chip PLL.
12:14 BITRATE	**FlexRay Bus Bit Rate** This field defines the FlexRay bus bit rate. 000 10.0 Mbit/s 001 5.0 Mbit/s 010 2.5 Mbit/s 011 8.0 Mbit/s 100 Reserved 101 Reserved 110 Reserved 111 Reserved

Table 1305. FlexRay channel selection

SCM	CHB	CHA	Description
Dual channel device modes			
0	0	0	ports FR_A_RX, FR_A_TX, and $\overline{\text{FR_A_TX_EN}}$ not driven by CC ports FR_B_RX, FR_B_TX, and $\overline{\text{FR_B_TX_EN}}$ not driven by CC
	0	1	ports FR_A_RX, FR_A_TX, and $\overline{\text{FR_A_TX_EN}}$ driven by CC—connected to FlexRay channel A ports FR_B_RX, FR_B_TX, and $\overline{\text{FR_B_TX_EN}}$ not driven by CC
	1	0	ports FR_A_RX, FR_A_TX, and $\overline{\text{FR_A_TX_EN}}$ not driven by CC ports FR_B_RX, FR_B_TX, and $\overline{\text{FR_B_TX_EN}}$ driven by CC—connected to FlexRay channel B
	1	1	ports FR_A_RX, FR_A_TX, and $\overline{\text{FR_A_TX_EN}}$ driven by CC—connected to FlexRay channel A ports FR_B_RX, FR_B_TX, and $\overline{\text{FR_B_TX_EN}}$ driven by CC—connected to FlexRay channel B

Table 1305. FlexRay channel selection (continued)

SCM	CHB	CHA	Description
Single channel device mode			
1	0	0	ports FR_A_RX, FR_A_TX, and $\overline{\text{FR_A_TX_EN}}$ not driven by CC ports FR_B_RX, FR_B_TX, and $\overline{\text{FR_B_TX_EN}}$ not driven by CC
	0	1	ports FR_A_RX, FR_A_TX, and $\overline{\text{FR_A_TX_EN}}$ driven by CC—connected to FlexRay channel A ports FR_B_RX, FR_B_TX, and $\overline{\text{FR_B_TX_EN}}$ not driven by CC
	1	0	ports FR_A_RX, FR_A_TX, and $\overline{\text{FR_A_TX_EN}}$ driven by CC—connected to FlexRay channel B ports FR_B_RX, FR_B_TX, and $\overline{\text{FR_B_TX_EN}}$ not driven by CC
	1	1	Reserved

57.5.2.5 System Memory Base Address Register (FR_SYMBADR)

Offset: 0x0004 Access: Read/Write
Write: Disabled mode

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	SMBA															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1230. System Memory Base Address High Register (FR_SYMBADHR)

Offset: 0x0006 Access: Read/Write
Write: Disabled mode

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	SMBA												0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1231. System Memory Base Address Low Register (FR_SYMBADLR)

Note: The system memory base address must be set before the CC is enabled.

The system memory base address registers define the base address of the FlexRay memory area within the system memory. The base address is used by the bus master interface (BMIF) to calculate the physical memory address for system memory accesses.

Table 1306. FR_SYMBADR field descriptions

Field	Description
SMBA	**System Memory Base Address** This is the value of the system memory base address for the individual message buffers and sync frame table. This is the value of the system memory base address for the receive FIFO if FR_MCR[FAM] is set to 1. It is defined as a byte address.

57.5.2.6 Strobe Signal Control Register (FR_STBSCR)

Offset: 0x0008

Access: Read/Write(1)
Write: Anytime

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	SEL				0	0	0	ENB	0	0	STBPSEL	
W	WMD															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. 16-bit write access required

Figure 1232. Strobe Signal Control Register (FR_STBSCR)

This register is used to assign the individual protocol timing-related strobe signals given in *Table 1308* to the external strobe ports. Each strobe signal can be assigned to at most one strobe port. Each write access to registers overwrites the previously-written ENB and STBPSEL values for the signal indicated by SEL. If more than one strobe signal is assigned to one strobe port, the current values of the strobe signals are combined with a binary OR and presented at the strobe port. If no strobe signal is assigned to a strobe port, the strobe port carries logic 0. For more detailed and timing information refer to *Section 57.6.16: Strobe signal support*.

Note: In Single-channel device mode, channel B-related strobe signals are undefined and should not be assigned to the strobe ports.

Table 1307. FR_STBSCR field descriptions

Field	Description
0 WMD	**Write Mode** This control bit defines the write mode of this register. 0 Write to all fields in this register on write access. 1 Write to SEL field only on write access.
4:7 SEL	**Strobe Signal Select** This control field selects one of the strobe signals given in *Table 1308* to be enabled or disabled and assigned to one of the four strobe ports.
11 ENB	**Strobe Signal Enable** The control bit is used to enable and disable the strobe signal selected by STBSSEL. 0 Strobe signal is disabled and not assigned to any strobe port. 1 Strobe signal is enabled and assigned to the strobe port selected by STBPSEL.
14:15 STBPSEL	**Strobe Port Select** This field selects the strobe port that the strobe signal selected by the SEL is assigned to. All strobe signals that are enabled and assigned to the same strobe port are combined with a binary OR operation. 00 Assign selected signal to FR_DBG[0]. 01 Assign selected signal to FR_DBG[1]. 10 Assign selected signal to FR_DBG[2]. 11 Assign selected signal to FR_DBG[3].

Table 1308. Strobe signal mapping

SEL		Description	Channel	Type	Offset(1)	Reference
Dec	Hex					
0	0x0	Arm	—	value	+1	MT start
1	0x1	mt	—	value	+1	MT start
2	0x2	Cycle start	—	pulse	0	MT start
3	0x3	Minislot start	—	pulse	0	MT start
4	0x4	Slot start	A	pulse	0	MT start
5	0x5		B			
6	0x6	Receive data after glitch filtering	A	value	+4	FR_A_RX
7	0x7		B			FR_B_RX
8	0x8	Channel idle indicator	A	level	+5	FR_A_RX
9	0x9		B			FR_B_RX
10	0xA	Syntax error detected	A	pulse	+4	FR_A_RX
11	0xB		B			FR_B_RX
12	0xC	Content error detected	A	level	+4	FR_A_RX
13	0xD		B			FR_B_RX
14	0xE	Receive FIFO almost-full interrupt signals	A	value	n.a.	RX FIFO A Almost Full Interrupt
15	0xF		B			RX FIFO B Almost Full Interrupt

1. Given in PE clock cycles

57.5.2.7 Strobe Port Control Register (FR_STBPCR)

Offset: 0x000A Access: Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	STB3EN	STB2EN	STB1EN	STB0EN
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1233. Strobe Port Control Register (FR_STBPCR)

This register is used to enable and disable the strobe port signals. Each disabled port will stay disabled even when strobe signals are assigned to it.

Table 1309. FR_STBPCR field descriptions

Field	Description
12 STB3EN	**Strobe Port 3 Enable** This control bit defines whether the FR_DBG[3] port is enabled or disabled. 0 Strobe port FR_DBG[3] disabled 1 Strobe port FR_DBG[3] enabled
13 STB2EN	**Strobe Port 2 Enable** This control bit defines whether the FR_DBG[2] port is enabled or disabled. 0 Strobe port FR_DBG[2] disabled 1 Strobe port FR_DBG[2] enabled
14 STB1EN	**Strobe Port 1 Enable** This control bit defines whether the FR_DBG[1] port is enabled or disabled. 0 Strobe port FR_DBG[1] disabled 1 Strobe port FR_DBG[1] enabled
15 STB0EN	**Strobe Port 0 Enable** This control bit defines whether the FR_DBG[0] port is enabled or disabled. 0 Strobe port FR_DBG[0] disabled 1 Strobe port FR_DBG[0] enabled

57.5.2.8 Message Buffer Data Size Register (FR_MBDSR)

Offset: 0x000C

Access: Read/Write
Write: *POC:config*

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	MBSEG2DS							0	MBSEG1DS						
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1234. Message Buffer Data Size Register (FR_MBDSR)

This register defines the size of the message buffer data section for the two message buffer segments in a number of two-byte entities.

The CC provides two independent segments for the individual message buffers. All individual message buffers within one segment have to have the same size for the message buffer data section. This size can be different for the two message buffer segments.

Table 1310. FR_MBDSR field descriptions

Field	Description
1:7 MBSEG2DS	**Message Buffer Segment 2 Data Size** The field defines the size of the message buffer data section in two-byte entities for message buffers within the *second* message buffer segment.
9:15 MBSEG1DS	**Message Buffer Segment 1 Data Size** The field defines the size of the message buffer data section in two-byte entities for message buffers within the *first* message buffer segment.

57.5.2.9 Message Buffer Segment Size and Utilization Register (FR_MBSSUTR)

Offset: 0x000E

Access: Read/Write
Write: *POC:config*

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	LAST_MB_SEG1							0	LAST_MB_UTIL						
W																
Reset	0	1	1	1	1	1	1	1	0	1	1	1	1	1	1	1

Figure 1235. Message Buffer Segment Size and Utilization Register (FR_MBSSUTR)

This register defines the last individual message buffer that belongs to the first message buffer segment and the number of the last used individual message buffer.

Table 1311. FR_MBSSUTR field descriptions

Field	Description
1:7 LAST_MB_SEG1	**Last Message Buffer In Segment 1** This field defines the message buffer number of the last individual message buffer that is assigned to the *first* message buffer segment. The individual message buffers in the *first* segment correspond to the message buffer control registers FR_MBCCSRn, FR_MBCCFRn, FR_MBFIDRn, and FR_MBIDXRn with n ≤ LAST_MB_SEG1. The first message buffer segment contains LAST_MB_SEG1 + 1 individual message buffers. **Note:** The first message buffer segment contains at least one individual message buffer. The individual message buffers in the *second* message buffer segment correspond to the message buffer control registers FR_MBCCSRn, FR_MBCCFRn, FR_MBFIDRn, FR_MBIDXRn with LAST_MB_SEG1 < n < 128. **Note:** If LAST_MB_SEG1 = 127, all individual message buffers belong to the first message buffer segment and the second message buffer segment is empty.
9:15 LAST_MB_UTIL	**Last Message Buffer Utilized** This field defines the message buffer number of last utilized individual message buffer. The message buffer search engine examines all individual message buffer with a message buffer number n ≤ LAST_MB_UTIL. **Note:** If LAST_MB_UTIL=LAST_MB_SEG1, all individual message buffers belong to the first message buffer segment and the second message buffer segment is empty.

57.5.2.10 PE DRAM Access Register (FR_PEDRAR)

Offset: 0x0010

Access: Read/Write(1)
Write: Normal mode

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	INST				ADDR											DAD
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. 16-bit write access required

Figure 1236. PE DRAM Access Register (FR_PEDRAR)

The FR_PEDRAR is used to trigger write and read operations on the PE data memory (PE DRAM). These operations are used for memory error injection and memory error observation.

Each write access to this register initiates a read or write operation on the PE DRAM. The access done status bit, DAD, is cleared after the write access and is set if the PE DRAM access has been finished.

In case of an PE DRAM write access, the data provided in FR_PEDRDR is written into the PE DRAM, read back from the PE DRAM, and are stored into the FR_PEDRDR.

In case of an PE DRAM read access, the requested data is read from PE DRAM and stored into the FR_PEDRDR.

For a detailed description, refer to *Section 57.6.24: Memory content error detection*.

Table 1312. FR_PEDRAR field descriptions

Field	Description
0:3 INST	**PE DRAM Access Instruction** This field defines the operation to be executed on the PE DRAM. 0011 PE DRAM write: Write FR_PEDRDR[DATA] to PE DRAM address ADDR (16-bit) 0101 PE DRAM read: Read Data from PE DRAM address ADDR (16-bit) into FR_PEDRDR[DATA] other reserved
4:14 ADDR	**PE DRAM Access Address** This field defines the address in the PE DRAM to be written to or read from.
15 DAD	**PE DRAM Access Done** This status bit is cleared when the application has written to this register and is set when the PE DRAM access has finished. 0 PE DRAM access running 1 PE DRAM access done

57.5.2.11 PE DRAM Data Register (FR_PEDRDR)

Offset: 0x0012

Access: Read/Write(1)
Write: Normal mode

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	DATA															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. 16-bit write access required.

Figure 1237. PE DRAM Data Register (FR_PEDRDR)

This register provides the data to be written to or read from the PE DRAM by the access initiated by write access to the FR_PEDRAR.

Table 1313. FR_PEDRDR field descriptions

Field	Description
0:15 DATA	Data

57.5.2.12 Protocol Operation Control Register (FR_POCR)

Offset: 0x0014

Access: Read/Write
Write: Normal mode

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	EOC_AP		ERC_AP		BSY_ WMC	0	0	0	POCCMD			
W	WME															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1238. Protocol Operation Control Register (FR_POCR)

The application uses this register to issue:

- Protocol control commands
- External clock correction commands

Protocol control commands are issued by writing to the POCCMD field. For more information on protocol control commands, refer to *Section 57.7.6: Protocol control command execution*.

External clock correction commands are issued by writing to the EOC_AP and ERC_AP fields. For more information on external clock correction, refer to *Section 57.6.11: External clock synchronization*.

Table 1314. FR_POCR field descriptions

Field	Description
0 WME	**Write Mode External Correction** This bit controls the write mode of the EOC_AP and ERC_AP fields. 0 Write to EOC_AP and ERC_AP fields on register write 1 No write to EOC_AP and ERC_AP fields on register write
4:5 EOC_AP	**External Offset Correction Application** This field is used to trigger the application of the external offset correction value defined in the Protocol Configuration Register 29 (FR_PCR29). 00 Do not apply external offset correction value 01 Reserved 10 Subtract external offset correction value 11 Add external offset correction value
6:7 ERC_AP	**External Rate Correction Application** This field is used to trigger application of the external rate correction value defined in the Protocol Configuration Register 21 (FR_PCR21). 00 Do not apply external rate correction value 01 Reserved 10 Subtract external rate correction value 11 Add external rate correction value

Table 1314. FR_POCR field descriptions (continued)

Field	Description
8 BSY_WMC	**Protocol Control Command Write Busy (BSY)** This status bit indicates the acceptance of the protocol control command issued by the application via the POCCMD field. The CC sets this status bit when the application has issued a protocol control command via the POCCMD field. The CC clears this status bit when protocol control command was accepted by the PE.When the application issues a protocol control command while the BSY bit is asserted, the CC ignores this command, sets the protocol command ignored error flag PCMI_EF in the CHI Error Flag Register (FR_CHIERFR), and will not change the value of the POCCMD field. (0) Command write idle, command accepted and ready to receive new protocol command (1) Command write busy, command not yet accepted, not ready to receive new protocol command **Write Mode Command (WMC)** This bit controls the write mode of the POCCMD field. (0) Write to POCCMD field on register write (1) Do not write to POCCMD field on register write
12:15 POCCMD	**Protocol Control Command** The application writes to this field to issue a protocol control command to the PE. The CC sends the protocol command to the PE immediately. While the transfer is running, the BSY bit is set. 0000 ALLOW_COLDSTAR: Immediately activate capability of node to cold start cluster. 0001 ALL_SLOTS: Delayed(1) transition to the All Slots Transmission mode. 0010 CONFIG: Immediately transition to the *POC:config* state. 0011 FREEZE: Immediately transition to the *POC:halt* state. 0100 READY, CONFIG_COMPLETE: Immediately transition to the *POC:ready* state. 0101 RUN: Immediately transition to the *POC:startup start* state. 0110 DEFAULT_CONFIG: Immediately transition to the *POC:default config* state. 0111 HALT: Delayed transition to the *POC:halt* state 1000 WAKEUP: Immediately initiate the wakeup procedure. 1001 Reserved 1010 Reserved 1011 Reserved 1100 Reserved 1101 Reserved 1110 Reserved 1111 Reserved

1. Delayed means on completion of current communication cycle.

57.5.2.13 Global Interrupt Flag and Enable Register (FR_GIFER)

Offset: 0x0016

Access: Read/Write
Write: Normal mode

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	MIF	PRIF	CHIF	WUPIF	FAFBIF	FAFAIF	RBIF	TBIF	MIE	PRIE	CHIE	WUPIE	FAFBIE	FAFAIE	RBIE	TBIE
W				w1c	w1c	w1c										
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1239. Global Interrupt Flag and Enable Register (FR_GIFER)

This register controls some of the interrupt request lines and provides the corresponding interrupt flags. The interrupt flags MIF, PRIF, CHIF, RBIF, and TBIF are the outcome of a binary OR of the related individual interrupt flags and interrupt enables. The generation scheme for these flags is depicted in *Figure 1387*. For more details on interrupt generation, refer to *Section 57.6.20: Interrupt support*. These flags are cleared automatically when all of the corresponding interrupt flags or interrupt enables in the related interrupt flag and enable registers are cleared by the application.

Table 1315. FR_GIFER field descriptions

Field	Description
0 MIF	**Module Interrupt Flag** This interrupt flag is set if at least one of the other interrupt flags in this register and the related interrupt enable bit are set. 0 No interrupt flag and related interrupt enable bit are set 1 At least one of the other interrupt flags in this register and the related interrupt bit are set
1 PRIF	**Protocol Interrupt Flag** This interrupt flag is set if at least one of the individual flags in the Protocol Interrupt Flag Register 0 (FR_PIFR0) and Protocol Interrupt Flag Register 1 (FR_PIFR1) and the related interrupt enable bit are set. 0 No individual protocol interrupt flag and related interrupt enable bit are set 1 At least one of the individual protocol interrupt flags and the related interrupt enable bit are set
2 CHIF	**CHI Interrupt Flag** This interrupt flag is set if at least one of the error flags in the CHI Error Flag Register (FR_CHIERFR) and the CHI error interrupt enable bit FR_GIFER[CHIE] are set. 0 All CHI error flags are equal to 0 or the CHI error interrupt is disabled 1 At least one CHI error flag and the CHI error interrupt enable are is set
3 WUPIF	**Wakeup Interrupt Flag** This interrupt flag is set when the CC has received a wakeup symbol on the FlexRay bus. The application can determine on which channel the wakeup symbol was received by reading the related wakeup flags WUB and WUA in the Protocol Status Register 3 (FR_PSR3). 0 No wakeup symbol received on FlexRay bus 1 Wakeup symbol received on FlexRay bus

Table 1315. FR_GIFER field descriptions (continued)

Field	Description
4 FAFBIF	**Receive FIFO Channel B Almost Full Interrupt Flag** This interrupt flag is set when one of the following events occurs: a) The current number of FIFO B entries is equal to or greater than the watermark defined by the WM field in the Receive FIFO Watermark and Selection Register (FR_RFWMSR), and the CC writes a received message into the FIFO B, or b) The current number of FIFO B entries is at least 1 and the periodic timer as defined by Receive FIFO Periodic Timer Register (FR_RFPTR) expires. 0 No such event 1 FIFO B almost full event has occurred **Note:** If the bit MCR[FUM] is set to 0, to clear FAFAIF or FAFBIF will also remove one element from the FIFO.
5 FAFAIF	**Receive FIFO Channel A Almost Full Interrupt Flag** This interrupt flag is set when one of the following events occurs: a) The current number of FIFO A entries is equal to or greater than the watermark defined by the WM field in the Receive FIFO Watermark and Selection Register (FR_RFWMSR), and the CC writes a received message into the FIFO A, or b) The current number of FIFO A entries is at least 1 and the periodic timer as defined by Receive FIFO Periodic Timer Register (FR_RFPTR) expires. 0 No such event 1 FIFO A almost full event has occurred
6 RBIF	**Receive Message Buffer Interrupt Flag** This interrupt flag is set if for at least one of the individual receive message buffers (FR_MBCCSRn[MTD] = 0) both the MBIF interrupt flag and the MBIE interrupt enable bit in the corresponding Message Buffer Configuration, Control, Status Registers (FR_MBCCSRn) are asserted. The application cannot clear this interrupt flag directly; instead, it is cleared by the CC when all of the MBIF interrupt flags of the individual receive message buffers are cleared by the application, or when the application has cleared the related MBIE interrupt enable bit. 0 None of the individual receive message buffers has the MBIF and MBIE flag set 1 At least one individual receive message buffer has the MBIF and MBIE flag set
7 TBIF	**Transmit Message Buffer Interrupt Flag** This flag is set if for at least one of the individual message buffers (FR_MBCCSRn[MTD] = 1) both the MBIF interrupt flag and the MBIE interrupt enable bit in the corresponding Message Buffer Configuration, Control, Status Registers (FR_MBCCSRn) are equal to 1. The application cannot clear this interrupt flag directly; instead, this interrupt flag is cleared by the CC when all of the individual interrupt flags MBIF of the individual transmit message buffers are cleared by the application, or when the application has cleared the related MBIE interrupt enable bit. 0 None of the individual transmit message buffers has the MBIF and MBIE flag set 1 At least one individual transmit message buffer has the MBIF and MBIE flag set
8 MIE	**Module Interrupt Enable** This bit controls whether the Module Interrupt line is asserted when the MIF flag is set. 0 Disable interrupt line 1 Enable interrupt line
9 PRIE	**Protocol Interrupt Enable** This bit controls whether the Protocol Interrupt line is asserted when the PRIF flag is set. 0 Disable interrupt line 1 Enable interrupt line

Table 1315. FR_GIFER field descriptions (continued)

Field	Description
10 CHIE	**CHI Interrupt Enable** This bit controls whether the CHI Interrupt line is asserted when the CHIF flag is set. 0 Disable interrupt line 1 Enable interrupt line
11 WUPIE	**Wakeup Interrupt Enable** This bit controls whether the Wakeup Interrupt line is asserted when the WUPIF flag is set. 0 Disable interrupt line 1 Enable interrupt line
12 FAFBIE	**Receive FIFO Channel B Almost Full Interrupt Enable** This bit controls whether the RX FIFO B Almost Full Interrupt line is asserted when the FAFBIF flag is set. 0 Disable interrupt line 1 Enable interrupt line
13 FAFAIE	**Receive FIFO Channel A Almost Full Interrupt Enable** This bit controls whether the RX FIFO A Almost Full Interrupt line is asserted when the FAFAIF flag is set. 0 Disable interrupt line 1 Enable interrupt line
14 RBIE	**Receive Message Buffer Interrupt Enable** This bit controls whether the Receive Message Buffer Interrupt line is asserted when the RBIF flag is set. 0 Disable interrupt line 1 Enable interrupt line
15 TBIE	**Transmit Message Buffer Interrupt Enable** This bit controls whether the Transmit Message Buffer Interrupt line is asserted when the TBIF flag is set. 0 Disable interrupt line 1 Enable interrupt line

57.5.2.14 Protocol Interrupt Flag Register 0 (FR_PIFR0)

Offset: 0x0018

Access: Read/Write
Write: Normal mode

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	FATL_IF	INTL_IF	ILCF_IF	CSA_IF	MRC_IF	MOC_IF	CCL_IF	MXS_IF	MTX_IF	LTXB_IF	LTXA_IF	TBVB_IF	TBVA_IF	TI2_IF	TI1_IF	CYS_IF
W	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1240. Protocol Interrupt Flag Register 0 (FR_PIFR0)

This register holds one set of the protocol-related individual interrupt flags.

Table 1316. FR_PIFR0 field descriptions

Field	Description
0 FATL_IF	**Fatal Protocol Error Interrupt Flag** This flag is set when the protocol engine has detected a fatal protocol error. In this case, the protocol engine goes into the *POC:halt* state immediately. The fatal protocol errors are: – *pLatestTx* violation, as described in the MAC process of the FlexRay protocol – Transmission across slot boundary violation, as described in the FSP process of the FlexRay protocol 0 No such event 1 Fatal protocol error detected
1 INTL_IF	**Internal Protocol Error Interrupt Flag** This flag is set when the protocol engine has detected an internal protocol error. In this case, the protocol engine goes into the *POC:halt* state immediately. An internal protocol error occurs when the protocol engine has not finished a calculation and a new calculation is requested. This can be caused by a hardware error. 0 No such event 1 Internal protocol error detected
2 ILCF_IF	**Illegal Protocol Configuration Interrupt Flag** This flag is set when the protocol engine has detected an illegal protocol configuration parameter setting. In this case, the protocol engine goes into the *POC:halt* state immediately. The protocol engine checks the *listen_timeout* value programmed into the Protocol Configuration Register 14 (FR_PCR14) and Protocol Configuration Register 15 (FR_PCR15) when the CONFIG_COMPLETE command was sent by the application via the Protocol Operation Control Register (FR_POCR). If the value of *listen_timeout* is equal to 0, the protocol configuration setting is considered as illegal. 0 No such event 1 Illegal protocol configuration detected
3 CSA_IF	**Cold Start Abort Interrupt Flag** This flag is set when the configured number of allowed cold start attempts is reached and none of these attempts was successful. The number of allowed cold start attempts is configured by the coldstart_attempts field in the Protocol Configuration Register 3 (FR_PCR3). 0 No such event 1 Cold start aborted and no more coldstart attempts allowed
4 MRC_IF	**Missing Rate Correction Interrupt Flag** This flag is set when an insufficient number of measurements is available for rate correction at the end of the communication cycle. 0 No such event 1 Insufficient number of measurements for rate correction detected
5 MOC_IF	**Missing Offset Correction Interrupt Flag** This flag is set when an insufficient number of measurements is available for offset correction. This is related to the MISSING_TERM event in the CSP process for offset correction in the FlexRay protocol. 0 No such event 1 Insufficient number of measurements for offset correction detected

Table 1316. FR_PIFR0 field descriptions (continued)

Field	Description
6 CCL_IF	**Clock Correction Limit Reached Interrupt Flag** This flag is set when the internal calculated offset or rate calculation values have reached or exceeded their configured thresholds as given by the *offset_correction_out* field in the Protocol Configuration Register 9 (FR_PCR9) and the *rate_correction_out* field in the Protocol Configuration Register 14 (FR_PCR14). 0 No such event 1 Offset or rate correction limit reached
7 MXS_IF	**Max Sync Frames Detected Interrupt Flag** This flag is set when the number of synchronization frames detected in the current communication cycle exceeds the value of the *node_sync_max* field in the Protocol Configuration Register 30 (FR_PCR30). 0 No such event 1 More than *node_sync_max* sync frames detected **Note:** Only synchronization frames that have passed the synchronization frame acceptance and rejection filters are taken into account
8 MTX_IF	**Media Access Test Symbol Received Interrupt Flag** This flag is set when the MTS symbol was received on channel A or channel B. 0 No such event 1 MTS symbol received
9 LTXB_IF	*pLatestTx* **Violation on Channel B Interrupt Flag** This flag is set when the frame transmission on channel B in the dynamic segment exceeds the dynamic segment boundary. This is related to the *pLatestTx* violation, as described in the MAC process of the FlexRay protocol. 0 No such event 1 *pLatestTx* violation occurred on channel B
10 LTXA_IF	*pLatestTx* **Violation on Channel A Interrupt Flag** This flag is set when the frame transmission on channel A in the dynamic segment exceeds the dynamic segment boundary. This is related to the *pLatestTx* violation as described in the MAC process of the FlexRay protocol. 0 No such event 1 *pLatestTx* violation occurred on channel A
11 TBVB_IF	**Transmission across boundary on channel B Interrupt Flag** This flag is set when the frame transmission on channel B crosses the slot boundary. This is related to the transmission across slot boundary violation as described in the FSP process of the FlexRay protocol. 0 No such event 1 Transmission across boundary violation occurred on channel B
12 TBVA_IF	**Transmission across boundary on channel A Interrupt Flag** This flag is set when the frame transmission on channel A crosses the slot boundary. This is related to the transmission across slot boundary violation as described in the FSP process of the FlexRay protocol. 0 No such event 1 Transmission across boundary violation occurred on channel A
13 TI2_IF	**Timer 2 Expired Interrupt Flag** This flag is set whenever timer 2 expires. 0 No such event 1 Timer 2 has reached its time limit

Table 1316. FR_PIFR0 field descriptions (continued)

Field	Description
14 TI1_IF	**Timer 1 Expired Interrupt Flag** This flag is set whenever timer 1 expires. 0 No such event 1 Timer 1 has reached its time limit
15 CYS_IF	**Cycle Start Interrupt Flag** This flag is set when a communication cycle starts. 0 No such event 1 Communication cycle started

57.5.2.15 Protocol Interrupt Flag Register 1 (FR_PIFR1)

Offset: 0x001A

Access: Read/Write
Write: Normal mode

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	EMC_IF	IPC_IF	PECF_IF	PSC_IF	SSI3_IF	SSI2_IF	SSI1_IF	SSI0_IF	0	0	EVT_IF	ODT_IF	0	0	0	0
W	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c			w1c	w1c				
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1241. Protocol Interrupt Flag Register 1 (FR_PIFR1)

The register holds one set of the protocol-related individual interrupt flags.

Table 1317. FR_PIFR1 field descriptions

Field	Description
0 EMC_IF	**Error Mode Changed Interrupt Flag** This flag is set when the value of the ERRMODE bit field in the Protocol Status Register 0 (FR_PSR0) is changed by the CC. 0 No such event 1 ERRMODE field changed
1 IPC_IF	**Illegal Protocol Control Command Interrupt Flag** This flag is set when the PE tries to execute a protocol control command that was issued via the POCCMD field of the Protocol Operation Control Register (FR_POCR), and detects that this protocol control command is not allowed in the current protocol state. In this case, the command is not executed. For more details, refer to *Section 57.7.6: Protocol control command execution*. 0 No such event 1 Illegal protocol control command detected
2 PECF_IF	**Protocol Engine Communication Failure Interrupt Flag** This flag is set if the CC has detected a communication failure between the PE and the CHI. 0 No such event 1 Protocol Engine Communication Failure detected

Table 1317. FR_PIFR1 field descriptions (continued)

Field	Description
3 PSC_IF	**Protocol State Changed Interrupt Flag** This flag is set when the protocol state in the PROTSTATE field in the Protocol Status Register 0 (FR_PSR0) has changed. 0 No such event 1 Protocol state changed
4 SSI3_IF	**Slot Status Counter Incremented Interrupt Flag** This flag is set when the SLOTSTATUSCNT field in the corresponding Slot Status Counter Register (FR_SSCR3) is incremented. 0 No such event 1 The corresponding slot status counter has incremented
5 SSI2_IF	**Slot Status Counter Incremented Interrupt Flag** This flag is set when the SLOTSTATUSCNT field in the corresponding Slot Status Counter Register (FR_SSCR2) is incremented. 0 No such event 1 The corresponding slot status counter has incremented
6 SSI1_IF	**Slot Status Counter Incremented Interrupt Flag** This flag is set when the SLOTSTATUSCNT field in the corresponding Slot Status Counter Register (FR_SSCR1) is incremented. 0 No such event 1 The corresponding slot status counter has incremented
7 SSI0_IF	**Slot Status Counter Incremented Interrupt Flag** This flag is set when the SLOTSTATUSCNT field in the corresponding Slot Status Counter Register (FR_SSCR0) is incremented. 0 No such event 1 The corresponding slot status counter has incremented
10 EVT_IF	**Even Cycle Table Written Interrupt Flag** This flag is set if the CC has written the sync frame measurement/ID tables into the FlexRay memory area for the even cycle. 0 No such event 1 Sync frame measurement table written
11 ODT_IF	**Odd Cycle Table Written Interrupt Flag** This flag is set if the CC has written the sync frame measurement/ID tables into the FlexRay memory area for the odd cycle. 0 No such event 1 Sync frame measurement table written

57.5.2.16 Protocol Interrupt Enable Register 0 (FR_PIER0)

Offset: 0x001C

Access: Read/Write
Write: Anytime

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	FATL_IE	INTL_IE	ILCF_IE	CSA_IE	MRC_IE	MOC_IE	CCL_IE	MXS_IE	MTX_IE	LTXB_IE	LTXA_IE	TBVB_IE	TBVA_IE	TI2_IE	TI1_IE	CYS_IE
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1242. Protocol Interrupt Enable Register 0 (FR_PIER0)

This register defines whether or not the individual interrupt flags defined in the Protocol Interrupt Flag Register 0 (FR_PIFR0) can generate a protocol interrupt request.

Table 1318. FR_PIER0 field descriptions

Field	Description
0 FATL_IE	**Fatal Protocol Error Interrupt Enable** This bit controls FATL_IF interrupt request generation. 0 Interrupt request generation disabled 1 Interrupt request generation enabled
1 INTL_IE	**Internal Protocol Error Interrupt Enable** This bit controls INTL_IF interrupt request generation. 0 Interrupt request generation disabled 1 Interrupt request generation enabled
2 ILCF_IE	**Illegal Protocol Configuration Interrupt Enable** This bit controls ILCF_IF interrupt request generation. 0 Interrupt request generation disabled 1 Interrupt request generation enabled
3 CSA_IE	**Cold Start Abort Interrupt Enable** This bit controls CSA_IF interrupt request generation. 0 Interrupt request generation disabled 1 Interrupt request generation enabled
4 MRC_IE	**Missing Rate Correction Interrupt Enable** This bit controls MRC_IF interrupt request generation. 0 Interrupt request generation disabled 1 Interrupt request generation enabled
5 MOC_IE	**Missing Offset Correction Interrupt Enable** This bit controls MOC_IF interrupt request generation. 0 Interrupt request generation disabled 1 Interrupt request generation enabled
6 CCL_IE	**Clock Correction Limit Reached Interrupt Enable** This bit controls CCL_IF interrupt request generation. 0 Interrupt request generation disabled 1 Interrupt request generation enabled

Table 1318. FR_PIER0 field descriptions (continued)

Field	Description
7 MXS_IE	**Max Sync Frames Detected Interrupt Enable** This bit controls MXS_IF interrupt request generation. 0 Interrupt request generation disabled 1 Interrupt request generation enabled
8 MTX_IE	**Media Access Test Symbol Received Interrupt Enable** This bit controls MTX_IF interrupt request generation. 0 Interrupt request generation disabled 1 Interrupt request generation enabled
9 LTXB_IE	*pLatestTx* **Violation on Channel B Interrupt Enable** This bit controls LTXB_IF interrupt request generation. 0 Interrupt request generation disabled 1 Interrupt request generation enabled
10 LTXA_IE	*pLatestTx* **Violation on Channel A Interrupt Enable** This bit controls LTXA_IF interrupt request generation. 0 Interrupt request generation disabled 1 Interrupt request generation enabled
11 TBVB_IE	**Transmission across boundary on channel B Interrupt Enable** This bit controls TBVB_IF interrupt request generation. 0 Interrupt request generation disabled 1 Interrupt request generation enabled
12 TBVA_IE	**Transmission across boundary on channel A Interrupt Enable** This bit controls TBVA_IF interrupt request generation. 0 Interrupt request generation disabled 1 Interrupt request generation enabled
13 TI2_IE	**Timer 2 Expired Interrupt Enable** This bit controls TI1_IF interrupt request generation. 0 Interrupt request generation disabled 1 Interrupt request generation enabled
14 TI1_IE	**Timer 1 Expired Interrupt Enable** This bit controls TI1_IF interrupt request generation. 0 Interrupt request generation disabled 1 Interrupt request generation enabled
15 CYS_IE	**Cycle Start Interrupt Enable** This bit controls CYC_IF interrupt request generation. 0 Interrupt request generation disabled 1 Interrupt request generation enabled

57.5.2.17 Protocol Interrupt Enable Register 1 (FR_PIER1)

Offset: 0x001E

Access: Read/Write
Write: Anytime

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	EMC_IE	IPC_IE	PECF_IE	PSC_IE	SSI3_IE	SSI2_IE	SSI1_IE	SSI0_IE	0	0	EVT_IE	ODT_IE	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1243. Protocol Interrupt Enable Register 1 (FR_PIER1)

This register defines whether or not the individual interrupt flags defined in Protocol Interrupt Flag Register 1 (FR_PIFR1) can generate a protocol interrupt request.

Table 1319. FR_PIER1 field descriptions

Field	Description
0 EMC_IE	**Error Mode Changed Interrupt Enable** This bit controls EMC_IF interrupt request generation. 0 Interrupt request generation disabled 1 Interrupt request generation enabled
1 IPC_IE	**Illegal Protocol Control Command Interrupt Enable** This bit controls IPC_IF interrupt request generation. 0 Interrupt request generation disabled 1 Interrupt request generation enabled
2 PECF_IE	**Protocol Engine Communication Failure Interrupt Enable** This bit controls PECF_IF interrupt request generation. 0 Interrupt request generation disabled 1 Interrupt request generation enabled
3 PSC_IE	**Protocol State Changed Interrupt Enable** This bit controls PSC_IF interrupt request generation. 0 Interrupt request generation disabled 1 Interrupt request generation enabled
4 SSI3_IE	**Slot Status Counter Incremented Interrupt Enable** This bit controls SSI3_IF interrupt request generation. 0 Interrupt request generation disabled 1 Interrupt request generation enabled
5 SSI2_IE	**Slot Status Counter Incremented Interrupt Enable** This bit controls SSI2_IF interrupt request generation. 0 Interrupt request generation disabled 1 Interrupt request generation enabled
6 SSI1_IE	**Slot Status Counter Incremented Interrupt Enable** This bit controls SSI1_IF interrupt request generation. 0 Interrupt request generation disabled 1 Interrupt request generation enabled

Table 1319. FR_PIER1 field descriptions (continued)

Field	Description
7 SSI0_IE	**Slot Status Counter Incremented Interrupt Enable** This bit controls SSI0_IF interrupt request generation. 0 Interrupt request generation disabled 1 Interrupt request generation enabled
10 EVT_IE	**Even Cycle Table Written Interrupt Enable** This bit controls EVT_IF interrupt request generation. 0 Interrupt request generation disabled 1 Interrupt request generation enabled
11 ODT_IE	**Odd Cycle Table Written Interrupt Enable** This bit controls ODT_IF interrupt request generation. 0 Interrupt request generation disabled 1 Interrupt request generation enabled

57.5.2.18 CHI Error Flag Register (FR_CHIERFR)

Offset: 0x0020

Access: Read/Write
Write: Normal mode

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	FRLB_EF	FRLA_EF	PCMI_EF	FOVB_EF	FOVA_EF	MBS_EF	MBU_EF	LCK_EF	0	SBCF_EF	FID_EF	DPL_EF	SPL_EF	NML_EF	NMF_EF	ILSA_EF
W	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c		w1c	w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1244. CHI Error Flag Register (FR_CHIERFR)

This register holds the CHI-related error flags. The interrupt generation for each of these error flags is controlled by the CHI interrupt enable bit, CHIE, in the Global Interrupt Flag and Enable Register (FR_GIFER).

Table 1320. FR_CHIERFR field descriptions

Field	Description
0 FRLB_EF	**Frame Lost Channel B Error Flag** This flag is set if a complete frame was received on channel B but it could not be stored in the selected individual message buffer because the message buffer is currently locked by the application. In this case, the frame and the related slot status information are lost. 0 No such event 1 Frame lost on channel B detected
1 FRLA_EF	**Frame Lost Channel A Error Flag** This flag is set if a complete frame was received on channel A but it could not be stored in the selected individual message buffer because the message buffer is currently locked by the application. In this case, the frame and the related slot status information are lost. 0 No such error 1 Frame lost on channel A detected

Table 1320. FR_CHIERFR field descriptions (continued)

Field	Description
2 PCMI_EF	**Protocol Command Ignored Error Flag** This flag is set if the application has issued a POC command by writing to the POCCMD field in the Protocol Operation Control Register (FR_POCR) while the BSY flag is equal to 1. In this case, the command is ignored by the CC and is lost. 0 No such error 1 POC command ignored
3 FOVB_EF	**Receive FIFO Overrun Channel B Error Flag** This flag is set when an overrun of the FIFO for channel B has occurred. This error occurs if a semantically valid frame was received on channel B and matches all criteria to be appended to the FIFO for channel B but the FIFO is full. In this case, the received frame and its related slot status information are lost. 0 No such error 1 FIFO overrun on channel B has been detected
4 FOVA_EF	**Receive FIFO Overrun Channel A Error Flag** This flag is set when an overrun of the FIFO for channel A has occurred. This error occurs if a semantically valid frame was received on channel A and matches the all criteria to be appended to the FIFO for channel A but the FIFO is full. In this case, the received frame and its related slot status information are lost. 0 No such error 1 FIFO overrun on channel B has been detected
5 MBS_EF	**Message Buffer Search Error Flag** This flag is set if at least one of the following events occurs: a) The message buffer search engine is still running while the next search must be started due to the FlexRay protocol timing. b) A message buffer index greater than 131 is detected in the FR_MBIDXR[MBIDX] field of a found message buffer or in one of the FR_RSBIR[RSBIDX] fields. Refer to *Section 57.6.7.4: Message buffer search error* for details. 0 No such event 1 Search engine active while search start appears or illegal message buffer index detected
6 MBU_EF	**Message Buffer Utilization Error Flag** This flag is asserted if the application writes to a message buffer control field that is beyond the number of utilized message buffers programmed in the Message Buffer Segment Size and Utilization Register (FR_MBSSUTR). If the application writes to a FR_MBCCSRn register with n > LAST_MB_UTIL, the CC ignores the write attempt and asserts the message buffer utilization error flag MBU_EF in the CHI Error Flag Register (FR_CHIERFR). 0 No such event 1 Non-utilized message buffer enabled
7 LCK_EF	**Lock Error Flag** This flag is set if the application tries to lock a message buffer that is already locked by the CC due to internal operations. In that case, the CC does not grant the lock to the application. The application must issue the lock request again. 0 No such error 1 Lock error detected

Table 1320. FR_CHIERFR field descriptions (continued)

Field	Description
9 SBCF_EF	**System Bus Communication Failure Error Flag** This flag is set if a system bus access was not finished within the required amount of time (refer to *Section 57.6.19.1.2: System bus access timeout*). 0 No such event 1 System bus access not finished in time
10 FID_EF	**Frame ID Error Flag** This flag is set if the frame ID stored in the message buffer header area differs from the frame ID stored in the message buffer control register. 0 No such error occurred 1 Frame ID error occurred
11 DPL_EF	**Dynamic Payload Length Error Flag** This flag is set if the payload length written into the message buffer header field of a transmit message buffer assigned to the dynamic segment is greater than the maximum payload length for the dynamic segment as it is configured in the corresponding max_payload_length_dynamic field in the Protocol Configuration Register 24 (FR_PCR24). 0 No such error occurred 1 Dynamic payload length error occurred
12 SPL_EF	**Static Payload Length Error Flag** This flag is set if the payload length written into the message buffer header field of a transmit message buffer assigned to the static segment is different from the payload length for the static segment as it is configured in the corresponding payload_length_static field in the Protocol Configuration Register 19 (FR_PCR19). 0 No such error occurred 1 Static payload length error occurred
13 NML_EF	**Network Management Length Error Flag** This flag is set if the payload length written into the header structure of a receive message buffer assigned to the static segment is less than the configured length of the Network Management Vector as configured in the Network Management Vector Length Register (FR_NMVLR). In this case, the received part of the Network Management Vector will be used to update the Network Management Vector. 0 No such error occurred 1 Network management length error occurred
14 NMF_EF	**Network Management Frame Error Flag** This flag is set if a received message in the static segment with a Preamble Indicator flag (PP) asserted has its Null Frame indicator flag (NF) asserted as well. In this case, the Global Network Management Registers (refer to *Section 57.5.2.40: Network Management Vector Registers (FR_NMVRn)*) are not updated. 0 No such error occurred 1 Network management frame error occurred
15 ILSA_EF	**Illegal System Bus Address Error Flag** This flag is set if the external system bus subsystem has detected an access to an illegal system bus address from the CC (refer to *Section 57.6.19.1.1: System bus illegal address access*). 0 No such event 1 Illegal system bus address accessed

57.5.2.19 Message Buffer Interrupt Vector Register (FR_MBIVEC)

Offset: 0x0022 Access: Read-only

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	TBIVEC							0	RBIVEC						
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1245. Message Buffer Interrupt Vector Register (FR_MBIVEC)

This register indicates the lowest numbered receive message buffer and the lowest numbered transmit message buffer that have their MBIF interrupt status flag and MBIE interrupt enable bits asserted. This means that message buffers with lower message buffer numbers have higher priority.

Table 1321. FR_MBIVEC field descriptions

Field	Description
1:7 TBIVEC	**Transmit Buffer Interrupt Vector** This field provides the number of the lowest numbered enabled transmit message buffer that has its MBIF interrupt status flag and its MBIE interrupt enable bit set. If there is no transmit message buffer with the MBIF interrupt status flag and the MBIE interrupt enable bits asserted, the value in this field is set to 0.
9:15 RBIVEC	**Receive Buffer Interrupt Vector** This field provides the message buffer number of the lowest numbered receive message buffer that has its MBIF interrupt flag and its MBIE interrupt enable bit asserted. If there is no receive message buffer with the MBIF interrupt status flag and the MBIE interrupt enable bits asserted, the value in this field is set to 0.

57.5.2.20 Channel A Status Error Counter Register (FR_CASERCR)

Offset: 0x0024 Access: Read-only

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	CHAERSCNT															
W																
Reset(1)	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. **A**dditional Reset: RUN Command

Figure 1246. Channel A Status Error Counter Register (FR_CASERCR)

This register provides the channel status error counter for channel A. The protocol engine generates a slot status vector for each static slot, each dynamic slot, the symbol window, and the NIT. The slot status vector contains the four protocol related error indicator bits *vSS!SyntaxError, vSS!ContentError, vSS!BViolation*, and *vSS!TxConflict*. The CC increments the status error counter by 1 if, for a slot or segment, at least one error indicator bit is set to 1. The counter wraps around after it has reached the maximum value. For more information on slot status monitoring, refer to *Section 57.6.18: Slot status monitoring*.

Table 1322. FR_CASERCR field descriptions

Field	Description
0:15 CHAERSCNT	**Channel Status Error Counter** This field provides the current value channel status error counter. The counter value is updated within the first macrotick of the following slot or segment.

57.5.2.21 Channel B Status Error Counter Register (FR_CBSERCR)

Offset: 0x0026 Access: Read-only

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	CHBERSCNT															
W																
Reset(1)	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. **A**dditional Reset: RUN Command

Figure 1247. Channel B Status Error Counter Register (FR_CBSERCR)

This register provides the channel status error counter for channel B. The protocol engine generates a slot status vector for each static slot, each dynamic slot, the symbol window, and the network idle time (NIT). The slot status vector contains the four protocol-related error indicator bits *vSS!SyntaxError*, *vSS!ContentError*, *vSS!BViolation*, and *vSS!TxConflict*. The CC increments the status error counter by 1 if, for a slot or segment, at least one error indicator bit is set to 1. The counter wraps around after it has reached the maximum value. For more information on slot status monitoring, refer to *Section 57.6.18: Slot status monitoring*.

Table 1323. FR_CBSERCR field descriptions

Field	Description
0:15 CHBERSCNT	**Channel Status Error Counter** This field provides the current channel status error count. The counter value is updated within the first macrotick of the following slot or segment.

57.5.2.22 Protocol Status Register 0 (FR_PSR0)

Offset: 0x0028 Access: Read-only

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	ERRMODE		SLOTMODE		0	PROTSTATE			STARTUPSTATE				0	WAKEUPSTATUS		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1248. .Protocol Status Register 0 (FR_PSR0)

This register provides information about the current protocol status.

Table 1324. FR_PSR0 field descriptions

Field	Description
0:1 ERRMODE	**Error Mode** protocol-related variable: *vPOC!ErrorMode*. This field indicates the error mode of the protocol. 00 ACTIVE 01 PASSIVE 10 COMM_HALT 11 Reserved
2:3 SLOTMODE	**Slot Mode** protocol-related variable: *vPOC!SlotMode*. This field indicates the slot mode of the protocol. 00 SINGLE 01 ALL_PENDING 10 ALL 11 Reserved
5:7 PROTSTATE	**Protocol State** protocol-related variable: *vPOC!State.* This field indicates the state of the protocol. 000 *POC:default config* 001 *POC:config* 010 *POC:wakeup* 011 *POC:ready* 100 *POC:normal passive* 101 *POC:normal active* 110 *POC:halt* 111 *POC:startup*

Table 1324. FR_PSR0 field descriptions (continued)

Field	Description
8:11 STARTUP STATE	**Startup State** protocol-related variable: *vPOC!StartupState*. This field indicates the current sub-state of the startup procedure. 0000 Reserved 0001 Reserved 0010 *POC:coldstart collision resolution* 0011 *POC:coldstart listen* 0100 *POC:integration consistency check* 0101 *POC:integration listen* 0110 Reserved 0111 *POC:initialize schedule* 1000 Reserved 1001 Reserved 1010 *POC:coldstart consistency check* 1011 Reserved 1100 Reserved 1101 *POC:integration coldstart check* 1110 *POC:coldstart gap* 1111 *POC:coldstart join*
13:15 WAKEUP STATUS	**Wakeup Status** protocol-related variable: *vPOC!WakeupStatus*. This field provides the outcome of the execution of the wakeup mechanism. 000 UNDEFINED 001 RECEIVED_HEADER 010 RECEIVED_WUP 011 COLLISION_HEADER 100 COLLISION_WUP 101 COLLISION_UNKNOWN 110 TRANSMITTED 111 Reserved

57.5.2.23 Protocol Status Register 1 (FR_PSR1)

Offset: 0x002A

Access: Read/Write
Write: Normal mode

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	CSAA	CSP	0	REMCSAT					CPN	HHR	FRZ	APTAC				
W	w1c															
Reset(1)	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. Additional Reset: CSAA, CSP, CPN: RUN Command

Figure 1249. Protocol Status Register 1 (FR_PSR1)

Table 1325. FR_PSR1 field descriptions

Field	Description
0 CSAA	**Cold Start Attempt Aborted Flag** protocol-related event: 'set coldstart abort indicator in CHI.' This flag is set when the CC has aborted a cold start attempt. 0 No such event 1 Cold start attempt aborted
1 CSP	**Leading Cold Start Path** This status bit is set when the CC has reached the *POC:normal active* state via the leading cold start path. This indicates that this node has started the network 0 No such event 1 *POC:normal active* reached from *POC:startup* state via leading cold start path
3:7 REMCSAT	**Remaining Coldstart Attempts** protocol-related variable: *vRemainingColdstartAttempts.* This field provides the number of remaining cold start attempts that the CC will execute.
8 CPN	**Leading Cold Start Path Noise**—protocol-related variable: *vPOC!ColdstartNoise.* This status bit is set if the CC has reached the *POC:normal active* state via the leading cold start path under noise conditions. This indicates there was some activity on the FlexRay bus while the CC was starting up the cluster. 0 No such event 1 *POC:normal active* state was reached from *POC:startup* state via noisy leading cold start path
9 HHR	**Host Halt Request Pending** protocol related variable: *vPOC!CHIHaltRequest.* This status bit is set when CC receives the HALT command from the application via the Protocol Operation Control Register (FR_POCR). The CC clears this status bit after a hard reset condition or when the protocol is in the *POC:default config* state. 0 No such event 1 HALT command received
10 FRZ	**Freeze Occurred** protocol related variable: *vPOC!Freeze.* This status bit is set when the CC has reached the *POC:halt* state due to the host FREEZE command or due to an internal error condition requiring immediate halt. The CC clears this status bit after a hard reset condition or when the protocol is in the *POC:default config* state. 0 No such event 1 Immediate halt due to FREEZE or internal error condition
11:15 APTAC	**Allow Passive to Active Counter** protocol-related variable: *vPOC!vAllowPassivetoActive.* This field provides the number of consecutive even/odd communication cycle pairs that have passed with valid rate and offset correction terms, but the protocol is still in the *POC:normal passive* state due to an application-configured delay to enter *POC:normal active* state. This delay is defined by the allow_passive_to_active field in Protocol Configuration Register 12 (FR_PCR12).

57.5.2.24 Protocol Status Register 2 (FR_PSR2)

Offset: 0x002C Access: Read-only

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	NBVB	NSEB	STCB	SBVB	SSEB	MTB	NBVA	NSEA	STCA	SBVA	SSEA	MTA	CKCORFCNT			
W																
Reset(1)	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. Additional Reset: RUN Command

Figure 1250. Protocol Status Register 2 (FR_PSR2)

This register provides a snapshot of status information about the network idle time (NIT), the symbol window, and the clock synchronization. The NIT-related status bits NBVB, NSEB, NBVA, and NSEA are updated by the CC after the end of the NIT and before the end of the first slot of the next communication cycle. The symbol-window-related status bits STCB, SBVB, SSEB, MTB, STCA, SBVA, SSEB, and MTA are updated by the CC after the end of the symbol window and before the end of the current communication cycle. If no symbol window is configured, the symbol-window-related status bits remain in their reset state. The clock synchronization-related CKCORFCNT is updated by the CC after the end of the static segment and before the end of the current communication cycle.

Table 1326. FR_PSR2 field descriptions

Field	Description
0 NBVB	**NIT Boundary Violation on Channel B** protocol-related variable: *vSS!BViolation* for NIT on channel B. This status bit is set when there was some media activity on the FlexRay bus channel B at the end of the NIT. 0 No such event 1 Media activity at boundaries detected
1 NSEB	**NIT Syntax Error on Channel B** protocol-related variable: *vSS!SyntaxError* for NIT on channel B. This status bit is set when a syntax error was detected during NIT on channel B. 0 No such event 1 Syntax error detected
2 STCB	**Symbol Window Transmit Conflict on Channel B** protocol-related variable: *vSS!TxConflict* for symbol window on channel B. This status bit is set if there was a transmission conflict during the symbol window on channel B. 0 No such event 1 Transmission conflict detected
3 SBVB	**Symbol Window Boundary Violation on Channel B** protocol-related variable: *vSS!BViolation* for symbol window on channel B. This status bit is set if there was some media activity on the FlexRay bus channel B at the start or at the end of the symbol window. 0 No such event 1 Media activity at boundaries detected

Table 1326. FR_PSR2 field descriptions (continued)

Field	Description
4 SSEB	**Symbol Window Syntax Error on Channel B** protocol related variable: *vSS!SyntaxError* for symbol window on channel B. This status bit is set when a syntax error was detected during the symbol window on channel B. 0 No such event 1 Syntax error detected
5 MTB	**Media Access Test Symbol MTS Received on Channel B** protocol-related variable: *vSS!ValidMTS* for Symbol Window on channel B. This status bit is set if the Media Access Test Symbol MTS was received in the symbol window on channel B. 0 No such event 1 MTS symbol received
6 NBVA	**NIT Boundary Violation on Channel A** protocol-related variable: *vSS!BViolation* for NIT on channel A. This status bit is set when there was some media activity on the FlexRay bus channel A at the end of the NIT. 0 No such event 1 Media activity at boundaries detected
7 NSEA	**NIT Syntax Error on Channel A** protocol-related variable: *vSS!SyntaxError* for NIT on channel A This status bit is set when a syntax error was detected during NIT on channel A. 0 No such event 1 Syntax error detected
8 STCA	**Symbol Window Transmit Conflict on Channel A** protocol related variable: *vSS!TxConflict* for symbol window on channel A. This status bit is set if there was a transmission conflicts during the symbol window on channel A. 0 No such event 1 Transmission conflict detected
9 SBVA	**Symbol Window Boundary Violation on Channel A** protocol-related variable: *vSS!BViolation* for symbol window on channel A. This status bit is set if there was some media activity on the FlexRay bus channel A at the start or at the end of the symbol window. 0 No such event 1 Media activity at boundaries detected
10 SSEA	**Symbol Window Syntax Error on Channel A** protocol-related variable: *vSS!SyntaxError* for symbol window on channel A. This status bit is set when a syntax error was detected during the symbol window on channel A. 0 No such event 1 Syntax error detected

Table 1326. FR_PSR2 field descriptions (continued)

Field	Description
11 MTA	**Media Access Test Symbol MTS Received on Channel A** protocol-related variable: *vSS!ValidMTS* for symbol window on channel A. This status bit is set if the Media Access Test Symbol (MTS) was received in the symbol window on channel A. 0 No such event 1 MTS symbol received
12:15 CKCORFCNT	**Clock Correction Failed Counter** protocol -elated variable: *vClockCorrectionFailed.* This field provides the number of consecutive even/odd communication cycle pairs that have passed without a clock synchronization offset or rate correction due to lack of synchronization frames. It is not incremented when it has reached the configured value of either max_without_clock_correction_fatal or max_without_clock_correction_passive as defined in Protocol Configuration Register 8 (FR_PCR8). The CC resets this counter on a hard reset condition, when the protocol enters the *POC:normal active* state, or when both the rate and offset correction terms have been calculated successfully.

57.5.2.25 Protocol Status Register 3 (FR_PSR3)

Offset: 0x002E

Access: Read/Write
Write: Normal mode

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	WUB	ABVB	AACB	ACEB	ASEB	AVFB	0	0	WUA	ABVA	AACA	ACEA	ASEA	AVFA
W			w1c	w1c	w1c	w1c	w1c	w1c			w1c	w1c	w1c	w1c	w1c	w1c
Reset(1)	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. Additional Reset: RUN Command

Figure 1251. Protocol Status Register 3 (FR_PSR3)

This register provides aggregated channel status information as an accrued status of channel activity for all communication slots, regardless of whether they are assigned for transmission or subscribed for reception. It provides accrued information for the symbol window, the NIT, and the wakeup status.

Table 1327. FR_PSR3 field descriptions

Field	Description
2 WUB	**Wakeup Symbol Received on Channel B** This flag is set when a wakeup symbol was received on channel B. 0 No wakeup symbol received 1 Wakeup symbol received
3 ABVB	**Aggregated Boundary Violation on Channel B** This flag is set when a boundary violation was detected on channel B. Boundary violations are detected in the communication slots, the symbol window, and the NIT. 0 No boundary violation detected 1 Boundary violation detected

Table 1327. FR_PSR3 field descriptions (continued)

Field	Description
4 AACB	**Aggregated Additional Communication on Channel B** This flag is set when at least one valid frame was received on channel B in a slot that also contained an additional communication with either syntax error, content error, or boundary violations. 0 No additional communication detected 1 Additional communication detected
5 ACEB	**Aggregated Content Error on Channel B** This flag is set when a content error was detected on channel B. Content errors are detected in the communication slots, the symbol window, and the NIT. 0 No content error detected 1 Content error detected
6 ASEB	**Aggregated Syntax Error on Channel B** This flag is set when a syntax error was detected on channel B. Syntax errors are detected in the communication slots, the symbol window, and the NIT. 0 No syntax error detected 1 Syntax errors detected
7 AVFB	**Aggregated Valid Frame on Channel B** This flag is set when a syntactically correct valid frame has been received in any static or dynamic slot through channel B. 0 No syntactically valid frames received 1 At least one syntactically valid frame received
10 WUA	**Wakeup Symbol Received on Channel A** This flag is set when a wakeup symbol was received on channel A. 0 No wakeup symbol received 1 Wakeup symbol received
11 ABVA	**Aggregated Boundary Violation on Channel A** This flag is set when a boundary violation was detected on channel A. Boundary violations are detected in the communication slots, the symbol window, and the NIT. 0 No boundary violation detected 1 Boundary violation detected
12 AACA	**Aggregated Additional Communication on Channel A** This flag is set when a valid frame was received in a slot on channel A that also contained an additional communication with either syntax error, content error, or boundary violations. 0 No additional communication detected 1 Additional communication detected
13 ACEA	**Aggregated Content Error on Channel A** This flag is set when a content error has been detected on channel A. Content errors are detected in the communication slots, the symbol window, and the NIT. 0 No content error detected 1 Content error detected

Table 1327. FR_PSR3 field descriptions (continued)

Field	Description
14 ASEA	**Aggregated Syntax Error on Channel A** This flag is set when a syntax error has been detected on channel A. Syntax errors are detected in the communication slots, the symbol window, and the NIT. 0 No syntax error detected 1 Syntax errors detected
15 AVFA	**Aggregated Valid Frame on Channel A** This flag is set when a syntactically correct valid frame has been received in any static or dynamic slot through channel A. 0 No syntactically valid frames received 1 At least one syntactically valid frame received

57.5.2.26 Macrotick Counter Register (FR_MTCTR)

Offset: 0x0030 Access: Read-only

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	MTCT													
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1252. Macrotick Counter Register (FR_MTCTR)

This register provides the macrotick count of the current communication cycle.

Table 1328. FR_MTCTR field descriptions

Field	Description
2:15 MTCT	**Macrotick Counter** protocol-related variable: *vMacrotick.* This field provides the macrotick count of the current communication cycle.

57.5.2.27 Cycle Counter Register (FR_CYCTR)

Offset: 0x0032 Access: Read-only

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	CYCCNT					
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1253. Cycle Counter Register (FR_CYCTR)

This register provides the number of the current communication cycle.

Table 1329. FR_CYCTR field descriptions

Field	Description
10:15 CYCCNT	**Cycle Counter** protocol-related variable: *vCycleCounter.* This field provides the number of the current communication cycle. If the counter reaches the maximum value of 63, the counter wraps and starts from 0 again.

57.5.2.28 Slot Counter Channel A Register (FR_SLTCTAR)

Offset: 0x0034 Access: Read-only

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	SLOTCNTA										
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1254. Slot Counter Channel A Register (FR_SLTCTAR)

This register provides the number of the current slot in the current communication cycle for channel A.

Table 1330. FR_SLTCTAR field descriptions

Field	Description
5:15 SLOTCNTA	**Slot Counter Value for Channel A** protocol-related variable: *vSlotCounter* for channel A. This field provides the number of the current slot in the current communication cycle.

57.5.2.29 Slot Counter Channel B Register (FR_SLTCTBR)

Offset: 0x0036 Access: Read-only

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	SLOTCNTB										
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1255. Slot Counter Channel B Register (FR_SLTCTBR)

This register provides the number of the current slot in the current communication cycle for channel B.

Table 1331. FR_SLTCTBR field descriptions

Field	Description
5:15 SLOTCNTB	**Slot Counter Value for Channel B** protocol-related variable: *vSlotCounter* for channel B. This field provides the number of the current slot in the current communication cycle.

57.5.2.30 Rate Correction Value Register (FR_RTCORVR)

Offset: 0x0038 Access: Read-only

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	RATECORR															
W																
Reset(1)	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. Additional Reset: RUN Command

Figure 1256. Rate Correction Value Register (FR_RTCORVR)

This register provides the sign extended rate correction value in microticks as it was calculated by the clock synchronization algorithm. The CC updates this register during the NIT of each odd numbered communication cycle.

Table 1332. FR_RTCORVR field descriptions

Field	Description
0:15 RATECORR	**Rate Correction Value** protocol-related variable: *vRateCorrection* (before value limitation and external rate correction). This field provides the sign extended rate correction value in microticks as it was calculated by the clock synchronization algorithm. The value is represented in 2's complement format. This value does not include the value limitation and the application of the external rate correction. If the magnitude of the internally calculated rate correction value exceeds the limit given by rate_correction_out in the Protocol Configuration Register 13 (FR_PCR13), the clock correction reached limit interrupt flag (CCL_IF) is set in the Protocol Interrupt Flag Register 0 (FR_PIFR0). **Note:** If the CC was not able to calculate a new rate correction term due to a lack of synchronization frames, the RATECORR value is not updated.

57.5.2.31 Offset Correction Value Register (FR_OFCORVR)

Offset: 0x003A Access: Read-only

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	OFFSETCORR															
W																
Reset(1)	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. Additional Reset: RUN Command

Figure 1257. Offset Correction Value Register (FR_OFCORVR)

This register provides the sign extended offset correction value in microticks as it was calculated by the clock synchronization algorithm. The CC updates this register during the NIT.

Table 1333. FR_OFCORVR field descriptions

Field	Description
0:15 OFFSETCORR	**Offset Correction Value** protocol-related variable: *vOffsetCorrection* (before value limitation and external offset correction). This field provides the sign extended offset correction value in microticks as it was calculated by the clock synchronization algorithm. The value is represented in 2's complement format. This value does not include the value limitation and the application of the external offset correction. If the magnitude of the internally calculated rate correction value exceeds the limit given by offset_correction_out field in the Protocol Configuration Register 29 (FR_PCR29), the clock correction reached limit interrupt flag CCL_IF is set in the Protocol Interrupt Flag Register 0 (FR_PIFR0). **Note:** If the CC was not able to calculate an new offset correction term due to a lack of synchronization frames, the OFFSETCORR value is not updated.

57.5.2.32 Combined Interrupt Flag Register (FR_CIFR)

Offset: 0x003C Access: Read-only

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	MIF	PRIF	CHIF	WUPIF	FAFBIF	FAFAIF	RBIF	TBIF
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1258. Combined Interrupt Flag Register (FR_CIFR)

This register provides five combined interrupt flags and a copy of three individual interrupt flags. The combined interrupt flags are the result of a binary OR of the values of other interrupt flags regardless of the state of the interrupt enable bits. The generation scheme for the combined interrupt flags is depicted in *Figure 1389*. The individual interrupt flags WUPIF, FAFBIF, and FAFAIF are copies of corresponding flags in the Global Interrupt Flag and Enable Register (FR_GIFER) and are provided here to simplify the application interrupt flag check. To clear the individual interrupt flags, the application must use the Global Interrupt Flag and Enable Register (FR_GIFER).

Note: The meanings of the combined status bits MIF, PRIF, CHIF, RBIF, and TBIF are different from those mentioned in the Global Interrupt Flag and Enable Register (FR_GIFER).

Table 1334. FR_CIFR field descriptions

Field	Description
8 MIF	**Module Interrupt Flag** This flag is set if there is at least one interrupt source that has its interrupt flag asserted. 0 No interrupt source has its interrupt flag asserted 1 At least one interrupt source has its interrupt flag asserted
9 PRIF	**Protocol Interrupt Flag** This flag is set if at least one of the individual protocol interrupt flags in the Protocol Interrupt Flag Register 0 (FR_PIFR0) or Protocol Interrupt Flag Register 1 (FR_PIFR1) is equal to 1. 0 All individual protocol interrupt flags are equal to 0 1 At least one of the individual protocol interrupt flags is equal to 1
10 CHIF	**CHI Interrupt Flag** This flag is set if at least one of the individual CHI error flags in the CHI Error Flag Register (FR_CHIERFR) is equal to 1. 0 All CHI error flags are equal to 0 1 At least one CHI error flag is equal to 1
11 WUPIF	**Wakeup Interrupt Flag** Provides the same value as FR_GIFER[WUPIF].
12 FAFBIF	**Receive FIFO Channel B Almost Full Interrupt Flag** Provides the same value as FR_GIFER[FAFBIF].
13 FAFAIF	**Receive FIFO Channel A Almost Full Interrupt Flag** Provides the same value as FR_GIFER[FAFAIF].
14 RBIF	**Receive Message Buffer Interrupt Flag** This flag is set if for at least one of the individual receive message buffers (FR_MBCCSRn[MTD] = 0) the interrupt flag MBIF in the corresponding Message Buffer Configuration, Control, Status Registers (FR_MBCCSRn) is equal to 1. 0 None of the individual receive message buffers has the MBIF flag asserted 1 At least one individual receive message buffers has the MBIF flag asserted
15 TBIF	**Transmit Message Buffer Interrupt Flag** This flag is set if for at least one of the individual transmit message buffers (FR_MBCCSRn[MTD] = 1) the interrupt flag MBIF in the corresponding Message Buffer Configuration, Control, Status Registers (FR_MBCCSRn) is equal to 1. 0 None of the individual transmit message buffers has the MBIF flag asserted 1 At least one individual transmit message buffers has the MBIF flag asserted

57.5.2.33 System Memory Access Timeout Register (FR_SYMATOR)

Offset: 0x003E

Access: Read/Write
Write: Disabled mode

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	TIMEOUT							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0

Figure 1259. System Memory Access Time-Out Register (FR_SYMATOR)

Table 1335. FR_SYMATOR field descriptions

Field	Description
8:15 TIMEOUT	**System Memory Access Timeout** This value defines when a system bus access timeout is detected. For a detailed description refer to *Section 57.7.1.1: Configure System Memory Access Timeout Register (FR_SYMATOR)* and *Section 57.6.19.1.2: System bus access timeout*.

57.5.2.34 Sync Frame Counter Register (FR_SFCNTR)

Offset: 0x0040 Access: Read-only

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	SFEVB				SFEVA				SFODB				SFODA			
W																
Reset(1)	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. Additional Reset: RUN Command

Figure 1260. Sync Frame Counter Register (FR_SFCNTR)

This register provides the number of synchronization frames that are used for clock synchronization in the last even and in the last odd numbered communication cycle. This register is updated after the start of the NIT and before 10 macroticks after offset correction start.

Note: If the application has locked the even synchronization table at the end of the static segment of an even communication cycle, the CC will not update the fields SFEVB and SFEVA.

If the application has locked the odd synchronization table at the end of the static segment of an odd communication cycle, the CC will not update the SFODB and SFODA values.

Table 1336. FR_SFCNTR field descriptions

Field	Description
0:3 SFEVB	Sync Frames Channel B, even cycle protocol-related variable: size of (*vsSyncIdListB* for even cycle). This field provides the size of the internal list of frame IDs of received synchronization frames used for clock synchronization.
4:7 SFEVA	Sync Frames Channel A, even cycle protocol related variable: size of (*vsSyncIdListA* for even cycle). This field provides the size of the internal list of frame IDs of received synchronization frames used for clock synchronization.
8:11 SFODB	Sync Frames Channel B, odd cycle protocol-related variable: size of (*vsSyncIdListB* for odd cycle). This field provides the size of the internal list of frame IDs of received synchronization frames used for clock synchronization.
12:15 SFODA	Sync Frames Channel A, odd cycle protocol-related variable: size of (*vsSyncIdListA* for odd cycle). This field provides the size of the internal list of frame IDs of received synchronization frames used for clock synchronization.

57.5.2.35 Sync Frame Table Offset Register (FR_SFTOR)

Offset: 0x0042

Access: Read/Write
Write: *POC:config*

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	SFT_OFFSET															0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1261. Sync Frame Table Offset Register (FR_SFTOR)

This register defines the FlexRay memory area related offset for sync frame tables. For more details, refer to *Section 57.6.12: Sync frame ID and sync frame deviation tables*.

Table 1337. FR_SFTOR field descriptions

Field	Description
0:14 SFT_OFFSET	**Sync Frame Table Offset** The offset of the sync frame tables in the FlexRay memory area. This offset is required to be 16-bit aligned. Thus, STF_OFFSET[0] is always 0.

57.5.2.36 Sync Frame Table Configuration, Control, Status Register (FR_SFTCCSR)

Offset: 0x0044

Access: Read/Write
Write: Normal mode

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	CYCNUM						ELKS	OLKS	EVAL	OVAL	0	0	SDVEN	SIDEN
W	ELKT	OLKT												OPT		
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1262. Sync Frame Table Configuration, Control, Status Register (FR_SFTCCSR)

This register provides configuration, control, and status information related to the generation and access of the clock sync ID tables and clock sync measurement tables. For a detailed description, refer to *Section 57.6.12: Sync frame ID and sync frame deviation tables*.

Table 1338. FR_SFTCCSR field descriptions

Field	Description
0 ELKT	**Even Cycle Tables Lock/Unlock Trigger** This trigger bit is used to lock and unlock the even cycle tables. 0 No effect 1 Triggers lock/unlock of the even cycle tables
1 OLKT	**Odd Cycle Tables Lock/Unlock Trigger** This trigger bit is used to lock and unlock the odd cycle tables. 0 No effect 1 Triggers lock/unlock of the odd cycle tables

Table 1338. FR_SFTCCSR field descriptions (continued)

Field	Description
2:7 CYCNUM	**Cycle Number** This field provides the number of the cycle in which the currently locked table was recorded. If none or both tables are locked, this value is related to the even cycle table.
8 ELKS	**Even Cycle Tables Lock Status** This status bit indicates whether the application has locked the even cycle tables. 0 Application has not locked the even cycle tables 1 Application has locked the even cycle tables
9 OLKS	**Odd Cycle Tables Lock Status** This status bit indicates whether the application has locked the odd cycle tables. 0 Application has not locked the odd cycle tables 1 Application has locked the odd cycle tables
10 EVAL	**Even Cycle Tables Valid** This status bit indicates whether the Sync Frame ID and Sync Frame Deviation Tables for the even cycle are valid. The CC clears this status bit when it starts updating the tables, and sets this bit when it has finished the table update. 0 Tables are not valid (update is ongoing) 1 Tables are valid (consistent)
11 OVAL	**Odd Cycle Tables Valid** This status bit indicates whether the Sync Frame ID and Sync Frame Deviation Tables for the odd cycle are valid. The CC clears this status bit when it starts updating the tables, and sets this bit when it has finished the table update. 0 Tables are not valid (update is ongoing) 1 Tables are valid (consistent)
13 OPT	**One Pair Trigger** This trigger bit controls whether the CC writes continuously or only one pair of Sync Frame Tables into the FlexRay memory area. If this trigger is set to 1 while SDVEN or SIDEN is set to 1, the CC writes only one pair of the enabled Sync Frame Tables corresponding to the next even-odd-cycle pair into the FlexRay memory area. In this case, the CC clears the SDVEN or SIDEN bits immediately. If this trigger is set to 0 while SDVEN or SIDEN is set to 1, the CC writes continuously the enabled Sync Frame Tables into the FlexRay memory area. 0 Write continuously pairs of enabled Sync Frame Tables into FlexRay memory area. 1 Write only one pair of enabled Sync Frame Tables into FlexRay memory area.

Table 1338. FR_SFTCCSR field descriptions (continued)

Field	Description
14 SDVEN	**Sync Frame Deviation Table Enable** This bit controls the generation of the Sync Frame Deviation Tables. The application must set this bit to request the CC to write the Sync Frame Deviation Tables into the FlexRay memory area. 0 Do not write Sync Frame Deviation Tables 1 Write Sync Frame Deviation Tables into FlexRay memory area **Note:** If SDVEN is set to 1, then SIDEN must also be set to 1.
15 SIDEN	**Sync Frame ID Table Enable** This bit controls the generation of the Sync Frame ID Tables. The application must set this bit to 1 to request the CC to write the Sync Frame ID Tables into the FlexRay memory area. 0 Do not write Sync Frame ID Tables 1 Write Sync Frame ID Tables into FlexRay memory area

57.5.2.37 Sync Frame ID Rejection Filter Register (FR_SFIDRFR)

Offset: 0x0046

Access: Read/Write(1)
Write: Normal mode

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	SYNFRID									
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. 16-bit write access required

Figure 1263. Sync Frame ID Rejection Filter Register (FR_SFIDRFR)

This register defines the Sync Frame Rejection Filter ID. The application must update this register outside of the static segment. If the application updates this register in the static segment, it can appear that the CC accepts the sync frame in the current cycle.

Table 1339. FR_SFIDRFR field descriptions

Field	Description
6:15 SYNFRID	**Sync Frame Rejection ID** This field defines the frame ID of a frame that must not be used for clock synchronization. For details, refer to *Section 57.6.15.2: Sync frame rejection filtering*.

57.5.2.38 Sync Frame ID Acceptance Filter Value Register (FR_SFIDAFVR)

Offset: 0x0048

Access: Read/Write
Write: *POC:config*

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	FVAL									
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1264. Sync Frame ID Acceptance Filter Value Register (FR_SFIDAFVR)

This register defines the sync frame acceptance filter value. For details on filtering, refer to *Section 57.6.15: Sync frame filtering*.

Table 1340. FR_SFIDAFVR field descriptions

Field	Description
6:15 FVAL	**Filter Value** This field defines the value for the sync frame acceptance filtering.

57.5.2.39 Sync Frame ID Acceptance Filter Mask Register (FR_SFIDAFMR)

Offset: 0x004A

Access: Read/Write
Write: *POC:config*

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	FMSK									
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1265. Sync Frame ID Acceptance Filter Mask Register (FR_SFIDAFMR)

This register defines the sync frame acceptance filter mask. For details on filtering, refer to *Section 57.6.15.1: Sync frame acceptance filtering*.

Table 1341. FR_SFIDAFMR field descriptions

Field	Description
6:15 FMSK	**Filter Mask** This field defines the mask for the sync frame acceptance filtering.

57.5.2.40 Network Management Vector Registers (FR_NMVR*n*)

Offset: 0x004C +*n**0x2 (*n*=0 to 5)

Access: Read-only

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	NMVP															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1266. Network Management Vector Registers (FR_NMVR*n*)

Each of these six registers holds one part of the Network Management Vector. The length of the Network Management Vector is configured in the Network Management Vector Length Register (FR_NMVLR). If FR_NMVLR is programmed with a value that is less than 12 bytes, the remaining bytes of the Network Management Vector Registers (FR_NMVR0–FR_NMVR5), which are not used for the Network Management Vector accumulating, will remain 0.

The NMVR provides accrued information over all received NMVs in the last communication cycle. All NMVs received in one cycle are ORed into the NMVR. The NMVR is updated at the end of the communication cycle.

Table 1342. NMVR*n* field descriptions

Field	Description
0:15 NMVP	**Network Management Vector Part** The mapping between the Network Management Vector Registers (FR_NMVR0–FR_NMVR5) and the receive message buffer payload bytes in NMV[0:11] is depicted in *Table 1343*.

Table 1343. Mapping of NMVRn to the received payload bytes NMV*x*

NMVR*n* register	NMV*x* received payload
FR_NMVR0.NMVP[0:7]	NMV0
FR_NMVR0.NMVP[8:15]	NMV1
FR_NMVR1.NMVP[0:7]	NMV2
FR_NMVR1.NMVP[[8:15]	NMV3
FR_NMVR2.NMVP[0:7]	NMV4
FR_NMVR2.NMVP[8:15]	NMV5
FR_NMVR3.NMVP[0:7]	NMV6
FR_NMVR3.NMVP[8:15]	NMV7
FR_NMVR4.NMVP[0:7]	NMV8
FR_NMVR4.NMVP[8:15]	NMV9
FR_NMVR5.NMVP[0:7]	NMV10
FR_NMVR5.NMVP[8:15]	NMV11

57.5.2.41 Network Management Vector Length Register (FR_NMVLR)

Offset: 0x0058

Access: Read/Write
Write: *POC:config*

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	NMVL			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1267. Network Management Vector Length Register (FR_NMVLR)

This register defines the length of the network management vector in bytes.

Table 1344. FR_NMVLR field descriptions

Field	Description
12:15 NMVL	**Network Management Vector Length** Protocol-related variable: *gNetworkManagementVectorLength.* This field defines the length of the Network Management Vector in bytes. Legal values are between 0 and 12.

57.5.2.42 Timer Configuration and Control Register (FR_TICCR)

Access: Read/Write
Offset: 0x005A Write: T2_CFG: *POC:config*
T2_REP, T1_REP, T1SP, T2SP, T1TR, T2TR: Normal mode

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	T2_CFG	T2_REP	0	0	0	T2ST	0	0	0	T1_REP	0	0	0	T1ST
W						T2SP	T2TR							T1SP	T1TR	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1268. Timer Configuration and Control Register (FR_TICCR)

This register is used to configure and control the T1 and T2 timers. For timer details, refer to *Section 57.6.17: Timer support*. The T1 timer is an absolute timer. The T2 timer can be configured as an absolute or relative timer.

Table 1345. FR_TICCR field descriptions

Field	Description
2 T2_CFG	**Timer T2 Configuration** This bit configures the timebase mode of timer T2. 0 T2 is absolute timer. 1 T2 is relative timer.
3 T2_REP	**Timer T2 Repetitive Mode** This bit configures the repetition mode of timer T2. 0 T2 is non repetitive. 1 T2 is repetitive.
5 T2SP	**Timer T2 Stop** This trigger bit is used to stop timer T2. 0 No effect 1 Stop timer T2
6 T2TR	**Timer T2 Trigger** This trigger bit is used to start timer T2. 0 No effect 1 Start timer T2

Table 1345. FR_TICCR field descriptions (continued)

Field	Description
7 T2ST	**Timer T2 State** This status bit provides the current state of timer T2. 0 Timer T2 is idle. 1 Timer T2 is running.
11 T1_REP	**Timer T1 Repetitive Mode** This bit configures the repetition mode of timer T1. 0 T1 is non repetitive. 1 T1 is repetitive.
13 T1SP	**Timer T1 Stop** This trigger bit is used to stop timer T1. 0 No effect 1 Stop timer T1
14 T1TR	**Timer T1 Trigger** This trigger bit is used to start timer T1. 0 No effect 1 Start timer T1
15 T1ST	**Timer T1 State** This status bit provides the current state of timer T1. 0 Timer T1 is idle. 1 Timer T1 is running.

Note: Both timers are deactivated immediately when the protocol enters a state different from POC:normal active or POC:normal passive.

57.5.2.43 Timer 1 Cycle Set Register (FR_TI1CYSR)

Offset: 0x005C

Access: Read/Write
Write: Anytime

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	T1_CYC_VAL						0	0	T1_CYC_MSK					
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1269. Timer 1 Cycle Set Register (FR_TI1CYSR)

This register defines the cycle filter value and the cycle filter mask for timer T1. For a detailed description of timer T1, refer to *Section 57.6.17.1: Absolute timer T1*.

Table 1346. FR_TI1CYSR field descriptions

Field	Description
2:7 T1_CYC_VAL	**Timer T1 Cycle Filter Value** This field defines the cycle filter value for timer T1.
10:15 T1_CYC_MSK	**Timer T1 Cycle Filter Mask** This field defines the cycle filter mask for timer T1.

Note: If the application modifies the value in this register while the timer is running, the change becomes effective immediately and timer T1 will expire according to the changed value.

57.5.2.44 Timer 1 Macrotick Offset Register (FR_TI1MTOR)

Offset: 0x005E

Access: Read/Write
Write: Anytime

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	T1_MTOFFSET													
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1270. Timer 1 Macrotick Offset Register (FR_TI1MTOR)

This register holds the macrotick offset value for timer T1. For a detailed description of timer T1, refer to *Section 57.6.17.1: Absolute timer T1*.

Table 1347. FR_TI1MTOR field descriptions

Field	Description
2:15 T1_MTOFFSET	**Timer 1 Macrotick Offset** This field defines the macrotick offset value for timer 1.

Note: If the application modifies the value in this register while the timer is running, the change becomes effective immediately and timer T1 will expire according to the changed value.

57.5.2.45 Timer 2 Configuration Register 0 (FR_TI2CR0)

Offset: 0x0060

Access: Read/Write
Write: Anytime

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	Reserved(1)		T2_CYC_VAL						Reserved(1)		T2_CYC_MSK					
W																
R	T2_MTCNT[31:16]															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. It is read as 0. Application must not write 1 to this bit.

Figure 1271. Timer 2 Configuration Register 0 (FR_TI2CR0)

The content of this register depends on the value of the T2_CFG bit in the Timer Configuration and Control Register (FR_TICCR). For a detailed description of timer T2, refer to *Section 57.6.17.2: Absolute/relative timer T2*.

Table 1348. FR_TI2CR0 field descriptions

Field	Description
Fields for absolute timer T2 (FR_TICCR[T2_CFG] = 0)	
2:7 T2_CYC_VAL	**Timer T2 Cycle Filter Value** This field defines the cycle filter value for timer T2.
10:15 T2_CYC_MSK	**Timer T2 Cycle Filter Mask** This field defines the cycle filter mask for timer T2.
Fields for relative timer T2 (FR_TICCR[T2_CFG] = 1)	
0:15 T2_MTCNT[31:16]	**Timer T2 Macrotick High Word** This field defines the high word of the macrotick count for timer T2.

Note: If timer T2 is configured as an absolute timer and the application modifies the values in this register while the timer is running, the change becomes effective immediately and timer T2 will expire according to the changed values.

If timer T2 is configured as a relative timer and the application changes the values in this register while the timer is running, the change becomes effective when the timer has expired according to the old values.

57.5.2.46 Timer 2 Configuration Register 1 (FR_TI2CR1)

Offset: 0x0062

Access: Read/Write
Write: Anytime

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	Reserved(1)				T2_MTOFFSET											
R W	T2_MTCNT															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. It is read as 0. Application must not write 1 to this bit.

Figure 1272. Timer 2 Configuration Register 1 (FR_TI2CR1)

The content of this register depends on the value of the T2_CFG bit in the Timer Configuration and Control Register (FR_TICCR). For a detailed description of timer T2, refer to *Section 57.6.17.2: Absolute/relative timer T2*.

Table 1349. FR_TI2CR1 field descriptions

Field	Description
Fields for absolute timer T2 (FR_TICCR[T2_CFG] = 0)	
2:15 T2_MTOFFSET	**Timer T2 Macrotick Offset** This field holds the macrotick offset value for timer T2.

Table 1349. FR_TI2CR1 field descriptions

Field	Description
Fields for relative timer T2 (FR_TICCR[T2_CFG] = 1)	
0:15 T2_MTCNT	**Timer T2 Macrotick Low Word** This field defines the low word of the macrotick value for timer T2.

Note: If timer T2 is configured as an absolute timer and the application modifies the values in this register while the timer is running, the change becomes effective immediately and the timer T2 will expire according to the changed values.

If timer T2 is configured as a relative timer and the application changes the values in this register while the timer is running, the change becomes effective when the timer has expired according to the old values.

57.5.2.47 Slot Status Selection Register (FR_SSSR)

Offset: 0x0064

Access: Read/Write(1)
Write: Anytime

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	SEL		0	SLOTNUMBER										
W	WMD															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. 16-bit write access required

Figure 1273. Slot Status Selection Register (FR_SSSR)

This register is used to access the four internal non-memory-mapped Slot Status Selection Registers FR_SSSR0 to FR_SSSR3. Each internal register selects a slot, or symbol window/NIT, whose status vector will be saved in the corresponding Slot Status Registers (FR_SSR0–FR_SSR7) according to *Table 1351*. For a detailed description of slot status monitoring, refer to *Section 57.6.18: Slot status monitoring*.

Table 1350. FR_SSSR field descriptions

Field	Description
0 WMD	**Write Mode** This control bit defines the write mode of this register. 0 Write to all fields in this register on write access 1 Write to SEL field only on write access

Table 1350. FR_SSSR field descriptions (continued)

Field	Description
2:3 SEL	**Selector** This field selects one of the four internal slot status selection registers for access. 00 Select FR_SSSR0 01 Select FR_SSSR1 10 Select FR_SSSR2 11 Select FR_SSSR3
5:15 SLOTNUMBER	**Slot Number** This field specifies the number of the slot whose status will be saved in the corresponding slot status registers. **Note:** If this value is set to 0, the related slot status register provides the status of the symbol window after the NIT start, and provides the status of the NIT after the cycle start.

Table 1351. Mapping between FR_SSSRn and FR_SSRn

Internal slot status selection register	Write the slot status of the slot selected by FR_SSSRn for each			
	Even communication cycle		Odd communication cycle	
	For channel B to	For channel A to	For channel B to	For channel A to
FR_SSSR0	FR_SSR0[0:7]	FR_SSR0[8:15]	FR_SSR1[0:7]	FR_SSR1[8:15]
FR_SSSR1	FR_SSR2[0:7]	FR_SSR2[8:15]	FR_SSR3[0:7]	FR_SSR3[8:15]
FR_SSSR2	FR_SSR4[0:7]	FR_SSR4[8:15]	FR_SSR5[0:7]	FR_SSR5[8:15]
FR_SSSR3	FR_SSR6[0:7]	FR_SSR6[8:15]	FR_SSR7[0:7]	FR_SSR7[8:15]

57.5.2.48 Slot Status Counter Condition Register (FR_SSCCR)

Offset: 0x0066

Access: Read/Write(1)
Write: Anytime

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	SEL		0	CNTCFG		MCY	VFR	SYF	NUF	SUF	STATUSMASK			
W	WMD															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. 16-bit write access required

Figure 1274. Slot Status Counter Condition Register (FR_SSCCR)

This register is used to access and program the four internal non-memory-mapped Slot Status Counter Condition Registers FR_SSCCR0 to FR_SSCCR3. Each of these four internal Slot Status Counter Condition Registers defines the mode and the conditions for incrementing the counter in the corresponding Slot Status Counter Registers (FR_SSCR0–FR_SSCR3). The correspondence is given in *Table 1353*. For a detailed description of slot status counters, refer to *Section 57.6.18.4: Slot status counter registers*.

Table 1352. FR_SSCCR field descriptions

Field	Description
0 WMD	**Write Mode** This control bit defines the write mode of this register. 0 Write to all fields in this register on write access. 1 Write to SEL field only on write access.
2:3 SEL	**Selector** This field selects one of the four internal slot counter condition registers for access. 00 select FR_SSCCR0 01 select FR_SSCCR1 10 select FR_SSCCR2 11 select FR_SSCCR3
5:6 CNTCFG	**Counter Configuration** This bit field controls the channel related incrementing of the slot status counter. 00 Increment by 1 if condition is fulfilled on channel A 01 Increment by 1 if condition is fulfilled on channel B 10 Increment by 1 if condition is fulfilled on at least one channel 11 Increment by 2 if condition is fulfilled on both channels Increment by 1 if condition is fulfilled on only one channel.
7 MCY	**Multi Cycle Selection** This bit defines whether the slot status counter accumulates over multiple communication cycles or provides information for the previous communication cycle only. 0 The Slot Status Counter provides information for the previous communication cycle only. 1 The Slot Status Counter accumulates over multiple communication cycles.
8 VFR	**Valid Frame Restriction** This bit is used to restrict the counter to received valid frames. 0 The counter is not restricted to valid frames only. 1 The counter is restricted to valid frames only.
9 SYF	**Sync Frame Restriction** This bit is used to restrict the counter to received frames with the sync frame indicator bit set to 1. 0 The counter is not restricted with respect to the sync frame indicator bit. 1 The counter is restricted to frames with the sync frame indicator bit set to 1.
10 NUF	**Null Frame Restriction** This bit is used to restrict the counter to received frames with the null frame indicator bit set to 0. 0 The counter is not restricted with respect to the null frame indicator bit. 1 The counter is restricted to frames with the null frame indicator bit set to 0.

Table 1352. FR_SSCCR field descriptions (continued)

Field	Description
11 SUF	**Startup Frame Restriction** This bit is used to restrict the counter to received frames with the startup frame indicator bit set to 1. 0 The counter is not restricted with respect to the startup frame indicator bit. 1 The counter is restricted to received frames with the startup frame indicator bit set to 1.
12:15 STATUSMASK	**Slot Status Mask** This bit field is used to enable the counter with respect to the four slot status error indicator bits. STATUSMASK[3]: This bit enables the counting for slots with the syntax error indicator bit set to 1. STATUSMASK[2]: This bit enables the counting for slots with the content error indicator bit set to 1. STATUSMASK[1]: This bit enables the counting for slots with the boundary violation indicator bit set to 1. STATUSMASK[0]: This bit enables the counting for slots with the transmission conflict indicator bit set to 1.

Table 1353. Mapping between internal FR_SSCCR*n* and FR_SSCR*n*

Condition Register	Condition Defined for Register
FR_SSCCR0	FR_SSCR0
FR_SSCCR1	FR_SSCR1
FR_SSCCR2	FR_SSCR2
FR_SSCCR3	FR_SSCR3

57.5.2.49 Slot Status Registers (FR_SSR*n*)

Offset: 0x0068 +*n**0x2 (*n* = 0 to 7) Access: Read-only

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	VFB	SYB	NFB	SUB	SEB	CEB	BVB	TCB	VFA	SYA	NFA	SUA	SEA	CEA	BVA	TCA
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1275. Slot Status Registers (FR_SSR*n*)

Each of these eight registers holds the status vector of the slot specified in the corresponding internal Slot Status Selection Register, which can be programmed using the Slot Status Selection Register (FR_SSSR). Each register is updated after the end of the corresponding slot as shown in *Figure 1385*. The register bits are directly related to the protocol variables and described in more detail in *Section 57.6.18: Slot status monitoring*.

Table 1354. FR_SSR*n* field descriptions

Field	Description
0 VFB	**Valid Frame on Channel B** Protocol-related variable: *vSS!ValidFrame* channel B. 0 *vSS!ValidFrame* = 0 1 *vSS!ValidFrame* = 1
1 SYB	**Sync Frame Indicator Channel B** Protocol-related variable: *vRF!Header!SyFIndicator* channel B. 0 *vRF!Header!SyFIndicator* = 0 1 *vRF!Header!SyFIndicator* = 1
2 NFB	**Null Frame Indicator Channel B** Protocol-related variable: *vRF!Header!NFIndicator* channel B. 0 *vRF!Header!NFIndicator* = 0 1 *vRF!Header!NFIndicator* = 1
3 SUB	**Startup Frame Indicator Channel B** Protocol-related variable: *vRF!Header!SuFIndicator* channel B. 0 *vRF!Header!SuFIndicator* = 0 1 *vRF!Header!SuFIndicator* = 1
4 SEB	**Syntax Error on Channel B** Protocol-related variable: *vSS!SyntaxError* channel B. 0 *vSS!SyntaxError* = 0 1 *vSS!SyntaxError* = 1
5 CEB	**Content Error on Channel B** Protocol-related variable: *vSS!ContentError* channel B. 0 *vSS!ContentError* = 0 1 *vSS!ContentError* = 1
6 BVB	**Boundary Violation on Channel B** Protocol-related variable: *vSS!BViolation* channel B. 0 *vSS!BViolation* = 0 1 *vSS!BViolation* = 1
7 TCB	**Transmission Conflict on Channel B** Protocol-related variable: *vSS!TxConflict* channel B. 0 *vSS!TxConflict* = 0 1 *vSS!TxConflict* = 1
8 VFA	**Valid Frame on Channel A** Protocol-related variable: *vSS!ValidFrame* channel A. 0 *vSS!ValidFrame* = 0 1 *vSS!ValidFrame* = 1
9 SYA	**Sync Frame Indicator Channel A** Protocol-related variable: *vRF!Header!SyFIndicator* channel A. 0 *vRF!Header!SyFIndicator* = 0 1 *vRF!Header!SyFIndicator* = 1
10 NFA	**Null Frame Indicator Channel A** Protocol-related variable: *vRF!Header!NFIndicator* channel A. 0 *vRF!Header!NFIndicator* = 0 1 *vRF!Header!NFIndicator* = 1

Table 1354. FR_SSR*n* field descriptions (continued)

Field	Description
11SUA	**Startup Frame Indicator Channel A** Protocol-related variable: *vRF!Header!SuFIndicator* channel A. 0 *vRF!Header!SuFIndicator* = 0 1 *vRF!Header!SuFIndicator* = 1
12 SEA	**Syntax Error on Channel A** Protocol-related variable: *vSS!SyntaxError* channel A. 0 *vSS!SyntaxError* = 0 1 *vSS!SyntaxError* = 1
13 CEA	**Content Error on Channel A** Protocol-related variable: *vSS!ContentError* channel A. 0 *vSS!ContentError* = 0 1 *vSS!ContentError* = 1
14 BVA	**Boundary Violation on Channel A** Protocol-related variable: *vSS!BViolation* channel A. 0 *vSS!BViolation* = 0 1 *vSS!BViolation* = 1
15 TCA	**Transmission Conflict on Channel A** Protocol-related variable: *vSS!TxConflict* channel A. 0 *vSS!TxConflict* = 0 1 *vSS!TxConflict* = 1

57.5.2.50 Slot Status Counter Registers (FR_SSCR*n*)

Offset: 0x0078 +*n**0x2 (*n* = 0 to 3) Access: Read-only

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	SLOTSTATUSCNT															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1276. Slot Status Counter Registers (FR_SSCR*n*)

Each of these four registers provides the slot status counter value for the previous communication cycle(s) and is updated at the cycle start. The provided value depends on the control bits and fields in the related internal Slot Status Counter Condition Register (FR_SSCCRn), which can be programmed by using the Slot Status Counter Condition Register (FR_SSCCR). For more details, refer to *Section 57.6.18.4: Slot status counter registers*.

Note: If the counter has reached its maximum value 0xFFFF and is in Multicycle mode (that is, FR_SSCCRn[MCY] = 1), the counter is not reset to 0x0000. The application can reset the counter by clearing the FR_SSCCRn[MCY] bit and waiting for the next cycle start, when the CC clears the counter. Subsequently, the counter can be put into Multicycle mode again.

Table 1355. FR_SSCR*n* field descriptions

Field	Description
0:15 SLOTSTATUSCNT	**Slot Status Counter** This field provides the current value of the Slot Status Counter.

57.5.2.51 MTS A Configuration Register (FR_MTSACFR)

Offset: 0x0080

Access: Read/Write
Write: MTE: Anytime
CYCCNTMSK,CYCCNTVAL:*POC:config*

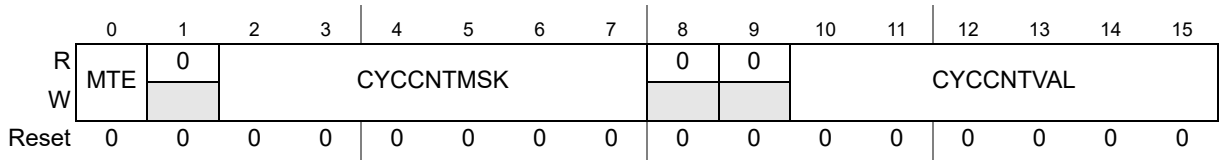


Figure 1277. MTS A Configuration Register (FR_MTSACFR)

This register controls the transmission of the Media Access Test Symbol (MTS) on channel A. For more details, refer to *Section 57.6.13: MTS generation*.

Table 1356. FR_MTSACFR field descriptions

Field	Description
0 MTE	**Media Access Test Symbol Transmission Enable** This control bit is used to enable and disable the transmission of the MTS in the selected set of cycles. 0 MTS transmission disabled 1 MTS transmission enabled
2:7 CYCCNTMSK	**Cycle Counter Mask** This field provides the filter mask for the MTS cycle count filter.
10:15 CYCCNTVAL	**Cycle Counter Value** This field provides the filter value for the MTS cycle count filter.

57.5.2.52 MTS B Configuration Register (FR_MTSBCFR)

Offset: 0x0082

Access: Read/Write
Write: MTE: Anytime
CYCCNTMSK,CYCCNTVAL:*POC:config*

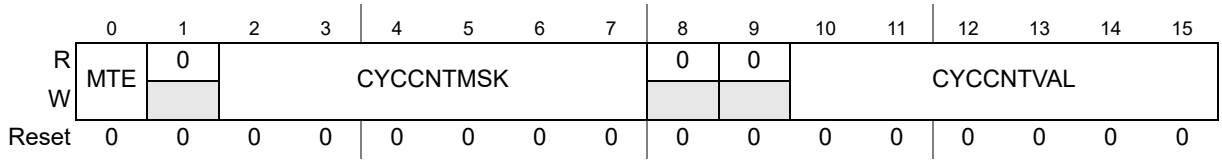


Figure 1278. MTS B Configuration Register (FR_MTSBCFR)

This register controls the transmission of the Media Access Test Symbol (MTS) on channel B. For more details, refer to *Section 57.6.13: MTS generation*.

Table 1357. FR_MTSBCFR field descriptions

Field	Description
0 MTE	**Media Access Test Symbol Transmission Enable** This control bit is used to enable and disable the transmission of the MTS in the selected set of cycles. 0 MTS transmission disabled 1 MTS transmission enabled
2:7 CYCCNTMSK	**Cycle Counter Mask** This field provides the filter mask for the MTS cycle count filter.
10:15 CYCCNTVAL	**Cycle Counter Value** This field provides the filter value for the MTS cycle count filter.

57.5.2.53 Receive Shadow Buffer Index Register (FR_RSBIR)

Access: Read/Write(1)
Write: WMD, SEL: Any Time
RSBIDX: *POC:config*

Offset: 0x0084

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	SEL		0	0	0	0	$RSBIDX_{A1}$/$RSBIDX_{A2}$/ $RSBIDX_{B1}$/$RSBIDX_{B2}$							
W	WMD															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. 16-bit write access required

Figure 1279. Receive Shadow Buffer Index Register (FR_RSBIR)

This register is used to provide and retrieve the indices of the message buffer header fields currently associated with the receive shadow buffers. For more details on the receive shadow buffer concept, refer to *Section 57.6.6.3.5: Receive shadow buffer concept*.

Table 1358. FR_RSBIR field descriptions

Field	Description
0 WMD	**Write Mode** This bit controls the write mode for this register. 0 Update SEL and RSBIDX field on register write 1 Update only SEL field on register write

Table 1358. FR_RSBIR field descriptions (continued)

Field	Description
2:3 SEL	**Selector** This field is used to select the internal receive shadow buffer index register for access. 00 FR_RSBIR_A1: receive shadow buffer index register for channel A, segment 1 01 FR_RSBIR_A2: receive shadow buffer index register for channel A, segment 2 10 FR_RSBIR_B1: receive shadow buffer index register for channel B, segment 1 11 FR_RSBIR_B2: receive shadow buffer index register for channel B, segment 2
8:15 $RSBIDX_{A1}$/ $RSBIDX_{A2}$/ $RSBIDX_{B1}$/ $RSBIDX_{B2}$	**Receive Shadow Buffer Index** This field contains the current index of the message buffer header field of the receive shadow message buffer selected by the SEL field. The CC uses this index to determine the physical location of the shadow buffer header field in the FlexRay memory area. The CC will update this field during receive operation.The application provides initial message buffer header index value in the configuration phase. - CC: Updates the message buffer header index after successful reception. Application: Provides initial message buffer header index. Legal Values are $0 \le i \le 131$. Illegal values will be detected during the message buffer search.

57.5.2.54 Receive FIFO Watermark and Selection Register (FR_RFWMSR)

Offset: 0x0086

Access: Read/Write

Write: *WM_A/WM_B: POC:config, SEL: Anytime*

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	WM_A/WM_B								0	0	0	0	0	0	0	SEL
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1280. Receive FIFO Watermark and Selection Register (FR_RFWMSR)

Table 1359. FR_RFWMSR field descriptions

Field	Description
0:7 WM_A/WM_B	**Watermark** This field defines the watermark value for the selected FIFO. This value is used to control the generation of the almost full interrupt flags.
15 SEL	**Select** This control bit selects the receiver FIFO for subsequent programming. 0 Receiver FIFO for channel A selected 1 Receiver FIFO for channel B selected

This register is used to:

- Select a receiver FIFO for subsequent programming access through the receiver FIFO configuration registers summarized in *Table 1360*.
- Define the watermark for the selected FIFO.

Table 1360. SEL-controlled receiver FIFO registers

Register
Receive FIFO Start Index Register (FR_RFSIR)
Receive FIFO Depth and Size Register (FR_RFDSR)
Receive FIFO Message ID Acceptance Filter Value Register (FR_RFMIDAFVR)
Receive FIFO Message ID Acceptance Filter Mask Register (FR_RFMIDAFMR)
Receive FIFO Frame ID Rejection Filter Value Register (FR_RFFIDRFVR)
Receive FIFO Frame ID Rejection Filter Mask Register (FR_RFFIDRFMR)
Receive FIFO Range Filter Configuration Register (FR_RFRFCFR)
Receive FIFO Range Filter Control Register (FR_RFRFCTR)

57.5.2.55 Receive FIFO Start Index Register (FR_RFSIR)

Offset: 0x0088

Access: Read/Write
Write: *POC:config*

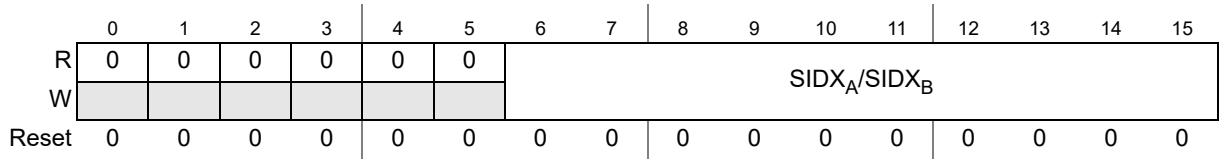


Figure 1281. Receive FIFO Start Index Register (FR_RFSIR)

This register defines the message buffer header index of the first message buffer of the selected FIFO.

Table 1361. FR_RFSIR field descriptions

Field	Description
6:15 $SIDX_A/SIDX_B$	**Start Index** This field defines the number of the message buffer header field of the selected FIFO's first message buffer. The CC uses the value of the SIDX field to determine the physical location of the receiver FIFO's first message buffer header field.

57.5.2.56 Receive FIFO Depth and Size Register (FR_RFDSR)

Offset: 0x008A

Access: Read/Write
Write: *POC:config*

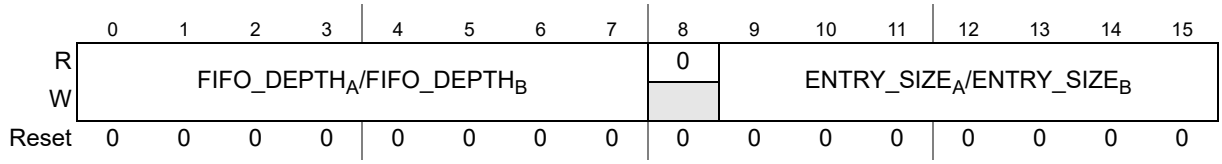


Figure 1282. Receive FIFO Depth and Size Register (FR_RFDSR)

This register defines the structure of the selected FIFO, that is the number of entries and the size of each entry.

Table 1362. RFDSR field descriptions

Field	Description
0:7 $FIFO_DEPTH_A$/ $FIFO_DEPTH_B$	**FIFO Depth** This field defines the depth (the number of entries) of the selected FIFO. **Note:** If the FIFO_DEPTH is configured to 0, FR_RFFIDRFMR[FIDRFMSK] must be configured to 0 as well to ensure that no frames are received into the FIFO.
9:15 $ENTRY_SIZE_A$/ $ENTRY_SIZE_B$	**Entry Size** This field defines the size of the frame data sections for the selected FIFO in two-byte entities.

57.5.2.57 Receive FIFO A Read Index Register (FR_RFARIR)

Offset: 0x008C Access: Read-only

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	RDIDX									
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1283. Receive FIFO A Read Index Register (FR_RFARIR)

This register provides the message buffer header index of the next available FIFO A entry that the application can read.

Table 1363. FR_RFARIR field descriptions

Field	Description
6:15 RDIDX	**Read Index** This field provides the message buffer header index of the next available FIFO message buffer that the application can read.

Note: If the FIFO is empty, the RDIDX field points to a physical message buffer with invalid content.

57.5.2.58 Receive FIFO B Read Index Register (FR_RFBRIR)

Offset: 0x008E Access: Read-only

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	RDIDX									
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1284. Receive FIFO B Read Index Register (FR_RFBRIR)

This register provides the message buffer header index of the next available FIFO B entry that the application can read.

Table 1364. FR_RFBRIR field descriptions

Field	Description
6:15 RDIDX	**Read Index** This field provides the message buffer header index of the next available FIFO message buffer that the application can read.

Note: If the FIFO is empty, the RDIDX field points to a physical message buffer with invalid content.

57.5.2.59 Receive FIFO Message ID Acceptance Filter Value Register (FR_RFMIDAFVR)

Offset: 0x0090 Access: Read/Write
Write: *POC:config*

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	$MIDAFVAL_A$/$MIDAFVAL_B$															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1285. Receive FIFO Message ID Acceptance Filter Value Register (FR_RFMIDAFVR)

This register defines the filter value for the message ID acceptance filter of the selected FIFO. For details on message ID filtering, refer to *Section 57.6.9.9: FIFO filtering*.

Table 1365. FR_RFMIDAFVR field descriptions

Field	Description
0:15 $MIDAFVAL_A$/ $MIDAFVAL_B$	**Message ID Acceptance Filter Value** Filter value for the message ID acceptance filter.

57.5.2.60 Receive FIFO Message ID Acceptance Filter Mask Register (FR_RFMIDAFMR)

Offset: 0x0092 Access: Read/Write
Write: *POC:config*

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	$MIDAFMSK_A$/$MIDAFMSK_B$															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1286. Receive FIFO Message ID Acceptance Filter Mask Register (FR_RFMIDAFMR)

This register defines the filter mask for the message ID acceptance filter of the selected FIFO. For details on message ID filtering, refer to *Section 57.6.9.9: FIFO filtering*.

Table 1366. FR_RFMIDAFMR field descriptions

Field	Description
0:15 $MIDAFMSK_A$/ $MIDAFMSK_B$	**Message ID Acceptance Filter Mask** Filter mask for the message ID acceptance filter.

57.5.2.61 Receive FIFO Frame ID Rejection Filter Value Register (FR_RFFIDRFVR)

Offset: 0x0094

Access: Read/Write
Write: *POC:config*

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	$FIDRFVAL_A$/$FIDRFVAL_B$										
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1287. Receive FIFO Frame ID Rejection Filter Value Register (FR_RFFIDRFVR)

This register defines the filter value for the frame ID rejection filter of the selected FIFO. For details on frame ID filtering, refer to *Section 57.6.9.9: FIFO filtering*.

Table 1367. FR_RFFIDRFVR field descriptions

Field	Description
5:15 $FIDRFVAL_A$/ $FIDRFVAL_B$	**Frame ID Rejection Filter Value** Filter value for the frame ID rejection filter.

57.5.2.62 Receive FIFO Frame ID Rejection Filter Mask Register (FR_RFFIDRFMR)

Offset: 0x0096

Access: Read/Write
Write: *POC:config*

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	$FIDRFMSK_A$/$FIDRFMSK_B$										
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1288. Receive FIFO Frame ID Rejection Filter Mask Register (FR_RFFIDRFMR)

This register defines the filter mask for the frame ID rejection filter of the selected FIFO. For details on frame ID filtering, refer to *Section 57.6.9.9: FIFO filtering*.

Table 1368. FR_RFFIDRFMR field descriptions

Field	Description
5:15 $FIDRFMSK_A$/ $FIDRFMSK_B$	**Frame ID Rejection Filter Mask** Filter mask for the frame ID rejection filter.

57.5.2.63 Receive FIFO Range Filter Configuration Register (FR_RFRFCFR)

Offset: 0x0098

Access: Read/Write(1)
Write: WMD, IBD, SEL: Any Time
SID: *POC:config*

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	IBD	SEL		0	SID_A/SID_B										
W	WMD															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. 16-bit write access required

Figure 1289. Receive FIFO Range Filter Configuration Register (FR_RFRFCFR)

This register provides access to the four internal frame ID range filter boundary registers of the selected FIFO. For details on frame ID range filter, refer to *Section 57.6.9.9: FIFO filtering*.

Table 1369. FR_RFRFCFR field descriptions

Field	Description
0 WMD	**Write Mode** This control bit defines the write mode of this register. 0 Write to all fields in this register on write access 1 Write to SEL and IBD field only on write access
1 IBD	**Interval Boundary** This control bit selects the interval boundary to be programmed with the SID value. 0 Program lower interval boundary 1 Program upper interval boundary
2:3 SEL	**Filter Selector** This control field selects the frame ID range filter to be accessed. 00 Select frame ID range filter 0 01 Select frame ID range filter 1 10 Select frame ID range filter 2 11 Select frame ID range filter 3
5:15 SID_A/SID_B	**Slot ID** Defines the IBD-selected frame ID boundary value for the SEL-selected range filter.

57.5.2.64 Receive FIFO Range Filter Control Register (FR_RFRFCTR)

Offset: 0x009A

Access: Read/Write
Write: Anytime

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	F3MD	F2MD	F1MD	F0MD	0	0	0	0	F3EN	F2EN	F1EN	F0EN
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1290. Receive FIFO Range Filter Control Register (FR_RFRFCTR)

This register is used to enable and disable each frame ID range filter and to define whether it is running as an acceptance or rejection filter.

Table 1370. FR_RFRFCTR field descriptions

Field	Description
4 F3MD	**Range Filter 3 Mode** This control bit defines the filter mode of the frame ID range filter 3. 0 Range filter 3 runs as acceptance filter 1 Range filter 3 runs as rejection filter
5 F2MD	**Range Filter 2 Mode** This control bit defines the filter mode of the frame ID range filter 2. 0 Range filter 2 runs as acceptance filter 1 Range filter 2 runs as rejection filter
6 F1MD	**Range Filter 1 Mode** This control bit defines the filter mode of the frame ID range filter 1. 0 Range filter 1 runs as acceptance filter 1 Range filter 1 runs as rejection filter
7 F0MD	**Range Filter 0 Mode** This control bit defines the filter mode of the frame ID range filter 0. 0 Range filter 0 runs as acceptance filter 1 Range filter 0 runs as rejection filter
12 F3EN	**Range Filter 3 Enable** This control bit is used to enable and disable the frame ID range filter 3. 0 Range filter 3 disabled 1 Range filter 3 enabled
13 F2EN	**Range Filter 2 Enable** This control bit is used to enable and disable the frame ID range filter 2. 0 Range filter 2 disabled 1 Range filter 2 enabled
14 F1EN	**Range Filter 1 Enable** This control bit is used to enable and disable the frame ID range filter 1. 0 Range filter 1 disabled 1 Range filter 1 enabled
15 F0EN	**Range Filter 0 Enable** This control bit is used to enable and disable the frame ID range filter 0. 0 Range filter 0 disabled 1 Range filter 0 enabled

57.5.2.65 Last Dynamic Transmit Slot Channel A Register (FR_LDTXSLAR)

Offset: 0x009C Access: Read-only

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	LDYNTXSLOTA										
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1291. Last Dynamic Transmit Slot Channel A Register (FR_LDTXSLAR)

This register provides the number of the last transmission slot in the dynamic segment for channel A. This register is updated after the end of the dynamic segment and before the start of the next communication cycle.

Table 1371. FR_LDTXSLAR field descriptions

Field	Description
5:15 LDYNTXSLOTA	**Last Dynamic Transmission Slot Channel A** Protocol-related variable: *zLastDynTxSlot* channel A. Number of the last transmission slot in the dynamic segment for channel A. If no frame was transmitted during the dynamic segment on channel A, the value of this field is set to 0.

57.5.2.66 Last Dynamic Transmit Slot Channel B Register (FR_LDTXSLBR)

Offset: 0x009E Access: Read-only

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	LDYNTXSLOTB										
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1292. Last Dynamic Transmit Slot Channel B Register (FR_LDTXSLBR)

This register provides the number of the last transmission slot in the dynamic segment for channel B. This register is updated after the end of the dynamic segment and before the start of the next communication cycle.

Table 1372. FR_LDTXSLBR field descriptions

Field	Description
5:15 LDYNTXSLOTB	**Last Dynamic Transmission Slot Channel B** Protocol-related variable: *zLastDynTxSlot* channel B. Number of the last transmission slot in the dynamic segment for channel B. If no frame was transmitted during the dynamic segment on channel B the value of this field is set to 0.

57.5.2.67 Protocol configuration registers

The following configuration registers provide the necessary configuration information to the protocol engine. The individual values in the registers are described in *Table 1373*. For more

details about the FlexRay related configuration parameters and the allowed parameter ranges, refer to *FlexRay Communications System Protocol Specification, Version 2.1 Rev A*.

Table 1373. Protocol configuration register fields

Name	Description(1)	Min	Max	Unit	FR_PCR
coldstart_attempts	gColdstartAttempts	—	—	number	3
action_point_offset	gdActionPointOffset - 1	—	—	MT	0
cas_rx_low_max	gdCASRxLowMax - 1	—	—	gdBit	4
dynamic_slot_idle_phase	gdDynamicSlotIdlePhase	—	—	minislot	28
minislot_action_point_offset	gdMinislotActionPointOffset - 1	—	—	MT	3
minislot_after_action_point	gdMinislot - gdMinislotActionPointOffset - 1	—	—	MT	2
static_slot_length	gdStaticSlot	—	—	MT	0
static_slot_after_action_point	gdStaticSlot - gdActionPointOffset - 1	—	—	MT	13
symbol_window_exists	gdSymbolWindow!=0	0	1	bool	9
symbol_window_after_action_point	gdSymbolWindow - gdActionPointOffset - 1	—	—	MT	6
tss_transmitter	gdTSSTransmitter	—	—	gdBit	5
wakeup_symbol_rx_idle	gdWakeupSymbolRxIdle	—	—	gdBit	5
wakeup_symbol_rx_low	gdWakeupSymbolRxLow	—	—	gdBit	3
wakeup_symbol_rx_window	gdWakeupSymbolRxWindow	—	—	gdBit	4
wakeup_symbol_tx_idle	gdWakeupSymbolTxIdle	—	—	gdBit	8
wakeup_symbol_tx_low	gdWakeupSymbolTxLow	—	—	gdBit	5
noise_listen_timeout	(gListenNoise * pdListenTimeout) - 1	—	—	µT	16/17
macro_initial_offset_a	pMacroInitialOffset[A]	—	—	MT	6
macro_initial_offset_b	pMacroInitialOffset[B]	—	—	MT	16
macro_per_cycle	gMacroPerCycle	—	—	MT	10
macro_after_first_static_slot	gMacroPerCycle - gdStaticSlot	—	—	MT	1
macro_after_offset_correction	gMacroPerCycle - gOffsetCorrectionStart	—	—	MT	28
max_without_clock_correction_fatal	gMaxWithoutClockCorrectionFatal	—	—	cyclepairs	8
max_without_clock_correction_passive	gMaxWithoutClockCorrectionPassive	—	—	cyclepairs	8
minislot_exists	gNumberOfMinislots!=0	0	1	bool	9
minislots_max	gNumberOfMinislots - 1	—	—	minislot	29
number_of_static_slots	gNumberOfStaticSlots	—	—	static slot	2
offset_correction_start	gOffsetCorrectionStart	—	—	MT	11
payload_length_static	gPayloadLengthStatic	—	—	2 bytes	19

Table 1373. Protocol configuration register fields (continued)

Name	Description[1]	Min	Max	Unit	FR_PCR
max_payload_length_dynamic	pPayloadLengthDynMax	—	—	2 bytes	24
first_minislot_action_point_offset	max(gdActionPointOffset, gdMinislotActionPointOffset) - 1	—	—	MT	13
allow_halt_due_to_clock	pAllowHaltDueToClock	—	—	bool	26
allow_passive_to_active	pAllowPassiveToActive	—	—	cyclepairs	12
cluster_drift_damping	pClusterDriftDamping	—	—	µT	24
comp_accepted_startup_range_a	pdAcceptedStartupRange - pDelayCompensation[A]	—	—	µT	22
comp_accepted_startup_range_b	pdAcceptedStartupRange - pDelayCompensation[B]	—	—	µT	26
listen_timeout	pdListenTimeout - 1	—	—	µT	14/15
key_slot_id	pKeySlotId	—	—	number	18
key_slot_used_for_startup	pKeySlotUsedForStartup	—	—	bool	11
key_slot_used_for_sync	pKeySlotUsedForSync	—	—	bool	11
latest_tx	gNumberOfMinislots - pLatestTx	—	—	minislot	21
sync_node_max	gSyncNodeMax	—	—	number	30
micro_initial_offset_a	pMicroInitialOffset[A]	—	—	µT	20
micro_initial_offset_b	pMicroInitialOffset[B]	—	—	µT	20
micro_per_cycle	pMicroPerCycle	—	—	µT	22/23
micro_per_cycle_min	pMicroPerCycle - pdMaxDrift	—	—	µT	24/25
micro_per_cycle_max	pMicroPerCycle + pdMaxDrift	—	—	µT	26/27
micro_per_macro_nom_half	round(pMicroPerMacroNom / 2)	—	—	µT	7
offset_correction_out	pOffsetCorrectionOut	—	—	µT	9
rate_correction_out	pRateCorrectionOut	—	—	µT	14
single_slot_enabled	pSingleSlotEnabled	—	—	bool	10
wakeup_channel	pWakeupChannel	Refer to *Table 1374*			10
wakeup_pattern	pWakeupPattern	—	—	number	18
decoding_correction_a	pDecodingCorrection + pDelayCompensation[A] + 2	—	—	µT	19
decoding_correction_b	pDecodingCorrection + pDelayCompensation[B] + 2	—	—	µT	7
key_slot_header_crc	header CRC for key slot	0x000	0x7FF	number	12
extern_offset_correction	pExternOffsetCorrection	—	—	µT	29
extern_rate_correction	pExternRateCorrection	—	—	µT	21

1. Refer to *FlexRay Communications System Protocol Specification, Version 2.1 Rev A* for detailed protocol parameter definitions.

Table 1374. Wake up channel selection

wakeup_channel	Wake up Channel
0	A
1	B

57.5.2.67.1 Protocol Configuration Register 0 (FR_PCR0)

Offset: 0x00A0 Access: Read/Write
Write: *POC:config*

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R/W	action_point_offset								static_slot_length							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1293. Protocol Configuration Register 0 (FR_PCR0)

57.5.2.67.2 Protocol Configuration Register 1 (FR_PCR1)

Offset: 0x00A2 Access: Read/Write
Write: *POC:config*

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	macro_after_first_static_slot													
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1294. Protocol Configuration Register 1 (FR_PCR1)

57.5.2.67.3 Protocol Configuration Register 2 (FR_PCR2)

Offset: 0x00A4 Access: Read/Write
Write: *POC:config*

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R/W	minislot_after_action_point								number_of_static_slots							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1295. Protocol Configuration Register 2 (FR_PCR2)

57.5.2.67.4 Protocol Configuration Register 3 (FR_PCR3)

Offset: 0x00A6 Access: Read/Write
Write: *POC:config*

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R/W	wakeup_symbol_rx_low						minislot_action_point_offset[4:0]						coldstart_attempts			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1296. Protocol Configuration Register 3 (FR_PCR3)

57.5.2.67.5 Protocol Configuration Register 4 (FR_PCR4)

Offset: 0x00A8

Access: Read/Write
Write: *POC:config*

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	cas_rx_low_max								wakeup_symbol_rx_window							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1297. Protocol Configuration Register 4 (FR_PCR4)

57.5.2.67.6 Protocol Configuration Register 5 (FR_PCR5)

Offset: 0x00AA

Access: Read/Write
Write: *POC:config*

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	tss_transmitter				wakeup_symbol_tx_low								wakeup_symbol_rx_idle			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1298. Protocol Configuration Register 5 (FR_PCR5)

57.5.2.67.7 Protocol Configuration Register 6 (FR_PCR6)

Offset: 0x00AC

Access: Read/Write
Write: *POC:config*

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	symbol_window_after_action_point								macro_initial_offset_a						
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1299. Protocol Configuration Register 6 (FR_PCR6)

57.5.2.67.8 Protocol Configuration Register 7 (FR_PCR7)

Offset: 0x00AE

Access: Read/Write
Write: *POC:config*

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	decoding_correction_b								micro_per_macro_nom_half							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1300. Protocol Configuration Register 7 (FR_PCR7)

57.5.2.67.9 Protocol Configuration Register 8 (FR_PCR8)

Offset: 0x00B0

Access: Read/Write
Write: *POC:config*

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	max_without_clock_correction_fatal				max_without_clock_correction_passive				wakeup_symbol_tx_idle							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1301. Protocol Configuration Register 8 (FR_PCR8)

57.5.2.67.10 Protocol Configuration Register 9 (FR_PCR9)

Offset: 0x00B2

Access: Read/Write
Write: *POC:config*

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	minislot_exists	symbol_window_exists	offset_correction_out													
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1302. Protocol Configuration Register 9 (FR_PCR9)

57.5.2.67.11 Protocol Configuration Register 10 (FR_PCR10)

Offset: 0x00B4

Access: Read/Write
Write: *POC:config*

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	single_slot_enabled	wakeup_channel	macro_per_cycle													
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1303. Protocol Configuration Register 10 (FR_PCR10)

57.5.2.67.12 Protocol Configuration Register 11 (FR_PCR11)

Offset: 0x00B6

Access: Read/Write
Write: *POC:config*

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	key_slot_used_for_startup	key_slot_used_for_sync	offset_correction_start													
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1304. Protocol Configuration Register 11 (FR_PCR11)

57.5.2.67.13 Protocol Configuration Register 12 (FR_PCR12)

Offset: 0x00B8

Access: Read/Write
Write: *POC:config*

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	allow_passive_to_active				key_slot_header_crc											
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1305. Protocol Configuration Register 12 (FR_PCR12)

57.5.2.67.14 Protocol Configuration Register 13 (FR_PCR13)

Offset: 0x00BA

Access: Read/Write
Write: *POC:config*

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	first_minislot_action_point_offset								static_slot_after_action_point							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1306. Protocol Configuration Register 13 (FR_PCR13)

57.5.2.67.15 Protocol Configuration Register 14 (FR_PCR14)

Offset: 0x00BC

Access: Read/Write
Write: *POC:config*

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	rate_correction_out												listen_timeout[20:16]			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1307. Protocol Configuration Register 14 (FR_PCR14)

57.5.2.67.16 Protocol Configuration Register 15 (FR_PCR15)

Offset: 0x00BE

Access: Read/Write
Write: *POC:config*

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	listen_timeout[15:0]															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1308. Protocol Configuration Register 15 (FR_PCR15)

57.5.2.67.17 Protocol Configuration Register 16 (FR_PCR16)

Offset: 0x00C0

Access: Read/Write
Write: *POC:config*

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	macro_initial_offset_b								noise_listen_timeout[24:16]							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1309. Protocol Configuration Register 16 (FR_PCR16)

57.5.2.67.18 Protocol Configuration Register 17 (FR_PCR17)

Offset: 0x00C2

Access: Read/Write
Write: *POC:config*

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	noise_listen_timeout[15:0]															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1310. Protocol Configuration Register 17 (FR_PCR17)

57.5.2.67.19 Protocol Configuration Register 18 (FR_PCR18)

Offset: 0x00C4

Access: Read/Write
Write: *POC:config*

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	wakeup_pattern								key_slot_id							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1311. Protocol Configuration Register 18 (FR_PCR18)

57.5.2.67.20 Protocol Configuration Register 19 (FR_PCR19)

Offset: 0x00C6

Access: Read/Write
Write: *POC:config*

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	decoding_correction_a								payload_length_static							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1312. Protocol Configuration Register 19 (FR_PCR19)

57.5.2.67.21 Protocol Configuration Register 20 (FR_PCR20)

Offset: 0x00C8

Access: Read/Write
Write: *POC:config*

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	micro_initial_offset_b								micro_initial_offset_a							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1313. Protocol Configuration Register 20 (FR_PCR20)

57.5.2.67.22 Protocol Configuration Register 21 (FR_PCR21)

Offset: 0x00CA

Access: Read/Write
Write: *POC:config*

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	extern_rate_ correction				latest_tx											
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1314. Protocol Configuration Register 21 (FR_PCR21)

57.5.2.67.23 Protocol Configuration Register 22 (FR_PCR22)

Offset: 0x00CC

Access: Read/Write
Write: *POC:config*

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	Reserved(1)				comp_accepted_startup_range_a								micro_per_cycle[19:16]			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1315. Protocol Configuration Register 22 (FR_PCR22)

1. Reserved bit or field. Application must not write any value different from the reset value.

57.5.2.67.24 Protocol Configuration Register 23 (FR_PCR23)

Offset: 0x00CE

Access: Read/Write
Write: *POC:config*

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	micro_per_cycle[15:0]															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1316. Protocol Configuration Register 23 (FR_PCR23)

57.5.2.67.25 Protocol Configuration Register 24 (FR_PCR24)

Offset: 0x00D0

Access: Read/Write
Write: *POC:config*

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	cluster_drift_damping				max_payload_length_dynamic								micro_per_cycle_min [19:16]			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1317. Protocol Configuration Register 24 (FR_PCR24)

57.5.2.67.26 Protocol Configuration Register 25 (FR_PCR25)

Offset: 0x00D2

Access: Read/Write
Write: *POC:config*

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	micro_per_cycle_min[15:0]															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1318. Protocol Configuration Register 25 (FR_PCR25)

57.5.2.67.27 Protocol Configuration Register 26 (FR_PCR26)

Offset: 0x00D4

Access: Read/Write
Write: *POC:config*

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	allow_halt_ due_to_clock	comp_accepted_startup_range_b											micro_per_cycle_max [19:16]			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1319. Protocol Configuration Register 26 (FR_PCR26)

57.5.2.67.28 Protocol Configuration Register 27 (FR_PCR27)

Offset: 0x00D6

Access: Read/Write
Write: *POC:config*

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	micro_per_cycle_max[15:0]															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1320. Protocol Configuration Register 27 (FR_PCR27)

57.5.2.67.29 Protocol Configuration Register 28 (FR_PCR28)

Offset: 0x00D8

Access: Read/Write
Write: *POC:config*

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	dynamic_slot_idle_phase				macro_after_offset_correction											
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1321. Protocol Configuration Register 28 (FR_PCR28)

57.5.2.67.30 Protocol Configuration Register 29 (FR_PCR29)

Offset: 0x00DA

Access: Read/Write
Write: *POC:config*

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	extern_offset_correction				minislots_max											
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1322. Protocol Configuration Register 29 (FR_PCR29)

57.5.2.67.31 Protocol Configuration Register 30 (FR_PCR30)

Offset: 0x00DC

Access: Read/Write
Write: *POC:config*

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	sync_node_max			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1323. Protocol Configuration Register 30 (FR_PCR30)

57.5.2.68 Stop Watch Count Register (FR_STPWR)

Offset: 0x00DE

Access: Read-only
Read Reset Only: Anytime

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	STPW															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1324. Stop Watch Count High Register (FR_STPWHR)

Offset: 0x00E0

Access: Read-only
Read Reset Only: Anytime

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	STPW															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1325. Stop Watch Count Low Register (FR_STPWLR)

Table 1375. FR_STPWR field descriptions

Field	Description
0:15 STPW	Stop Watch Count Register This is the value of the stop watch counter running on system clock. This starts on cycle start even and stops on an external event “OS_tick”. For details, refer to *Section 57.6.26*. This is a read reset type register, which means that when all the 32 bits are read, it automatically clears.

57.5.2.69 Protocol Event Output Enable and StopWatch Control Register (FR_PEOER)

Offset: 0x00E2

Access: Read/Write
Write: Anytime

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	STPW_EN	0	0	0	0	0	TIM2_EE	TIM1_EE	CYS_EE
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1326. Protocol Event Output Enable and StopWatch Control Register (FR_PEOER)

This register defines whether or not the event output ports are enabled based on the events such as Stop watch count Cycle start and Timer1 and Timer2 expiry (refer to *Section 57.6.17: Timer support* for details).

Table 1376. FR_PEOER field descriptions

Field	Description
7 STPW_EN	Stop watch count Enable This bit controls the stopwatch counter. 0 Stopwatch counter disabled 1 Stopwatch counter enabled
13 TIM2_EE	Timer 2 expired Event Output Enable This bit controls the event on port “fr_evt_tim2”. 0 Timer 2 expired event out disabled 1 Timer 2 expired event out enabled

Table 1376. FR_PEOER field descriptions

Field	Description
14 TIM1_EE	Timer 1 expired Event Output Enable This bit controls the event on port “fr_evt_tim1”. 0 Timer 1 expired event out disabled 1 Timer 1 expired event out enabled
15 CYS_EE	Cycle Start Event Output Enable This bit controls the event on port “fr_evt_cyc”. 0 Cycle start event out disabled 1 Cycle start event out enabled

57.5.2.70 Receive FIFO Start Data Offset Register (FR_RFSDOR)

Offset: 0x00E6

Access: Read/Write
Write: *POC:config*

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	SDO_A/SDO_B															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1327. Receive FIFO Start Data Offset Register (FR_RFSDOR)

Table 1377. FR_RFSDOR field descriptions

Field	Description
0:15 SDO_A/SDO_B	Start Data Field Offset This field defines the data field offset of the header field of the first message buffer of the selected FIFO. The CC uses the value of the SDO field to determine the physical location of the receiver FIFO’s first message buffer header field. For configuration constraints, refer to *Section 57.7.1.2: Configure data field offsets*.

Note: Since all data fields of the FIFO are of equal length and are located at subsequent system memory addresses, the content of the FR_RFSDOR register corresponds to the start address of the payload area of the selected FIFO.

57.5.2.71 Receive FIFO System Memory Base Address Register (FR_RFSYMBADR)

Offset: 0x00E8

Access: Read/Write
Write: Disabled mode

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	SMBA															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1328. Receive FIFO System Memory Base Address High Register (FR_RFSYMBADHR)

Offset: 0x00EA

Access: Read/Write
Write: Disabled mode

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	SMBA												0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1329. Receive FIFO System Memory Base Address Low Register (FR_RFSYMBADLR)

These registers define the system memory base address for the receive FIFO if the FIFO address mode bit FR_MCR[FAM] is set to 1. The system memory base address is used by the BMIF to calculate the physical memory address for system memory accesses for the FIFOs.

Table 1378. FR_RFSYMBADR field descriptions

Field	Description
SMBA	System Memory Base Address This is the value of the system memory base address for the receive FIFO if the FIFO address mode bit FR_MCR[FAM] is set to 1. It is defined as a byte address.

57.5.2.72 Receive FIFO Periodic Timer Register (FR_RFPTR)

Offset: 0x00EC

Access: Read/Write
Write: *POC:config*

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	PTD													
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1330. Receive FIFO Periodic Timer Register (FR_RFPTR)

This register holds the periodic timer duration for the periodic FIFO timer. The periodic timer applies to both FIFOs (refer to *Section 57.6.9.3: FIFO periodic timer*).

Table 1379. FR_RFPTR field descriptions

Field	Description
2:15 PTD	Periodic Timer Duration This value defines the periodic timer duration in terms of macroticks. 0000 Timer stays expired 3FFF Timer never expires other Timer expires after specified number of macroticks, expires and is restarted at each cycle start

57.5.2.73 Receive FIFO Fill Level and POP Count Register (FR_RFFLPCR)

Offset: 0x00EE Access: Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	FLB_PCB								FLA_PCA							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1331. Receive FIFO Fill Level and POP Count Register (FR_RFFLPCR)

This register provides the current fill level of the two receiver FIFOs and is used to pop a number of entries from the FIFOs.

Table 1380. FR_RFFLPCR field descriptions

Field	Description
0:7 FLB_PCB	Fill Level FIFO B (FLB) This field provides the current number of entries in the FIFO B. Pop Count FIFO B (PCB) This field defines the number of entries to be removed from FIFO B.
8:15 FLA_PCA	Fill Level FIFO A (FLA) This field provides the current number of entries in the FIFO A. Pop Count FIFO A (PCA) This field defines the number of entries to be removed from FIFO A.

Note: If the pop count value PCA/PCB is greater than the current FIFO fill level FLB/FLA, then the FIFO is empty after the update. No notification is given that not the required number of entries was removed.

57.5.2.74 ECC Error Interrupt Flag and Enable Register (FR_EEIFER)

Offset: 0x00F0 Access: Read/Write
Write: Normal Mode

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	LRNE_OF	LRCE_OF	DRNE_OF	DRCE_OF	LRNE_IF	LRCE_IF	DRNE_IF	DRCE_IF	0	0	0	0	LRNE_IE	LRCE_IE	DRNE_IE	DRCE_IE
W	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c								
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1332. ECC Error Interrupt Flag and Enable Register (FR_EEIFER)

This register provides the means to control the ECC-related interrupt request lines and provides the corresponding interrupt flags. The interrupt flags are cleared by writing 1, which resets the corresponding report registers. For a detailed description, refer to *Section 57.6.24.2*.

Table 1381. FR_EEIFER field descriptions

Field	Description
Error overflow flags	
0 LRNE_OF	LRAM Non-Corrected Error Overflow Flag This flag is set to 1 when at least one of the following events occurs: a) Memory errors are detected *but not corrected* on CHI LRAM and interrupt flag LRNE_IF is already 1. b) Memory errors are detected *but not corrected* on at least two banks of CHI LRAM. 0 No such event 1 Non-Corrected Error overflow detected on CHI LRAM
1 LRCE_OF	LRAM Corrected Error Overflow Flag This flag is set to 1 when at least one of the following events occurs: a) Memory errors are detected *and corrected* on CHI LRAM and interrupt flag LRCE_IF is already 1. b) Memory errors are detected *and corrected* on at least two banks of CHI LRAM. 0 No such event 1 Corrected Error overflow detected on CHI LRAM **Note:** Error Correction is not implemented on CHI LRAM, so the flag will never be asserted.
2 DRNE_OF	DRAM Non-Corrected Error Overflow Flag This flag is set to 1 when at least one of the following events appears: a) Memory errors are detected *but not corrected* on PE DRAM and interrupt flag DRNE_IF is already 1. b) Memory errors are detected *but not corrected* on at least two banks of the PE DRAM. 0 No such event 1 Non-Corrected Error overflow detected on PE DRAM
3 DRCE_OF	DRAM Corrected Error Overflow Flag This flag is set to 1 when at least one of the following events appears: a) Memory errors are detected *and corrected* on PE DRAM and interrupt flag DRCE_IF is already 1. b) Memory errors are detected *and corrected* on at least two banks of PE DRAM. 0 No such event 1 Corrected Error overflow detected on PE DRAM

Table 1381. FR_EEIFER field descriptions (continued)

Field	Description
Error interrupt flags	
4 LRNE_IF	LRAM Non-Corrected Error Interrupt Flag This interrupt flag is set to 1 when a memory error is *detected but not corrected* on the CHI LRAM. 0 No such event 1 Non-Corrected Error detected on CHI LRAM
5 LRCE_IF	LRAM Corrected Error Interrupt Flag This interrupt flag is set to 1 when a memory error is *detected and corrected* on the CHI LRAM. 0 No such event 1 Corrected Error detected on CHI LRAM **Note:** Error Correction is not implemented on CHI LRAM, so the flag will never be asserted.
6 DRNE_IF	DRAM Non-Corrected Error Interrupt Flag This interrupt flag is set to 1 when a memory error is *detected but not corrected* on PE DRAM. 0 No such event 1 Non-Corrected Error detected on PE DRAM
7 DRCE_IF	DRAM Corrected Error Interrupt Flag This interrupt flag is set to 1 when a memory error is *detected and corrected* on PE DRAM. 0 No such event 1 Corrected Error detected on PE DRAM
Error interrupt enables	
12 LRNE_IE	**LRAM Non-Corrected Error Interrupt Enable** This flag controls if the LRAM Non-Corrected Error Interrupt line is asserted when the LRNE_IF flag is set. 0 Disable interrupt line 1 Enable interrupt line
13 LRCE_IE	**LRAM Corrected Error Interrupt Enable** This flag controls if the LRAM Corrected Error Interrupt line is asserted when the LRCE_IF flag is set. 0 Disable interrupt line 1 Enable interrupt line
14 DRNE_IE	**DRAM Non-Corrected Error Interrupt Enable** This flag controls if the DRAM Non-Corrected Error Interrupt line is asserted when the DRNE_IF flag is set. 0 Disable interrupt line 1 Enable interrupt line
15 DRCE_IE	**DRAM Corrected Error Interrupt Enable** This flag controls if the DRAM Corrected Error Interrupt line is asserted when the DRCE_IF flag is set. 0 Disable interrupt line 1 Enable interrupt line

57.5.2.75 ECC Error Report and Injection Control Register (FR_EERICR)

Offset: 0x00F2

Access: Read/Write
Write: ERS: Anytime
ERM, EIM, EIE: IDL

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	BSY	0	0	0	0	0	ERS		0	0	0	ERM	0	0	EIM	EIE
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1333. ECC Error Report and Injection Control Register (FR_EERICR)

This register configures the error injection and error reporting and provides the selector for the content of the report registers.

Table 1382. FR_EERICR field descriptions

Field	Description
0 BSY	Register Update Busy This field indicates the current state of the ECC configuration update and controls the register write access condition IDL specified in *Section 57.5.2.2*. 0 ECC configuration is idle 1 ECC configuration is running
6:7 ERS	Error Report Select This field selects the content of the ECC Error reporting registers. 00 Show PE DRAM non-corrected error information 01 Show PE DRAM corrected error information 10 Show CHI LRAM non-corrected error information 11 Show CHI LRAM corrected error information
11 ERM	Error Report Mode This bit configures the type of data written into the internal error report registers on the detection of a memory error. 0 Store data and code as delivered by ECC decoding logic 1 Store data and code as read from the memory
14 EIM	Error Injection Mode This bit configures the ECC error injection mode. 0 Use FR_EEIDR[DATA] and FR_EEICR[CODE] as XOR distortion pattern for error injection 1 Use FR_EEIDR[DATA] and FR_EEICR[CODE] as write value for error injection
15 EIE	Error Injection Enable This bit configures the ECC error injection on the memories. 0 Error injection disabled 1 Error injection enabled

57.5.2.76 ECC Error Report Address Register (FR_EERAR)

Offset: 0x00F4 Access: Read-only

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	MID	BANK			ADDR											
W																
Reset	0	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1334. ECC Error Report Address Register (FR_EERAR)

This register provides the memory identifier, bank, and address for which the memory error is reported.

Table 1383. FR_EERAR field descriptions

Field	Description
0 MID	Memory Identifier This flag provides the memory instance for which the memory error is reported. 0 PE DRAM 1 CHI LRAM
1:3 BANK	Memory Bank This field provides the BANK for which the memory error is reported. 111 Reset value, indicates no error found after reset. For MID = 0: 000 PE DRAM [7:0] 001 PE DRAM [15:8] Others are not used. For MID = 1: Refer to *Table 1384* for the assignment of the LRAM banks.
4:15 ADDR	Memory Address This field provides the address of the failing memory location.

Table 1384. LRAM bank value for MID = 1

BANK	Register		
000	FR_MBCCFR(2n)	FR_MBDOR(6n)	FR_LEETR0
001	FR_MBFIDR(2n)	FR_MBDOR(6n + 1)	FR_LEETR1
010	FR_MBIDXR(2n)	FR_MBDOR(6n + 2)	FR_LEETR2
011	FR_MBCCFR(2n+1)	FR_MBDOR(6n + 3)	FR_LEETR3
100	FR_MBFIDR(2n+1)	FR_MBDOR(6n + 4)	FR_LEETR4
101	FR_MBIDXR(2n+1)	FR_MBDOR(6n + 5)	FR_LEETR5
110	Not Used	Not Used	Not Used
111			

57.5.2.77 ECC Error Report Data Register (FR_EERDR)

Offset: 0x00F6 Access: Read-only

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	DATA															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1335. ECC Error Report Data Register (FR_EERDR)

This register provides the data related information of the reported memory read access. The assignment of the bits depends on the selected memory and memory bank as shown in *Table 1386*.

Table 1385. FR_EERDR field descriptions

Field	Description
0:15 DATA	Data The content of this field depends on the report mode selected by FR_EERICR[ERM]. ERM = 0: ECC Data, shows data as generated by the ECC decoding logic ERM = 1: Memory Data, shows data as read from the memory

Table 1386. Valid bits in FR_EERDR[DATA] / FR_EEIDR[DATA] field

MEM	BANK	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PE DRAM	0									PE DRAM[7:0]							
PE DRAM	1									PE DRAM[15:8]							
CHI LRAM	0	FR_MBCCFR(2n)															
CHI LRAM	0	FR_MBDOR(6n)															
CHI LRAM	0	FR_LEETR0															
CHI LRAM	1						FR_MBFIDR(2n)[FID]										
CHI LRAM	1	FR_MBDOR(6n+1)															
CHI LRAM	1	FR_LEETR1															
CHI LRAM	2									FR_MBIDXR(2n)[MBIDX]							
CHI LRAM	2	FR_MBDOR(6n+2)															
CHI LRAM	2	FR_LEETR2															
CHI LRAM	3	FR_MBCCFR(2n+1)															
CHI LRAM	3	FR_MBDOR(6n+3)															
CHI LRAM	3	FR_LEETR3															
CHI LRAM	4						FR_MBFIDR(2n+1)[FID]										
CHI LRAM	4	FR_MBDOR(6n+4)															
CHI LRAM	4	FR_LEETR4															
CHI LRAM	5									FR_MBIDXR(2n+1)[MBIDX]							

Table 1386. Valid bits in FR_EERDR[DATA] / FR_EEIDR[DATA] field (continued)

MEM	BANK	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CHI LRAM	5	FR_MBDOR(6n+5)															
CHI LRAM	5	FR_LEETR5															

57.5.2.78 ECC Error Report Code Register (FR_EERCR)

Offset: 0x00F8 Access: Read-only

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	CODE				
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1336. ECC Error Report Code Register (FR_EERCR)

This register provides the ECC-related information of the reported memory read access.

Table 1387. FR_EERSR field descriptions

Field	Description
11:15 CODE	Code The content of this field depends on the report mode selected by FR_EERICR[ERM]. ERM = 0: Syndrome. Shows the ECC syndrome generated by the ECC decoding logic. The coding of the PE DRAM syndrome is shown in *Note:*. The coding of the CHI LRAM syndrome is shown in *Section 57.6.24.2.4*. ERM = 1: Checkbits. Shows the ECC checkbits read from the memory.

57.5.2.79 ECC Error Injection Address Register (FR_EEIAR)

Offset: 0x00FA Access: Read/Write
Write: IDL

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	MID		BANK		ADDR											
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1337. ECC Error Injection Address Register (FR_EEIAR)

This register defines the memory module, bank, and address where the ECC error has to be injected.

Table 1388. FR_EEIAR field descriptions

Field	Description
0 MID	Memory Identifier This flag defines the memory instance for ECC error injection. 0 PE DRAM 1 CHI LRAM
1:3 BANK	Memory Bank This field defines the memory bank for ECC error injection. For MID = 0: 000 BANK0: PE DRAM [7:0] 001 BANK1: PE DRAM [15:8] Others reserved For MID = 1: Refer to *Table 1384* for the assignment of the LRAM banks.
4:15 ADDR	Memory Address This flag defines the memory address for ECC error injection.

57.5.2.80 ECC Error Injection Data Register (FR_EEIDR)

Offset: 0x00FC

Access: Read/Write
Write: IDL

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	DATA															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1338. ECC Error Injection Data Register (FR_EEIDR)

This register defines the data distortion pattern for the error injection write. The number of valid bits depends on the selected memory and memory bank as shown in *Table 1386*.

Table 1389. FR_EEIDR field descriptions

Field	Description
0:15 DATA	Data The content of this field depends on the error injection mode selected by FR_EERICR[EIM]. EIM = 0: This field defines the XOR distortion pattern for the data written into the memory. EIM = 1: This field defines the data to be written into the memory.

Note: The effect of the error injected depends on the LRAM content at the address accessed and on the module internal usage of the data. Refer to Section 57.6.24.3: Memory error response for details.

57.5.2.81 ECC Error Injection Code Register (FR_EEICR)

Offset: 0x00FE

Access: Read/Write
Write: IDL

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	CODE				
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1339. ECC Error Injection Code Register (FR_EEICR)

This register defines the ECC code distortion pattern for the error injection write.

Table 1390. FR_EEICR field descriptions

Field	Description
11:15 CODE	Code The content of this field depends on the error injection mode selected by FR_EERICR[EIM]. EIM = 0: This field defines the XOR distortion pattern for the ECC checkbits written into the memory. EIM = 1: This field defines the ECC checkbits written into the memory.

57.5.2.82 Message Buffer Configuration Control and Status Register *n* (FR_MBCCSR*n*)

Offset: 0x0800 + *n**0x8 (*n* = 0 to 127)

Access: Read/Write(1)
Write: MTD: *POC:config* or MB_DIS
CMT: MB_LCK or MB_DIS
EDT, LCKT, MBIE, MBIF: Normal mode

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	MTD	CMT	0	0	MBIE	0	0	0	DUP	DVAL	EDS	LCKS	MBIF
W					rwm	EDT	LCKT									w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. Additional Reset: CMT, DUP, DVAL, MBIF: Message Buffer Disable

Figure 1340. Message Buffer Configuration, Control and Status Register *n* (FR_MBCCSR*n*)

The content of these registers comprises message buffer configuration data, message buffer control data, message buffer status information, and message buffer interrupt flags. A detailed description of all flags can be found in *Section 57.6.6: Individual message buffer functional description*.

If the application writes 1 to the EDT bit, no write access to the other register bits is performed.

If the application writes 0 to the EDT bit and 1 to the LCKT bit, no write access to the other bits is performed.

Table 1391. FR_MBCCSR*n* field descriptions

Field	Description
Message buffer configuration	
3 MTD	Message Buffer Transfer Direction This bit configures the transfer direction of a message buffer. 0 Receive message buffer 1 Transmit message buffer
Message buffer control	
4 CMT	Commit for Transmission This bit indicates if the transmit message buffer data are ready for transmission. 0 Message buffer data not ready for transmission 1 Message buffer data ready for transmission
5 EDT	Enable/Disable Trigger If the application writes 1 to this bit, a message buffer enable or disable is triggered, depending on the current value of the EDS status bit. 0 No effect 1 Message buffer enable or disable is triggered
6 LCKT	Lock/Unlock Trigger If the application writes 1 to this bit and writes 0 to the EDT bit, a message buffer lock or unlock is triggered, depending on the current value of the LCKS status bit. 0 No effect 1 Message buffer lock or unlock is triggered
7 MBIE	Message Buffer Interrupt Enable This control bit defines whether the message buffer will generate an interrupt request when its MBIF flag is set. 0 Interrupt request generation disabled 1 Interrupt request generation enabled
Message buffer status	
11 DUP	Data Updated This status bit indicates whether the frame header in the message buffer header field and the data in the message buffer data field were updated after a frame reception. 0 Frame Header and message buffer data field not updated 1 Frame Header and message buffer data field updated
12 DVAL	Data Valid For receive message buffers this status bit indicates whether the message buffer data field contains valid frame data. For transmit message buffers, the status bit indicates if a message is transferred again due to the state transmission mode of the message buffer. 0 Receive message buffer contains no valid frame data / message is transmitted for the first time. 1 Receive message buffer contains valid frame data / message will be transferred again.
13 EDS	Enable/Disable Status This status bit indicates whether the message buffer is enabled or disabled. 0 Message buffer is disabled 1 Message buffer is enabled

Table 1391. FR_MBCCSR*n* field descriptions (continued)

Field	Description
14 LCKS	Lock Status This status bit indicates the current lock status of the message buffer. 0 Message buffer is not locked by the application 1 Message buffer is locked by the application
15 MBIF	Message Buffer Interrupt Flag This flag is set when the slot status field of the message buffer was updated after frame transmission or reception, or when a transmit message buffer was just enabled by the application. 0 No such event 1 Slot status field updated or transmit message buffer just enabled

57.5.2.83 Message Buffer Cycle Counter Filter Registers (FR_MBCCFR*n*)

Offset: 0x0802 +*n**0x8 (*n* = 0 to 127)

Access: Read/Write(1)
Write: *POC:config* or MB_DIS

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	MTM	CHA	CHB	CCFE	CCFMSK								CCFVAL			
Reset	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

1. 16-bit write access required

Figure 1341. Message Buffer Cycle Counter Filter Registers (FR_MBCCFR*n*)

This register contains message buffer configuration data for the transmission mode, the channel assignment, and for the cycle counter filtering. For detailed information on cycle counter filtering, refer to *Section 57.6.7.1: Message buffer cycle counter filtering*.

Table 1392. FR_MBCCFR*n* field descriptions

Field	Description
0 MTM	Message Buffer Transmission Mode This control bit applies only to transmit message buffers and defines the transmission mode. 0 Event Transmission mode 1 State Transmission mode
1 CHA	Channel Assignment This control bits defines the channel assignment and control the receive and transmit behavior of the message buffer according to *Table 1393*.
2 CHB	Channel Assignment This control bits defines the channel assignment and control the receive and transmit behavior of the message buffer according to *Table 1393*.
3 CCFE	Cycle Counter Filtering Enable This control bit is used to enable and disable the cycle counter filtering. 0 Cycle counter filtering disabled 1 Cycle counter filtering enabled

Table 1392. FR_MBCCFR*n* field descriptions (continued)

Field	Description
4:9 CCFMSK	Cycle Counter Filtering Mask This field defines the filter mask for the cycle counter filtering.
10:15 CCFVAL	Cycle Counter Filtering Value This field defines the filter value for the cycle counter filtering.

Table 1393. Channel assignment description

CHA	CHB	Transmit message buffer		Receive message buffer	
		Static segment	Dynamic segment	Static segment	Dynamic segment
1	1	transmit on both channel A and channel B	transmit on channel A only	store first valid frame received on either channel A or channel B	store first valid frame received on channel A, ignore channel B
0	1	transmit on channel B	transmit on channel B	store first valid frame received on channel B	store first valid frame received on channel B
1	0	transmit on channel A	transmit on channel A	store first valid frame received on channel A	store first valid frame received on channel A
0	0	no frame transmission	no frame transmission	no frame stored	no frame stored

Note: If at least one message buffer assigned to a certain slot is assigned to both channels, then all message buffers assigned to this slot have to be assigned to both channels. Otherwise, the message buffer configuration is illegal and the result of the message buffer search is not defined.

57.5.2.84 Message Buffer Frame ID Registers (FR_MBFIDR*n*)

Offset: 0x0804 +*n**0x8 (*n* = 0 to 127)

Access: Read/Write(1)
Write: *POC:config* or MB_DIS

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

1. 16-bit write access required

Figure 1342. Message Buffer Frame ID Registers (FR_MBFIDR*n*)

Table 1394. FR_MBFIDR*n* field descriptions

Field	Description
5:15 FID	Frame ID The semantic of this field depends on the message buffer transfer type. – *Receive Message Buffer:* This field is used as a filter value to determine if the message buffer is used for reception of a message received in a slot with the slot ID equal to FID. – *Transmit Message Buffer:* This field is used to determine the slot in which the message in this message buffer should be transmitted.

57.5.2.85 Message Buffer Index Registers (FR_MBIDXR*n*)

Offset: 0x0806 +*n**0x8 (*n* = 0 to 127)

Access: Read/Write(1)
Write: *POC:config* or MB_DIS

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

1. 16-bit write access required

Figure 1343. Message Buffer Index Registers (FR_MBIDXR*n*)

Table 1395. FR_MBIDXR*n* field descriptions

Field	Description
8:15 MBIDX	Message Buffer Index This field provides the index of the message buffer header field of the physical message buffer that is currently associated with this message buffer. The application writes the index of the initially associated message buffer header field into this register. The CC updates this register after frame reception or transmission. Legal Values are 0 ≤ i ≤ 131. Illegal values will be detected during the message buffer search.

57.5.2.86 Message Buffer Data Field Offset Registers (FR_MBDORn)

Offset: 0x1000 + *n**0x2 (n = 0 to 131)

Access: Read/Write(1)
Write: *POC:config*

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

1. 16-bit write access required

Figure 1344. Message Buffer Data Field Offset Registers (FR_MBDORn)

Table 1396. FR_MBDORn field descriptions

Field	Description
0:15 MBDO	Message Buffer Data Field Offset This field provides the data field offset belonging to a particular Message Buffer Index. For configuration constraints, refer to *Section 57.7.1.2: Configure data field offsets*.

57.5.2.87 LRAM ECC Error Test Registers (FR_LEETR*n*)

Offset: 0x1108 + *n**0x2 (n = 0 to 5)

Access: Read/Write(1)
Write: Anytime

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

1. 16-bit write access required

Figure 1345. LRAM ECC Error Test Registers (FR_LEETR*n*)

Table 1397. FR_LEETR*n* field descriptions

Field	Description
0:15 LEETD	LRAM ECC Error Test Data This field contains the LRAM data belonging to the test register located in LRAM Bank n.

57.6 Functional description

This section provides a detailed description of the functionality implemented in the CC.

57.6.1 Message buffer concept

The CC uses a data structure called a *message buffer* to store frame data, configuration, control, and status data. Each message buffer consists of two parts: the *message buffer control data* and the *physical message buffer.* The message buffer control data is located in dedicated registers. The structure of the message buffer control data depends on the message buffer type, described in *Section 57.6.3: Message buffer types*. The physical message buffer is located in the FlexRay memory area and is described in *Section 57.6.2: Physical message buffer*.

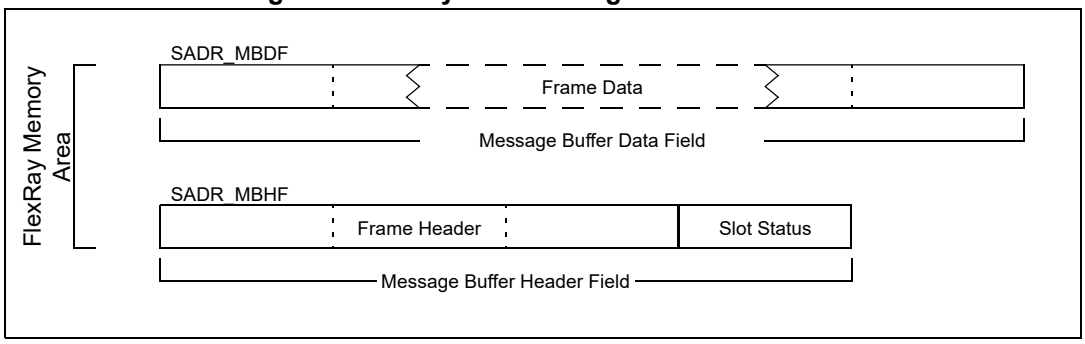
57.6.2 Physical message buffer

All FlexRay messages and related frame and slot status information of received frames and of frames to be transmitted to the FlexRay bus are stored in data structures called *physical message buffers*. The physical message buffers are located in the FlexRay memory area.The structure of a physical message buffer is depicted in *Figure 1346*.

A physical message buffer consists of two fields, the *message buffer header field* and the *message buffer data field*. The message buffer header field contains the *frame header* and the *slot status*. The message buffer data field contains the *frame data*.

The connection between the two fields is established by the *data field offset*.

Figure 1346. Physical message buffer structure



57.6.2.1 Message buffer header field

The message buffer header field is a contiguous region in the FlexRay memory area that occupies eight bytes. It contains the frame header and the slot status. Its structure is shown in *Figure 1346*. The physical start address SADR_MBHF of the message buffer header field must be 16-bit aligned.

57.6.2.1.1 Frame header

The frame header occupies the first six bytes in the message buffer header field. It contains all FlexRay frame header-related information according to the *FlexRay Communications System Protocol Specification, Version 2.1 Rev A*. A detailed description of the usage and the content of the frame header is provided in *Section 57.6.5.2.1: Frame header description*.

57.6.2.1.2 Slot status

The slot status occupies the last 2 bytes of the message buffer header field. It provides the slot and frame status-related information according to the *FlexRay Communications System Protocol Specification, Version 2.1 Rev A*. A detailed description of the content and usage of the slot status is provided in *Section 57.6.5.2.2: Slot status description*.

57.6.2.2 Message buffer data field

The message buffer data field is a contiguous area of 2-byte entities. This field contains the frame payload data, or a part of it, of the frame to be transmitted to or received from the FlexRay bus. The minimum length of this field depends on the specific message buffer configuration and is specified in the message buffer descriptions given in *Section 57.6.3: Message buffer types*.

57.6.3 Message buffer types

The CC provides three different types of message buffers.

- Individual Message Buffers
- Receive Shadow Buffers
- Receive FIFO Buffers

For each message buffer type, the structure of the physical message buffer is identical. The message buffer types differ only in the structure and content of message buffer control data, which control the related physical message buffer. The message buffer control data are described in the following sections.

57.6.3.1 Individual message buffers

The individual message buffers are used for all types of frame transmission and for dedicated frame reception based on individual filter settings for each message buffer. The CC supports three types of individual message buffers, which are described in *Section 57.6.6: Individual message buffer functional description*.

Each individual message buffer consists of two parts, the physical message buffer, which is located in the FlexRay memory area, and the message buffer control data, which are located in dedicated registers. The structure of an individual message buffer is given in *Figure 1347*.

Each individual message buffer has a message buffer number *n* assigned, which determines the set of message buffer control registers associated to this individual message buffer. The individual message buffer with message buffer number *n* is controlled by the registers FR_MBCCSR*n*, FR_MBCCFR*n*, FR_MBFIDR*n* and FR_MBIDXR*n*.

The connection between the message buffer control registers and the physical message buffer is established by the message buffer index field, MBIDX, in the Message Buffer Index Registers (FR_MBIDXRn). The start address SADR_MBHF of the related message buffer header field in the FlexRay memory area is determined according to *Equation 34*.

Equation 34 $\quad$ **SADR_MBHF = (FR_MBIDXR*n*[MBIDX] $\times$ 8) + SMBA**

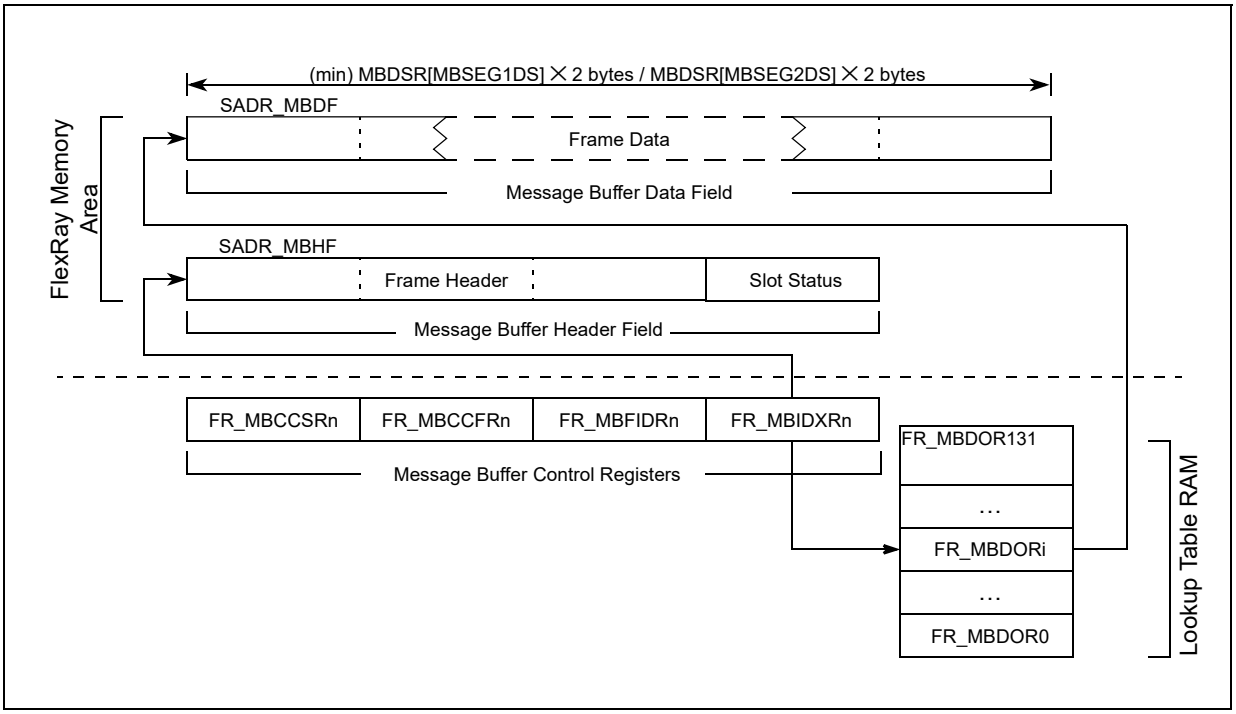
The data field belonging to a particular physical message buffer is characterized by the data field offset. For each physical message buffer with MBIDX i, the FR_MBDORi contains the offset of the corresponding message buffer data field with respect to the CC FlexRay memory area base address as provided by SMBA field in the System Memory Base Address Register (FR_SYMBADR).

The data field offset is used to determine the start address SADR_MBDF of the corresponding message buffer data field in the FlexRay memory area according to *Equation 35*.

Equation 35 $\quad$ **SADR_MBDF = [Data Field Offset] + SMBA**

The FR_MBDOR*n* are stored in the module internal memory LRAM. Refer to *Section 57.7.2.3: CHI LRAM initialization* for the setup of the data field offset values.

Figure 1347. Individual message buffer structure



57.6.3.1.1 Individual message buffer segments

The set of the individual message buffers can be split up into two message buffer segments using the Message Buffer Segment Size and Utilization Register (FR_MBSSUTR). All individual message buffers with a message buffer number n ≤ FR_MBSSUTR[LAST_MB_SEG1] belong to the first message buffer segment. All individual message buffers with a message buffer number n > FR_MBSSUTR[LAST_MB_SEG1] belong to the second message buffer segment. The following rules apply to the length of the message buffer data field:

- All physical message buffers associated to individual message buffers that belong to the same message buffer segment must have message buffer data fields of the same length.
- The minimum length of the message buffer data field for individual message buffers in the first message buffer segment is 2 × FR_MBDSR[MBSEG1DS] bytes.
- The minimum length of the message buffer data field for individual message buffers assigned to the second segment is 2 × FR_MBDSR[MBSEG2DS] bytes.

57.6.3.2 Receive shadow buffers

The receive shadow buffers are required for the individual message buffers' frame reception process. The CC provides four receive shadow buffers: one receive shadow buffer per channel and per message buffer segment.

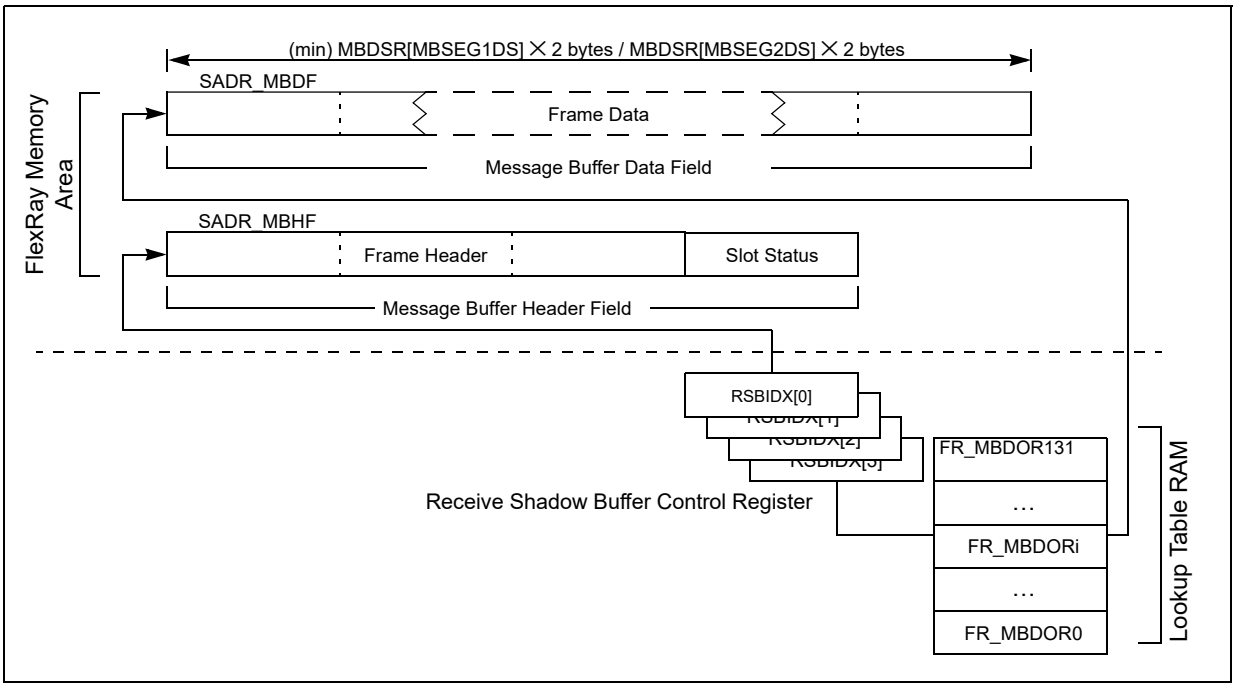
Each receive shadow buffer consists of two parts: the physical message buffer located in the FlexRay memory area and the receive shadow buffer control registers located in dedicated registers. The structure of a receive shadow buffer is shown in *Figure 1348*. The four internal shadow buffer control registers can be accessed by the Receive Shadow Buffer Index Register (FR_RSBIR).

The connection between the receive shadow buffer control register and the physical message buffer for the selected receive shadow buffer is established by the receive shadow buffer index field RSBIDX in the Receive Shadow Buffer Index Register (FR_RSBIR). The start address SADR_MBHF of the related message buffer header field in the FlexRay memory area is determined according to *Equation 36*.

Equation 36 $\quad SADR_MBHF = (FR_RSBIR[RSBIDX] \times 8) + SMBA$

The length required for the message buffer data field depends on the message buffer segment that the receive shadow buffer is assigned to. For the receive shadow buffers assigned to the first message buffer segment, the length must be the same as for the individual message buffers assigned to the first message buffer segment. For the receive shadow buffers assigned to the second message buffer segment, the length must be the same as for the individual message buffers assigned to the second message buffer segment. The receive shadow buffer assignment is described in Receive Shadow Buffer Index Register (FR_RSBIR).

Figure 1348. Receive shadow buffer structure



57.6.3.3 Receive FIFO

The receive FIFO implements a frame reception system based on the FIFO concept. The CC provides two independent receive FIFOs, one per channel.

A receive FIFO consists of a set of physical message buffers in the FlexRay memory area and a set of receive FIFO control registers located in dedicated registers. The structure of a receive FIFO is given in *Figure 1349*.

The connection between the receive FIFO control registers and the set of physical message buffers is established by the Receive FIFO Start Index Register (FR_RFSIR), the Receive FIFO Depth and Size Register (FR_RFDSR), and the Receive FIFO A Read Index Register (FR_RFARIR)/Receive FIFO B Read Index Register (FR_RFBRIR).

The system memory base address SMBA valid for the receive FIFOs is defined by the system memory base address register selected by the FIFO address mode bit FR_MCR[FAM], refer to *Section 57.5.2.4: Module Configuration Register (FR_MCR)*.

The start byte address SADR_MBHF[1] of the first message buffer header field that belongs to the receive FIFO is determined according to *Equation 37*.

Equation 37 SADR_MBHF[1] = (8 × FR_RFSIR[SIDX]) + SMBA

The start byte address SADR_MBHF[n] of the last message buffer header field that belongs to the receive FIFO in the FlexRay memory area is determined according to *Equation 38*.

Equation 38 SADR_MBHF[n] =
(8 × (FR_RFSIR[SIDX] + FR_RFDSR[FIFO_DEPTH])) + SMBA

The required information to access the current entry of the FIFO is given in the following registers:

- The registers Receive FIFO A Read Index Register (FR_RFARIR) and Receive FIFO B Read Index Register (FR_RFBRIR) provide the index of the physical message buffer belonging to the current entry.
- The data field offset belonging to the current FIFO entry RF_DFO[X] must be calculated using the current read index i according to the following formula:

Equation 39 RF_DFO[X] =
FR_RFSDOR[X] + (FR_RFDSR[X][ENTRY_SIZE] × 2) × (i - R_RFSIR[SIDX])

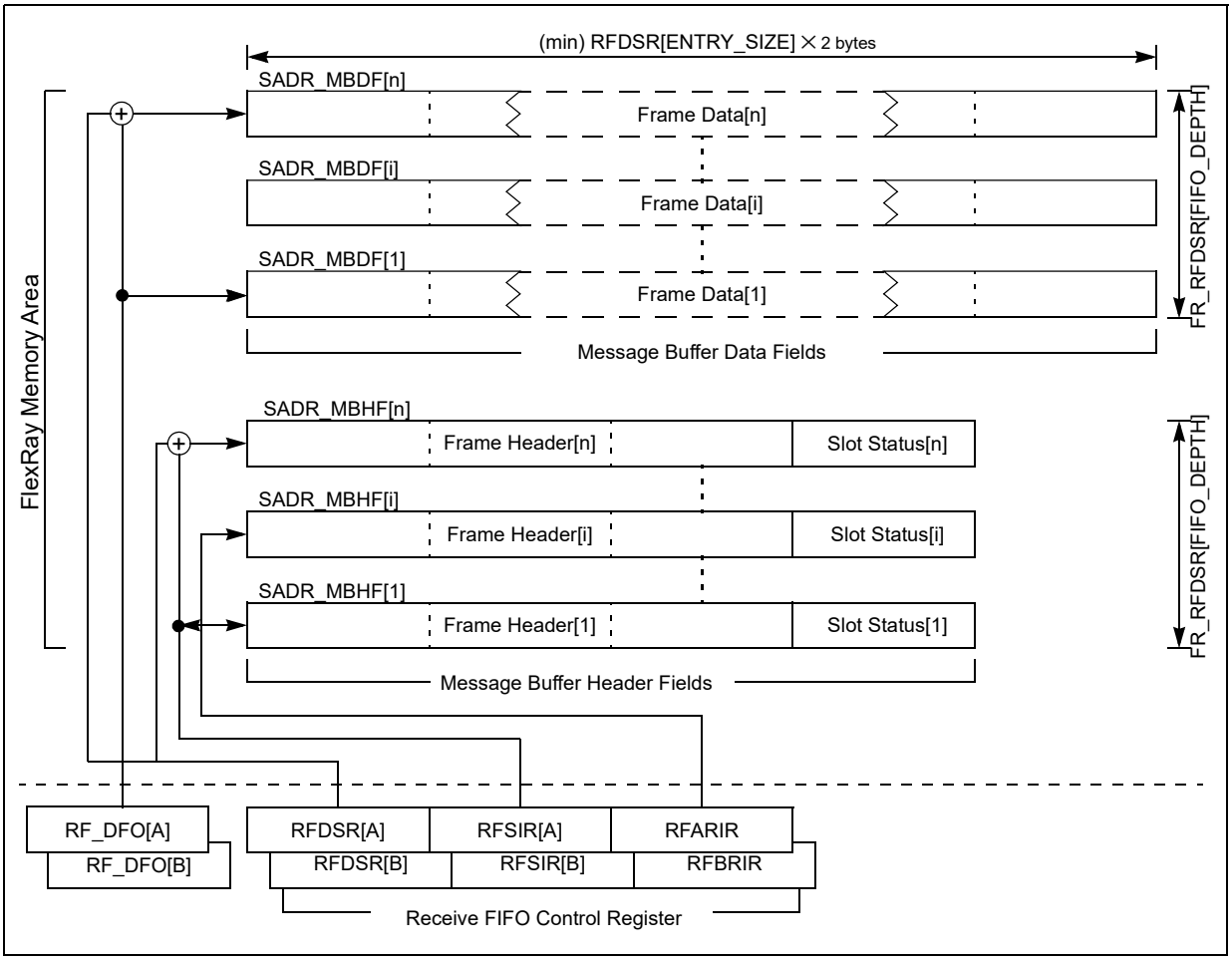
Note: The current read index loops up starting at the number given in the FR_RD[A/B]RDIDX register for the required number of entries.

Refer to Section 57.6.9.8: FIFO update for details about updating the FIFO read pointer.

All message buffer header fields assigned to a receive FIFO are within a contiguous region defined by FR_RFSIR[SIDX] and FR_RFDSR[FIFO_DEPTH].

The data sections of all FIFO entries within on receive FIFO are of the same length defined by FR_RFDSR[FIFO_SIZE].

Figure 1349. Receive FIFO structure



Note: The actual values of the data field offsets RF_DFO[A/B] need to be calculated according to Equation 39. They are not stored in a register.

57.6.3.4 Message buffer configuration and control data

This section describes the configuration and control data for each message buffer type.

57.6.3.4.1 Individual message buffer configuration data

Before an individual message buffer can be used for transmission or reception, it must be configured. There is a set of common configuration parameters that applies to all individual message buffers and a set of configuration parameters that applies to each message buffer individually.

57.6.3.4.2 Common configuration data

The set of common configuration data for individual message buffers is located in the following registers:

- Message Buffer Data Size Register (FR_MBDSR)
  The MBSEG2DS and MBSEG1DS fields define the minimum length of the message buffer data field with respect to the message buffer segment.
- Message Buffer Segment Size and Utilization Register (FR_MBSSUTR)
  The LAST_MB_SEG1 and LAST_MB_UTIL fields define the segmentation of the individual message buffers and the number of individual message buffers that are used. For more details, refer to *Section 57.6.3.1.1: Individual message buffer segments*.

57.6.3.4.3 Specific configuration data

The set of message buffer specific configuration data for individual message buffers is located in the following registers:

- Message Buffer Configuration, Control, Status Registers (FR_MBCCSRn)
  The MTD bit configures the message buffer type.
- Message Buffer Cycle Counter Filter Registers (FR_MBCCFRn)
  The MTM, CHA, CHB bits configure the transmission mode and the channel assignment. The CCFE, CCFMSK, and CCFVAL bits and fields configure the cycle counter filter.
- Message Buffer Frame ID Registers (FR_MBFIDRn)
  For a transmit message buffer, the FID field is used to determine the slot in which the message in this message buffer will be transmitted.
- Message Buffer Index Registers (FR_MBIDXRn)
  This MBIDX field provides the index of the message buffer header field of the physical message buffer that is currently associated with this message buffer.

57.6.3.5 Individual message buffer control data

During normal operation, each individual message buffer can be controlled by the CMT, LCKT, EDT, and MBIE control and trigger bits in the Message Buffer Configuration, Control, Status Registers (FR_MBCCSRn).

57.6.3.6 Receive shadow buffer configuration data

Before the individual message buffers can receive frames, the receive shadow buffers must be configured. The configuration data are provided by the Receive Shadow Buffer Index Register (FR_RSBIR). For each receive shadow buffer, the application provides the message buffer header index. When the protocol is in the *POC:normal active* or *POC:normal passive* state, the receive shadow buffers are under full CC control.

57.6.3.7 Receive FIFO control and configuration data

This section describes the configuration and control data for the two receive FIFOs.

57.6.3.7.1 Receive FIFO configuration data

The CC provides two functional independent receive FIFOs, one per channel. The FIFOs have a common subset of configuration data:

- Receive FIFO Periodic Timer Register (FR_RFPTR)

Each FIFO has its own set of configuration data. The configuration data are located in the following registers:

- Receive FIFO Watermark and Selection Register (FR_RFWMSR)
- Receive FIFO Start Index Register (FR_RFSIR)
- Receive FIFO Start Data Offset Register (FR_RFSDOR)
- Receive FIFO Depth and Size Register (FR_RFDSR)
- Receive FIFO Message ID Acceptance Filter Value Register (FR_RFMIDAFVR)
- Receive FIFO Message ID Acceptance Filter Mask Register (FR_RFMIDAFMR)
- Receive FIFO Frame ID Rejection Filter Value Register (FR_RFFIDRFVR)
- Receive FIFO Frame ID Rejection Filter Mask Register (FR_RFFIDRFMR)
- Receive FIFO Range Filter Configuration Register (FR_RFRFCFR)

57.6.3.7.2 Receive FIFO control data

The application can access the FIFOs at any time using the control bits in the following registers:

- Global Interrupt Flag and Enable Register (FR_GIFER)
- Receive FIFO Fill Level and POP Count Register (FR_RFFLPCR)

57.6.3.7.3 Receive FIFO status data

The current status of the receive FIFO is provided in the following registers:

- Global Interrupt Flag and Enable Register (FR_GIFER)
- Receive FIFO A Read Index Register (FR_RFARIR)
- Receive FIFO B Read Index Register (FR_RFBRIR)
- Receive FIFO Fill Level and POP Count Register (FR_RFFLPCR)

57.6.4 FlexRay memory area layout

The CC supports a wide range of possible layouts for the FlexRay memory area. Two basic layout modes can be selected by the FIFO address mode bit FR_MCR[FAM].

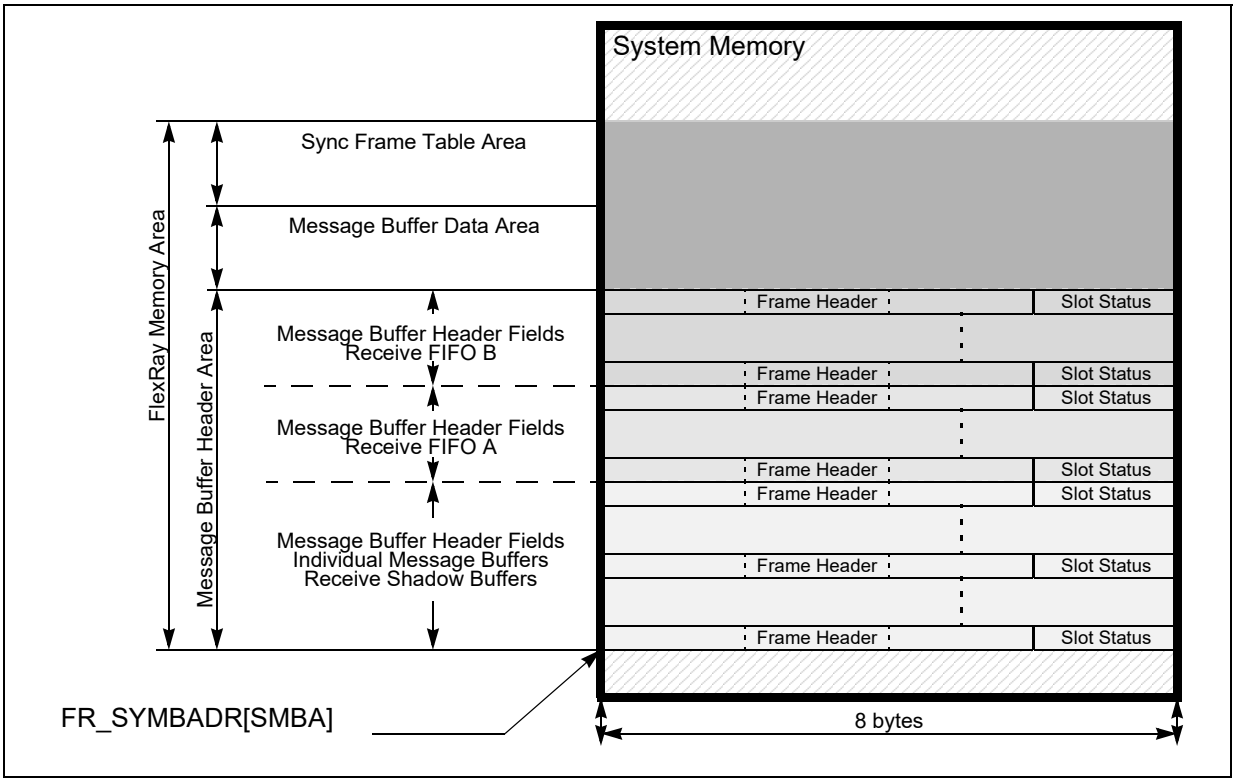
57.6.4.1 FlexRay memory area layout (FR_MCR[FAM] = 0)

Figure 1350 shows an example layout for the FIFO address mode FR_MCR[FAM] = 0. In this mode, the following set of rules applies to the layout of the FlexRay memory area:

- The FlexRay memory area is one contiguous region.
- The FlexRay memory area size is a maximum of 64 KB.
- The FlexRay memory area starts at a 16 byte boundary.

The FlexRay memory area contains three areas: the *message buffer header area*, the *message buffer data area*, and the *sync frame table area*.

Figure 1350. Example of FlexRay memory area layout (FR_MCR[FAM] = 0)

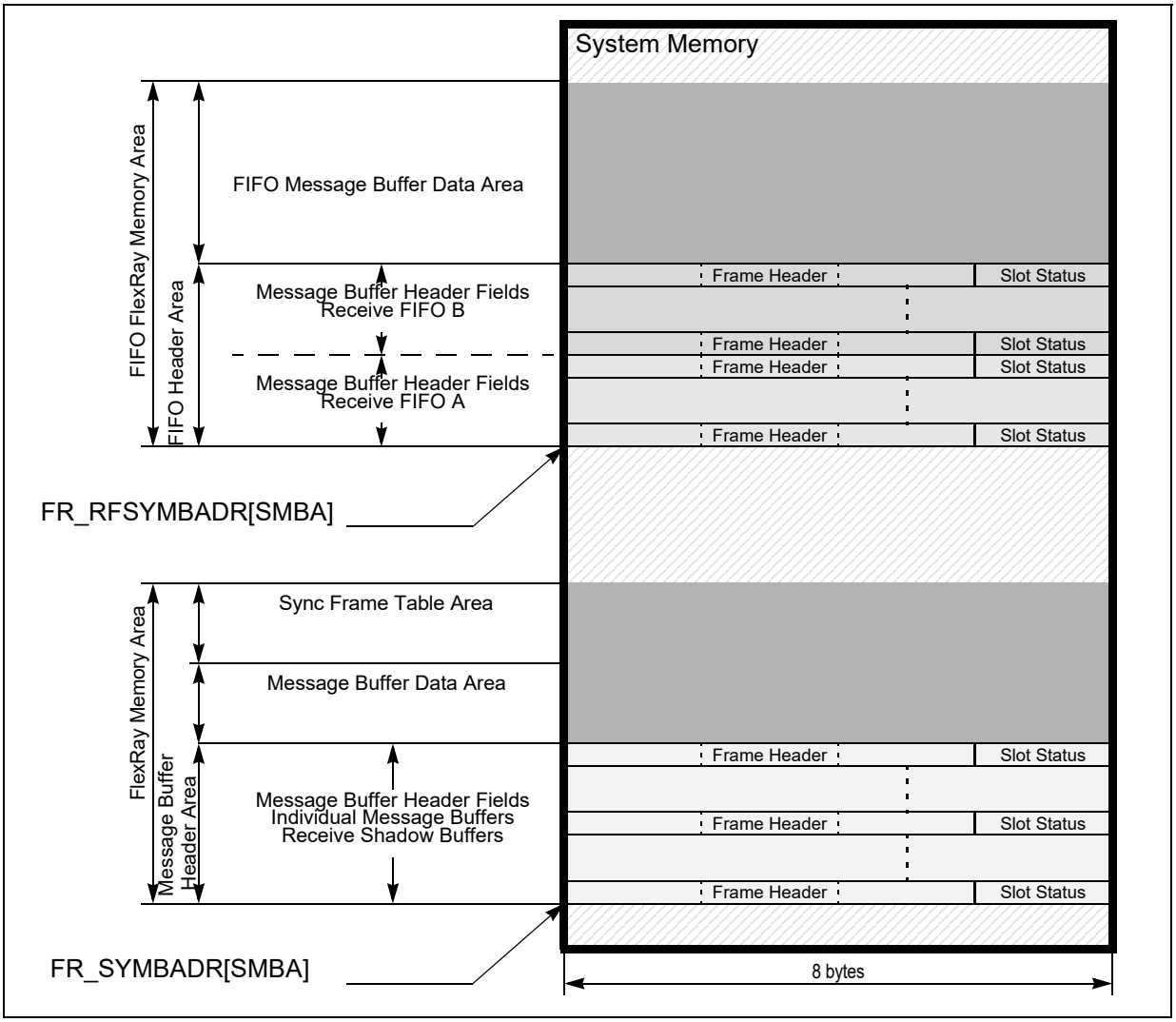


57.6.4.2 FlexRay memory area layout (FR_MCR[FAM] = 1)

Figure 1351 shows an example layout for the FIFO address mode FR_MCR[FAM] = 1. The following set of rules applies to the layout of the FlexRay memory area:

- The FlexRay memory area consists of two contiguous regions.
- The size of each region is maximum 64 KB.
- Each region start at a 16 byte boundary.

Figure 1351. Area example of FlexRay memory area layout (FR_MCR[FAM] = 1)



57.6.4.3 Message buffer header area (FR_MCR[FAM] = 0)

The message buffer header area contains all message buffer header fields of the physical message buffers for all message buffer types. The following rule applies to the message buffer header fields for the three type of message buffers.

1. The start byte address SADR_MBHF of each message buffer header field for individual message buffers and receive shadow buffers must fulfill *Equation 40*.

Equation 40 $$SADR_MBHF = (i \times 8) + FR_SYMBADR[SMBA];\ (0 \leq i \leq 131)$$

2. The start byte address SADR_MBHF of each message buffer header field for the FIFO must fulfill *Equation 41*.

Equation 41 $$SADR_MBHF = (i \times 8) + FR_SYMBADR[SMBA];\ (0 \leq i \leq 1023)$$

3. The message buffer header fields for each FIFO must be a contiguous area.

57.6.4.4 Message buffer header area (FR_MCR[FAM] = 1)

The message buffer header area contains all message buffer header fields of the physical message buffers for the individual message buffers and receiver shadow buffers. The following rules apply to the message buffer header fields for the two type of message buffers.

1. The start address SADR_MBHF of each message buffer header field for individual message buffers and receive shadow buffers must fulfill *Equation 42*.

Equation 42 $\mathbf{SADR_MBHF = (i \times 8) + FR_SYMBADR[SMBA];\ (0 \leq i \leq 131)}$

57.6.4.5 FIFO message buffer header area (FR_MCR[FAM] = 1)

The FIFO message buffer header area contains all message buffer header fields of the physical message buffers for the FIFO. The following rules apply to the FIFO message buffer header fields.

1. The start byte address SADR_MBHF of each message buffer header field for the FIFO must fulfill *Equation 43*.

Equation 43 $\mathbf{SADR_MBHF = (i \times 8) + FR_RFSYMBADR[SMBA];\ (0 \leq i \leq 1023)}$

2. The message buffer header fields for each FIFO have to be a contiguous area.

57.6.4.6 Message buffer data area

The message buffer data area contains all the message buffer data fields of the physical message buffers. Each message buffer data field must start at a 16-bit boundary.

57.6.4.7 Sync frame table area

The sync frame table area is used to provide a copy of the internal sync frame tables for application access. Refer to *Section 57.6.12: Sync frame ID and sync frame deviation tables* for the description of the sync frame table area.

57.6.5 Physical message buffer description

This section provides a detailed description of the usage and the content of the two parts of a physical message buffer: the message buffer header field and the message buffer data field.

57.6.5.1 Message buffer protection and data consistency

The physical message buffers are located in the FlexRay memory area. The CC provides no means to protect the FlexRay memory area from uncontrolled or illegal host or other client write access. To ensure data consistency of the physical message buffers, the application must follow the write access scheme that is given in the description of each of the physical message buffer fields.

57.6.5.2 Message buffer header field description

This section provides a detailed description of the usage and content of the message buffer header field. A description of the structure of the message buffer header fields is given in *Section 57.6.2.1: Message buffer header field*. Each message buffer header field consists of two sections: the frame header section and the slot status section.

57.6.5.2.1 Frame header description

57.6.5.2.1.1 Frame header content

The semantic and content of the frame header section depend on the message buffer type.

For individual receive message buffers and receive FIFOs, the frame header receives the frame header data of the *first valid frame* received on the assigned channels.

For receive shadow buffers, the frame header receives the frame header data of the current frame received regardless of whether the frame is valid or not.

For transmit message buffers, the application writes the frame header of the frame to be transmitted into this location. The frame header will be read out when the frame is transferred to the FlexRay bus.

The structure of the frame header in the message buffer header field for receive message buffers and the receive FIFO is given in *Figure 1352*. A detailed description is given in *Table 1399*.

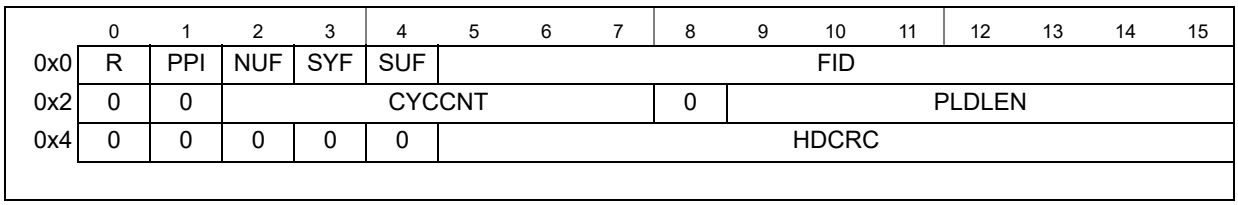


Figure 1352. Frame header structure (receive message buffer and receive FIFO)

The structure of the frame header in the message buffer header field for transmit message buffers is given in *Figure 1353*. A detailed description is given in *Table 1400*. The checks that will be performed are described in Frame Header Checks.

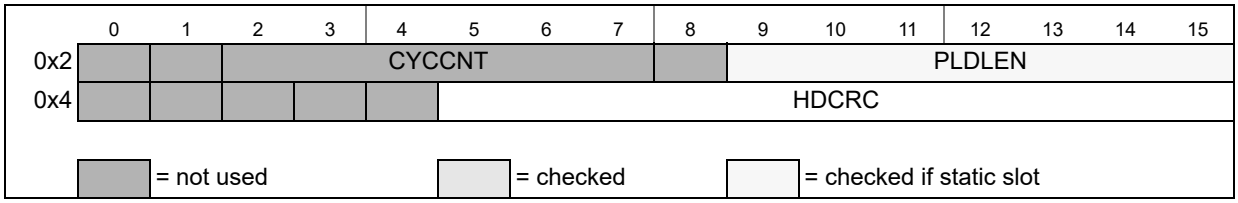


Figure 1353. Frame header structure (transmit message buffer)

The structure of the frame header in the message buffer header field for transmit message buffers assigned to key slot is given in *Figure 1354*.

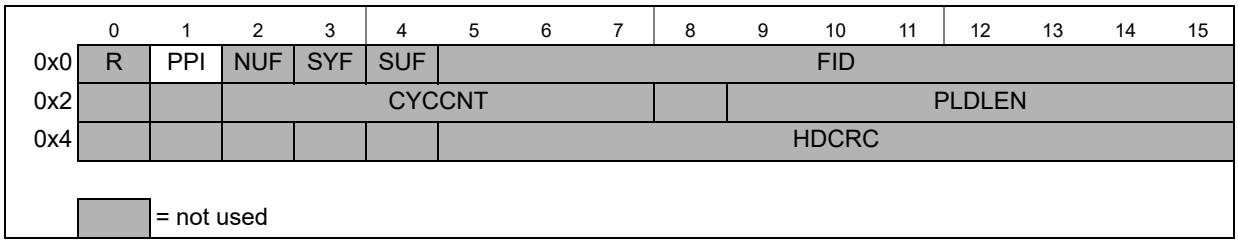


Figure 1354. Frame header structure (transmit message buffer for key slot)

57.6.5.2.1.2 Frame header access

The frame header is located in the FlexRay memory area. To ensure data consistency, the application must follow the write access scheme described below.

For receive message buffers, receive shadow buffers, and receive FIFOs, the application must not write to the frame header field.

For transmit message buffers, the application must follow the write access restrictions given in *Table 1398*. This table shows the condition under which the application can write to the frame header entries without corrupting the FlexRay message transmission.

Table 1398. Frame header write access constraints (transmit message buffer)

Field	Static segment	Dynamic segment
FID	*POC:config* or MB_DIS	
PPI, PLDLEN, HDCRC	*POC:config* or MB_DIS or	
		MB_LCK

57.6.5.2.1.3 Frame header checks

As shown in *Figure 1353* and *Figure 1354*, not all fields in the message buffer frame header are used for transmission. Some fields in the message buffer frame header are ignored, some are used for transmission, and some of them are checked for correct values. All checks that will be performed are described below.

For message buffers assigned to the key slot, no checks will be performed.

The value of the FID field must be equal to the value of the corresponding Message Buffer Frame ID Registers (FR_MBFIDRn). If the CC detects a mismatch while transmitting the frame header, it will set the frame ID error flag, FID_EF, in the CHI Error Flag Register (FR_CHIERFR). The value of the FID field will be ignored and replaced by the value provided in the Message Buffer Frame ID Registers (FR_MBFIDRn).

For transmit message buffers assigned to the static segment, the PLDLEN value must be equal to the value of the payload_length_static field in the Protocol Configuration Register 19 (FR_PCR19). If this is not fulfilled, the static payload length error flag SPL_EF in the CHI Error Flag Register (FR_CHIERFR) is set when the message buffer is under transmission. A syntactically and semantically correct frame is generated with payload_length_static payload words and the payload length field in the transmitted frame header set to payload_length_static.

For transmit message buffers assigned to the *dynamic* segment, the PLDLEN value must be less than or equal to the value of the max_payload_length_dynamic field in the Protocol Configuration Register 24 (FR_PCR24). If this is not fulfilled, the dynamic payload length error flag DPL_EF in the CHI Error Flag Register (FR_CHIERFR) is set when the message buffer is under transmission. A syntactically and semantically correct dynamic frame is generated with PLDLEN payload words and the payload length field in the frame header set to PLDLEN.

Table 1399. Frame header field descriptions (receive message buffer and receive FIFO)

Field	Description
R	Reserved Bit This is the value of the *Reserved bit* of the received frame stored in the message buffer.
PPI	Payload Preamble Indicator This is the value of the *Payload Preamble Indicator* of the received frame stored in the message buffer.
NUF	Null Frame Indicator This is the value of the *Null Frame Indicator* of the received frame stored in the message buffer.
SYF	Sync Frame Indicator This is the value of the *Sync Frame Indicator* of the received frame stored in the message buffer.
SUF	Startup Frame Indicator This is the value of the *Startup Frame Indicator* of the received frame stored in the message buffer.
FID	Frame ID This is the value of the *Frame ID* field of the received frame stored in the message buffer.
CYCCNT	Cycle Count This is the number of the communication cycle in which the frame stored in the message buffer was received.
PLDLEN	Payload Length This is the value of the *Payload Length* field of the received frame stored in the message buffer.
HDCRC	Header CRC This is the value of the *Header CRC* field of the received frame stored in the message buffer.

Table 1400. Frame header field descriptions (transmit message buffer)

Field	Description
R	Reserved Bit This bit is not used. The value of the Reserved bit is generated internally according to the FlexRay Communications System Protocol Specification, Version 2.1 Rev A.
PPI	Payload Preamble Indicator This bit provides the value of the Payload Preamble Indicator for the frame transmitted from the message buffer.
NUF	Null Frame Indicator This bit is not used, the value of the Null Frame Indicator is generated internally according to the FlexRay Communications System Protocol Specification, Version 2.1 Rev A.

Table 1400. Frame header field descriptions (transmit message buffer) (continued)

Field	Description
SYF	Sync Frame Indicator This bit is not used. The value of the Sync Frame Indicator is generated internally according to the FlexRay Communications System Protocol Specification, Version 2.1 Rev A.
SUF	Startup Frame Indicator This bit is not used. The value of the Startup Frame Indicator is generated internally according to the FlexRay Communications System Protocol Specification, Version 2.1 Rev A.
FID	Frame ID This field is checked as described in Frame Header Checks.
CYCCNT	Cycle Count This field is not used, the value of the transmitted Cycle Count field is taken from the internal communication cycle counter.
PLDLEN	Payload Length This field is checked and used as described in Frame Header Checks.
HDCRC	Header CRC This field provides the value of the Header CRC field for the frame transmitted from the message buffer.

57.6.5.2.2 Slot status description

The slot status is a read-only structure for the application and a write-only structure for the CC. The meaning and content of the slot status in the message buffer header field depend on the message buffer type.

57.6.5.2.2.1 Receive message buffer and receive FIFO slot status description

This section describes the slot status structure for the individual receive message buffers and receive FIFOs. The content of the slot status structure for receive message buffers depends on the message buffer type and on the channel assignment for individual receive message buffers as given by *Table 1401*.

Table 1401. Receive message buffer slot status content

Receive message buffer type	Slot status content
Individual Receive Message Buffer assigned to both channels FR_MBCCFR*n*[CHA] = 1 and FR_MBCCFR*n*[CHB] = 1	Refer to *Figure 1355*
Individual Receive Message Buffer assigned to channel A FR_MBCCFR*n*[CHA] = 1 and FR_MBCCFR*n*[CHB] = 0	Refer to *Figure 1356*
Individual Receive Message Buffer assigned to channel B FR_MBCCFR*n*[CHA] = 0 and FR_MBCCFR*n*[CHB] = 1	Refer to *Figure 1357*
Receive FIFO Channel A Message Buffer	Refer to *Figure 1356*
Receive FIFO Channel B Message Buffer	Refer to *Figure 1357*

The meaning of the bits in the slot status structure is explained in *Table 1402*.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	VFB	SYB	NFB	SUB	SEB	CEB	BVB	CH	VFA	SYA	NFA	SUA	SEA	CEA	BVA	0
Reset	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

Figure 1355. Receive message buffer slot status structure (ChAB)

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	VFA	SYA	NFA	SUA	SEA	CEA	BVA	0
Reset	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

Figure 1356. Receive message buffer slot status structure (ChA)

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	VFB	SYB	NFB	SUB	SEB	CEB	BVB	1	0	0	0	0	0	0	0	0
Reset	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

Figure 1357. Receive message buffer slot status structure (ChB)

Table 1402. Receive message buffer slot status field description

Field	Description
Common message buffer status bits	
VFB	Valid Frame on Channel B Protocol-related variable: *vSS!ValidFrame* channel B. 0 *vSS!ValidFrame* = 0 1 *vSS!ValidFrame* = 1
SYB	Sync Frame Indicator Channel B Protocol-related variable: *vRF!Header!SyFIndicator* channel B. 0 *vRF!Header!SyFIndicator* = 0 1 *vRF!Header!SyFIndicator* = 1
NFB	Null Frame Indicator Channel B Protocol-related variable: *vRF!Header!NFIndicator* channel B. 0 *vRF!Header!NFIndicator* = 0 1 *vRF!Header!NFIndicator* = 1
SUB	Startup Frame Indicator Channel B Protocol-related variable: *vRF!Header!SuFIndicator* channel B. 0 *vRF!Header!SuFIndicator* = 0 1 *vRF!Header!SuFIndicator* = 1
SEB	Syntax Error on Channel B Protocol-related variable: *vSS!SyntaxError* channel B. 0 *vSS!SyntaxError* = 0 1 *vSS!SyntaxError* = 1
CEB	Content Error on Channel B Protocol related variable: *vSS!ContentError* channel B. 0 *vSS!ContentError* = 0 1 *vSS!ContentError* = 1

Table 1402. Receive message buffer slot status field description (continued)

Field	Description
BVB	Boundary Violation on Channel B Protocol-related variable: *vSS!BViolation* channel B. 0 *vSS!BViolation* = 0 1 *vSS!BViolation* = 1
CH	Channel first valid received This status bit applies only to receive message buffers assigned to the static segment and to both channels. It indicates the channel that has received the *first valid* frame in the slot. This flag is set to 0 if no valid frame was received at all in the subscribed slot. 0 First valid frame received on channel A, or no valid frame received at all 1 First valid frame received on channel B
VFA	Valid Frame on Channel A Protocol-related variable: *vSS!ValidFrame* channel A. 0 *vSS!ValidFrame* = 0 1 *vSS!ValidFrame* = 1
SYA	Sync Frame Indicator Channel A Protocol-related variable: *vRF!Header!SyFIndicator* channel A. 0 *vRF!Header!SyFIndicator* = 0 1 *vRF!Header!SyFIndicator* = 1
NFA	Null Frame Indicator Channel A Protocol-related variable: *vRF!Header!NFIndicator* channel A. 0 *vRF!Header!NFIndicator* = 0 1 *vRF!Header!NFIndicator* = 1
SUA	Startup Frame Indicator Channel A Protocol-related variable: *vRF!Header!SuFIndicator* channel A. 0 *vRF!Header!SuFIndicator* = 0 1 *vRF!Header!SuFIndicator* = 1
SEA	Syntax Error on Channel A Protocol-related variable: *vSS!SyntaxError* channel A. 0 *vSS!SyntaxError* = 0 1 *vSS!SyntaxError* = 1
CEA	Content Error on Channel A Protocol-related variable: *vSS!ContentError* channel A. 0 *vSS!ContentError* = 0 1 *vSS!ContentError* = 1
BVA	Boundary Violation on Channel A Protocol-related variable: *vSS!BViolation* channel A. 0 *vSS!BViolation* = 0 1 *vSS!BViolation* = 1

57.6.5.2.2.2 Transmit message buffer slot status description

This section describes the slot status structure for transmit message buffers. Only the TCA and TCB status bits are directly related to the transmission process. All other status bits in this structure are related to a receive process that may have occurred. The content of the slot status structure for transmit message buffers depends on the channel assignment as given by *Table 1403*.

Table 1403. Transmit message buffer slot status content

Transmit Message Buffer Type	Slot Status Content
Individual Transmit Message Buffer assigned to both channels FR_MBCCFR*n*[CHA]=1 and FR_MBCCFR*n*[CHB]=1	Refer to *Figure 1358*
Individual Transmit Message Buffer assigned to channel A FR_MBCCFR*n*[CHA]=1 and FR_MBCCFR*n*[CHB]=0	Refer to *Figure 1359*
Individual Transmit Message Buffer assigned to channel B FR_MBCCFR*n*[CHA]=0 and FR_MBCCFR*n*[CHB]=1	Refer to *Figure 1360*

The meaning of the bits in the slot status structure is described in *Table 1404*.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	VFB	SYB	NFB	SUB	SEB	CEB	BVB	TCB	VFA	SYA	NFA	SUA	SEA	CEA	BVA	TCA
Reset	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

Figure 1358. Transmit message buffer slot status structure (ChAB)

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	VFA	SYA	NFA	SUA	SEA	CEA	BVA	TCA
Reset	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

Figure 1359. Transmit message buffer slot status structure (ChA)

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	VFB	SYB	NFB	SUB	SEB	CEB	BVB	TCB	0	0	0	0	0	0	0	0
Reset	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

Figure 1360. Transmit message buffer slot status structure (ChB)

Table 1404. Transmit message buffer slot status structure field description

Field	Description
VFB	Valid Frame on Channel B Protocol-related variable: *vSS!ValidFrame* channel B. 0 *vSS!ValidFrame* = 0 1 *vSS!ValidFrame* = 1
SYB	Sync Frame Indicator Channel B Protocol-related variable: *vRF!Header!SyFIndicator* channel B. 0 *vRF!Header!SyFIndicator* = 0 1 *vRF!Header!SyFIndicator* = 1
NFB	Null Frame Indicator Channel B Protocol-related variable: *vRF!Header!NFIndicator* channel B. 0 *vRF!Header!NFIndicator* = 0 1 *vRF!Header!NFIndicator* = 1
SUB	Startup Frame Indicator Channel B Protocol-related variable: *vRF!Header!SuFIndicator* channel B. 0 *vRF!Header!SuFIndicator* = 0 1 *vRF!Header!SuFIndicator* = 1

Table 1404. Transmit message buffer slot status structure field description

Field	Description
SEB	Syntax Error on Channel B Protocol-related variable: *vSS!SyntaxError* channel B. 0 *vSS!SyntaxError* = 0 1 *vSS!SyntaxError* = 1
CEB	Content Error on Channel B Protocol-related variable: *vSS!ContentError* channel B. 0 *vSS!ContentError* = 0 1 *vSS!ContentError* = 1
BVB	Boundary Violation on Channel B Protocol-related variable: *vSS!BViolation* channel B. 0 *vSS!BViolation* = 0 1 *vSS!BViolation* = 1
TCB	Transmission Conflict on Channel B Protocol-related variable: *vSS!TxConflict* channel B. 0 *vSS!TxConflict* = 0 1 *vSS!TxConflict* = 1
VFA	Valid Frame on Channel A Protocol-related variable: *vSS!ValidFrame* channel A. 0 *vSS!ValidFrame* = 0 1 *vSS!ValidFrame* = 1
SYA	Sync Frame Indicator Channel A Protocol-related variable: *vRF!Header!SyFIndicator* channel A. 0 *vRF!Header!SyFIndicator* = 0 1 *vRF!Header!SyFIndicator* = 1
NFA	Null Frame Indicator Channel A Protocol-related variable: *vRF!Header!NFIndicator* channel A. 0 *vRF!Header!NFIndicator* = 0 1 *vRF!Header!NFIndicator* = 1
SUA	Startup Frame Indicator Channel A Protocol-related variable: *vRF!Header!SuFIndicator* channel A. 0 *vRF!Header!SuFIndicator* = 0 1 *vRF!Header!SuFIndicator* = 1
SEA	Syntax Error on Channel A Protocol-related variable: *vSS!SyntaxError* channel A. 0 *vSS!SyntaxError* = 0 1 *vSS!SyntaxError* = 1
CEA	Content Error on Channel A Protocol-related variable: *vSS!ContentError* channel A. 0 *vSS!ContentError* = 0 1 *vSS!ContentError* = 1

Table 1404. Transmit message buffer slot status structure field description

Field	Description
BVA	Boundary Violation on Channel A Protocol-related variable: *vSS!BViolation* channel A. 0 *vSS!BViolation* = 0 1 *vSS!BViolation* = 1
TCA	Transmission Conflict on Channel A Protocol-related variable: *vSS!TxConflict* channel A. 0 *vSS!TxConflict* = 0 1 *vSS!TxConflict* = 1

57.6.5.3 Message buffer data field description

The message buffer data field is used to store the frame payload data, or a part of it, of the frame to be transmitted to or received from the FlexRay bus. The minimum required length of this field depends on the message buffer type that the physical message buffer is assigned to and is given in *Table 1405*. The structure of the message buffer data field is given in *Figure 1361*.

Table 1405. Message buffer data field minimum length

Physical message buffer assigned to	Minimum length defined by
Individual message buffer in segment 1	FR_MBDSR[MBSEG1DS]
Receive shadow buffer in segment 1	FR_MBDSR[MBSEG1DS]
Individual message buffer in segment 2	FR_MBDSR[MBSEG2DS]
Receive shadow buffer in segment 2	FR_MBDSR[MBSEG2DS]
Receive FIFO for channel A	FR_RFDSR[ENTRY_SIZE] (FR_RFWMSR[SEL] = 0)
Receive FIFO for channel B	FR_RFDSR[ENTRY_SIZE] (FR_RFWMSR[SEL] = 1)

Note: The CC will not access any locations outside the message buffer data field boundaries given by Table 1405.

	0 1 2 3	4 5 6 7	8 9 10 11	12 13 14 15
0x0	DATA0 / MID0 / NMV0		DATA1 / MID1 / NMV1	
0x2	DATA2 / NMV2		DATA3 / NMV3	
…	…		…	
0xN-2	DATA N-2		DATA N-1	

Figure 1361. Message buffer data field structure

The message buffer data field is located in the FlexRay memory area; thus, the CC has no means to control application write access to the field. To ensure data consistency, the application must follow a write and read access scheme.

57.6.5.3.1 Message buffer data field read access

For transmit message buffers, the CC will not modify the content of the message buffer data field. Thus the application can read back the data at any time without any impact on data consistency.

For receive message buffers the application must lock the related receive message buffer and retrieve the message buffer header index from the Message Buffer Index Registers (FR_MBIDXRn). While the message buffer is locked, the CC will not update the message buffer data field.

For receive FIFOs, the application can read the message buffer indicated by the Receive FIFO A Read Index Register (FR_RFARIR) or the Receive FIFO B Read Index Register (FR_RFBRIR) when the related fill levels in the Receive FIFO Fill Level and POP Count Register (FR_RFFLPCR) indicate an non-empty FIFO.

57.6.5.3.2 Message buffer data field write access

For receive message buffers, receive shadow buffers, and receive FIFOs, the application must not write to the message buffer data field.

For transmit message buffers, the application must follow the write access restrictions given in *Table 1406*.

Table 1406. Frame data write access constraints

Field	CC/MB State
DATA, MID, NMV	*POC:config* or MB_DIS or MB_LCK

Table 1407. Frame data field descriptions

Field	Description
DATA 0, DATA 1, … DATA N–1	Message Data Provides the message data received or to be transmitted. For receive message buffer and receive FIFOs, this field provides the message data received for this message buffer. For transmit message buffers, the field provides the message data to be transmitted.
MID 0, MID 1	Message Identifier If the payload preamble bit PPI is set in the message buffer frame header, the MID field holds the message ID of a dynamic frame located in the message buffer. The receive FIFO filter uses the received message ID for message ID filtering.
NMV 0, NMV 1, … NMV 11	Network Management Vector If the payload preamble bit PPI is set in the message buffer frame header, the network management vector field holds the network management vector of a static frame located in the message buffer. **Note:** The MID and NMV bytes replace the corresponding DATA bytes.

57.6.6 Individual message buffer functional description

The CC provides the following basic types of individual message buffers:

- Transmit message buffers
- Receive message buffers

Before an individual message buffer can be used, it must be configured by the application. After the initial configuration, the message buffer can be reconfigured later. The set of the configuration data for individual message buffers is given in *Section 57.6.3.4.1: Individual message buffer configuration data*.

57.6.6.1 Individual message buffer configuration

The individual message buffer configuration consists of two steps. The first step is the allocation of the required amount of memory for the FlexRay memory area. The second step is the programming of the message buffer configuration registers, which is described in this section.

57.6.6.1.1 Common configuration data

One part of the message buffer configuration data is common to all individual message buffers and the receive shadow buffers. This data can only be set when the protocol is in the *POC:config* state.

The application configures the number of utilized individual message buffers by writing the message buffer number of the last utilized message buffer into the LAST_MB_UTIL field in the Message Buffer Segment Size and Utilization Register (FR_MBSSUTR).

The application configures the size of the two segments of individual message buffers by writing the message buffer number of the last message buffer in the first segment into the LAST_MB_SEG1 field in the Message Buffer Segment Size and Utilization Register (FR_MBSSUTR).

The application configures the length of the message buffer data fields for both of the message buffer segments by writing to the MBSEG2DS and MBSEG1DS fields in the Message Buffer Data Size Register (FR_MBDSR).

Depending on the current receive functionality of the CC, the application must configure the receive shadow buffers. For each segment and for each channel with at least one individual receive message buffer assigned, the application must configure the related receive shadow buffer using the Receive Shadow Buffer Index Register (FR_RSBIR).

57.6.6.1.2 Specific configuration data

The second part of the message buffer configuration data is specific for each message buffer.

This data can be changed only when either

- The protocol is in the POC:config state, or
- The message buffer is disabled (that is, FR_MBCCSR*n*[EDS] = 0)

The individual message buffer type is defined by the MTD bit in the Message Buffer Configuration, Control, Status Registers (FR_MBCCSRn) as given in *Table 1408*.

Table 1408. Individual message buffer types

FR_MBCCSRn[MTD]	Individual message buffer description
0	Receive message buffer
1	Transmit message buffer

The message-buffer-specific configuration data includes the following:

1. MTD bits in the Message Buffer Configuration, Control, Status Registers (FR_MBCCSRn)
2. All fields in the Message Buffer Cycle Counter Filter Registers (FR_MBCCFRn)
3. All fields in the Message Buffer Frame ID Registers (FR_MBFIDRn)
4. All fields in the Message Buffer Index Registers (FR_MBIDXRn)

The meaning of the specific configuration data depends on the message buffer type, as given in the detailed message buffer type descriptions. Refer to *Section 57.6.6.2: Transmit message buffers* and *Section 57.6.6.3: Receive message buffers* for further information.

57.6.6.2 Transmit message buffers

The section provides a detailed description of the functionality of single-buffered transmit message buffers.

A transmit message buffer is used by the application to provide message data to the CC that will be transmitted over the FlexRay Bus. The CC uses the transmit message buffers to provide information about the transmission process and status information about the slot in which message was transmitted.

The individual message buffer with message buffer number *n* is configured to be a transmit message buffer by the following setting:

- FR_MBCCSR*n*[MTD] = 1 (transmit message buffer)

57.6.6.2.1 Access regions

For certain message buffer fields, both the application and the CC have access. To ensure data consistency, a message buffer locking scheme is used to control the access to the data, control, and status bits of a message buffer. The access regions for transmit message buffers are depicted in *Figure 1362*. A description of the regions is given in *Table 1410*. If a region is active as indicated in *Table 1411*, the access scheme given for that region applies to the message buffer.

Figure 1362. Transmit message buffer access regions

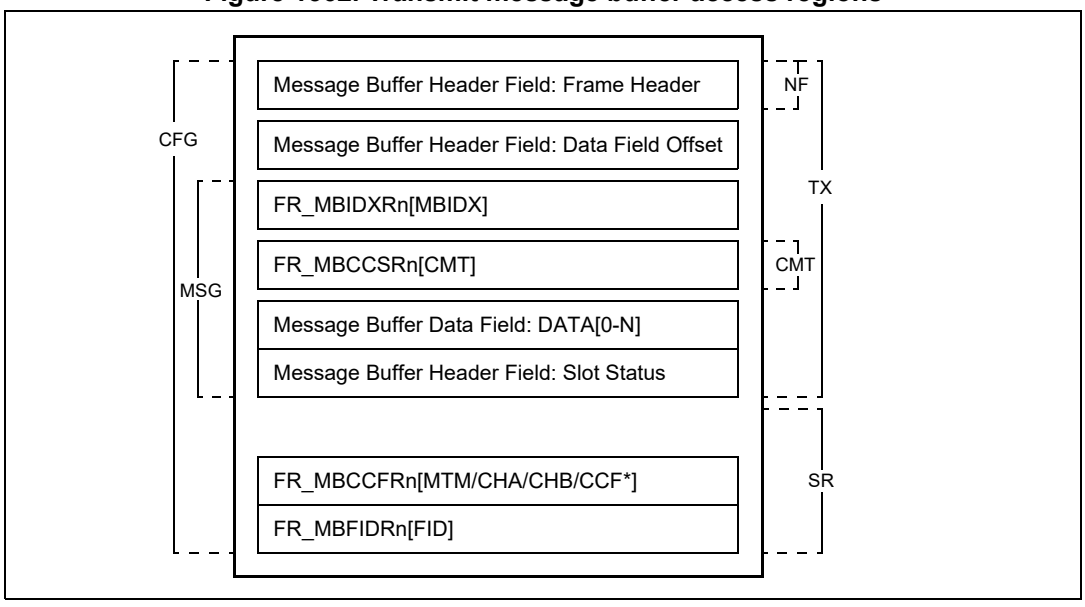


Table 1409. Transmit message buffer access regions description

Region	Access from		Region used for
	Application	Module	
CFG	Read/write	—	Message buffer configuration
MSG	Read/write	—	Message data and slot status access
NF	—	Read-only	Message header access for null frame transmission
TX	—	Read/write	Message transmission and slot status update
CMT	—	Read-only	Message buffer validation
SR	—	Read-only	Message buffer search

The trigger bits FR_MBCCSR*n*[EDT] and FR_MBCCSR*n*[LCKT], and the interrupt enable bit FR_MBCCSR*n*[MBIE], are not under access control and can be accessed from the application at any time. The status bits FR_MBCCSR*n*[EDS] and FR_MBCCSR*n*[LCKS] are not under access control and can be accessed from the CC at any time.

The interrupt flag FR_MBCCSR*n*[MBIF] is not under access control and can be accessed from the application and the CC at any time. CC clear access has higher priority.

The CC restricts its access to the regions depending on the current state of the message buffer. The application must adhere to these restrictions in order to ensure data consistency. The transmit message buffer states are given in *Figure 1363*. A description of the states is given in *Table 1410*, which also provides the access scheme for the access regions.

The status bits FR_MBCCSR*n*[EDS] and FR_MBCCSR*n*[LCKS] provide the application with the required message buffer status information. The internal status information is not visible to the application.

57.6.6.2.2 Message buffer states

This section describes the transmit message buffer states and provides a state diagram.

Figure 1363. Transmit message buffer states

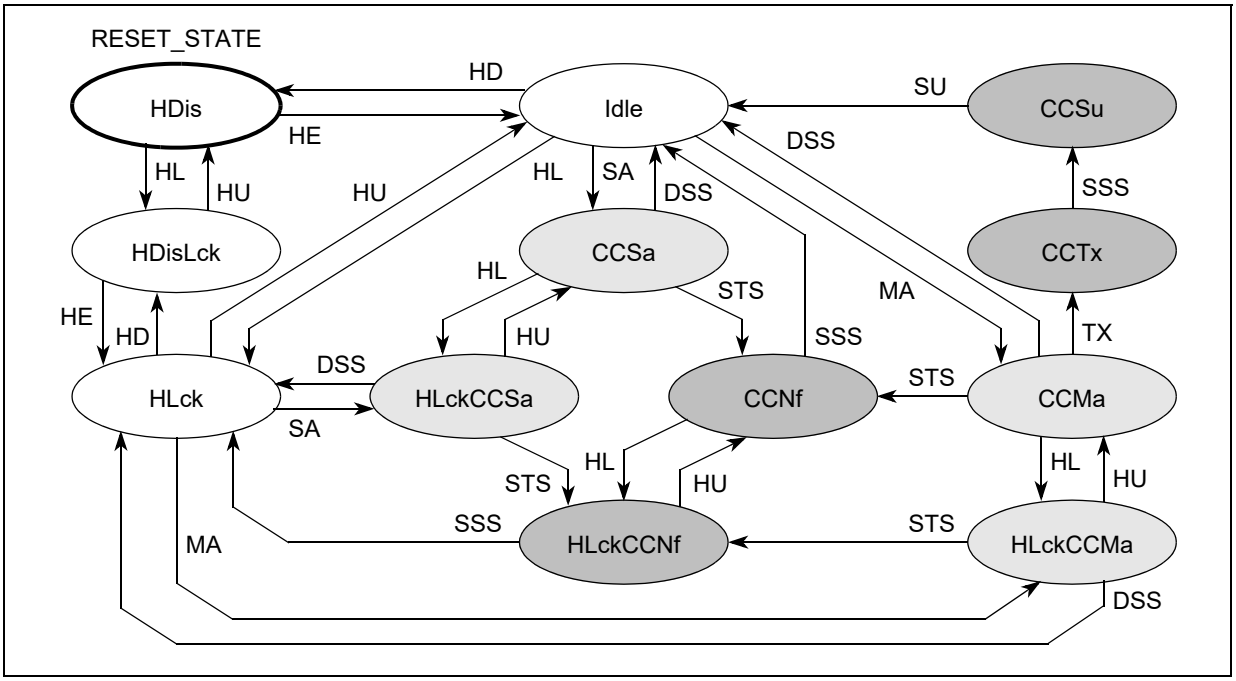


Table 1410. Transmit message buffer state description

State	FR_MBCCSR*n*		Access region		Description
	EDS	LCKS	Appl.	Module	
Idle	1	0	—	CMT, SR	Idle – Message Buffer is idle. Included in message buffer search
HDis	0	0	CFG	—	Disabled – Message Buffer under configuration. Excluded from message buffer search.
HDisLck	0	1	CFG	—	Disabled and Locked – Message Buffer under configuration. Excluded from message buffer search.
HLck	1	1	MSG	SR	Locked – Application access to data, control, and status. Included in message buffer search.
CCSa	1	0	—	—	Slot Assigned – Message buffer assigned to next static slot. Ready for Null Frame transmission.
HLckCCSa	1	1	MSG	—	Locked and Slot Assigned – Application access to data, control, and status.Message buffer assigned to next static slot.
CCNf	1	0	—	NF	Null Frame Transmission – Header is used for null frame transmission.
HLckCCNf	1	1	MSG	NF	Locked and Null Frame Transmission – Application access to data, control, and status. Header is used for null frame transmission.
CCMa	1	0	—	CMT	Message Available – Message buffer is assigned to next slot and cycle counter filter matches.

Table 1410. Transmit message buffer state description (continued)

State	FR_MBCCSR*n*		Access region		Description
	EDS	LCKS	Appl.	Module	
HLckCCMa	1	1	MSG	—	Locked and Message Available – Applications access to data, control, and status. Message buffer is assigned to next slot and cycle counter filter matches.
CCTx	1	0	—	TX	Message Transmission – Message buffer data transmit. Payload data from buffer transmitted.
CCSu	1	0	—	TX	Status Update – Message buffer status update. Update of status flags, the slot status field, and the header index.

57.6.6.2.3 Message buffer transitions

57.6.6.2.3.1 Application transitions

The application transitions can be triggered by the application using the commands described in *Table 1416*. The application issues the commands by writing to the Message Buffer Configuration, Control, Status Registers (FR_MBCCSRn). Only one command can be issued with one write access. Each command is executed immediately. If the command is ignored, it must be issued again.

- Message Buffer Enable and Disable

The enable and disable commands are issued by writing 1 to the trigger bit FR_MBCCSR*n*[EDT]. The transition that will be triggered by each of these command depends on the current value of the status bit FR_MBCCSR*n*[EDS]. If the command triggers the disable transition HD and the message buffer is in the CCSa, HLckCCSa, CCMa, HLckCCMa, CCNf, HLckCCNf, or CCTx state, the disable transition has no effect (command is ignored) and the message buffer state is not changed. No notification is given to the application.

- Message Buffer Lock and Unlock

The lock and unlock commands are issued by writing 1 to the trigger bit FR_MBCCSR*n*[LCKT]. The transition that will be triggered by each of these commands depends on the current value of the status bit FR_MBCCSRn[LCKS]. If the command triggers the lock transition HL and the message buffer is in the state CCTx, the lock transition has no effect (command is ignored) and message buffer state is not changed. In this case, the message buffer lock error flag LCK_EF in the CHI Error Flag Register (FR_CHIERFR) is set.

Table 1411. Transmit message buffer application transitions

Transition	Command	Condition	Description
HE	FR_MBCCSR*n*[EDT] = 1	FR_MBCCSR*n*[EDS] = 0	Application triggers message buffer enable.
HD		FR_MBCCSR*n*[EDS] = 1	Application triggers message buffer disable.

Table 1411. Transmit message buffer application transitions (continued)

Transition	Command	Condition	Description
HL	FR_MBCCSR*n*[LCKT] = 1	FR_MBCCSR*n*[LCKS] = 0	Application triggers message buffer lock.
HU		FR_MBCCSR*n*[LCKS] = 1	Application triggers message buffer unlock.

57.6.6.2.3.2 Module transitions

The module transitions that can be triggered by the CC are described in *Table 1412*. Each transition will be triggered for certain message buffers when the related condition is fulfilled.

Table 1412. Transmit message buffer module transitions

Transition	Condition	Description
SA	Slot match and static slot	Slot Assigned – Message buffer is assigned to next static slot.
MA	Slot match and CycleCounter match	Message Available – Message buffer is assigned to next slot and cycle counter filter matches.
TX	Slot start and FR_MBCCSR*n*[CMT] = 1	Transmission Slot Start – Slot Start and commit bit CMT is set. In case of a dynamic slot, pLatestTx is not exceeded.
SU	Status updated	Status Updated – Slot Status field and message buffer status flags updated. Interrupt flag set.
STS	Static slot start	Static Slot Start – Start of static slot.
DSS	Dynamic slot start or symbol window start or NIT start	Dynamic Slot or Segment Start – Start of dynamic slot or symbol window or NIT.
SSS	Slot start or symbol window start or NIT start	Slot or Segment Start – Start of static slot or dynamic slot or symbol window or NIT.

57.6.6.2.3.3 Transition priorities

The application can trigger only one transition at a time. There is no need to specify priorities among them.

As shown in the first part of *Table 1413*, the module transitions have a higher priority than the application transitions. For all states except the CCMa state, both a lock/unlock transition HL/HD and a module transition can be executed at the same time. The result state is reached by first applying the application transition and, subsequently, the module transition to the intermediately reached state. For example, if the message buffer is in the HLck state and the application unlocks the message buffer by the HU transition and the module triggers the slot assigned transition SA, the intermediate state is Idle and the resulting state is CCSa.

The priorities among the module transitions are given in the second part of *Table 1413*.

Table 1413. Transmit message buffer transition priorities

State	Priorities	Description
Module vs. application		
Idle, HLck	SA > HD MA > HD	Slot Assigned > Message Buffer Disable Message Available > Message Buffer Disable
CCMa	TX > HL	Transmission Start > Message Buffer Lock
Module internal		
Idle, HLck	MA > SA	Message Available > Slot Assigned
CCMa	TX > STS TX > DSS	Transmission Slot Start > Static Slot Start Transmission Slot Start > Dynamic Slot Start

57.6.6.2.4 Transmit message setup

To transmit a message over the FlexRay bus, the application writes the message data into the message buffer data field and sets the commit bit, CMT, in the Message Buffer Configuration, Control, Status Registers (FR_MBCCSRn). The physical access to the message buffer data field is described in *Section 57.6.3.1: Individual message buffers*.

As indicated by *Table 1404*, the application will write to the message buffer data field and change the commit bit CMT only if the transmit message buffer is in one of the following states: HDis, HDisLck, HLck, HLckCCSa, HLckCCMa, or HLckCCMa. The application can change the state of a message buffer if it issues the appropriate commands shown in *Table 1404*. The state change is indicated through the FR_MBCCSR*n*[EDS] and FR_MBCCSRn[LCKS] status bits.

If the transmit message buffer enters the HDis, HDisLck, HLck, HLckCCSa, HLckCCMa, or HLckCCMa state, the FR_MBCCSR*n*[DVAL] flag is negated.

57.6.6.2.5 Message transmission

As a result of the message buffer search described in *Section 57.6.7: Individual message buffer search*, the CC triggers the message available transition MA for up to two transmit message buffers. This changes the message buffer state from Idle to CCMa and the message buffers can be used for message transmission in the next slot.

The CC transmits a message from a message buffer if both of the following two conditions are fulfilled at the start of the transmission slot:

1. The message buffer is in the message available state CCMa
2. The message data is still valid (that is, FR_MBCCSR*n*[CMT] = 1)

In this case, the CC triggers the TX transition and changes the message buffer state to CCTx. A transmit message buffer timing and state change diagram for message transmission is given in *Figure 1364*. In this example, the message buffer with message buffer number *n* is Idle at the start of the search slot, matches the slot and cycle number of the next slot, and message buffer data are valid (that is, FR_MBCCSR*n*[CMT] = 1).

Figure 1364. Message transmission timing

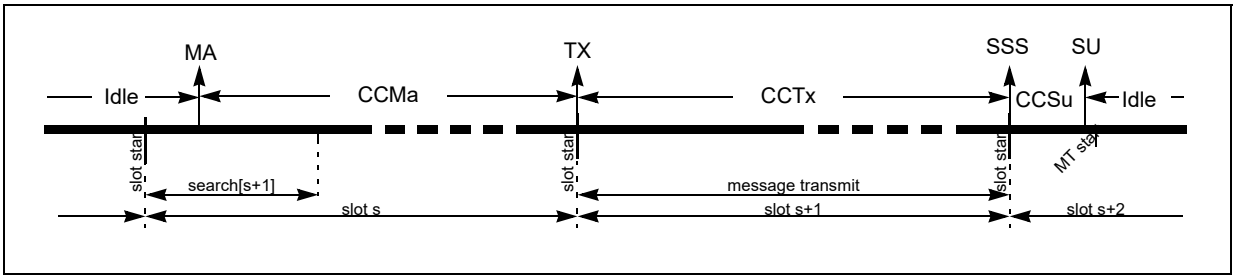
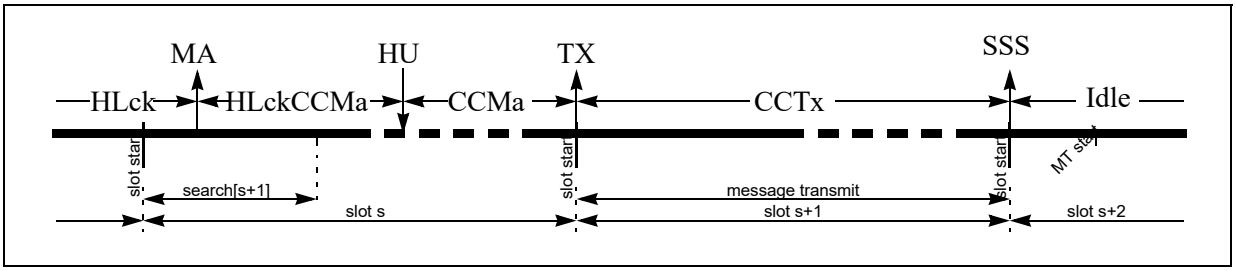


Figure 1365. Message transmission from HLck state with unlock



The amount of message data read from the FlexRay memory area and transferred to the FlexRay bus is determined by the following three items:

1. The message buffer segment that the message buffer is assigned to, as defined by the Message Buffer Segment Size and Utilization Register (FR_MBSSUTR).
2. The message buffer data field size, as defined by the related field of the Message Buffer Data Size Register (FR_MBDSR)
3. The value of the PLDLEN field in the message buffer header field, as described in *Section 57.6.5.2.1: Frame header description*

If a message buffer is assigned to message buffer segment 1, and PLDLEN > MBSEG1DS, then 2 × MBSEG1DS bytes will be read from the message buffer data field and zero padding is used for the remaining bytes for the FlexRay bus transfer. If PLDLEN ≤ MBSEG1DS, the CC reads and transfers 2 ×PLDLEN bytes. The same holds for segment 2 and MBSEG2DS.

57.6.6.2.6 Null frame transmission

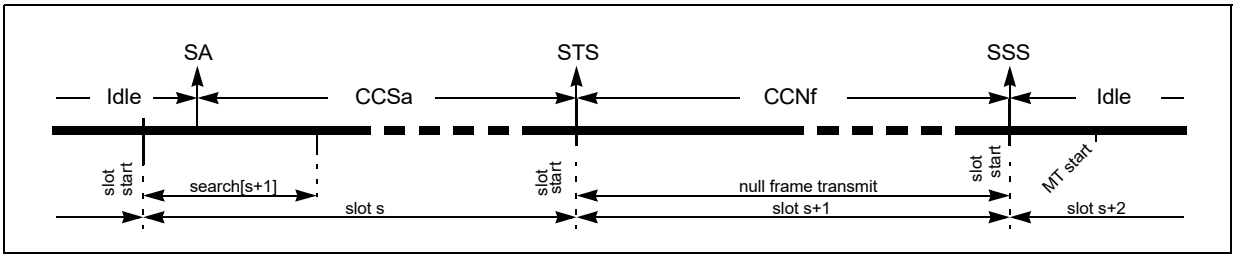
A static slot with slot number S is assigned to the CC for channel A, if at least one transmit message buffer is configured with the FR_MBFIDR*n*[FID] set to S and FR_MBCCFR*n*[CHA] set to 1. A null frame is transmitted in the static slot *s* on channel A, if this slot is assigned to the CC for channel A, and all transmit message buffers with FR_MBFIDR*n*[FID] = S and FR_MBCCFR*n*[CHA] = 1 are either not committed (that is FR_MBCCSR*n*[CMT] = 0), or are locked by the application (that is, FR_MBCCSR*n*[LCKS] = 1), or the cycle counter filter is enabled and does not match.

Additionally, the application can clear the commit bit of a message buffer that is in the CCMa state, which is called *uncommit* or *transmit abort*. This message buffer will be used for null frame transmission.

As a result of the message buffer search described in *Section 57.6.7: Individual message buffer search*, the CC triggers the slot assigned transition SA for up to two transmit message buffers if at least one of the conditions mentioned above is fulfilled for these message buffers. The transition SA changes the message buffer states from either Idle to CCSa or

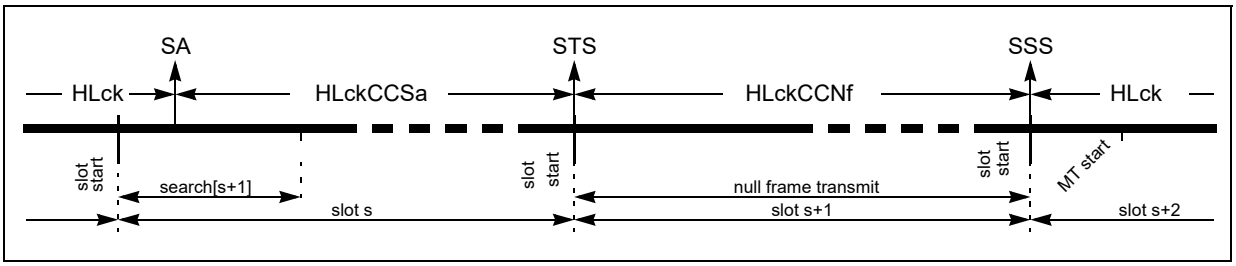
from HLck to HLckCCSa. In each case, these message buffers will be used for null frame transmission in the next slot. A message buffer timing and state change diagram for null frame transmission from Idle state is given in *Figure 1366*.

Figure 1366. Null frame transmission from Idle state



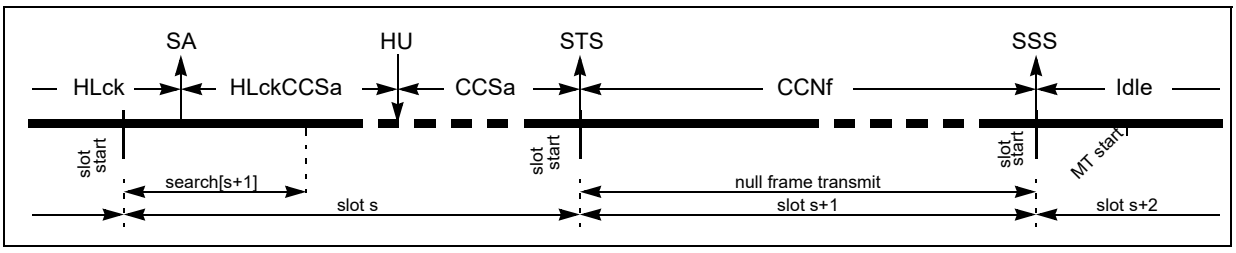
A message buffer timing and state change diagram for null frame transmission from HLck state is given in *Figure 1367*.

Figure 1367. Null frame transmission from HLck state



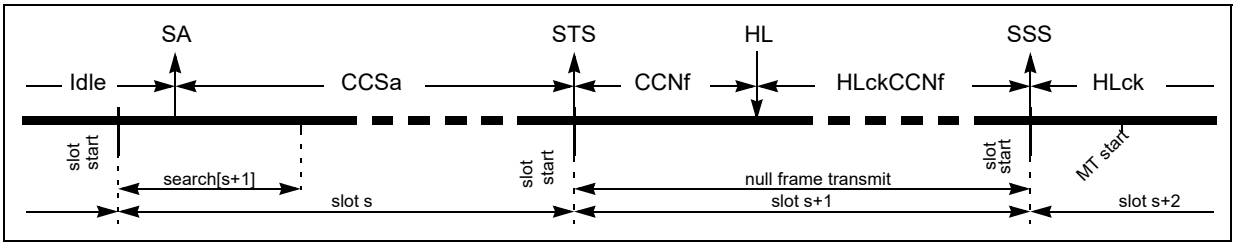
If a transmit message buffer is in the CCSa or HLckCCSa state at the start of the transmission slot, a null frame is transmitted in any case, even if the message buffer is unlocked or committed before the transmission slot starts. A transmit message buffer timing and state change diagram for null frame transmission for this case is given in *Figure 1368*.

Figure 1368. Null frame transmission from HLck state with unlock



Since the null frame transmission will not use the message buffer data, the application can lock/unlock the message buffer during null frame transmission. A transmit message buffer timing and state change diagram for null frame transmission for this case is given in *Figure 1369*.

Figure 1369. Null frame transmission from Idle state with locking



57.6.6.2.7 Message buffer status update

After the end of each slot, the PE generates the slot status vector. Depending on the status, the transmitted frame type, and the amount of transmitted data, the message buffer status is updated.

57.6.6.2.7.1 Message buffer status update after complete message transmission

The term complete message transmission refers to the fact that all payload data stored in the message buffer was sent to the FlexRay bus. In this case, the CC updates the slot status field of the message buffer and triggers the status updated transition SU. With the SU transition, the CC sets the message buffer interrupt flag FR_MBCCSR*n*[MBIF] to indicate the successful message transmission.

Depending on the Transmission Mode flag FR_MBCCFR*n*[MTM], the CC changes the commit flag FR_MBCCSR*n*[CMT] and the valid flag FR_MBCCSR*n*[DVAL]. If the FR_MBCCFRn[MTM] flag is negated, the message buffer is in *Event Transmission mode.* In this case, each committed message is transmitted only once. The commit flag FR_MBCCSR*n*[CMT] is cleared with the SU transition. If the FR_MBCCFR*n*[MTM] flag is asserted, the message buffer is in the *State Transmission mode*. In this case, each committed message is transmitted as long as the application provides new data or locks the message buffers. The CC will not clear the FR_MBCCSR*n*[CMT] flag at the end of transmission and will set the valid flag FR_MBCCSR*n*[DVAL] to indicate that the message will be transmitted again.

57.6.6.2.7.2 Message buffer status update after incomplete message transmission

The term *incomplete message transmission* refers to the fact that not all payload data that should be transmitted was sent to FlexRay bus. This may be caused by the following regular conditions in the dynamic segment:

1. The transmission slot starts in a minislot with a minislot number greater than *pLatestTx*.
2. The transmission slot did not exist in the dynamic segment at all.

Additionally, an incomplete message transmission can be caused by internal communication errors. If those errors occur, the Protocol Engine Communication Failure Interrupt Flag PECF_IF is set in the Protocol Interrupt Flag Register 1 (FR_PIFR1).

In either of these two cases, the status of the message buffer is not changed at all with the SU transition. The slot status field is not updated, the status and control flags are not changed, and the interrupt flag is not set.

57.6.6.2.7.3 Message buffer status update after null frame transmission

After the transmission of a null frame, the status of the message buffer that was used for the null frame transmission is not changed at all. The slot status field is not updated, the status and control flags are not changed, and the interrupt flag is not set.

57.6.6.3 Receive message buffers

The section provides a detailed description of the functionality of the receive message buffers. If receive message buffers are used, the related receive shadow buffer must be configured as described in *Section 57.6.3.2: Receive shadow buffers*.

A receive message buffer is used to receive a message from the FlexRay bus based on individual filter criteria. The CC uses the receive message buffer to provide the following data to the application:

- Message data received
- Information about the reception process
- Status information about the slot in which the message was received

A individual message buffer with message buffer number *n* is configured as a receive message buffer by the following configuration setting:

- FR_MBCCSR*n*[MTD] = 0 (receive message buffer)

Both the application and the CC have access to certain message buffers. To ensure data consistency, a message buffer locking scheme controls the access to the data, control, and status bits of a message buffer. The access regions for receive message buffers are depicted in *Figure 1370*. A description of the regions is given in *Table 1414*. If a region is active as indicated in *Table 1415*, the access scheme given for that region applies to the message buffer.

Figure 1370. Receive message buffer access regions

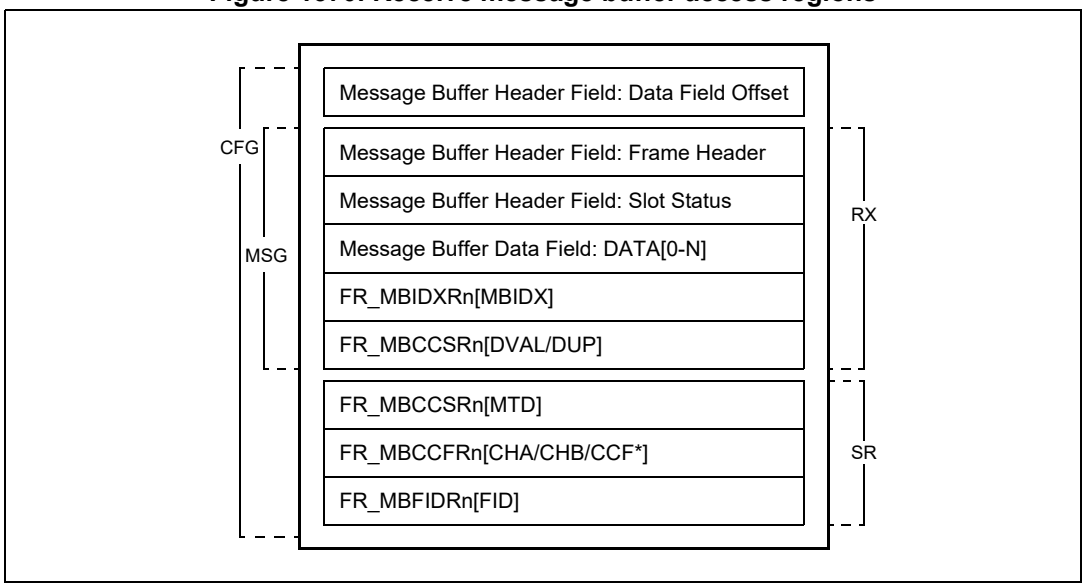


Table 1414. Receive message buffer access region description

Region	Access from		Region used for
	Application	Module	
CFG	Read/write	—	Message buffer configuration, message data, and status access
MSG	Read/write	—	Message data, header, and status access
RX	—	Write-only	Message reception and status update
SR	—	Read-only	Message buffer search data

The trigger bits FR_MBCCSR*n*[EDT] and FR_MBCCSR*n*[LCKT] and the interrupt enable bit FR_MBCCSR*n*[MBIE] are not under access control and can be accessed from the application at any time. The status bits FR_MBCCSR*n*[EDS] and FR_MBCCSR*n*[LCKS] are not under access control and can be accessed from the CC at any time.

The interrupt flag FR_MBCCSR*n*[MBIF] is not under access control and can be accessed from the application and the CC at any time. CC set access has higher priority.

The CC restricts its access to the regions depending on the current state of the message buffer. The application must adhere to these restrictions in order to ensure data consistency. The receive message buffer states are given in *Figure 1371*. A description of the message buffer states is given in *Table 1410*, which also provides the access scheme for the access regions.

The status bits FR_MBCCSR*n*[EDS] and FR_MBCCSR*n*[LCKS] provide the application with the required status information. The internal status information is not visible to the application.

Figure 1371. Receive message buffer states

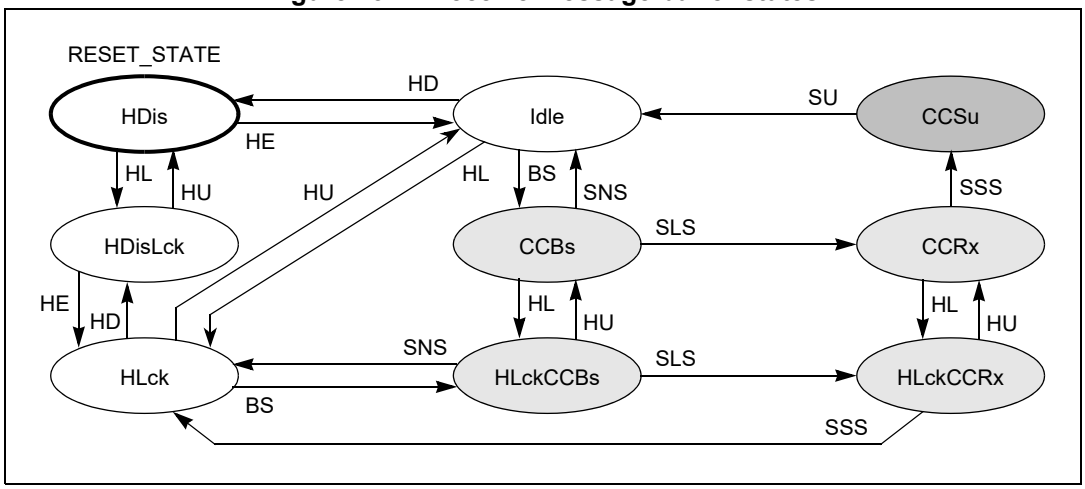


Table 1415. Receive message buffer states and access

State	FR_MBCCSR*n*		Access from		Description
	EDS	LCKS	Appl.	Module	
Idle	1	0	—	SR	Idle – Message buffer is idle. Included in message buffer search.
HDis	0	0	CFG	—	Disabled – Message buffer under configuration. Excluded from message buffer search.
HDisLck	0	1	CFG	—	Disabled and Locked – Message buffer under configuration. Excluded from message buffer search.
HLck	1	1	MSG	—	Locked – Application access to data, control, and status. Included in message buffer search.
CCBs	1	0	—	—	Buffer Subscribed – Message buffer subscribed for reception. Filter matches next (slot, cycle, channel) tuple.
HLckCCBs	1	1	MSG	—	Locked and Buffer Subscribed – Application access to data, control, and status. Message buffer subscribed for reception.
CCRx	1	0	—	—	Message Receive – Message data received into related shadow buffer.
HLckCCRx	1	1	MSG	—	Locked and Message Receive – Application access to data, control, and status. Message data received into related shadow buffer.
CCSu	1	0	—	RX	Status Update – Message buffer status update. Update of status flags, the slot status field, and the header index.

57.6.6.3.1 Message buffer transitions

57.6.6.3.1.1 Application transitions

The application transitions that can be triggered by the application using the commands are described in *Table 1411*. The application issues the commands by writing to the Message Buffer Configuration, Control, Status Registers (FR_MBCCSRn). Only one command can be issued with one write access. Each command is executed immediately. If the command is ignored, it must be issued again.

Message buffer enable and disable

The enable and disable commands are issued by writing 1 to the trigger bit FR_MBCCSR*n*[EDT]. The transition that will be triggered by each of these commands depends on the current value of the status bit FR_MBCCSR*n*[EDS]. If the command triggers the disable transition HD and the message buffer is in the CCBs, HLckCCBs, or CCRx state, the disable transition has no effect (command is ignored) and the message buffer state is not changed. No notification is given to the application.

Message buffer lock and unlock

The lock and unlock commands are issued by writing 1 to the trigger bit FR_MBCCSR*n*[LCKT]. The transition that will be triggered by each of these commands depends on the current value of the status bit FR_MBCCSR*n*[LCKS]. If the command triggers the lock transition HL while the message buffer is in the CCRx state, the lock transition has no effect (command is ignored) and the message buffer state is not changed.

In this case, the message buffer lock error flag LCK_EF in the CHI Error Flag Register (FR_CHIERFR) is set.

Table 1416. Receive message buffer application transitions

Transition	Host command	Condition	Description
HE	FR_MBCCSR*n*[EDT] = 1	FR_MBCCSR*n*[EDS] = 0	Application triggers message buffer enable
HD		FR_MBCCSR*n*[EDS] = 1	Application triggers message buffer disable
HL	FR_MBCCSR*n*[LCKT] = 1	FR_MBCCSR*n*[LCKS = 0	Application triggers message buffer lock
HU		FR_MBCCSR*n*[LCKS] = 1	Application triggers message buffer unlock

Module Transitions

The module transitions that can be triggered by the CC are described in *Table 1417*. Each transition will be triggered for certain message buffers when the related condition is fulfilled.

Table 1417. Receive message buffer module transitions

Transition	Condition	Description
BS	Slot match and cycle counter match	Buffer Subscribed – The message buffer filter matches next slot and cycle.
SLS	Slot start	Slot Start – Start of either Static Slot or Dynamic Slot.
SNS	Symbol window start or NIT start	Symbol Window or NIT Start – Start of either Symbol Window or NIT.
SSS	Slot start, symbol window start, or NIT start	Slot or Segment Start – Start of either Static Slot, Dynamic Slot, Symbol Window, or NIT.
SU	Status updated	Status Updated – Slot Status field, message buffer status flags, header index updated. Interrupt flag set.

57.6.6.3.1.2 Transition priorities

The application can trigger only one transition at a time. There is no need to specify priorities among them.

As shown in *Table 1418*, the module transitions have a higher priority than the application transitions. For all states except the CCRx state, a module transition and the application lock/unlock transition HL/HU can be executed at the same time. The result state is reached by first applying the module transition and subsequently the application transition to the intermediately reached state. For example, if the message buffer is in the CCBs buffer subscribed state and the module triggers the SLS slot start transition at the same time that the application locks the message buffer through the HL transition, the intermediate state is CCRx and the resulting state is locked buffer subscribed state, HLckCCRx.

Table 1418. Receive message buffer transition priorities

State	Priorities	Description
Module vs. Application		
Idle	BS > HD	Buffer subscribed > Message buffer disable

Table 1418. Receive message buffer transition priorities (continued)

State	Priorities	Description
HLck	BS > HD	Buffer subscribed > Message buffer disable
CCRx	SSS > HL	Slot or segment start > Message buffer lock

57.6.6.3.2 Message reception

As a result of the message buffer search, the CC changes the state of up to two enabled receive message buffers from either idle state Idle or locked state HLck to either subscribed state CCBs or locked buffer subscribed state HLckCCBs by triggering the buffer subscribed transition BS.

If the receive message buffers for the next slot are assigned to both channels, then at most one receive message buffer is changed to a buffer subscribed state.

If more than one matching message buffer is assigned to a certain channel, then only the message buffer with the lowest message buffer number is in one of the states mentioned above.

With the start of the next static or dynamic slot, the module triggers the slot start transition SLS. This changes the state of the subscribed receive message buffers from either CCBs to CCRx or from HLckCCBs to HLckCCRx, respectively.

During the reception slot, the received frame data is written into the shadow buffers. For details on receive shadow buffers, refer to *Section 57.6.6.3.5: Receive shadow buffer concept*. The data and status of the receive message buffers that are the CCRx or HLckCCRx are not modified in the reception slot.

57.6.6.3.3 Message buffer update

With the start of the next static or dynamic slot or with the start of the symbol window or NIT, the module triggers the slot or segment start transition SSS. This transition changes the state of the receiving receive message buffers from either CCRx to CCSu or from HLckCCRx to HLck, respectively.

If a message buffer is in the locked state HLckCCRx, no update will be performed. The received data is lost. This is indicated by setting the Frame Lost Channel A/B Error Flag FRLA_EF/FRLB_EF in the CHI Error Flag Register (FR_CHIERFR).

If a message buffer was in the CCRx state, it is now in the CCSu state. After the evaluation of the slot status provided by the PE, the message buffer is updated. The message buffer update depends on the slot status bits and the segment the message buffer is assigned to, as described in *Table 1419*.

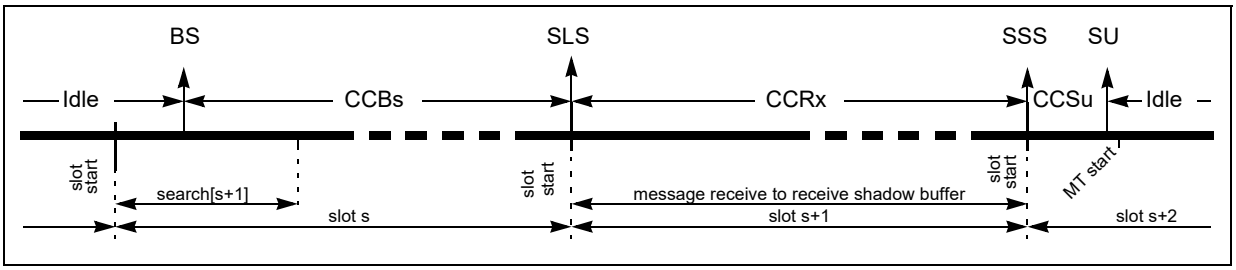
Table 1419. Receive message buffer update

vSS!ValidFrame	vRF!Header!NFIndicator	Update description
1	1	Valid non-null frame received. – Message Buffer Data field updated. – Frame Header field updated. – Slot Status field updated. – DUP = 1 – DVAL = 1 – MBIF = 1
1	0	Valid null frame received. – Message Buffer Data field *not* updated. – Frame Header field *not* updated. – Slot Status field updated. – DUP = 0 – DVAL *not* changed – MBIF = 1
0	x	No valid frame received. – Message Buffer Data field not updated. – Frame Header field not updated. – Slot Status field updated. – DUP = 0 – DVAL *not* changed. – MBIF = 1, if the slot was not an empty dynamic slot. **Note:** An empty dynamic slot is indicated by the following frame and slot status bit values: vSS!ValidFrame = 0 and vSS!SyntaxError = 0 and vSS!ContentError = 0 and vSS!BViolation = 0.

Note: If the number of the last slot in the current communication cycle on a given channel is n, then all receive message buffers assigned to this channel with FR_MBFIDRn[FID] > n will not be updated at all.

When the receive message buffer update has finished the status updated transition SU is triggered, which changes the buffer state from CCSu to Idle. An example receive message buffer timing and state change diagram for a normal frame reception is given in *Figure 1372*.

Figure 1372. Message reception timing



The amount of message data written into the message buffer data field of the receive shadow buffer is determined by the following items:

- The message buffer segment that the message buffer is assigned to, as defined by the Message Buffer Segment Size and Utilization Register (FR_MBSSUTR).
- The message buffer data field size, as defined by the related field of the Message Buffer Data Size Register (FR_MBDSR).
- The number of bytes received over the FlexRay bus.

If the message buffer is assigned to the message buffer segment 1, and the number of received bytes is greater than 2 ×FR_MBDSR.MBSEG1DS, the CC writes only 2 × FR_MBDSR.MBSEG1DS bytes into the message buffer data field of the receive shadow buffer. If the number of received bytes is less than 2 × FR_MBDSR.MBSEG1DS, the CC writes only the received number of bytes and will not change the trailing bytes in the message buffer data field of the receive shadow buffer. The same holds for the message buffer segment 2 with FR_MBDSR.MBSEG2DS.

57.6.6.3.4 Received message access

To access the message data received over the FlexRay bus, the application reads the message data stored in the message buffer data field of the corresponding receive message buffer. The access to the message buffer data field is described in *Section 57.6.3.1: Individual message buffers*.

The application can read the message buffer data field if the receive message buffer is one of the states HDis, HDisLck, or HLck. If the message buffer is in one of these states, the CC will not change the content of the message buffer.

57.6.6.3.5 Receive shadow buffer concept

The receive shadow buffer concept applies only to individual receive message buffers. The intention of this concept is to ensure that only syntactically and semantically valid received non-null frames are presented to the application in a receive message buffer. The basic structure of a receive shadow buffer is described in *Section 57.6.3.2: Receive shadow buffers*.

The receive shadow buffers temporarily store the received frame header and message data. The slot status information is generated after the slot boundary. If the slot status information indicates the reception of the valid non-null frame (refer to *Table 1419*), the CC writes the slot status into the slot status field of the receive shadow buffer and exchanges the content of the Message Buffer Index Registers (FR_MBIDXRn) with the content of the corresponding internal shadow buffer index register. In all cases, the CC writes the slot status into the identified receive message buffer, depending on the slot status and the FlexRay segment the message buffer is assigned to.

The shadow buffer concept, with its index exchange, results in the fact that the FlexRay memory area located message buffer associated to an individual receive message buffer changes after successful reception of a valid frame. This means that the message buffer area in the FlexRay memory area accessed by the application for reading the received message is different from the initial setting of the message buffer. Therefore, the application must not rely on the index information written initially into the Message Buffer Index Registers (FR_MBIDXRn). Instead, the index of the message buffer header field must be fetched from the Message Buffer Index Registers (FR_MBIDXRn).

57.6.7 Individual message buffer search

This section provides a detailed description of the message buffer search algorithm.

The message buffer search determines for each enabled channel whether a slot *s* in a communication cycle *c* is assigned for frame or null frame transmission or if it is subscribed for frame reception on that channel.

The message buffer search is a sequential algorithm invoked at the following protocol-related events:

- NIT start
- Slot start in the static segment
- Minislot start in the dynamic segment

The message buffer search within the NIT searches for message buffers assigned or subscribed to slot 1. The message buffer search within slot *s* searches for message buffers assigned or subscribed to slot *s* + *1*.

In general, the message buffer search for the next slot *s* considers only message buffers that

1. Are enabled (that is, FR_MBCCSR*n*[EDS] = 1), and
2. Match the next slot *s*, (that is, FR_MBFIDR*n*[FID] = *s*)

On top of that, for the static segment, only those message buffers are considered that match the condition of at least one row of *Table 1420*. For the dynamic segment, only those message buffers are considered that match the condition of at least one row of *Table 1421*. These message buffers are called *matching* message buffers.

For each enabled channel, the message buffer search may identify multiple *matching* message buffers. Among all matching message buffers, the message buffers with highest priority according to *Table 1420* for the static segment and according to *Table 1421* for the dynamic segment are selected.

Table 1420. Message buffer search priority (static segment)

Priority	MTD	LCKS	CMT	CCFM(1)	Description	Transition
(Highest) 0	1	0	1	1	Transmit buffer, matches cycle count, not locked and committed	MA
1	1	—	0	1	Transmit buffer, matches cycle count, not committed	SA
	1	1	—	1	Transmit buffer, matches cycle count, locked	SA
2	1	—	—	—	Transmit buffer	SA
3	0	0	n/a	1	Receive buffer, matches cycle count, not locked	BS
(Lowest) 4	0	1	n/a	1	Receive buffer, matches cycle count, locked	BS

1. Cycle Counter Filter Match, refer to *Section 57.6.7.1: Message buffer cycle counter filtering.*

Table 1421. Message buffer search priority (dynamic segment)

Priority	MTD	LCKS	CMT	CCFM(1)	Description	Transition
(Highest) 0	1	0	1	1	Transmit buffer, matches cycle count, not locked and committed	MA
1	0	0	n/a	1	Receive buffer, matches cycle count, not locked	BS
(Lowest) 2	0	1	n/a	1	Receive buffer, matches cycle count, locked	BS

1. Cycle Counter Filter Match, refer to *Section 57.6.7.1: Message buffer cycle counter filtering*

If there are multiple message buffers with highest priority, the message buffer with the lowest message buffer number is selected. All message buffers that have the highest priority must have a consistent channel assignment as specified in *Section 57.6.7.2: Message buffer channel assignment consistency*.

Depending on the message buffer channel assignment, the same message buffer can be found for both channel A and channel B. In this case, this message buffer is used as described in *Section 57.6.3.1: Individual message buffers*.

57.6.7.1 Message buffer cycle counter filtering

The message buffer cycle counter filter is a value-mask filter defined by the CCFE, CCFMSK, and CCFVAL fields in the Message Buffer Cycle Counter Filter Registers (FR_MBCCFRn). This filter determines a set of communication cycles in which the message buffer is considered for message reception or message transmission. If the cycle counter filter is disabled (CCFE = 0), this set of cycles consists of all communication cycles.

If the cycle counter filter of a message buffer does not match a certain communication cycle number, this message buffer is not considered for message transmission or reception in that communication cycle. In case of a transmit message buffer assigned to a slot in the static segment, though, this buffer is added to the matching message buffers to indicate the slot assignment and to trigger the null frame transmission.

The cycle counter filter of a message buffer matches the communication cycle with the number CYCCNT if at least one of the following condition is true.

Equation 44

$$MBCCFRn[CCFE] = 0$$

Equation 45

$$CYCCNT \& MBCCFRn[CCFMSK] = MBCCFRn[CCFVAL] \& MBCCFRn[CCFMSK]$$

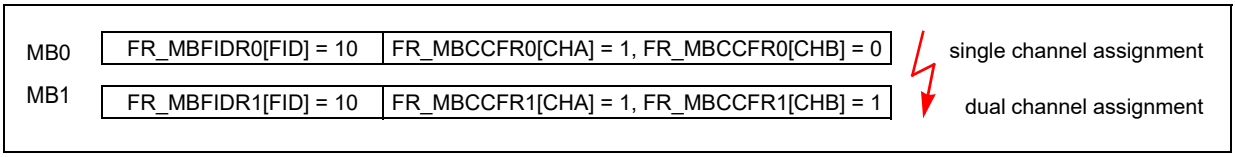
57.6.7.2 Message buffer channel assignment consistency

The message buffer channel assignment given by the CHA and CHB bits in the Message Buffer Cycle Counter Filter Registers (FR_MBCCFRn) defines the channels on which the message buffer will receive or transmit. The message buffer with number *n* transmits or receives on channel A if FR_MBCCFR*n*[CHA] = 1 and transmits or receives on channel B if FR_MBCCFR*n*[CHB] = 1.

To ensure correct message buffer operation, all message buffers assigned to the same slot and with the same priority must have a *consistent* channel assignment. That means they must be either assigned to one channel only, or must be assigned to *both* channels. The

behavior of the message buffer search is not defined if both types of channel assignments occur for one slot and priority. An inconsistent channel assignment for message buffer 0 and message buffer 1 is depicted in *Figure 1373*.

Figure 1373. Inconsistent channel assignment



57.6.7.3 Node-related slot multiplexing

The term *node-related slot multiplexing* applies to the dynamic segment only and refers to the functionality if transmit as well as receive message buffers are configured for the same slot.

According to *Table 1421* the transmit buffer is only found if the cycle counter filter matches, and the buffer is not locked and committed. In all other cases, the receive buffer will be found. Thus, if the block has no data to transmit in a dynamic slot, it is able to receive frames on that slot.

57.6.7.4 Message buffer search error

There are two kinds of error that may occur during message buffer search[25].

57.6.7.4.1 Message buffer search start while running

If the message buffer search is running in slot *s – 1* and the next message buffer search start event appears due to the start of slot *s*, the message buffer search engine is stopped and the Message Buffer Search Error Flag MBS_EF is set in the CHI Error Flag Register (FR_CHIERFR). As a result of this stop, no individual message buffer is identified for transmission or reception in slot *s*. Additionally, the search engine will not be started in slot *s*, and consequently no individual message buffer is identified for transmission or reception in slot *s + 1*.

A message buffer search error appears only if the CHI frequency is too slow to allow the search through all message buffers to be completed within the NIT or a minislot.

For more details of minimum required CHI frequency, refer to *Section 57.7.5: Number of usable message buffers*.

57.6.7.4.2 Illegal message buffer index found

If the message buffer search has finished the message buffer search in slot *s – 1*, it retrieves the data offset values for the found message buffers and the receive shadow buffers. If one of these message buffers contains an illegal message buffer index, the Message Buffer Search Error Flag MBS_EF is set in the CHI Error Flag Register (FR_CHIERFR) and no individual message buffer is identified for transmission or reception in slot *s*. The legal message buffer index values for the individual and receive shadow

25. The FIFO reception is not affected by the search errors. Additionally, if no rx buffer has been found due to a search error, the received frame is considered for FIFO reception.

buffers are specified in *Section 57.5.2.53: Receive Shadow Buffer Index Register (FR_RSBIR)* and *Section 57.5.2.85: Message Buffer Index Registers (FR_MBIDXRn)*.

57.6.8 Individual message buffer reconfiguration

The initial configuration of each individual message buffer can be changed even when the protocol is not in the *POC:config* state. This is referred to as individual message buffer *reconfiguration*. The configuration bits and fields that can be changed are given in the section on Specific Configuration Data. The common configuration data given in the section on Specific Configuration Data cannot be reconfigured when the protocol is out of the *POC:config* state.

57.6.8.1 Reconfiguration schemes

Depending on the target and destination basic state of the message buffer that is to be reconfigured, there are three reconfiguration schemes.

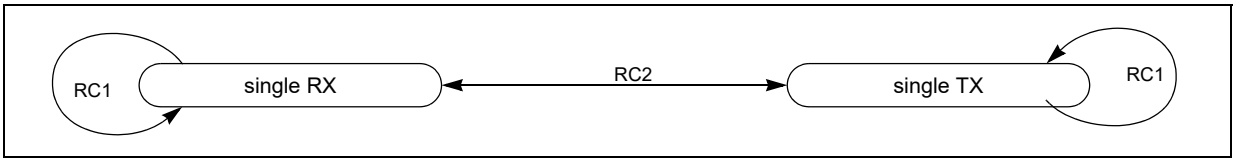
57.6.8.1.1 Basic type not changed (RC1)

A reconfiguration will not change the basic type of the individual message buffer, if the message buffer transfer direction bit FR_MBCCSR*n*[MTD] is not changed. This type of reconfiguration is denoted by RC1 in *Figure 1374*. Transmit and receive message buffers can be RC1-reconfigured when in the HDis or HDisLck state.

57.6.8.1.2 Buffer type not changed (RC2)

A reconfiguration will not change the buffer type of the individual message buffer. This type of reconfiguration is denoted by RC2 in *Figure 1374*. It applies to transmit and receive message buffers. Transmit and receive message buffers can be RC2-reconfigured when in the HDis or HDisLck state.

Figure 1374. Message buffer reconfiguration scheme



57.6.9 Receive FIFOs

This section provides the functional description of the two receive FIFOs.

57.6.9.1 Overview

The two receive FIFOs implement the queued message buffer concept defined by the *FlexRay Communications System Protocol Specification, Version 2.1 Rev A.* One FIFO is assigned to channel A, and the other FIFO is assigned to channel B. Both FIFOs work independently from each other.

The message buffer structure of each FIFO is described in *Section 57.6.3.3: Receive FIFO*. The area in the FlexRay memory area for each of the two FIFOs is characterized by:

- The FIFO system memory base address.
- The index of the first FIFO entry given by the Receive FIFO Start Index Register (FR_RFSIR).
- The data field offset of the data field belonging to the first FIFO entry given by the Receive FIFO Start Data Offset Register (FR_RFSDOR).
- The number of FIFO entries and the length of each FIFO entry as given by the Receive FIFO Depth and Size Register (FR_RFDSR).

57.6.9.2 FIFO configuration

The FIFOs can be configured for two different locations of the system memory base address via the FIFO Address Mode bit FAM in the Module Configuration Register (FR_MCR).

57.6.9.2.1 Single System Memory Base Address mode

This mode is configured when the FIFO address mode flag FR_MCR[FAM] is set to 0. In this mode, the location of the system memory base address for the FIFO buffers is the System Memory Base Address Register (FR_SYMBADR).

57.6.9.2.2 Dual System Memory Base Address mode

This mode is configured when the FIFO address mode flag FR_MCR[FAM] is set to 1. In this mode, the location of the system memory base address for the FIFO buffers is the Receive FIFO System Memory Base Address Register (FR_RFSYMBADR).

The FIFO control and configuration data is given in *Section 57.6.3.7: Receive FIFO control and configuration data*. The configuration of the FIFOs consists of two steps.

The first step is the allocation of the required amount of FlexRay memory area for the FlexRay window. This includes the allocation of the message buffer header area and the allocation of the message buffer data fields. For more details, refer to *Section 57.6.4: FlexRay memory area layout*.

The second step is the programming of the configuration data register while the PE is in *POC:config*.

The following steps configure the layout of the FIFO.

1. Configure the FIFO update and address modes in Module Configuration Register (FR_MCR).
2. Configure the FIFO system memory base address.
3. Configure the Receive FIFO Start Index Register (FR_RFSIR) with the first message buffer header index that belongs to the FIFO.
4. Configure the Receive FIFO Start Data Offset Register (FR_RFSDOR) with the data field offset of the data field belonging to the first message buffer that belongs to the FIFO.
5. Configure the Receive FIFO Depth and Size Register (FR_RFDSR) with FIFO entry size.
6. Configure the Receive FIFO Depth and Size Register (FR_RFDSR) with FIFO depth.
7. Configure the FIFO filters.

57.6.9.3 FIFO periodic timer

The FIFO periodic timer is used to generate a FIFO almost-full interrupt at a certain point in time, if the almost-full watermark is not reached but the FIFO is not empty. This can be used to prevent frames from getting stuck in the FIFO for a long time.

The FIFO periodic timer is configured via the Receive FIFO Periodic Timer Register (FR_RFPTR). If the periodic timer duration FR_RFPTR[PTD] is configured to 0x0000, the periodic timer is continuously expired. If the periodic timer duration FR_RFPTR[PTD] is configured to 0x3FFF, the periodic timer never expires. If the periodic timer is configured to a value *ptd,* greater than 0x0000 and smaller 0x3FFF, the periodic timer expires and is restarted at the start of every communication cycle, and expires and is restarted after *ptd* macroticks have been elapsed.

57.6.9.4 FIFO reception

The FIFO reception is a CC internal operation.

A message frame reception is directed into the FIFO if no individual message buffer is assigned for transmission or subscribed for reception for the current slot. In this case, the FIFO filter path shown in *Figure 1375* is activated.

If the FIFO filter path indicates that the received frame has to be appended to the FIFO and the FIFO is not full, the CC writes the received frame header into the message buffer header field indicated by the CC internal FIFO write index. The frame payload data are written into the corresponding message buffer data field. If the status of the received frame indicates a valid non-null frame, the slot status information is written into the message buffer header field and the CC internal FIFO write index is updated by 1 and the FIFO fill level FLA (FLB) in the Receive FIFO Fill Level and POP Count Register (FR_RFFLPCR) is incremented.If the status of the received frame indicates an invalid or null frame, the frame is not appended to the FIFO.

57.6.9.5 FIFO almost-full interrupt generation

If the FIFO fill level FLA (FLB) is updated after a frame reception and exceeds the FIFO watermark level WM (that is, FLA > WM_A (FLB > WM_B)), then the FIFO almost-full interrupt flag FR_GIFER[FAFAIF] (FR_GIFER[FAFBIF]) is asserted.If the periodic timer expires, and FIFOA (FIFOB) is not empty (that is, FLA > 0 (FLB > 0), then the FIFO almost-full interrupt flag FR_GIFER[FAFAIF] (FR_GIFER[FAFBIF]) is asserted.

57.6.9.6 FIFO overflow error generation

If the FIFOA (FIFOB) is full (that is, FLA = $FIFO_DEPTH_A$ (FLB=$FIFO_DEPTH_B$)) and the conditions for a FIFO reception as described in *Section 57.6.9.4: FIFO reception* are fulfilled, then the FIFO overflow error flag FR_CHIERFR[FOVA_EF] (FR_CHIERFR[FOVB_EF]) is asserted.

57.6.9.7 FIFO message access

The FIFOA (FIFOB) contains valid messages if the FIFO fill level given in the fields FLA (FLB) in the Receive FIFO Fill Level and POP Count Register (FR_RFFLPCR) is greater than 0. The Receive FIFO A Read Index Register (FR_RFARIR) and the (Receive FIFO B Read Index Register (FR_RFBRIR)) point to a message buffer with valid content and the oldest frames stored in the FIFO.The respective read data field offsets can be calculated according to *Equation 39*.

If the FIFO fill level FLA (FLB) in the Receive FIFO Fill Level and POP Count Register (FR_RFFLPCR) is 0, then the FIFOA (FIFOB) contains no valid messages and the corresponding read index register Receive FIFO A Read Index Register (FR_RFARIR) or (Receive FIFO B Read Index Register (FR_RFBRIR) point to a message buffer with invalid content. In this case, the application must not read data from this FIFO.

To access the oldest message in the FIFOA (FIFOB), the application first reads the read index RDIDX out of the Receive FIFO A Read Index Register (FR_RFARIR) (Receive FIFO B Read Index Register (FR_RFBRIR)). This read index points to the message buffer header field of the oldest message buffer that contains valid received message data. The data field offset belonging to this message buffer must be calculated by the application according to *Equation 39*. The application can access the message data as described in *Section 57.6.3.3: Receive FIFO*. When the application has read the message buffer data and status information, it can update the FIFO as described in *Section 57.6.9.8: FIFO update*.

57.6.9.8 FIFO update

The application updates the FIFOA (FIFOB) by writing a pop count value *pc* different from 0 to the PCA (PCB) field in the Receive FIFO Fill Level and POP Count Register (FR_RFFLPCR).

As a result of the this operation, the CC removes the oldest *pc* entries from FIFOA (FIFOB).

If the specified pop count value *pc* is greater than the current fill level *fl* provided in FLA (FAB) field, then only *fl* entries are removed from the FIFOA (FIFOB), the remaining *fl-pc* requested pop operations are discarded without any notification. In this case FIFOA (FIFOB) is empty after the update operation.

The read index in the Receive FIFO A Read Index Register (FR_RFARIR) (Receive FIFO B Read Index Register (FR_RFBRIR)) is incremented by the number of removed items. If the read index reaches the top of the FIFO, it wraps around to the FIFO start index defined in Receive FIFO Start Index Register (FR_RFSIR) automatically.

57.6.9.8.1 FIFO interrupt flag update

The FIFO Interrupt Flag Update mode is configured when the FIFO update mode flag FR_MCR[FUM] is set to 0. In this mode, FIFOA (FIFOB) will be updated by one entry when the interrupt flag FR_GIFER[FAFAIF] (FR_GIFER[FAFBIF]) is written with 1 by the application.

If the FIFO is empty, the update request is ignored without any notification.

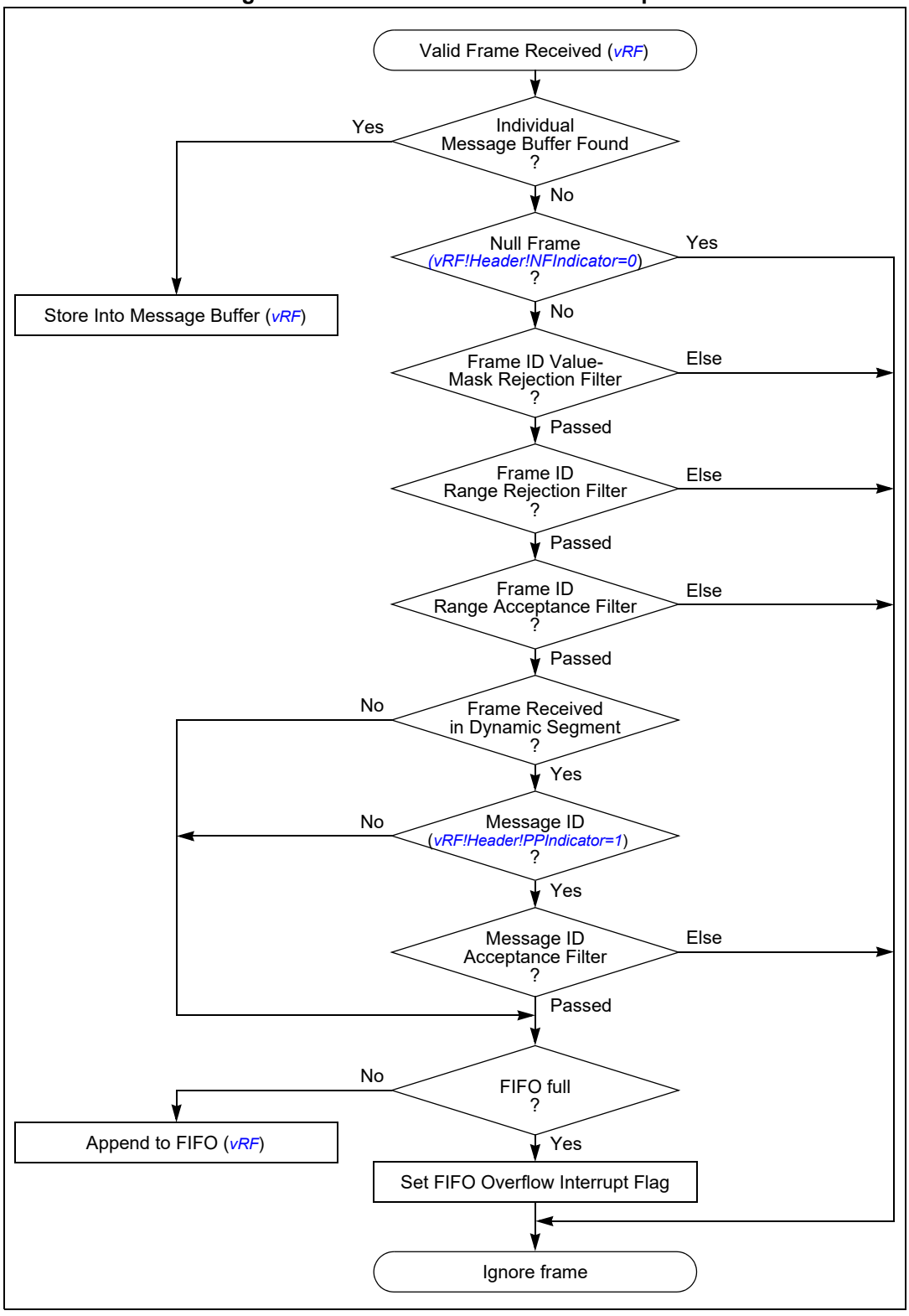
The read index in the Receive FIFO A Read Index Register (FR_RFARIR) (Receive FIFO B Read Index Register (FR_RFBRIR)) is incremented by 1 if the FIFO was not empty. If the read index reaches the top of the FIFO, it wraps around to the FIFO start index automatically.

57.6.9.9 FIFO filtering

The FIFO filtering is activated after all enabled individual receive message buffers have been searched without success for a message buffer to receive the current frame.

The CC provides four sets of FIFO filters. The FIFO filters are applied to valid non-null frames only. The FIFO will not receive invalid or null-frames. For each FIFO filter, the pass criteria is specified in the related section given below. Only frames that have passed all filters will be appended to the FIFO. The FIFO filter path is depicted in *Figure 1375*.

Figure 1375. Received frame FIFO filter path



A received frame passes the FIFO filtering if it has passed all four types of filter.

57.6.9.9.1 RX FIFO Frame ID Value-mask Rejection filter

The Frame ID Value-mask Rejection filter is a value-mask filter and is defined by the fields in the Receive FIFO Frame ID Rejection Filter Value Register (FR_RFFIDRFVR) and the Receive FIFO Frame ID Rejection Filter Mask Register (FR_RFFIDRFMR). Each received frame with a frame ID FID that does not match the value-mask filter value passes the filter (that is, it is not rejected).

Consequently, a received valid frame with the frame ID FID passes the RX FIFO Frame ID Value-mask Rejection filter if *Equation 46* is fulfilled.

Equation 46

$$FID \,\&\, FR_RFFIDRFMR[FIDRFMSK] \neq FR_RFFIDRFVR[FIDRFVAL] \,\&\, FR_RFFIDRFMR[FIDRFMSK]$$

The RX FIFO Frame ID Value-mask Rejection filter can be configured to pass all frames by using the following setting:

FR_RFFIDRFVR[FIDRFVAL] = 0x000 and FR_RFFIDRFMR[FIDRFMSK] = 0x7FF

Using the setting above, only the frame with frame ID 0 will be rejected, which is an invalid frame. All other frames will pass.

The RX FIFO Frame ID Value-mask Rejection filter can be configured to reject all frames by using the following setting:

FR_RFFIDRFMR[FIDRFMSK] = 0x000

Using the setting above, *Equation 46* can never be fulfilled (0!= 0) and thus all frames are rejected; no frame will pass. This is the reset value for the RX FIFO.

57.6.9.9.2 RX FIFO Frame ID Range Rejection filter

Each of the four RX FIFO Frame ID Range filters can be configured as a rejection filter. The filters are configured by the Receive FIFO Range Filter Configuration Register (FR_RFRFCFR) and controlled by the Receive FIFO Range Filter Control Register (FR_RFRFCTR). The RX FIFO frame ID range filters apply to all received valid frames. A received frame with the frame ID FID passes the RX FIFO Frame ID Range Rejection filters if either no rejection filter is enabled, or, if for all of the enabled RX FIFO Frame ID Range Rejection filters (that is, FR_RFRFCTR[FiMD] = 1 and FR_RFRFCTR[FiEN] = 1), *Equation 47* is fulfilled.

Equation 47

$$(FID < FR_RFRFCFR_{SEL}[SID_{IBD=0}]) \text{ or } (FR_RFRFCFR_{SEL}[SID_{IBD=1}] < FID)$$

Consequently, all frames with a frame ID that fulfills *Equation 48* for at least one of the enabled rejection filters will be rejected and thus not pass.

Equation 48

$$FR_RFRFCFR_{SEL}[SID_{IBD=0}] \leq FID \leq FR_RFRFCFR_{SEL}[SID_{IBD=1}]$$

57.6.9.9.3 RX FIFO Frame ID Range Acceptance filter

Each of the four RX FIFO Frame ID Range filters can be configured as an acceptance filter. The filters are configured by the Receive FIFO Range Filter Configuration Register (FR_RFRFCFR) and controlled by the Receive FIFO Range Filter Control Register (FR_RFRFCTR). The RX FIFO Frame ID range filters apply to all received valid frames. A received frame with the frame ID FID passes the RX FIFO Frame ID Range acceptance filters if either no acceptance filter is enabled, or, if for at least one of the enabled RX FIFO Frame ID Range acceptance filters (that is, FR_RFRFCTR[FiMD] = 0 and FR_RFRFCTR[FiEN] = 1), *Equation 49* is fulfilled.

Equation 49

$$FR_RFRFCFR_{SEL}[SID_{IBD=0}] \leq FID \leq FR_RFRFCFR_{SEL}[SID_{IBD=1}]$$

57.6.9.9.4 RX FIFO Message ID Acceptance filter

The RX FIFO Message ID Acceptance filter is a value-mask filter and is defined by the Receive FIFO Message ID Acceptance Filter Value Register (FR_RFMIDAFVR) and the Receive FIFO Message ID Acceptance Filter Mask Register (FR_RFMIDAFMR). This filter applies only to valid frames received in the dynamic segment with the payload preamble indicator bit PPI set to 1. All other frames will pass this filter.

A received valid frame in the dynamic segment with the payload preamble indicator bit PPI set to 1 and with the message ID MID (the first two bytes of the payload) will pass the RX FIFO Message ID Acceptance filter if *Equation 50* is fulfilled.

Equation 50

$$MID \,\&\, FR_RFMIDAFMR[MIDAFMSK] = FR_RFMIDAFMR[MIDAFVAL] \,\&\, FR_RFMIDAFMR[MIDAFMSK]$$

The RX FIFO Message ID Acceptance filter can be configured to accept all frames by setting

- FR_RFMIDAFMR[MIDAFMSK]:= 0x000

Using the setting above, *Equation 50* is always fulfilled and all frames will pass.

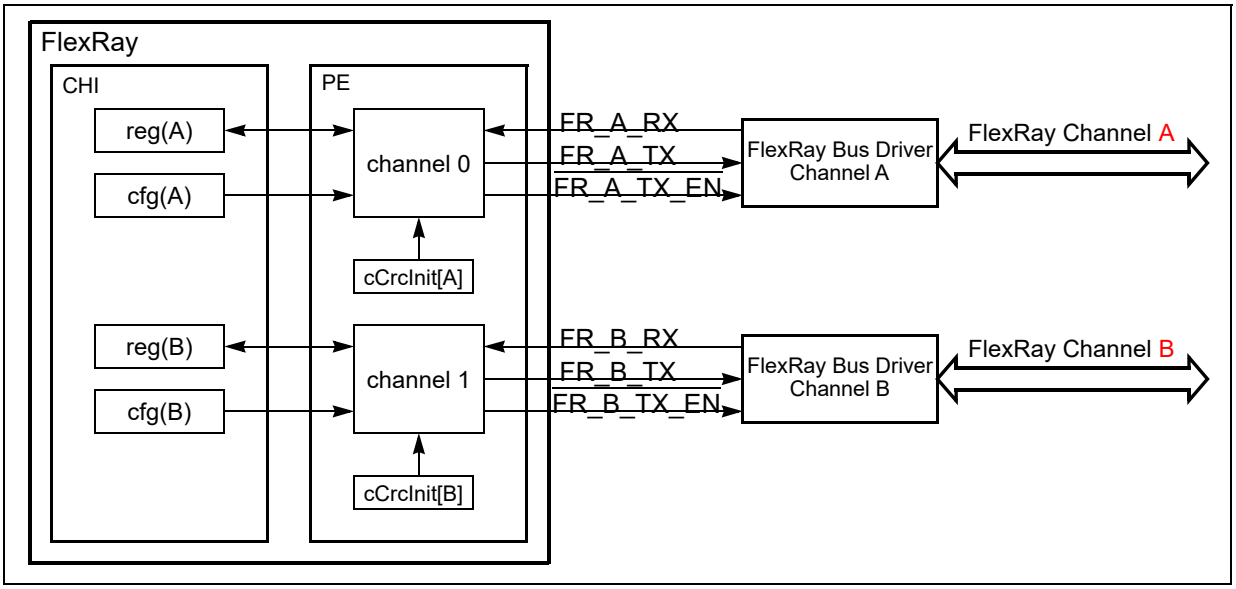
57.6.10 Channel device modes

This section describes the two FlexRay channel device modes that are supported by the CC.

57.6.10.1 Dual-Channel Device mode

In Dual-Channel Device mode, both FlexRay ports are connected to physical FlexRay bus lines. The FlexRay port consisting of FR_A_RX, FR_A_TX, and FR_A_TX_EN is connected to the physical bus channel A and the FlexRay port consisting of FR_B_RX, FR_B_TX, and FR_B_TX_EN is connected to the physical bus channel B. The dual-channel system is shown in *Figure 1376*.

Figure 1376. Dual-Channel Device mode



57.6.10.2 Single-channel device mode

Single-Channel Device mode supports devices that have only one FlexRay port available. This FlexRay port consists of the signals FR_A_RX, FR_A_TX, and FR_A_TX_EN and can be connected to either the physical bus channel A (shown in *Figure 1377*) or the physical bus channel B (shown in *Figure 1378*).

If the device is configured as a single-channel device by setting FR_MCR[SCM] to 1, only the internal channel A and FlexRay port A are used. Depending on the setting of FR_MCR[CHA] and FR_MCR[CHB], the internal channel A behaves either as a FlexRay channel A or FlexRay channel B. The bit FR_MCR[CHA] must be set, if FlexRay port A is connected to FlexRay channel A. The bit FR_MCR[CHB] must be set if FlexRay port B is connected to FlexRay channel B. The two FlexRay channels differ only in the initial value for the frame CRC *cCrcInit*. For a single-channel device, the application can access and configure only the registers related to internal channel A.

Figure 1377. Single-channel device mode (channel A)

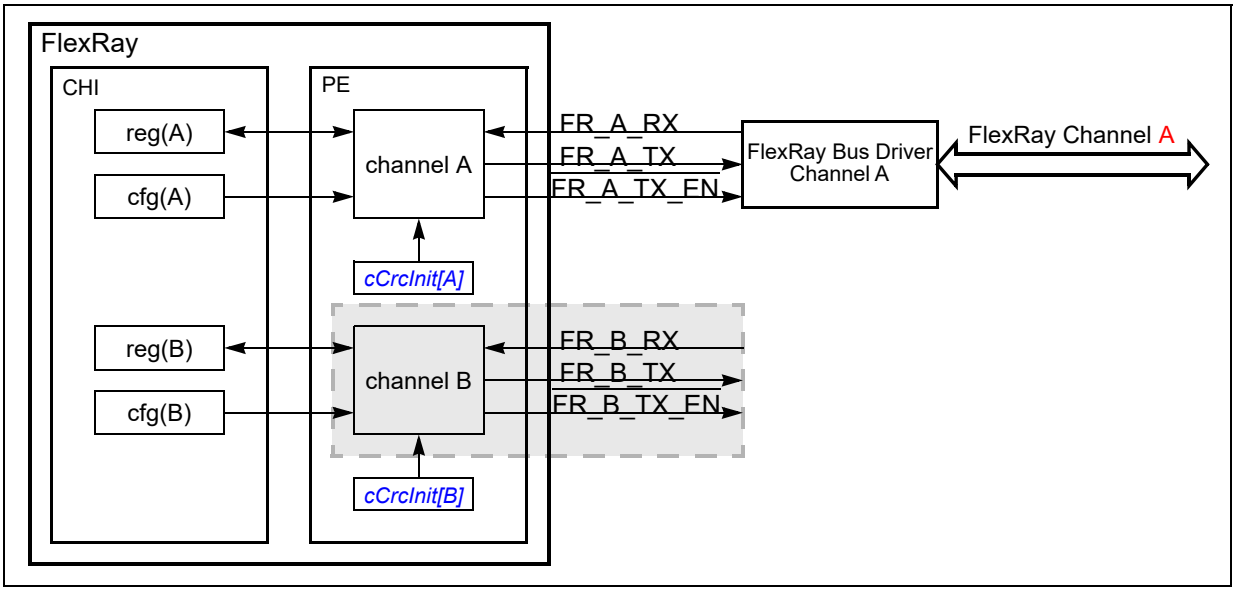
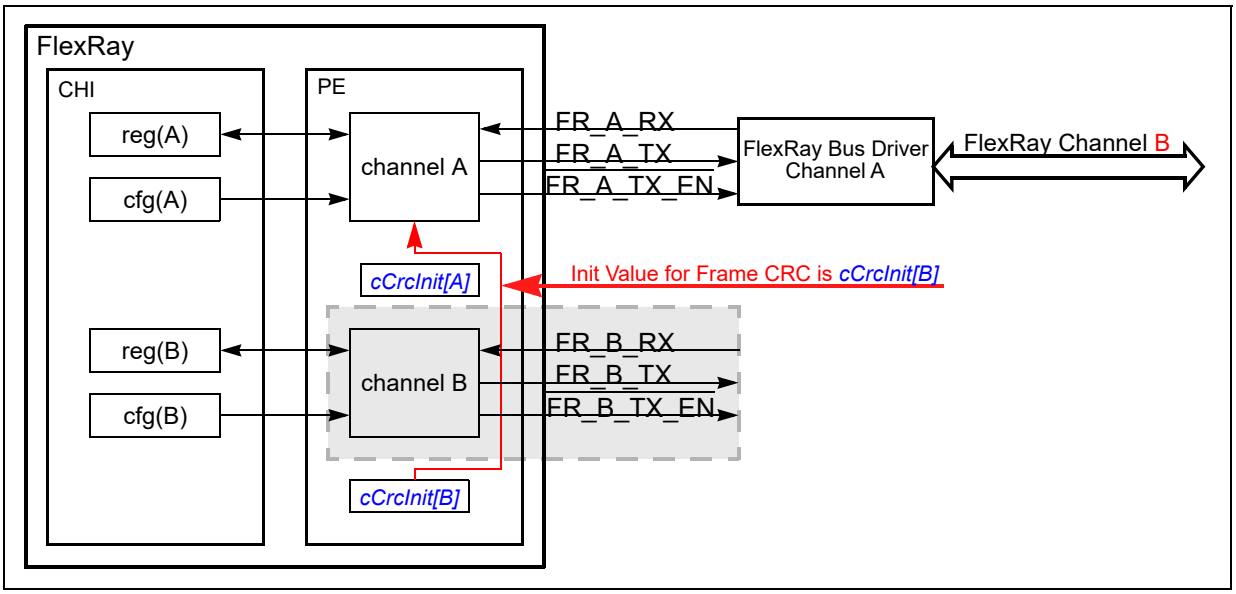


Figure 1378. Single-channel device mode (channel B)



57.6.11 External clock synchronization

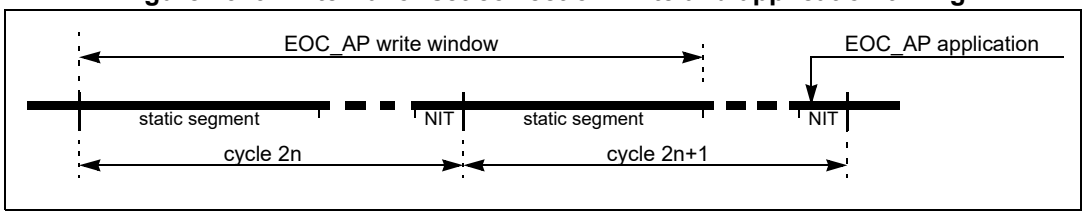
The application of the external rate and offset correction is triggered when the application writes to the EOC_AP and ERC_AP fields in the Protocol Operation Control Register (FR_POCR). The PE applies the external correction values in the next even-odd cycle pair as shown in *Figure 1379* and *Figure 1380*.

Note: The values provided in the EOC_AP and ERC_AP fields are the values that were written from the application most recently. If these values were already applied, they will not be applied in the current cycle pair again.

If the offset correction applied in the NIT of cycle 2n + 1 will be affected by the external offset correction, the EOC_AP field must be written to after the start of cycle 2n and before the end

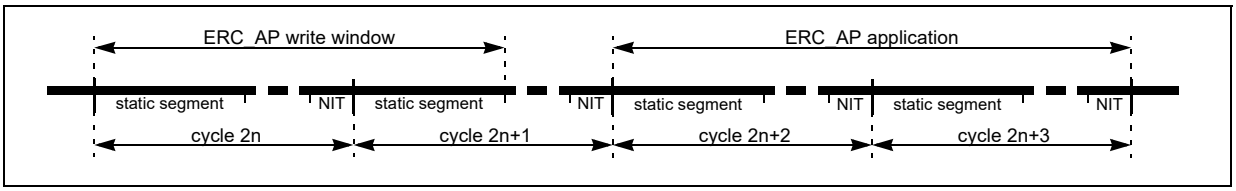
of the static segment of cycle 2n + 1. If this field is written to after the end of the static segment of cycle 2n + 1, it is not guaranteed that the external correction value is applied in cycle 2n + 1. If the value is not applied in cycle 2n + 1, then the value will be applied in the cycle 2n+3. Refer to *Figure 1379* for timing details.

Figure 1379. External offset correction write and application timing



If the rate correction for the cycle pair [2n + 2, 2n + 3] will be affected by the external offset correction, the ERC_AP field must be written to after the start of cycle 2n and before the end of the static segment start of cycle 2n + 1. If this field is written to after the end of the static segment of cycle 2n + 1, it is not guaranteed that the external correction value is applied in cycle pair [2n + 2, 2n + 3]. If the value is not applied for cycle pair [2n + 2, 2n + 3], then the value will be applied for cycle pair [2n+4, 2n+5]. Refer to *Figure 1380* for details.

Figure 1380. External rate correction write and application timing



57.6.12 Sync frame ID and sync frame deviation tables

The FlexRay protocol requires the provision of a snapshot of the synchronization frame ID tables for the even and odd communication cycle for both channels. The CC provides the means to write a copy of these internal tables into the FlexRay memory area and ensures application access to consistent tables by means of table locking. Once the application has locked the table successfully, the CC will not overwrite these tables and the application can read a consistent snapshot.

Note: Only synchronization frames that have passed the synchronization frame filters are considered for clock synchronization and appear in the sync frame tables.

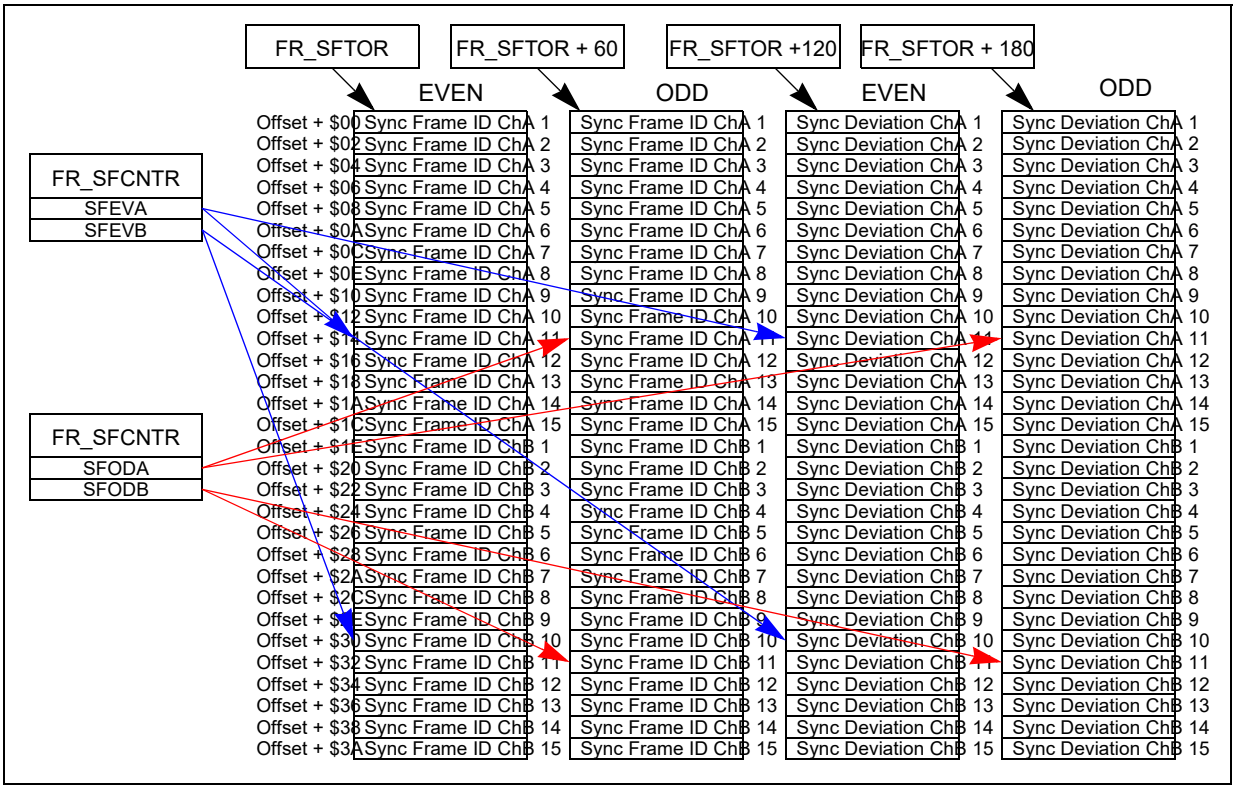
57.6.12.1 Sync frame ID table content

The sync frame ID table is a snapshot of the protocol-related variables *vsSyncIdListA* and *vsSyncIdListB* for each even and odd communication cycle. This table provides a list of the frame IDs of the synchronization frames received on the corresponding channel and cycle that are used for the clock synchronization.

57.6.12.2 Sync frame deviation table content

The sync frame deviation table is a snapshot of the protocol-related variable zsDev(id)(oe)(ch)!Value. Each sync frame deviation table entry provides the deviation value for the sync frame, with the frame ID presented in the corresponding entry in the sync frame ID table.

Figure 1381. Sync table memory layout



57.6.12.3 Sync frame ID and sync frame deviation table setup

The CC writes a copy of the internal synchronization frame ID and deviation tables into the FlexRay memory area if requested by the application. The application must provide the appropriate amount of FlexRay memory area for the tables. The memory layout of the tables is given in *Figure 1381*. Each table occupies 120 16-bit entries.

While the protocol is in *POC:config* state, the application must program the offsets for the tables into the Sync Frame Table Offset Register (FR_SFTOR).

57.6.12.4 Sync frame ID and sync frame deviation table generation

The application controls the generation process of the sync frame ID and sync frame deviation tables into the FlexRay memory area using the Sync Frame Table Configuration, Control, Status Register (FR_SFTCCSR). A summary of the copy modes is given in *Table 1422*.

Table 1422. Sync frame table generation modes

FR_SFTCCSR			Description
OPT	SDVEN	SIDEN	
0	0	0	No sync frame table copy
0	0	1	Sync frame ID tables will be copied continuously
0	1	0	Reserved

Table 1422. Sync frame table generation modes (continued)

FR_SFTCCSR			Description
OPT	SDVEN	SIDEN	
0	1	1	Sync frame ID tables and sync frame deviation tables will be copied continuously
1	0	0	No sync frame table copy
1	0	1	Sync frame ID tables for next even-odd-cycle pair will be copied
1	1	0	Reserved
1	1	1	Sync frame ID tables and sync frame deviation tables for next even-odd-cycle pair will be copied

The sync frame table generation process is described in the following for the even cycle. The same sequence applies to the odd cycle.

If the application has enabled the sync frame table generation by setting FR_SFTCCSR[SIDEN] to 1, the CC starts the update of the even-cycle-related tables after the start of the NIT of the next even cycle. The CC checks whether the application has locked the tables by reading the FR_SFTCCSR[ELKS] lock status bit. If this bit is set, the CC will not update the table in this cycle. If this bit is cleared, the CC locks this table and starts the table update. To indicate that these tables are currently updated and may contain inconsistent data, the CC clears the even table valid status bit FR_SFTCCSR[EVAL]. When all table entries related to the even cycle have been transferred into the FlexRay memory area, the CC sets the even table valid bit FR_SFTCCSR[EVAL] and the Even Cycle Table Written Interrupt Flag EVT_IF in the Protocol Interrupt Flag Register 1 (FR_PIFR1). If the interrupt enable flag EVT_IE is set, an interrupt request is generated.

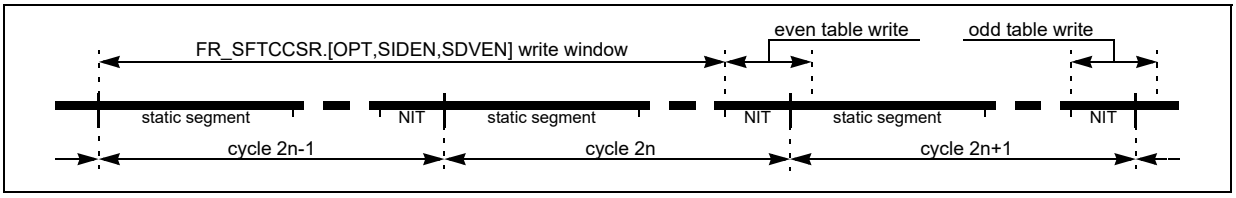
To read the generated tables, the application must lock the tables to prevent the CC from updating them. The locking is initiated by writing a 1 to the Even Table Lock trigger, FR_SFTCCSR[ELKT]. When the even table is not currently updated by the CC, the lock is granted and the even table lock status bit FR_SFTCCSR[ELKS] is set. This indicates that the application has successfully locked the even sync tables and the corresponding status information fields, SFRA and SFRB, in the Sync Frame Counter Register (FR_SFCNTR). The value in the FR_SFTCCSR[CYCNUM] field provides the number of the cycle that this table is related to.

The number of available table entries per channel is provided in the FR_SFCNTR[SFEVA] and FR_SFCNTR[SFEVB] fields. The application can now start to read the sync table data from the locations given in *Figure 1381*.

After reading all the data from the locked tables, the application must unlock the table by writing to the Even Table Lock trigger FR_SFTCCSR[ELKT] again. The even table lock status bit FR_SFTCCSR[ELKS] is reset immediately.

If the sync frame table generation is disabled, the table valid bits FR_SFTCCSR[EVAL] and FR_SFTCCSR[EVAL] are reset when the counter values in the Sync Frame Counter Register (FR_SFCNTR) are updated. This is done because the tables stored in the FlexRay memory area are no longer related to the values in the Sync Frame Counter Register (FR_SFCNTR).

Figure 1382. Sync frame table trigger and generation timing



57.6.12.5 Sync frame table access

The sync frame tables will be transferred into the FlexRay memory area during the table write windows shown in *Figure 1381*. During the table write, the application cannot lock the table that is currently written. If the application locks the table outside of the table write window, the lock is granted immediately.

57.6.12.5.1 Sync frame table locking and unlocking

The application locks the even/odd sync frame table by writing 1 to the lock trigger bit ELKT/OLKT in the Sync Frame Table Configuration, Control, Status Register (FR_SFTCCSR). If the affected table is not currently written to the FlexRay memory area, the lock is granted immediately, and the lock status bit ELKS/OLKS is set. If the affected table is currently written to the FlexRay memory area, the lock is not granted. In this case, the application must issue the lock request again until the lock is granted.

The application unlocks the even/odd sync frame table by writing 1 to the lock trigger bit ELKT/OLKT. The lock status bit ELKS/OLKS is cleared immediately.

57.6.13 MTS generation

The CC provides a flexible means to request the transmission of the Media Access Test Symbol (MTS) in the symbol window on channel A or channel B.

The application can configure the set of communication cycles that transmit the MTS over the FlexRay bus by programming the CYCCNTMSK and CYCCNTVAL fields in the MTS A Configuration Register (FR_MTSACFR) and MTS B Configuration Register (FR_MTSBCFR).

The application enables or disables the generation of the MTS on either channel by setting or clearing the MTE control bit in the MTS A Configuration Register (FR_MTSACFR) or MTS B Configuration Register (FR_MTSBCFR). If an MTS is to be transmitted in a certain communication cycle, the application must set the MTE control bit during the static segment of the preceding communication cycle.

The MTS is transmitted over channel A in the communication cycle with number CYCCNT if *Equation 51*, *Equation 52*, and *Equation 53* are fulfilled.

Equation 51

FR_PSR0[PROTSTATE] = *POC:normal active*

Equation 52

FR_MTSACRF[MTE] = 1

Equation 53

CYCCNT & FR_MTSACFR[CYCCNTMSK] = FR_MTSACFR[CYCCNTVAL] & FR_MTSACFR[CYCCNTMSK]

The MTS is transmitted over channel B in the communication cycle with number CYCCNT if *Equation 51*, *Equation 54* and *Equation 55* are fulfilled.

Equation 54

$$FR_MTSBCRF[MTE] = 1$$

Equation 55

$$CYCCNT \& FR_MTSBCFR[CYCCNTMSK] = FR_MTSBCFR[CYCCNTVAL] \& FR_MTSBCFR[CYCCNTMSK]$$

57.6.14 Key slot transmission

57.6.14.1 Key slot assignment

A key slot is assigned to the CC if the key_slot_id field in the Protocol Configuration Register 18 (FR_PCR18) is configured with a value greater than 0 and less than or equal to number_of_static_slots in Protocol Configuration Register 2 (FR_PCR2); otherwise no key slot is assigned.

57.6.14.2 Key slot transmission in *POC:startup*

If a key slot is assigned and the CC is in the *POC:startup* state, startup null frames will be transmitted as specified by FlexRay Communications System Protocol Specification, Version 2.1 Rev A.

57.6.14.3 Key slot transmission in *POC:normal active*

If a key slot is assigned and the CC is in *POC:normal active*, a frame of the type as shown in *Table 1423* is transmitted. If a transmit message buffer is configured for the key slot and a valid message is available, a message frame is transmitted (refer to *Section 57.6.6.2.5*). If no transmit message buffer is configured for the key slot or no valid message is available, a null frame is transmitted (refer to *Section 57.6.6.2.6*).

Table 1423. Key slot frame type

FR_PCR11[key_slot_used_for_sync]	FR_PCR11[key_slot_used_for_startup]	key slot frame type
0	0	Normal frame
0	1	Normal frame(1)
1	0	Sync frame
1	1	Startup frame

1. The frame transmitted has a semantically incorrect header and will be detected as an invalid frame at the receiver.

57.6.15 Sync frame filtering

Each received synchronization frame must pass the Sync Frame Acceptance filter and the Sync Frame Rejection filter before it is considered for clock synchronization. If the synchronization frame filtering is globally disabled (that is, the SFFE control bit in the Module Configuration Register (FR_MCR) is cleared), all received synchronization frames are considered for clock synchronization. If a received synchronization frame did not pass at

least one of the two filters, this frame is processed as a normal frame and is not considered for clock synchronization.

57.6.15.1 Sync frame acceptance filtering

The synchronization frame acceptance filter is implemented as a value-mask filter. The value is configured in the Sync Frame ID Acceptance Filter Value Register (FR_SFIDAFVR) and the mask is configured in the Sync Frame ID Acceptance Filter Mask Register (FR_SFIDAFMR). A received synchronization frame with the frame ID FID passes the sync frame acceptance filter, if *Equation 56* or *Equation 57* evaluates to true.

Equation 56

$$FR_MCR[SFFE] = 0$$

Equation 57

$$FID \,\&\, FR_SFIDAFMR[FMSK] = FR_SFIDAFVR[FVAL] \,\&\, FR_SFIDAFMR[FMSK]$$

Note: Sync frames are transmitted in the static segment only. Thus FID ≤ 1023.

57.6.15.2 Sync frame rejection filtering

The synchronization frame rejection filter is a comparator. The compare value is defined by the Sync Frame ID Rejection Filter Register (FR_SFIDRFR). A received synchronization frame with the frame ID FID passes the sync frame rejection filter if *Equation 58* or *Equation 59* evaluates to true.

Equation 58

$$FR_MCR[SFFE] = 0$$

Equation 59

$$FID \neq FR_SFIDRFR[SYNFRID]$$

Note: Sync frames are transmitted in the static segment only. Thus FID ≤ 1023.

57.6.16 Strobe signal support

The CC provides a number of strobe signals for observing internal protocol timing related signals in the protocol engine. The signals are listed and described in *Table 1308*.

57.6.16.1 Strobe signal assignment

Each of the strobe signals listed in *Table 1308* can be assigned to one of the four strobe ports using the Strobe Signal Control Register (FR_STBSCR). To assign multiple strobe signals, the application must write multiple times to the Strobe Signal Control Register (FR_STBSCR) with appropriate settings.

To read out the current settings for a strobe signal with number *n*, the application must execute the following sequence.

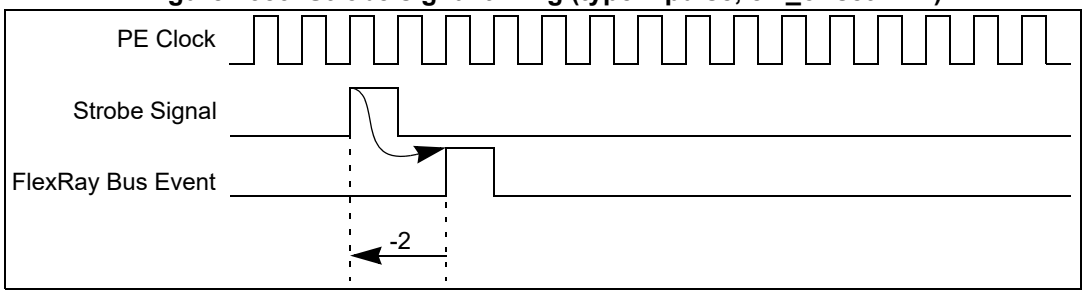
1. Write to FR_STBSCR with WMD = 1 and SEL = *n*. (Updates the SEL field only.)
2. Read STBCSR.
   The SEL field provides *n* and the ENB and STBPSEL fields provides the settings for signal *n*.

57.6.16.2 Strobe signal timing

This section provides detailed timing information of the strobe signals with respect to the protocol engine clock.

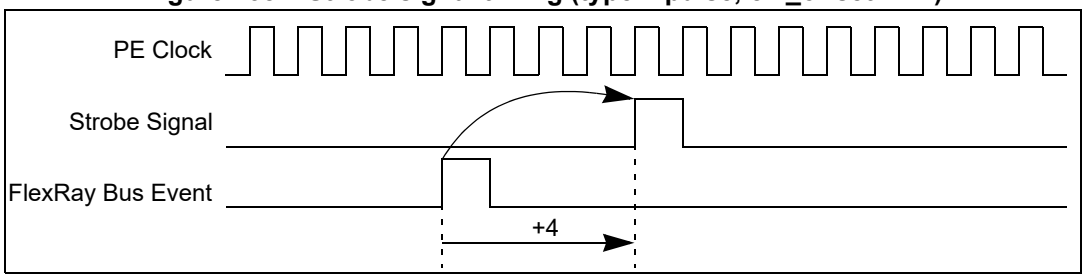
The strobe signals display internal PE signals. Due to the internal architecture of the PE, some signals are generated several PE clock cycles before the actual action is performed on the FlexRay bus. These signals are listed in *Table 1308* with a negative clock offset. An example waveform is given in *Figure 1383*.

Figure 1383. Strobe signal timing (type = pulse, clk_offset = –2)



Other signals refer to events that occurred on the FlexRay bus some cycles before the strobe signal is changed. These signals are listed in *Table 1308* with a positive clock offset. An example waveform is given in *Figure 1384*.

Figure 1384. Strobe signal timing (type = pulse, clk_offset = +4)



57.6.17 Timer support

The CC provides two timers that run on the FlexRay time base. Each timer generates a maskable interrupt when it reaches a configured point in time. Timer T1 is an absolute timer. Timer T2 can be configured to be an absolute or a relative timer. Both timers can be configured to be repetitive. In Non-repetitive mode, the timer stops if it expires. In Repetitive mode, the timer is restarted when it expires.

Both timers are active only when the protocol is in *POC:normal active* or *POC:normal passive* state. If the protocol is not in one of these modes, the timers are stopped. The application must restart the timers when the protocol has reached the *POC:normal active* or *POC:normal passive* state.

57.6.17.1 Absolute timer T1

Absolute timer T1 has the protocol cycle count and the macrotick count as the time base. The timer 1 interrupt flag TI1_IF in the Protocol Interrupt Flag Register 0 (FR_PIFR0) is set at the macrotick start event, if *Equation 60* and *Equation 61* are fulfilled.

Equation 60

$$CYCTR[CTCCNT] \& FR_TI1CYSR[T1_CYC_MSK] = FR_TI1CYSR[T1_CYC_VAL] \& FR_TI1CYSR[T1_CYC_MSK]$$

Equation 61

$$FR_MTCTR[MTCT] = FR_TI1MTOR[T1_MTOFFSET]$$

If the timer 1 interrupt enable bit TI1_IE in the Protocol Interrupt Enable Register 0 (FR_PIER0) is asserted, an interrupt request is generated.

The status bit T1ST is set when the timer is triggered, and is cleared when the timer expires and is non-repetitive. If the timer expires but is repetitive, the T1ST bit is not cleared and the timer is restarted immediately. The T1ST is cleared when the timer is stopped.

The corresponding interrupt condition (that is, when *Equation 60* and *Equation 61* are fulfilled) also leads to an event-out indication on an output port (fr_evt_tim1) based on the FR_PEOER[TIM1_EE].

57.6.17.2 Absolute/relative timer T2

Timer T2 can be configured to be an absolute or relative timer by setting the T2_CFG control bit in the Timer Configuration and Control Register (FR_TICCR). The status bit T2ST is set when the timer is triggered, and is cleared when the timer expires and is non-repetitive. If the timer expires but is repetitive, the T2ST bit is not cleared and the timer is restarted immediately. The T2ST is cleared when the timer is stopped.

57.6.17.2.1 Absolute timer T2

If timer T2 is configured as an absolute timer, it has the same functionality as timer T1 but the configuration from Timer 2 Configuration Register 0 (FR_TI2CR0) and Timer 2 Configuration Register 1 (FR_TI2CR1) is used. When timer T2 expires, the interrupt flag TI2_IF in the Protocol Interrupt Flag Register 0 (FR_PIFR0) is set. If the timer 1 interrupt enable bit TI1_IE in the Protocol Interrupt Enable Register 0 (FR_PIER0) is asserted, an interrupt request is generated.

The corresponding interrupt condition for T2 also leads to an event out indication on an output port (*fr_evt_tim2*) based on the FR_PEOER[TIM2_EE].

57.6.17.2.2 Relative Timer T2

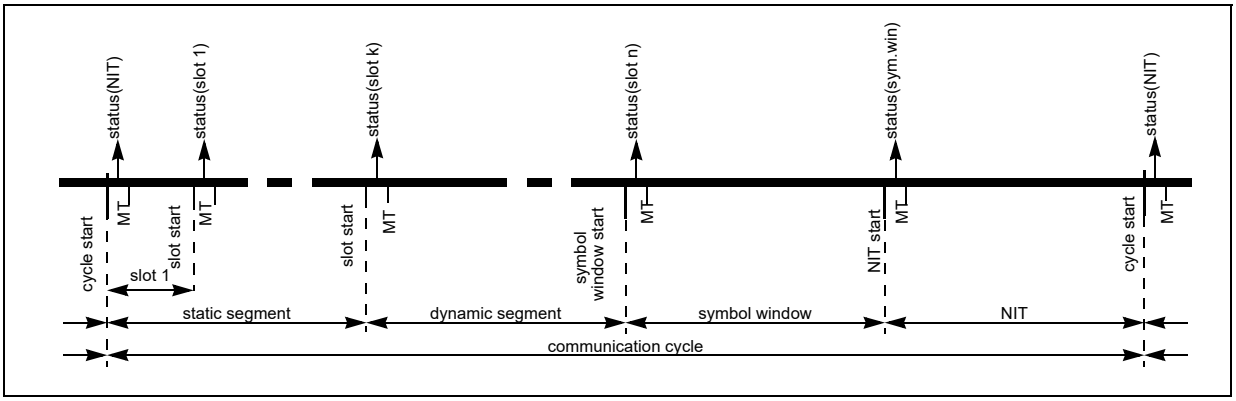
If timer T2 is configured as a relative timer, the interrupt flag TI2_IF in the Protocol Interrupt Flag Register 0 (FR_PIFR0) is set when the programmed amount of macroticks MT[31:0] – defined by Timer 2 Configuration Register 0 (FR_TI2CR0) and Timer 2 Configuration Register 1 (FR_TI2CR1) – has elapsed since the trigger or restart of timer T2. The relative timer is implemented as a down counter and expires when it has reached 0. At the macrotick start event, the value of MT[31:0] is checked and then decremented. Thus, if the timer is started with MT[31:0] == 0, it expires at the next macrotick start.The corresponding interrupt condition for T2 also leads to an event out indication on an output port (*fr_evt_tim2*) based on the FR_PEOER[TIM2_EE].

57.6.18 Slot status monitoring

The CC provides several means for slot status monitoring. All slot status monitors use the same slot status vector provided by the PE. The PE provides a slot status vector for each static slot, for each dynamic slot, for the symbol window, and for the NIT, on a per-channel base. The content of the slot status vector is described in *Table 1424*. The PE provides the

slot status vector within the first macrotick after the end of the related slot/window/NIT, as shown in *Figure 1385*.

Figure 1385. Slot status vector update



Note: The slot status for the NIT of cycle n is provided after the start of cycle n+1.

Table 1424. Slot status content

Slot	Status content

Static/dynamic slot	Slot-related status
	vSS!ValidFrame: valid frame received
	vSS!SyntaxError: syntax error occurred while receiving
	vSS!ContentError: content error occurred while receiving
	vSS!BViolation: boundary violation while receiving
	for slots in which the module transmits:
	vSS!TxConflict: reception ongoing while transmission starts
	for slots in which the module does not transmit:
	vSS!TxConflict: reception ongoing while transmission starts
	first valid - channel that has received the first valid frame
	Received-frame-related status
	extracted from
	a) header of valid frame, if vSS!ValidFrame = 1
	b) last received header, if vSS!ValidFrame = 0
	c) set to 0, if nothing was received
	vRF!Header!NFIndicator: null frame indicator (0 for null frame)
	vRF!Header!SuFIndicator: startup frame indicator
	vRF!Header!SyFIndicator: sync frame indicator

Symbol window	Window-related status *vSS!ValidFrame*: always 0 *vSS!ContentError*: content error occurred while receiving *vSS!SyntaxError*: syntax error occurred while receiving *vSS!BViolation*: boundary violation while receiving *vSS!TxConflict*: reception ongoing while transmission starts Received-symbol-related status *vSS!ValidMTS*: valid Media Test Access Symbol received Received-frame-related status see static/dynamic slot

Table 1424. Slot status content (continued)

Slot	Status content
NIT	NIT-related status *vSS!ValidFrame*: always 0 *vSS!ContentError*: content error occurred while receiving *vSS!SyntaxError*: syntax error occurred while receiving *vSS!BViolation*: boundary violation while receiving *vSS!TxConflict*: always 0 Received-frame-related status see static/dynamic slot

57.6.18.1 Channel status error counter registers

The two channel status error counter registers, Channel A Status Error Counter Register (FR_CASERCR) and Channel B Status Error Counter Register (FR_CBSERCR) are incremented by one if at least one of four slot status error bits, *vSS!SyntaxError*, *vSS!ContentError*, *vSS!BViolation*, or *vSS!TxConflict* is set to 1. The status vectors for all slots in the static and dynamic segment, in the symbol window, and in the NIT are taken into account. The counters wrap around after they have reached the maximum value.

57.6.18.2 Protocol status registers

The Protocol Status Register 2 (FR_PSR2) provides slot status information about the NIT and the symbol window. The Protocol Status Register 3 (FR_PSR3) provides aggregated slot status information.

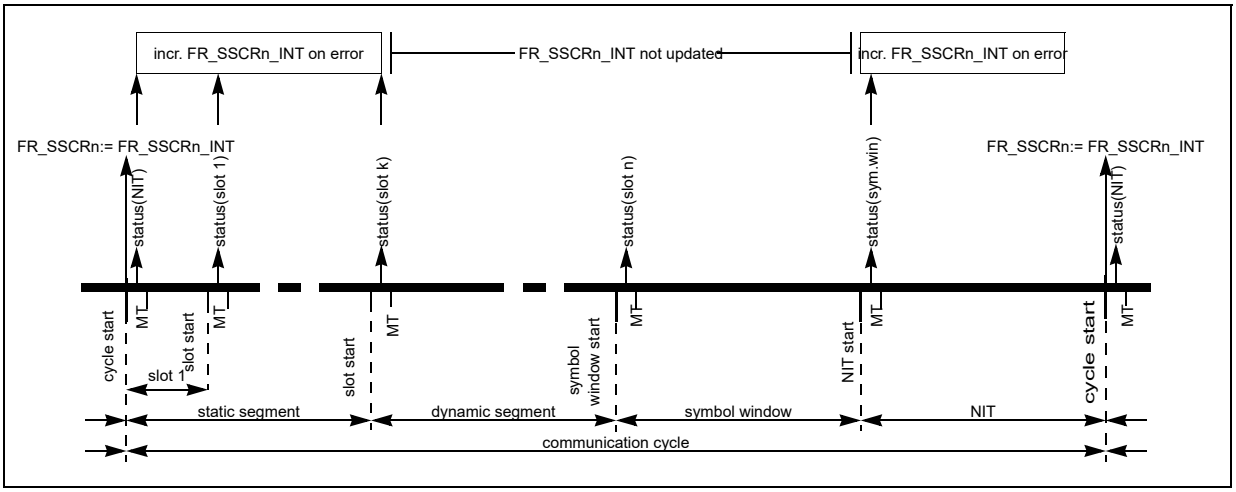
57.6.18.3 Slot status registers

The eight slot status registers, Slot Status Registers (FR_SSR0–FR_SSR7), can be used to observe the status of static slots, dynamic slots, the symbol window, or the NIT without individual message buffers. These registers provide all slot status-related and received frame/symbol-related status information, as given in *Table 1424*, except for the *first valid* indicator for non-transmission slots.

57.6.18.4 Slot status counter registers

The CC provides four slot status error counter registers, Slot Status Counter Registers (FR_SSCR0–FR_SSCR3). Each of these slot status counter registers is updated with the value of an internal slot status counter at the start of a communication cycle. The internal slot status counter is incremented if its increment condition, defined by the Slot Status Counter Condition Register (FR_SSCCR), matches the status vector provided by the PE. All static slots, the symbol window, and the NIT status are taken into account. *Dynamic* slots are *excluded*. The internal slot status counting and update timing is shown in *Figure 1386*.

Figure 1386. Slot status counting and FR_SSCR*n* update



The PE provides the status of the NIT in the first slot of the next cycle. Due to these facts, the FR_SSCR*n* register reflects, in cycle n, the status of the NIT of cycle n – 2, and the status of all static slots and the symbol window of cycle n – 1.

The increment condition for each slot status counter consists of two parts: the frame-related condition part and the slot-related condition part. The internal slot status counter FR_SSCR*n*_INT is incremented if at least one of the following conditions is fulfilled:

1. Frame-related condition:

- (FR_SSCCR*n*[VFR] | FR_SSCCR*n*[SYF] | FR_SSCCR*n*[NUF] | FR_SSCCR*n*[SUF]) // count on frame condition
  = 1;

and

- ((~FR_SSCCR*n*[VFR] | vSS!ValidFrame) & // valid frame restriction
  (~FR_SSCCR*n*[SYF] | vRF!Header!SyFIndicator) & // sync frame indicator restriction
  (~FR_SSCCR*n*[NUF] | ~vRF!Header!NFIndicator) & // null frame indicator restriction
  (~FR_SSCCR*n*[SUF] | vRF!Header!SuFIndicator)) // startup frame indicator restriction
  = 1;

Note: The indicator bits SYF, NUF, and SUF are valid only when a valid frame was received. Thus, the VFR must always be set whenever the count-on-frame condition is used.

2. Slot-related condition:

- ((FR_SSCCR*n*[STATUSMASK[3]] & vSS!ContentError) | // increment on content error
  (FR_SSCCR*n*[STATUSMASK[2]] & vSS!SyntaxError) | // increment on syntax error
  (FR_SSCCR*n*[STATUSMASK[1]] & vSS!BViolation) | // increment on boundary violation
  (FR_SSCCR*n*[STATUSMASK[0]] & vSS!TxConflict)) // increment on transmission conflict
  = 1;

If the slot status counter is in Single-cycle mode (that is, FR_SSCCR*n*[MCY] = 0), the internal slot status counter FR_SSCR*n*_INT is reset at each cycle start. If the slot status counter is in Multicycle mode (that is, FR_SSCCR*n*[MCY] = 1), the counter is not reset and incremented until the maximum value is reached.

57.6.18.5 Message buffer slot status field

Each individual message buffer and each FIFO message buffer provides a slot status field, which provides the information shown in *Table 1424* for the static/dynamic slot. The update conditions for the slot status field depend on the message buffer type. Refer to the Message Buffer Update sections in *Section 57.6.3.4.1*.

57.6.19 System bus access

This section provides a description of the system bus access failures and the related CC behavior. System bus access failures may occur when the CC transfers data to or from the FlexRay memory area.

The system bus access failure types are described in *Section 57.6.19.1*.

The behavior of the CC after the occurrence of a system bus access failure is described in *Section 57.6.19.2*.

57.6.19.1 System bus access failure types

This section describes the two types of system bus access failures.

57.6.19.1.1 System bus illegal address access

A system bus illegal address access is detected when the CC has used an illegal or invalid address to access the FlexRay system memory area. There are three conditions that are treated as a system bus illegal address access:

- The system bus subsystem detects a CC access to an illegal system memory address.
- The CC detects the usage of an data field offset with the value of 0.
- The CC detects a memory error while reading a data field offset from the CHI LRAM memory (refer to *Section 57.6.24.3.1*).

If a system bus illegal address access is detected, the CC sets the ILSA_EF flag in the CHI Error Flag Register (FR_CHIERFR).

57.6.19.1.2 System bus access timeout

A system bus access timeout is detected if an access to the FlexRay memory area is not finished in time. The timeout value is derived from the SYMATOR[TIMEOUT] setting (refer to *Section 57.7.1.1*).

If a system bus access timeout is detected, the CC sets the SBCF_EF flag in the CHI Error Flag Register (FR_CHIERFR).

57.6.19.2 System bus access failure response

This section describes the two types of behavior of the CC after the occurrence of a system bus access failure. The actual behavior is defined by the SBFF bit in the Module Configuration Register (FR_MCR).

57.6.19.2.1 Continue after system bus access failure

If FR_MCR[SBFF] is 0, the CC will continue its operation after the occurrence of the system bus access failure, but it will not generate any system bus accesses until the start of the next communication cycle. Since no data is read from or written to the FlexRay memory

area, no messages are received or transmitted. Consequently, none of the individual message buffers or receive FIFOs will be updated until the next communication cycle starts.

If a frame is under transmission when the system bus failure occurs, a correct frame is generated in which the remaining header and frame data are replaced by all zeros. Depending on the point in time, this can affect the PPI bit, the header CRC, the payload length in the case of a dynamic slot, and the payload data. Starting from the next slot in the current cycle, no frames will be transmitted or received, except for the key slot; a sync or startup null-frame is transmitted if the key slot is assigned.

If a frame is received when the system bus failure occurs, the reception is aborted and the related receive message buffer is not updated.

Normal operation is resumed after the start of next communication cycle.

57.6.19.2.2 Freeze after system bus access failure

If FR_MCR[SBFF] is set to 1, the CC will go into Freeze mode immediately after the occurrence of one of the system bus access failures.

57.6.20 Interrupt support

The CC provides 172 individual interrupt sources and five combined interrupt sources.

57.6.20.1 Individual interrupt sources

57.6.20.1.1 Message buffer interrupts

The CC provides 128 message buffer interrupt sources.

Each individual message buffer provides an interrupt flag FR_MBCCSR*n*[MBIF] and an interrupt enable bit FR_MBCCSR*n*[MBIE]. The CC sets the interrupt flag when the slot status of the message buffer was updated. If the interrupt enable bit is asserted, an interrupt request is generated.

57.6.20.1.2 FIFO interrupts

The CC provides two FIFO interrupt sources.

Each of the two FIFOs provides a Receive FIFO Almost Full Interrupt Flag. The CC sets the Receive FIFO Almost Full Interrupt Flags (FR_GIFER[FAFBIF], FR_GIFER[FAFAIF]) in the Global Interrupt Flag and Enable Register (FR_GIFER) if the corresponding receive FIFO fill level exceeds the defined watermark.

57.6.20.1.2.1 Wakeup interrupt

The CC provides one interrupt source related to the wakeup.

The CC sets the Wakeup Interrupt Flag FR_GIFER[WUPIF] when it has received a wakeup symbol on the FlexRay bus. The CC generates an interrupt request if the interrupt enable bit FR_GIFER[WUPIE] is asserted.

57.6.20.1.2.2 Protocol interrupts

The CC provides 25 interrupt sources for protocol-related events. For details, refer to the Protocol Interrupt Flag Register 0 (FR_PIFR0) and Protocol Interrupt Flag Register 1 (FR_PIFR1). Each interrupt source has its own interrupt enable bit.

57.6.20.1.2.3 CHI interrupts

The CC provides 16 interrupt sources for CHI-related error events. For details, refer to the CHI Error Flag Register (FR_CHIERFR). There is one common interrupt enable bit FR_GIFER[CHIE] for all CHI error interrupt sources.

57.6.20.2 Combined interrupt sources

Each combined interrupt source generates an interrupt request only when at least one of the interrupt sources that is combined generates an interrupt request.

57.6.20.2.1 Receive message buffer interrupt

The receive message buffer interrupt request is generated when at least one of the individual receive message buffers generates an interrupt request MBXIRQ[n] and the interrupt enable bit FR_GIFER[RBIE] is set.

57.6.20.2.2 Transmit message buffer interrupt

The transmit message buffer interrupt request is generated when at least one of the individual transmit message buffers generates an interrupt request MBXIRQ[n] and the interrupt enable bit FR_GIFER[TBIE] is asserted.

57.6.20.2.3 Protocol interrupt

The protocol interrupt request is generated when at least one of the individual protocol interrupt sources generates an interrupt request and the interrupt enable bit FR_GIFER[PRIE] is set.

57.6.20.2.4 CHI interrupt

The CHI interrupt request is generated when at least one of the individual chi error interrupt sources generates an interrupt request and the interrupt enable bit FR_GIFER[CHIE] is set.

57.6.20.2.5 Module interrupt

The module interrupt request is generated if at least one of the combined interrupt sources generates an interrupt request and the interrupt enable bit FR_GIFER[MIE] is set.

Figure 1387. Scheme of FR_GIFER interrupt signal generation

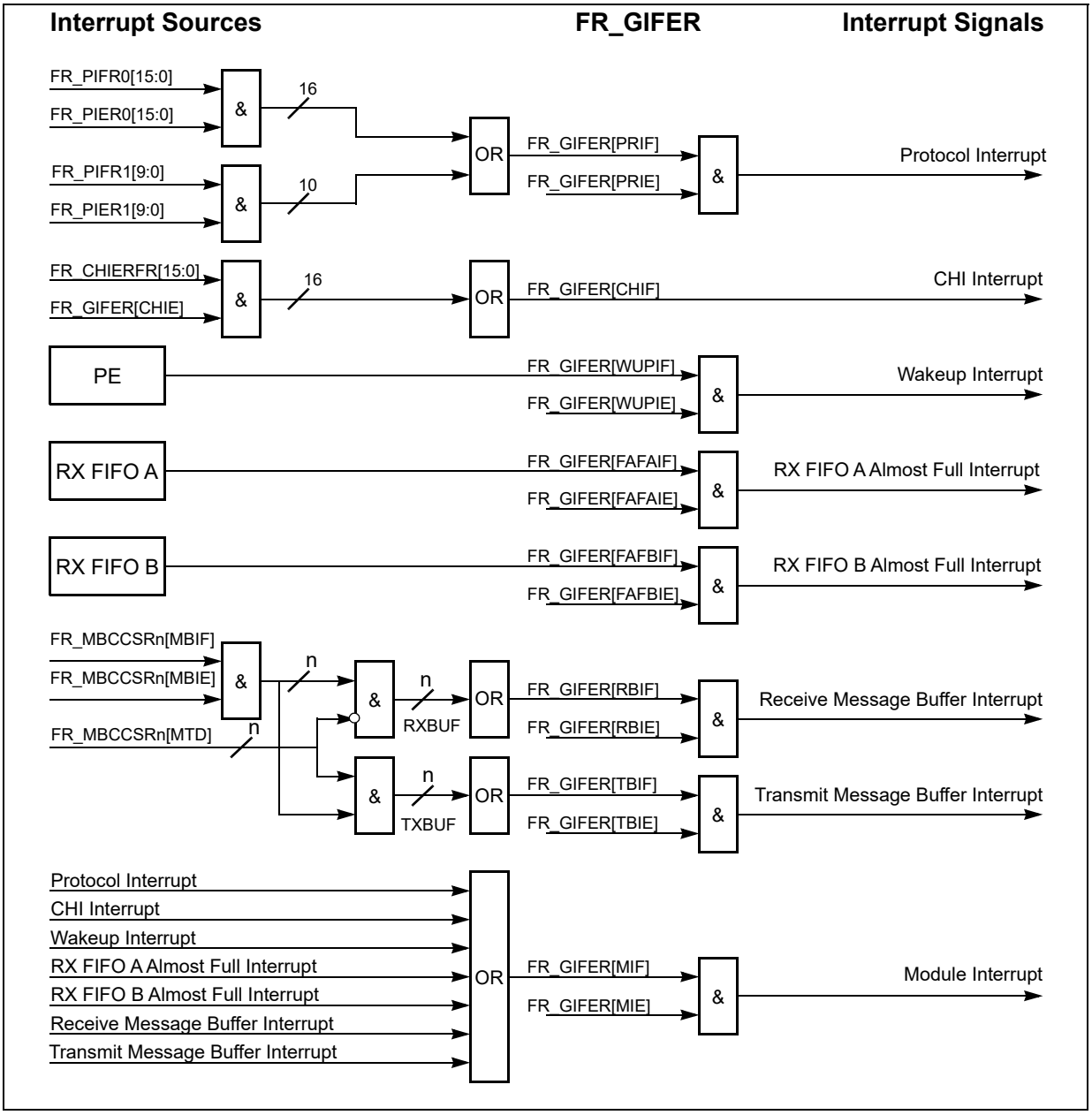


Figure 1388. FR_EEIFER interrupt signal generation scheme

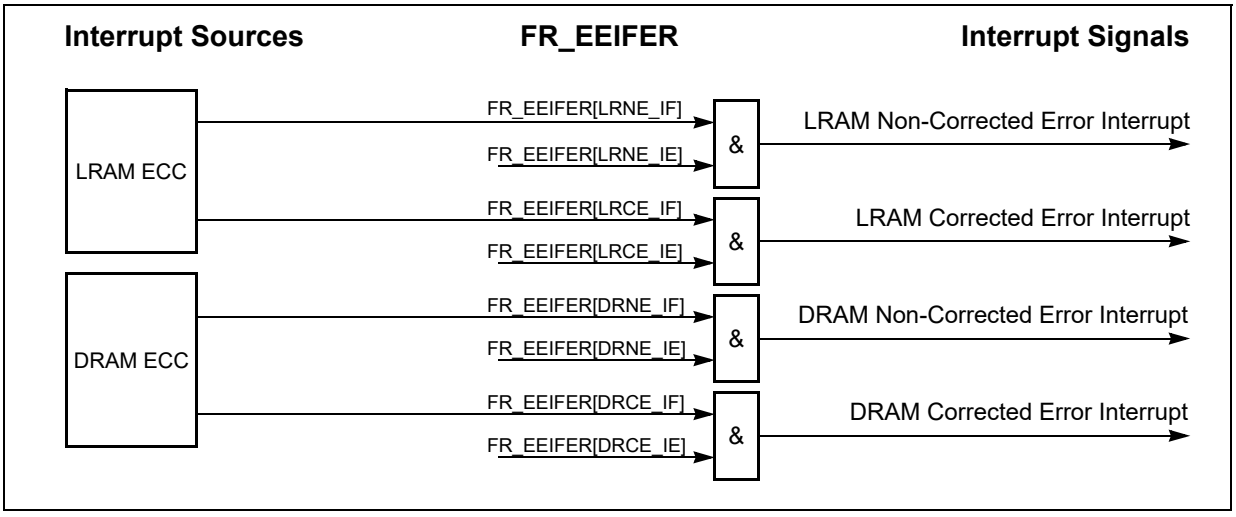
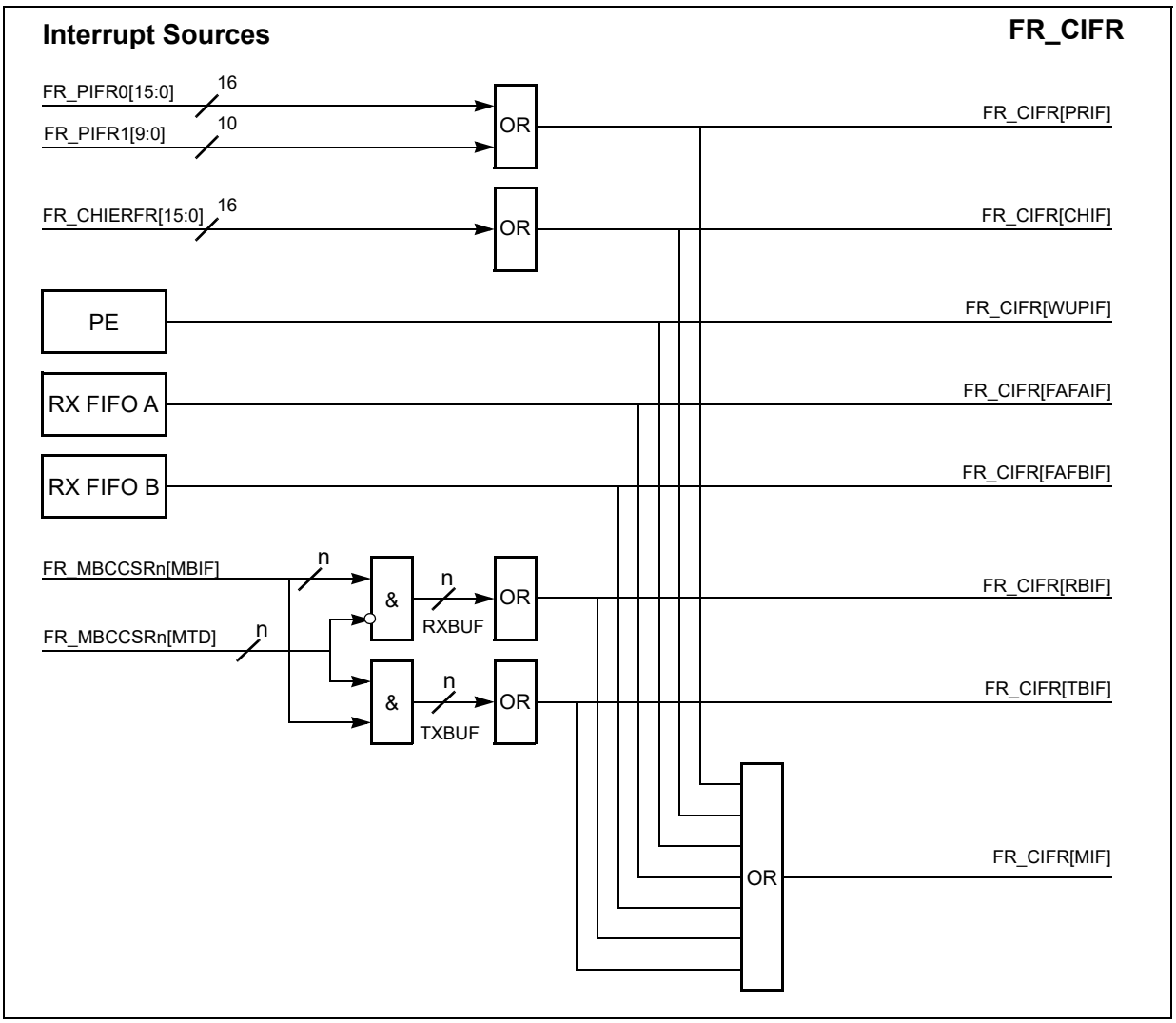


Figure 1389. FR_CIFR flag generation scheme



57.6.21 Lower bit rate support

The CC supports a number of lower bit rates on the FlexRay bus channels. The lower bit rates are implemented by modifying the duration of the microtick *pdMicrotick*, the number of samples per microtick *pSamplesPerMicrotick*, the number of samples per bit *cSamplesPerBit*, and the strobe offset *cStrobeOffset.* The application configures the FlexRay channel bit rate by setting the BITRATE field in the Module Configuration Register (FR_MCR). The protocol values are set internally. The available bit rates, the related BITRATE field configuration settings, and the related protocol parameter values are shown in *Table 1425*.

Table 1425. FlexRay channel bit rate control

FlexRay channel bit rate [Mbit/s]	FR_MCR[BITRATE]	*pdMicrotick* [ns]	*gdSampleClockPeriod* [ns]	pSamplesPerMicrotick	cSamplesPerBit	cStrobeOffset
10.0	000	25.0	12.5	2	8	5
8.0	011	25.0	12.5	2	10	6
5.0	001	25.0	25.0	1	8	5
2.5	010	50.0	50.0	1	8	5

Note: The bit rate of 8 Mbit/s is not defined by the FlexRay Communications System Protocol Specification, Version 2.1 Rev A.

57.6.22 PE data memory (PE DRAM)

The PE data memory (PE DRAM) is 128 word, 16-bit wide memory with byte access that contains the program data of the PE internal CPU. The PE DRAM is divided into two banks of 8 bits each. The memory data [7:0] is assigned to BANK0 and the memory data [15:8] is assigned to BANK1.

Table 1426. PE DRAM layout

ADDR	BANK1	BANK0
0x00	byte1	byte0
0x01	byte3	byte2
…		
0x7F	byte255	byte254

The FlexRay module provides means to access the PE DRAM from the application. The PE DRAM application access is initiated and controlled via *Section 57.5.2.10: PE DRAM Access Register (FR_PEDRAR)* and *Section 57.5.2.10: PE DRAM Access Register (FR_PEDRAR)*. This functionality is used to check the memory error detection.

57.6.22.1 PE DRAM read access

A read access from the PE DRAM can be initiated in any protocol state. The following sequence describes a read access from the PE DRAM address 0x70.

1. FR_PEDRAR:= 0x50E0;
   // INST=0x5; ADDR=070
2. Wait until FR_PEDRAR[DAD] == 1;
   // wait for end of PE DRAM access
3. val = FR_PEDRDR[DATA];
   // read PE DRAM data

The read access is handled by the PE internal CPU with the lowest execution priority. This may cause a response delay with a maximum of 1000 PE clock cycle (25 µs).

57.6.22.2 PE DRAM write access

A write access into the PE DRAM can be initiated in any protocol state. The following sequence describes a write access to the PE DRAM address 0x70.

1. FR_PEDRDR:= DATA;
   // write value to be written into data register
2. FR_PEDRAR:= 0x30E0;
   // INST=0x3; ADDR=0x70
3. wait until FR_PEDRAR[DAD] == 1;
   // wait for end of PE DRAM access
4. val = FR_PEDRDR[DATA];
   // read back PE DRAM data

The write access is handled by the PE internal CPU with the lowest execution priority. This may cause a response delay with a maximum of 1000 PE clock cycle (25 µs).

If the conditions given in *Section 57.6.22.3: PE DRAM write access limitations* are fulfilled, the data provided in PE DRAM Access Register (FR_PEDRAR) is written into the PE DRAM, read back in the next clock cycle, and stored into the PE DRAM Access Register (FR_PEDRAR). Otherwise, data is not written into the PE DRAM and 0x0000 is stored into the PE DRAM Access Register (FR_PEDRAR).

57.6.22.3 PE DRAM write access limitations

The PE DRAM is used by the protocol engine if the module is not in *POC:default config* state. The only address not used by the protocol engine is 0x70. To prevent the corruption of protocol engine data the following PE DRAM write access limitations apply for application writes.

1. When the module is in *POC:default config* state, all PE DRAM addresses are writable.
2. When the module is not in *POC:default config* state, only PE DRAM address 0x70 is writable.

57.6.23 CHI lookup-table memory (CHI LRAM)

The CHI lookup-table memory (CHI LRAM) is a CHI-internal memory that contains the message buffer configuration data and the data field offsets for the physical message buffers. The configuration data for two message buffers or six data field offsets are contained in one memory row. The CHI LRAM is divided into six memory banks.

Table 1427. CHI LRAM layout

ADR	BANK5	BANK4	BANK3	BANK2	BANK1	BANK0
0x00	FR_MBIDXR1	FR_MBFIDR1	FR_MBCCFR1	FR_MBIDXR0	FR_MBFIDR0	FR_MBCCFR0
0x01	FR_MBIDXR3	FR_MBFIDR3	FR_MBCCFR3	FR_MBIDXR2	FR_MBFIDR2	FR_MBCCFR2
…						
0x3F	FR_MBIDXR127	FR_MBFIDR127	FR_MBCCFR127	FR_MBIDXR126	FR_MBFIDR126	FR_MBCCFR126
0x40	FR_MBDOR5	FR_MBDOR4	FR_MBDOR3	FR_MBDOR2	FR_MBDOR1	FR_MBDOR0
…						
0x55	FR_MBDOR131	FR_MBDOR130	FR_MBDOR129	FR_MBDOR128	FR_MBDOR127	FR_MBDOR126
0x56	FR_LEETR5	FR_LEETR4	FR_LEETR3	FR_LEETR2	FR_LEETR1	FR_LEETR0

57.6.23.1 CHI LRAM read and write access

The CHI LRAM is accessed by the application via regular register read and write accesses.

57.6.24 Memory content error detection

The FlexRay module provides integrated memory content error detection for both the CHI LRAM and PE DRAM, and memory content error correction for the PE DRAM. The memory error detection for the CHI LRAM uses a standard Hamming code with a Hamming distance of 3 and detects all single-bit and double-bit errors (SEDDED). The memory error detection and correction for the PE DRAM uses an enhanced Hamming code with a Hamming distance of 4 and detects and corrects all single-bit errors and detects all double-bit errors (SECDED).

This section describes the reporting of the occurrence of memory content errors, the reaction of the module on the occurrence, and how the application can inject memory errors in order to trigger the report and response behavior.

57.6.24.1 Memory error types

A memory error is the distortion of one or more bits read out of the memory. The reading of the values of all zeros and all ones is considered as a special case. The FlexRay module detects and indicates the memory errors as shown in *Table 1428*. The entries on the top have higher priority.

Each memory read access reads out *all* banks of the addressed row and runs error detection on *all* banks, even in the case that the application has triggered a read from only one bank. This may lead to the reporting of an memory error if at least one bank contains a memory error, even if an error-free bank has been read.

Table 1428. Detected memory error types

Memory	Priority	Memory data	Indication
CHI LRAM	0 (highest)	All zeros	Non-corrected error
PE DRAM			Non-corrected error
CHI LRAM		All ones	Non-corrected error
PE DRAM			
CHI LRAM	1 (lowest)	One bit flipped	Non-corrected error
PE DRAM			Corrected error
CHI LRAM		Two bits flipped	Non-corrected error
PE DRAM			
CHI LRAM		Three or more bits flipped	One out of {no error, non-corrected error}, defined by coding given in *Section 57.6.24.2.3* and *Section 57.6.24.2.4*
PE DRAM			One out of {no error, corrected error, non-corrected error}, defined by coding given in *Section 57.6.24.2.1* and *Section 57.6.24.2.2*

57.6.24.2 Memory error reporting

The memory error reporting is enabled only if the ECC functionality enable bit ECCE in the Module Configuration Register (FR_MCR) is set.

For each of the two memories, there are two sets of internal registers to store the detection of one corrected and one non-corrected memory error.

If a memory error is detected, the module checks whether the related error interrupt flag in the ECC Error Interrupt Flag and Enable Register (FR_EEIFER) is set.

- If the error interrupt flag is set, the related internal error reporting register is not updated and the related error overflow flag is set to 1 to indicate a loss of error condition.
- If the error interrupt flag is not set, the internal reporting register is updated and the error interrupt flag is set to 1. If two or more memory errors of the same type are detected, the error for the bank with the lower bank number will be reported, and the error overflow flag will be set to 1.

If a memory error is detected for at least two banks of one memory, the related error overflow flag is set to 1 to indicate a loss of error condition.

In addition to the above, the error indications (corrected/non-corrected) along with the failing address are sent out from the module for external error logging and corrective actions by the software.

57.6.24.2.1 PE DRAM checkbits

The coding of the checkbits reported in ECC Error Report Code Register (FR_EERCR) for PE DRAM memory errors is shown in *Table 1428*. This table shows the implemented enhanced Hamming code. If the error injection was applied to distort the checkbits, then the distorted checkbits are reported.

Table 1429. PE DRAM checkbits coding

CODE	CODE				DATA							
	3	2	1	0	7	6	5	4	3	2	1	0
4(1)	X	X	X	X	X	X	X	X	X	X	X	X
3(2)	—	—	—	—	X	X	X	X	—	—	—	—
2	—	—	—	—	X	—	—	—	X	X	X	—
1	—	—	—	—	—	X	X	—	X	X	—	X
0	—	—	—	—	—	X	—	X	X	—	X	X

1. The checkbit CODE[4] is set to 1 if and only if there is an even number of 1's in columns with X.
2. The checkbits CODE[3]… CODE[0] are set to 1 if and only if there is an odd number of 1's in all columns with X.

Note: This coding of the checkbit ensures that neither 0x000 nor 0xFFF are valid code words written into the memory.

57.6.24.2.2 PE DRAM syndrome

The coding of the syndrome reported in the ECC Error Report Code Register (FR_EERCR) for PE DRAM memory errors is shown in *Table 1430*.

Table 1430. FR_EERCR[CODE] PE DRAM syndrome coding

FR_EERCR[CODE]		Description
[4]	[3:0]	
0x1	0x0	No Error (never appears in error report registers)
0x0	0x0	If data == 0: Non Corrected Error (Dedicated Handling of All Zero Code Word) If data!= 0: Corrected Error (Parity Bit 4)
0x0	0x1	Corrected error (parity bit 0)
0x0	0x2	Corrected error (parity bit 1)
0x0	0x3	Corrected error (data bit 0)
0x0	0x4	Corrected error (parity bit 2)
0x0	0x5	Corrected error (data bit 1)
0x0	0x6	Corrected error (data bit 2)
0x0	0x7	Corrected error (data bit 3)
0x0	0x8	Corrected error (parity bit 3)
0x0	0x9	Corrected error (data bit 4)
0x0	0xA	Corrected error (data bit 5)
0x0	0xB	Corrected error (data bit 6)
0x0	0xC	Corrected error (data bit 7)
0x0	0xD–0xF	Non-corrected error
0x1	0x1–0xF	Non-corrected error

57.6.24.2.3 CHI LRAM checkbits

The coding of the checkbits reported in ECC Error Report Code Register (FR_EERCR) for CHI LRAM memory errors is shown in *Table 1431*. This table shows the implemented Hamming code. If the error injection was applied to distort the checkbits, then the distorted checkbits are reported.

Table 1431. CHI LRAM checkbits coding

CODE(1)	DATA															
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
4	X	X	X	X	X	—	—	—	—	—	—	—	—	—	—	—
3	—	—	—	—	—	X	X	X	X	X	X	X	—	—	—	—
2	X	X	—	—	—	X	X	X	X	—	—	—	X	X	X	—
1	—	—	X	X	—	X	X	—	—	X	X	—	X	X	—	X
0	X	—	X	—	X	X	—	X	—	X	—	X	X	—	X	X

1. The checkbit CODE[n] is set to 1 if and only if there is an odd number of 1's in all columns with X.

57.6.24.2.4 CHI LRAM syndrome

The coding of the syndrome reported in ECC Error Report Code Register (FR_EERCR) for CHI LRAM memory errors is shown in *Table 1432*.

Table 1432. FR_EERCR[CODE] CHI LRAM syndrome coding

FR_EERCR[CODE]	Description
0x00	No Error (never appears in error report registers)
0x01–0x1F	Non-corrected error

57.6.24.3 Memory error response

The memory error response is enabled only when the ECC functionality enable bit ECCE in the Module Configuration Register (FR_MCR) is set.

In case a *corrected* memory error is detected, the FlexRay module continues its normal operation using the corrected data word. This section describes the behavior of the FlexRay module after the detection of a *non-corrected* memory error.

57.6.24.3.1 CHI LRAM error response after CC read

When the CC is out of the *POC:default config* state, it reads the configuration data and the data field offsets of all utilized message buffers in every slot and in the NIT. If a non-corrected memory error is detected during this module read access, the error response of the module depends on LRAM location where the error occurred.

- If the LRAM address belongs to physical message buffer configuration data the FlexRay module will consider the affected message buffer as disabled for the current

search and will exclude this buffer from the search. The configuration of the affected message buffer is not changed.

  – If the affected message buffer is a Tx message buffer, no frame will be transmitted from this message buffer in the next slot. If the affected message buffer is an Rx message buffer, no frame will be received to this message buffer in the next slot.

- If the LRAM address belongs to the data field offset area and the related physical message buffer is used for Rx or Tx, the first access to the system memory caused by payload read or write yields to the assertion of the FR_CHIERFR[ILSA_EF]. No memory access occurs with respect to payload access for the complete frame.

57.6.24.3.2 CHI LRAM error response after application read

The application can read the content of the CHI LRAM by reading the FR_MBCCFR*n*, FR_MBFIDR*n*, FR_MBIDXR*n*, FR_MBDOR*n* and FR_LEETR*n* registers. If a non-corrected memory error is detected during this kind of read access, the module indicates the detected memory error, delivers the non-corrected data read, and continues its normal operation.

57.6.24.3.3 PE DRAM error response after CC read

If the CC detects a non-corrected memory error during an internal read of program data contained in PE DRAM, this is considered a fatal protocol error and the module enters the protocol freeze state immediately.

57.6.24.3.4 PE DRAM error response after application read in *POC:default config* state

If the CC detects a non-corrected memory error during an application-triggered read from any PE DRAM address and the protocol is in the *POC:default config* state, this is considered a fatal protocol error and the module enters the protocol freeze state. This behavior allows for checking the freeze functionality in case of the detection of non-corrected errors.

57.6.24.3.5 PE DRAM error response after application read out of *POC:default config*

If the CC detects a non-corrected memory error during an application-triggered read from any PE DRAM address and the protocol is not in the *POC:default config* state, this error is not considered a fatal error and the protocol state is not changed. This prevents any interference of the running protocol by PE DRAM error injection reads.

57.6.25 Memory error injection

The error injection functionality is used by the application to inject data errors into the memories to trigger and check the memory error detection functionality.

The error injection is enabled only if the ECC functionality enable bit ECCE in the Module Configuration Register (FR_MCR) and the error injection enable control bit EIE in the ECC Error Report and Injection Control Register (FR_EERICR) are set.

Error Injection mode is configured by the EIM configuration bit in the ECC Error Report and Injection Control Register (FR_EERICR).When error injection is enabled, each write access to the configured memory location will be distorted.

The injector has the same behavior for FlexRay module memory writes and application memory writes.

57.6.25.1 CHI LRAM error injection

The following sequence describes a memory error injection sequence for the CHI LRAM memory. This sequence consists of the error injector setup followed by an application triggered write access to provoke a distortion of the memory content. The content of the CHI LRAM is described in *Table 1427*.

When the CC is in *POC:default config*, there are no limitations for the error injection and no impacts of error injection to the application. For error injection out of *POC:default config,* refer to *Section 57.7.3*.

Injector setup:

1. FR_MCR[ECCE]:= 1;
   // enable ECC functionality
2. FR_EERICR[EIE]:=I_MODE;
   // configure error injection mode
3. FR_EEIAR[MID]:= 1;
   // select CHI LRAM for error injection
4. FR_EEIAR[BANK]:= I_BANK;
   // select bank for error injection; I_BANK = {0,1,2,3,4,5}
5. FR_EEIAR[ADDR]:= I_ADDR;
   // select address for error injection; I_ADDR ≤ 0x56
6. FR_EEIDR[DATA]:= D_DIST;
   // define data distortion pattern
7. FR_EEICR[CODE]:= C_DIST;
   // define checkbit distortion pattern
8. FR_EERICR[EIE]:=1;
   // enable error injection

Application write access:

If (I_BANK==0) -> FR_MBCCFR(2n) / FR_MBDOR(6k) / FR_LEETR0 := DATA;
If (I_BANK==1) -> FR_MBFIDR(2n) / FR_MBDOR(6k+1) / FR_LEETR1 := DATA;
If (I_BANK==2) -> FR_MBIDXR(2n) / FR_MBDOR(6k+2) / FR_LEETR2 := DATA;
If (I_BANK==3) -> FR_MBCCFR(2n+1) / FR_MBDOR(6k+3) / FR_LEETR3 := DATA;
If (I_BANK==4) -> FR_MBFIDR(2n+1) / FR_MBDOR(6k+4) / FR_LEETR4 := DATA;
If (I_BANK==5) -> FR_MBIDXR(2n+1) / FR_MBDOR(6k+5) / FR_LEETR5 := DATA;
// write DATA to the defined injection bank and injection address (refer to *Table 1427*).

57.6.25.2 PE DRAM error injection

The following sequence describes a memory error injection sequence for the PE DRAM memory. This sequence consists of the error injector setup followed by an application triggered write access to provoke a distortion of the memory content.

When the FlexRay module is in *POC:default config*, there are no limitations for the error injection and no impacts of error injection to the application. For error injection out of *POC:default config,* refer to *Section 57.7.3.2*.

Injector setup:

1. FR_MCR[ECCE]:= 1;
   // enable ECC functionality
2. FR_EERICR[EIE]:=I_MODE;
   // configure error injection mode
3. FR_EEIAR[MID]:= 0;
   // select PE DRAM for error injection
4. FR_EEIAR[BANK]:= I_BANK;
   // define bank for error injection; I_BANK = {0,1}
5. FR_EEIAR[ADDR]:= I_ADDR;
   // define address for error injection; I_ADDR ≤ 0x7F
6. FR_EEIDR[DATA]:= D_DIST;
   // define data distortion pattern
7. FR_EEICR[CODE]:= C_DIST;
   // define checkbit distortion pattern
8. FR_EERICR[EIE]:=1;
   // enable error injection

Application write access (for example, I_ADDR=0x70):

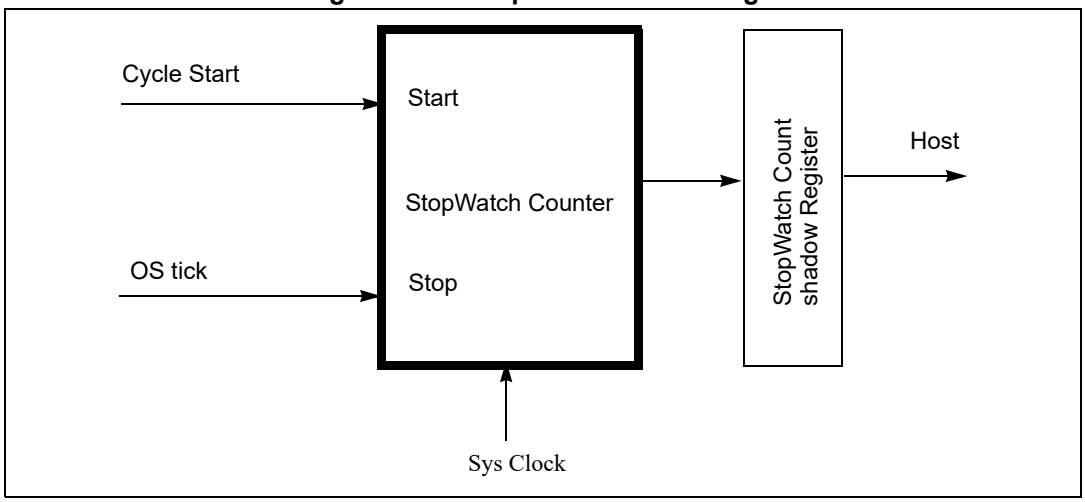
1. FR_PEDRAR:= 0x30E0;
   // INST=0x3; ADDR=0x70
2. wait until FR_PEDRAR[DAD] == 1;
   // wait for end of PE DRAM access
3. val = FR_PEDRDR[DATA]; |
   // get read back PE DRAM data

Note: The write access to the PE DRAM triggers a subsequent read access from PE DRAM in the next cycle, which triggers the detection of the distorted data.

57.6.26 StopWatch function

FlexRay supports a StopWatch function to facilitate OS synchronization to the FlexRay network. This provides a capability to capture the time difference in system clocks between the operating system timer tick and the FlexRay time base. This allows the software synchronization handler to adjust the OS clock to sync with the FR macrotick.

Figure 1390. StopWatch counter logic

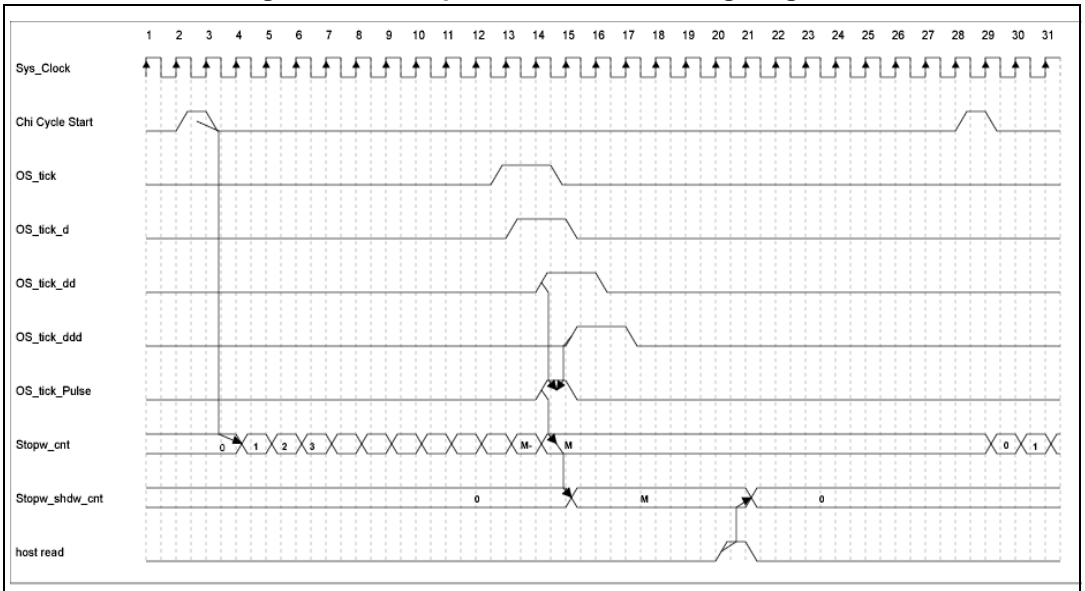


As shown here, there is a 32-bit stopwatch counter that resets and starts with each cycle start. This counter runs on the system clock and increments every cycle until an OS tick event is received, after which the counter halts and the count value is transferred to a shadow register. The shadow register (*Section 57.5.2.68: Stop Watch Count Register (FR_STPWR)*) holds the counter value till the host reads all the 32 bits, after which it automatically gets cleared.

The StopWatch logic is controlled through a control bit STPW_EN in the FR_PEOER (*Section 57.5.2.69: Protocol Event Output Enable and StopWatch Control Register (FR_PEOER)*. The counter runs only when this bit is set.

Figure 1391 shows the actual timing of the stopwatch counter logic operation.

Figure 1391. StopWatch function timing diagram



Note: 'Host read' implies that all 32 bits have been read out completely. Any partial read (that is for 8/16 bits access) will not clear the StopWatch count register.

If cycle start and OS tick occur simultaneously, timer will remain at 0. But OS_tick has a 2 stage synchronizer and Cycle_start has a single stage synchronizer into Chi clock domain. So simultaneously here means OS_tick_Pulse and Chi_Cycle_Start occurring at the same cycle.

57.7 Application information

57.7.1 Module configuration

This section describes the essential parts of the module configuration.

57.7.1.1 Configure System Memory Access Timeout Register (FR_SYMATOR)

To ensure reliable operation of the CC, the application must ensure that the TIMEOUT value in the System Memory Access Time-Out Register (FR_SYMATOR) and the CHI clock frequency f_{CHI} in MHz fulfill *Equation 62*[26].

Equation 62

$$0 \leq \text{SYMATOR[TIMEOUT]} \leq \left\lfloor 0.45 \cdot f_{CHI} - 8 \right\rfloor$$

For a given SYMATOR[TIMEOUT] value, f_{CHI} can be increased without causing unreliable operation of the CC. The same holds for reducing the SYMATOR[TIMEOUT] value for a given f_{CHI}.

Some examples for maximum values of the SYMATOR[TIMEOUT] for a minimum CHI frequency are given in *Table 1433*.

26. Refer to *Section 57.3: Controller host interface clocking* for all constraints of minimum CHI clock frequency.

Table 1433. Maximum SYMATOR[TIMEOUT] examples

f_{CHI}	SYMATOR[TIMEOUT]	f_{CHI}	SYMATOR[TIMEOUT]
≥ 18 MHz	0	≥ 100 MHz	≤ 37
≥ 23 MHz	≤ 2	≥ 120 MHz	≤ 46
≥ 27 MHz	≤ 4	≥ 140 MHz	≤ 55
≥ 32 MHz	≤ 6	≥ 160 MHz	≤ 64
≥ 60 MHz	≤ 19	≥ 180 MHz	≤ 73
≥ 80 MHz	≤ 28	≥ 200 MHz	≤ 82

57.7.1.1.1 System bus wait state constraints

The SYMATOR[TIMEOUT] value corresponds directly to a certain acceptable number of wait states on the system bus.

For single-channel configurations and situations in which the sync frame table generation functionality is *not* used (FR_SFTCCSR[SDVEN,SIDEN] = 0), no timeout will be detected if fewer than 2 × SYMATOR[TIMEOUT] + 1 wait states will be seen on the system bus for each system bus access.

For dual-channel configurations, or situations in which the sync frame table generation functionality is used, no timeout will be detected if fewer than SYMATOR[TIMEOUT] – 1 wait states will be seen on the system bus for each system bus access.

57.7.1.2 Configure data field offsets

The data field offsets are located in the Message Buffer Data Field Offset Registers (FR_MBDORn) and Receive FIFO Start Data Offset Register (FR_RFSDOR). The application has to configure the data field offset values for all message buffers that are used.

When the module is enabled, the initialization value of the FR_MBDOR*n*[MBDO] and FR_RFSDOR[SDO] is 0. This value is considered to be illegal (refer to *Section 57.6.19.1.1*).

57.7.2 Initialization sequence

This section describes the steps required to initialize the CC. The first subsection describes the steps required after a system reset; the second section describes the steps required after preceding shutdown of the CC.

57.7.2.1 Module initialization

This section describes the module-related initialization steps after a system reset.

1. Configure the CC.
   a) Configure the control bits in the Module Configuration Register (FR_MCR).
   b) Configure the system memory base address in the System Memory Base Address Register (FR_SYMBADR).
2. Enable the CC.
   a) Write 1 to the module enable bit MEN in the Module Configuration Register (FR_MCR).

The CC now enters Normal mode. The application can commence with the protocol initialization described in *Section 57.7.2.2*.

57.7.2.2 Protocol initialization

This section describes the protocol-related initialization steps.

1. Configure the Protocol Engine.
   a) Issue CONFIG command via the Protocol Operation Control Register (FR_POCR).
   b) Wait for *POC:config* in the Protocol Status Register 0 (FR_PSR0).
   c) Configure the FR_PCR0,..., FR_PCR30 registers to set all protocol parameters.
2. Configure the message buffers and FIFOs.
   a) Set the number of message buffers used and the message buffer segmentation in the Message Buffer Segment Size and Utilization Register (FR_MBSSUTR).
   b) Define the message buffer data size in the Message Buffer Data Size Register (FR_MBDSR).
   c) Configure each message buffer by setting the configuration values in the Message Buffer Configuration, Control, Status Registers (FR_MBCCSRn), Message Buffer Cycle Counter Filter Registers (FR_MBCCFRn), Message Buffer Frame ID Registers (FR_MBFIDRn), and Message Buffer Index Registers (FR_MBIDXRn).
   d) Configure the FIFOs.
   e) Issue CONFIG_COMPLETE command via the Protocol Operation Control Register (FR_POCR).
   f) Wait for *POC:ready* in the Protocol Status Register 0 (FR_PSR0).

After this sequence, the CC is configured as a FlexRay node and is ready to integrate into the FlexRay cluster.

57.7.2.3 CHI LRAM initialization

The CHI LRAM initialization is performed by the CC when it leaves Disabled mode. The initialization runs for 87 CHI clock cycles. All fields in the FR_MBCCSR*n*, FR_MBCCFR*n*, FR_MBFIDR*n*, FR_MBDOR*n* and LEETR*n* registers are initialized to 0. All application read or write accesses to these registers are delayed until the initialization is finished.

57.7.2.4 PE DRAM initialization

The PE DRAM initialization is performed by the CC in the *POC:default config* state. This initialization runs for 4.8 µs and will delay the state transition from *POC:default config* into *POC:config*.

57.7.3 Memory error injection out of *POC:default config*

This section provides information for application-driven memory error injection out if *POC:default config.* The CC provides the means to inject memory errors from the application without any impacts to the internal protocol operation of the CC.

57.7.3.1 CHI LRAM error injection out of *POC:default config*

The CC will never perform any internal read access from the LRAM ECC Error Test Registers (FR_LEETRn). Any memory errors injected into these CHI LRAM locations will never be detected by internal access, independently from the protocol state.

The application should use these registers and related CHI LRAM location to inject memory errors into the CHI LRAM. The injection sequence is described in *Section 57.6.25.1*.

57.7.3.2 PE DRAM error injection out of *POC:default config*

The CC will never perform any internal read access from the PE DRAM address 0x70. This is the only PE DRAM address writable by the application out of the *POC:default config* state.

The application should use these PE DRAM locations to inject memory errors into the PE DRAM. The injection sequence is described in *Section 57.6.25.2*.

57.7.4 Shutdown sequence

This section describes a secure shutdown sequence to stop the CC gracefully. The main targets of this sequence are:

- Finish all ongoing reception and transmission
- Do not corrupt FlexRay bus and do not disturb ongoing FlexRay bus communication

For a graceful shutdown, the application should perform the following tasks:

1. Disable all enabled message buffers.
   a) Repeatedly write 1 to FR_MBCCSR*n*[EDT] until FR_MBCCSR*n*[EDS] == 0.
2. Stop the Protocol Engine.
   a) Issue HALT command via Protocol Operation Control Register (FR_POCR).
   b) Wait for *POC:halt* in Protocol Status Register 0 (FR_PSR0).

57.7.5 Number of usable message buffers

This section describes the required minimum CHI clock frequency for a specified number of utilized message buffers configured in the Message Buffer Segment Size and Utilization Register (FR_MBSSUTR), a configured mini-slot length *gdMinislot*, and a configured nominal macrotick length *gdMacrotick*[27].

Additional constraints for the minimum CHI clock frequency are given in *Section 57.3*.

The CC uses a sequential search algorithm to determine the individual message buffer assigned or subscribed to the next slot. This search is started at the start of the slot and must be finished before the start of the next slot.

27. Refer to *Section 57.3: Controller host interface clocking* for all constraints of minimum CHI clock frequency.

The shortest FlexRay slot is a corrected empty dynamic slot. A corrected empty dynamic slot is a mini-slot and consists of *gdMinislot* corrected macroticks with a duration of *gdMacrotick*. The minimum duration of a corrected macrotick is $gdMacrotick_{min}$ = 39 µT. This results in a minimum length of a correct slot.

Equation 63

$$\Delta_{slotmin} = 39 \cdot pdMicrotick \cdot gdMinislot$$

The message buffer search engine runs on the CHI clock and evaluates one individual message buffer per CHI clock cycle. For internal status update operations and to account for clock domain crossing jitter, an additional amount of 27 CHI clock cycles is required to ensure correct search engine operation.

For a given number of utilized message buffers FR_MBSSUTR[LAST_MB_UTIL] + 1 and for a given CHI clock frequency f_{chi}, this results in a search duration of

Equation 64

$$\Delta_{search} = \frac{1}{f_{chi}} \cdot (FR_MBSSUTR[LAST_MB_UTIL]+27)$$

The message buffer search must be finished within one slot which requires that *Equation 65* must be fulfilled:

Equation 65

$$\Delta_{search} \leq \Delta_{slotmin}$$

This results in the formula given in *Equation 66* which determines the required minimum CHI frequency for a given number of message buffers that are utilized.

Equation 66

$$f_{chi} \geq \frac{(FR_MBSSUTR[LAST_MB_UTIL]+27)}{39 \cdot pdMicrotick \cdot gdMinislot}$$

The required minimum CHI Clock frequency for a selected set of relevant protocol parameters and for the LAST_MB_UTIL field in the Message Buffer Segment Size and Utilization Register (FR_MBSSUTR) set to 127 is given in *Table 1434*.

Table 1434. Minimum CHI clock frequency [MHz] examples (128 message buffers used)

pdMicrotick [ns]	gdMinislot					
	2	3	4	5	6	7
25.0	79.5	53	39.8	31.8	26.5	22.8
50.0	39.8	26.5	19.9	15.9	13.3	11.4

Note: If the minimum CHI frequency is not met the CHIERFR[MBS_EF] flag is set. Refer to Section 57.5.2.18: CHI Error Flag Register (FR_CHIERFR) for details.

57.7.6 Protocol control command execution

This section considers the issues of the protocol control command execution.

The application issues any of the protocol control commands listed in the POCCMD field of *Table 1314* by writing the command to the POCCMD field of the Protocol Operation Control Register (FR_POCR). As a result the CC sets the BSY bit while the command is transferred to the PE. When the PE has accepted the command, the BSY flag is cleared. All commands are accepted by the PE.

The PE maintains a protocol command vector. For each command that was accepted by the PE, the PE sets the corresponding command bit in the protocol command vector. If a command is issued while the corresponding command bit is set, the command is not queued and is lost.

If the command execution block of the PE is idle, it selects the next accepted protocol command with the highest priority from the current protocol command vector according to the protocol control command priorities given in *Table 1435*. If the current protocol state does not allow the execution of this protocol command (refer to POC state changes in *FlexRay Communications System Protocol Specification, Version 2.1 Rev A*) the CC asserts the illegal protocol command interrupt flag IPC_IF in the Protocol Interrupt Flag Register 1 (FR_PIFR1). The protocol command is not executed in this case.

Some protocol commands may be interrupted by other commands or the detection of a fatal protocol error as indicated by *Table 1435*. If the application issues the FREEZE or READY command, or if the PE detects a fatal protocol error, some commands already stored in the command vector will be removed from this vector.

Table 1435. Protocol control command priorities

Protocol command	Priority	Interrupted by	Cleared and terminated by
FREEZE	(Highest) 1	None	
READY	2		
CONFIG_COMPLETE	3		
ALL_SLOTS	4	FREEZE, READY, CONFIG_COMPLETE, fatal protocol error	FREEZE, READY, CONFIG_COMPLETE, fatal protocol error
ALLOW_COLDSTART	5		
RUN	6		FREEZE, fatal protocol error
WAKEUP	7		FREEZE, fatal protocol error
DEFAULT_CONFIG	8		FREEZE, fatal protocol error
CONFIG	9		
HALT	(Lowest) 10		FREEZE, READY, CONFIG_COMPLETE, fatal protocol error

57.7.7 Message buffer search on simple message buffer configuration

This section describes the message buffer search behavior for a simplified message buffer configuration. The FIFO behavior is not considered in this section.

57.7.7.1 Simple message buffer configuration

A simple message buffer configuration is a configuration that has at most one transmit message buffer and at most one receive message buffer assigned to a slot *s*. The simple configuration used in this section utilizes two message buffers, one single buffered transmit message buffer and one receive message buffer.

The transmit message buffer has the message buffer number *t* and has the following configuration.

Table 1436. Transmit buffer configuration

Register	Field	Value	Description
FR_MBCCSR*t*	MTD	1	Transmit buffer
FR_MBCCFR*t*	MTM	0	Event Transition mode
	CHA	1	Assigned to channel A
	CHB	0	Not assigned to channel B
	CCFE	1	Cycle counter filter enabled
	CCFMSK	000011	Cycle set = {4n} = {0,4,8,12,…}
	CCFVAL	000000	
FR_MBFIDR*t*	FID	S	Assigned to slot s

The availability of data in the transmit buffer is indicated by the commit bit FR_MBCCSRt[CMT] and the lock bit FR_MBCCSRt[LCKS].

The receive message buffer has the message buffer number *r* and has the following configuration.

Table 1437. Receive buffer configuration

Register	Field	Value	Description
FR_MBCCSR*r*	MTD	0	Receive buffer
FR_MBCCFR*r*	MTM	—	N/A
	CHA	1	Assigned to channel A
	CHB	0	Not assigned to channel B
	CCFE	1	Cycle counter filter enabled
	CCFMSK	000001	Cycle set = {2n} = {0,2,4,6,…}
	CCFVAL	000000	
FR_MBFIDR*r*	FID	S	Subscribed slot

Furthermore the assumption is that both message buffers are enabled (FR_MBCCSR*t*[EDS] = 1 and FR_MBCCSR*r*[EDS] = 1).

Note: The cycle set {4n+2} = {2,6,10,…} is assigned to the receive buffer only.

The cycle set {4n} = {0,4,8,12,…} is assigned to both buffers.

57.7.7.2 Behavior in static segment

In this case, both message buffers are assigned to a slot *s* in the *static* segment.

The configuration of a transmit buffer for a static slot *s* assigns this slot to the node as a transmit slot. The FlexRay protocol requires:

- When a slot occurs, if the slot is assigned to a node on a channel that node must transmit either a normal frame or a null frame on that channel. Specifically, a null frame will be sent if there is no data ready, or if there is no match on a transmit filter (cycle counter filtering, for example).

Regardless of the availability of data and the cycle counter filter, the node will transmit a frame in the static slot *s.* In any case, the result of the message buffer search will be the transmit message buffer *t.* The receive message buffer *r* will not be found, no reception is possible.

57.7.7.3 Behavior in dynamic segment

In this case, both message buffers are assigned to a slot *s* in the *dynamic* segment. The FlexRay protocol requires:

- When a slot occurs, if a slot is assigned to a node on a channel that node only transmits a frame on that channel if there is data ready and there is a match on relevant transmit filters (no null frames are sent).

The transmission of a frame in the dynamic segment is determined by the availability of data and the match of the cycle counter filter of the transmit message buffer.

57.7.7.3.1 Transmit Data Not Available

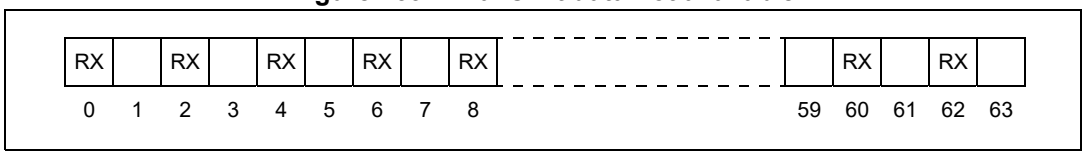
If transmit data are not available, that is the transmit buffer is not committed FR_MBCCSR*t*[CMT]=0 or locked FR_MBCCSR*t*[LCKS]=1, or both.

a) For the cycles in the set {4n}, which is assigned to both buffers, the receive buffer will be found and the node can receive data, and,

b) For the cycles in the set {4n+2}, which is assigned to the receive buffer only, the receive buffer will be found and the node can receive data.

The receive cycles are shown in *Figure 1392*.

Figure 1392. Transmit data not available



57.7.7.3.2 Transmit data available

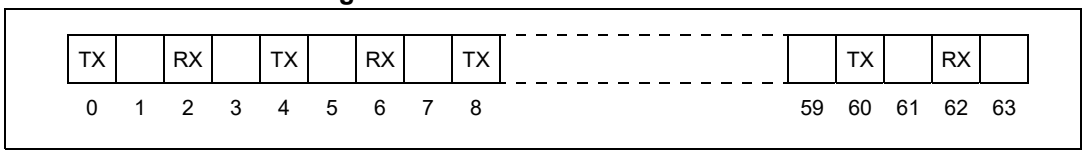
If transmit data are *available,* that is, the transmit buffer is committed FR_MBCCSR*t*[CMT] = 1 and not locked FR_MBCCSR*t*[LCKS] = 0,

a) For the cycles in the set {4n}, which is assigned to both buffers, the transmit buffer will be found and the node transmits data.

b) For the cycles in the set {4n+2}, which is assigned to the receive buffer only, the receive buffer will be found and the node can receive data.

The receive and transmit cycles are shown in *Figure 1393*.

Figure 1393. Transmit data available



58 Inter-Integrated Circuit (I^2C)

58.1 Overview

This chapter describes the Inter-Integrated Circuit (I^2C) bus module implemented on this chip and presents the following topics:

- *Section 58.2: Introduction to I2C*
- *Section 58.3: External signal descriptions*
- *Section 58.4: Memory map and register definition*
- *Section 58.5: Functional description*
- *Section 58.6: Initialization and application information*

58.2 Introduction to I^2C

This section presents the following topics:

- *Section 58.2.1: Definition: I^2C module*
- *Section 58.2.2: Advantages of the IC bus*
- *Section 58.2.3: Module block diagram*
- *Section 58.2.4: Features*
- *Section 58.2.5: Modes of operation*
- *Section 58.2.6: Definition: I2C conditions*

58.2.1 Definition: I^2C module

The I^2C module is a functional unit that provides a two-wire – serial data (SDA) and serial clock (SCL) – bidirectional serial bus that provides a simple and efficient method of data exchange between this chip and other devices, such as microcontrollers, EEPROMs, real-time clock devices, analog-to-digital converters, and LCDs.

58.2.2 Advantages of the IC bus

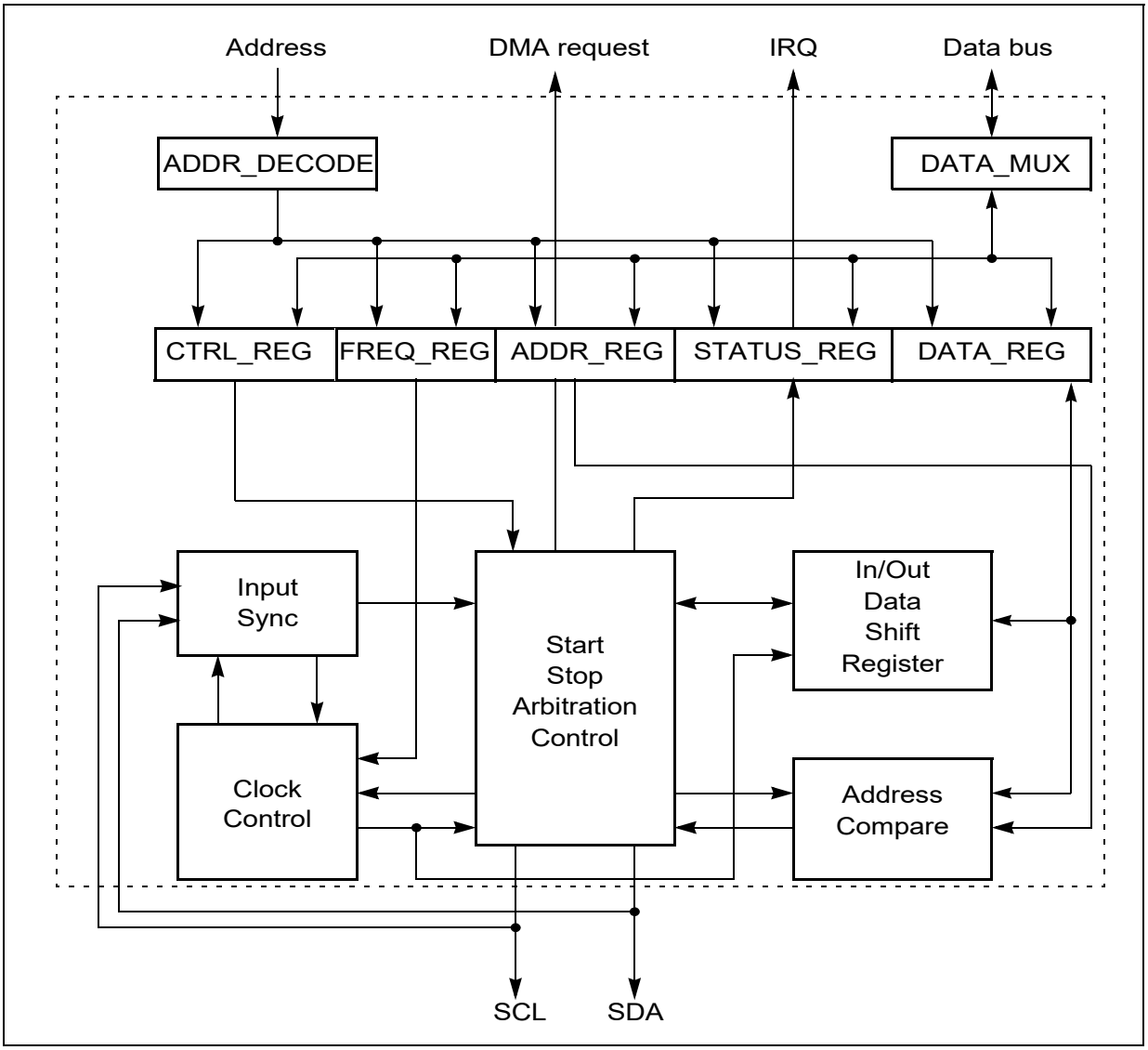
The synchronous, multiple-master two-wire I^2C bus:

- Minimizes interconnections between devices
- Allows the connection of additional devices to the bus for expansion and system development
- Includes collision detection and arbitration that prevent data corruption if two or more masters attempt to control the bus simultaneously
- Does not require an external address decoder

58.2.3 Module block diagram

Figure 1394 shows a block diagram of the I^2C module.

Figure 1394. I²C block diagram



58.2.4 Features

The I²C module has the following key features:

- Compatible with I²C bus standard[28]
- Operating speeds
  - Up to 100 kbps in Standard Mode
  - Up to 400 kbps in Fast Mode
  - Actual baud rate dependent on the SCL rise time (which depends on external pull-up resistor values and bus loading)
- Multi-master operation
- Software programmable for one of 256 different serial clock frequencies
- Software selectable acknowledge bit
- Interrupt driven byte-by-byte data transfer
- Arbitration lost interrupt with automatic mode switching from master to slave
- Calling address identification interrupt
- Start and stop signal generation/detection
- Repeated start signal generation
- Acknowledge bit generation/detection
- Bus busy detection
- Basic DMA interface
- Maximum communication length and the number of devices that can be connected are limited by a maximum bus capacitance of 400 pF

58.2.5 Modes of operation

The I²C module supports the chip modes described in *Table 1438*.

Table 1438. Chip modes supported by the I²C module

Chip mode	Description	Important notes
RUN	Basic mode of operation	—
STOP	The lowest-power mode that allows the chip to turn off all the clocks to the I²C module	The I²C module can enter this mode when there are no active transfers (active data between valid START and STOP conditions) on the bus. Refer to *Section 58.5.6: IPG STOP mode*.
IPG DEBUG	Allows the chip to freeze all ongoing activities (such as an ongoing transaction, counter values, and register status) for debugging.	Refer to *Section 58.5.7: IPG DEBUG mode*.

In addition to chip modes, the I²C module has several module-specific modes. These are described in *Table 1439*.

28. Compliant with I²C 2.0 standard with the exception that HS (high speed) mode is not supported

Table 1439. Module-specific modes supported by the I²C module

Module mode	Description	Important notes
Master mode	The I²C module is the driver of the SDA line.	– Do not use the I²C module's slave address as a calling address. – The I²C module cannot be a master and a slave simultaneously.
Slave mode	The I²C module is not the driver of the SDA line.	– Enable the I²C module before a START condition from a non-I²C master is detected. – By default the I²C module performs as a slave receiver.

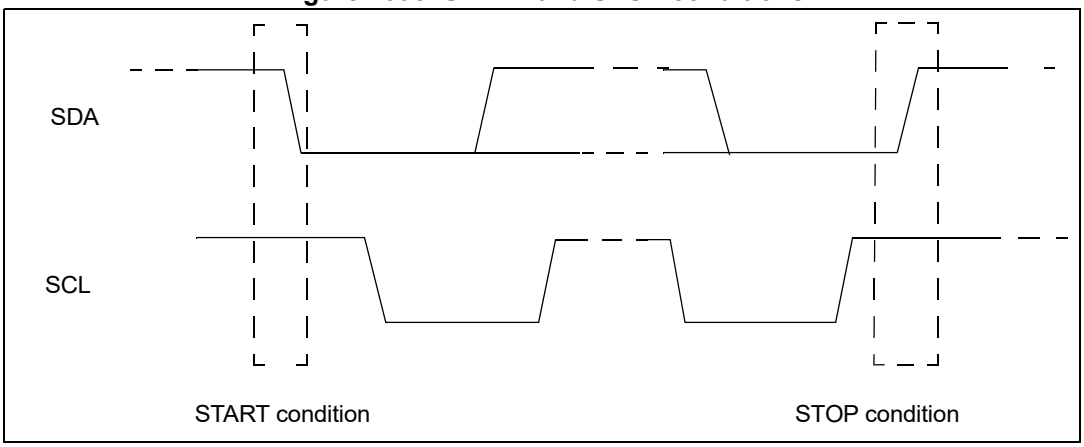
58.2.6 Definition: I²C conditions

Table 1440 shows the I²C-specific conditions defined for the I²C module.

Table 1440. I²C Conditions

Condition	Description
START	A condition that denotes the beginning of a new data transfer and awakens all slaves. Each data transfer contains several bytes of data. It is defined as a high-to-low transition of SDA while SCL is high, as shown in *Figure 1395*.
STOP	A condition generated by the master to terminate a transfer and free the bus. It is defined as a low-to-high transition of SDA while SCL is high, as shown in *Figure 1395*.
Repeated START	A START condition that is generated without a STOP condition to terminate the previous transfer.

Figure 1395. START and STOP conditions



58.3 External signal descriptions

This section presents the following topics:

- *Section 58.3.1: Signal overview*
- *Section 58.3.2: Detailed external signal descriptions*

58.3.1 Signal overview

The I^2C module uses the Serial Data (SDA) and Serial Clock (SCL) signals as a communication interconnect with other devices. The signal patterns driven on the SDA signal represent address, data, or read/write information at different stages of the protocol.

All devices connected to the SDA and SCL signals must have open-drain or open-collector outputs. The logical AND function is performed on both signals with external pull-up resistors. For the electrical characteristics of these signals, refer to the data sheet for this chip.

58.3.2 Detailed external signal descriptions

The SDA and SCL signals are described in *Table 1441*.

Table 1441. External signal descriptions

Signal	Description
SCL	Bidirectional serial clock line of the module, compatible with the I^2C bus specification
SDA	Bidirectional serial data line of the module, compatible with the I^2C bus specification

58.4 Memory map and register definition

This section provides a detailed description of all memory-mapped registers in the I^2C module. It presents the following topics:

- *Section 58.4.1: I2C memory map*
- *Section 58.4.2: Register accessibility*
- *Section 58.4.3: I2C Bus Address register (IBAD)*
- *Section 58.4.4: I2C Bus Frequency Divider register (IBFD)*
- *Section 58.4.5: I2C Bus Control Register (IBCR)*
- *Section 58.4.6: I2C Bus Status Register (IBSR)*
- *Section 58.4.7: I2C Bus Data I/O Register (IBDR)*
- *Section 58.4.8: I2C Bus Interrupt Config register (IBIC)*
- *Section 58.4.9: I2C Bus Debug register (IBDBG)*

58.4.1 I^2C memory map

The memory map for the I^2C module is given below in *Table 1442*. The total address for each register is the sum of the base address for the I^2C module and the address offset for each register.

Table 1442. I²C memory map

Address Offset	Register	Section
0x0000	I²C Bus Address Register (IBAD)	*Section 58.4.3*
0x0001	I²C Bus Frequency Divider Register (IBFD)	*Section 58.4.4*
0x0002	I²C Bus Control Register (IBCR)	*Section 58.4.5*
0x0003	I²C Bus Status Register (IBSR)	*Section 58.4.6*
0x0004	I²C Bus Data I/O Register (IBDR)	*Section 58.4.7*
0x0005	I²C Bus Interrupt Config Register (IBIC)	*Section 58.4.8*
0x0006	I²C Bus Debug Register (IBDBG)	*Section 58.4.9*
0x0007–0x3FFF	Reserved(1)	

1. If enabled at the SoC level, reads or writes to these registers will cause bus aborts. Refer to the System Services Module documentation for more details.

58.4.2 Register accessibility

All registers are accessible via 8-bit, 16-bit, or 32-bit accesses. However, 16-bit accesses must be aligned to 16-bit boundaries, and 32-bit accesses must be aligned to 32-bit boundaries.

As an example, the IBFD is at address offset 0x01, and can be accessed in any of the following ways:

- 8-bit R/W access of address offset 0x0001
- Second byte of 16-bit R/W access of address offset 0x0000
- Second byte of 32-bit R/W access of address offset 0x0000

The IBFD register cannot be accessed by a 16- or 32-bit RW operation at address offset 0x0001 because those operations require an address aligned to a 16- or 32-bit boundary.

58.4.3 I²C Bus Address register (IBAD)

This register contains the address the I²C Bus will respond to when addressed as a slave. This is not the address sent on the bus during the address transfer.

Access:
Supervisor mode only
(user mode accesses are ignored, and no error response is asserted while accessing this register in user mode)

Offset: 0x0000

	0	1	2	3	4	5	6	7
R	ADR							0
W								
Reset	0	0	0	0	0	0	0	0

Figure 1396. I²C Bus Address register (IBAD)

Table 1443. IBAD field descriptions

Field	Description
0:6 ADR	Slave Address Specific slave address to be used by the I^2C Bus module. **Note:** The default mode of I^2C Bus is slave mode for an address match on the bus.

58.4.4 I^2C Bus Frequency Divider register (IBFD)

Access:
Supervisor mode only
(user mode accesses are ignored, and no error response is asserted while accessing this register in user mode)

Offset: 0x0001

	0	1	2	3	4	5	6	7
R W	IBC							
Reset	0	0	0	0	0	0	0	0

Figure 1397. I^2C Bus Frequency Divider register (IBFD)

Table 1444. IBFD field descriptions

Field	Description
0:7 IBC	I-Bus Clock Rate This field is used to prescale the clock for bit rate selection. The bit clock generator is implemented as a prescale divider. The IBC bits are decoded to give the Tap and Prescale values as follows: 0:1 Selects the prescaled shift register (refer to *Table 1453*) 2:4 Selects the prescaler divider (refer to *Table 1454*) 5:7 Selects the shift register tap point (refer to *Table 1455*)

58.4.5 I^2C Bus Control Register (IBCR)

Access:
Supervisor mode only
(user mode accesses are ignored, and no error response is asserted while accessing this register in user mode)

Offset: 0x0002

	0	1	2	3	4	5	6	7
R	MDIS	IBIE	MSSL	TXRX	NOACK	0	DMAEN	0
W						RSTA		
Reset	1	0	0	0	0	0	0	0

Figure 1398. I^2C Bus Control Register (IBCR)

Table 1445. IBCR field descriptions

Field	Description
0 MDIS	Module disable This bit controls the software reset of the entire I^2C Bus module. 0 The I^2C Bus module is enabled. This bit must be cleared before any other IBCR bits have any effect. 1 The module is reset and disabled. This is the power-on reset situation. When high, the interface is held in reset, but registers can still be accessed. Status register bits (IBSR) are not valid when module is disabled. **Note:** If the I^2C Bus module is enabled in the middle of a byte transfer, the interface behaves as follows: slave mode ignores the current transfer on the bus and starts operating whenever a subsequent start condition is detected. Master mode will not be aware that the bus is busy, hence if a start cycle is initiated then the current bus cycle may become corrupt. This would ultimately result in either the current bus master or the I^2C Bus module losing arbitration, after which, bus operation would return to normal.
1 IBIE	I-Bus Interrupt Enable 0 Interrupts from the I^2C Bus module are disabled. This does not clear any currently pending interrupt condition. 1 Interrupts from the I^2C Bus module are enabled. An I^2C Bus interrupt occurs provided the IBIF bit in the status register is also set.
2 MSSL	Master/Slave mode select When this bit is changed from 0 to 1, a START signal is generated on the bus and the master mode is selected. When this bit is changed from 1 to 0, a STOP signal is generated and the operation mode changes from master to slave. A STOP signal should be generated only if the IBIF flag is set. This field is cleared without generating a STOP signal when the master loses arbitration. 0 Slave Mode 1 Master Mode
3 TXRX	Transmit/Receive mode select This bit selects the direction of master and slave transfers. When addressed as a slave this bit should be set by software according to the SRW bit in the status register. In master mode this bit should be set according to the type of transfer required. Therefore, for address cycles, this bit will always be high. 0 Receive 1 Transmit
4 NOACK	Data Acknowledge disable This bit specifies the value driven onto SDA during data acknowledge cycles for both master and slave receivers. The I^2C module will always acknowledge address matches, provided it is enabled, regardless of the value of NOACK. **Note:** Values written to this bit are only used when the I^2C Bus is a receiver, not a transmitter. 0 An acknowledge signal will be sent out to the bus at the 9th clock bit after receiving one byte of data. 1 No acknowledge signal response is sent (that is, acknowledge bit = 1).

Table 1445. IBCR field descriptions (continued)

Field	Description
5 RSTA	Repeat Start Writing a one to this bit will generate a repeated START condition on the bus, provided it is the current bus master. This bit will always be read as a low. Attempting a repeated start at the wrong time, if the bus is owned by another master, will result in loss of arbitration. 0 No effect 1 Generate repeat start cycle
6 DMAEN	DMA Enable When this bit is set, the DMA Tx and Rx lines will be asserted when the I^2C module requires data to be read or written to the data register. No Transfer Done interrupts will be generated when this bit is set, however an interrupt will be generated if the loss of arbitration or addressed as slave conditions occur. The DMA mode is only valid when the I^2C module is configured as a Master and the DMA transfer still requires CPU intervention at the start and the end of each frame of data. Refer to the DMA Application Information section for more details. 0 Disable the DMA TX/RX request signals 1 Enable the DMA TX/RX request signals

58.4.6 I^2C Bus Status Register (IBSR)

Access:
Supervisor mode only
(user mode accesses are ignored, and no error response is asserted while accessing this register in user mode)

Offset: 0x0003

	0	1	2	3	4	5	6	7
R	TCF	IAAS	IBB	IBAL	0	SRW	IBIF	RXAK
W				w1c			w1c	
Reset	1	0	0	0	0	0	0	0

Figure 1399. I^2C Bus Status Register (IBSR)

Table 1446. IBSR field descriptions

Field	Description
0 TCF	Transfer complete While one byte of data is being transferred, this bit is cleared. It is set by the falling edge of the 9th clock of a byte transfer. **Note:** This bit is only valid during or immediately following a transfer to the I^2C module or from the I^2C module. 0 Transfer in progress 1 Transfer complete
1 IAAS	Addressed as a slave When its own specific address (I-Bus Address Register) is matched with the calling address, this bit is set. The CPU is interrupted provided the IBIE is set. Then the CPU needs to check the SRW bit and set the TXRX field accordingly. Writing to the I-Bus Control Register clears this bit. 0 Not addressed 1 Addressed as a slave
2 IBB	Bus busy This bit indicates the status of the bus. When a START signal is detected, IBB is set. If a STOP signal is detected, IBB is cleared and the bus enters idle state. 0 Bus is Idle 1 Bus is busy
3 IBAL	Arbitration Lost The arbitration lost bit (IBAL) is set by hardware when the arbitration procedure is lost. Arbitration is lost in the following circumstances: – SDA is sampled low when the master drives a high during an address or data transmit cycle. – SDA is sampled low when the master drives a high during the acknowledge bit of a data receive cycle. – A start cycle is attempted when the bus is busy. – A repeated start cycle is requested in slave mode. – A stop condition is detected when the master did not request it. This bit must be cleared by software, by writing a one to it. A write of zero has no effect.
5 SRW	Slave Read/Write When the IAAS bit is set, this bit indicates the value of the R/W command bit of the calling address sent from the master. This bit is only valid when the I-Bus is in slave mode, a complete address transfer has occurred with an address match and no other transfers have been initiated. By reading this field, the CPU can detect slave transmit/receive mode according to the command of the master. 0 Slave receive, master writing to slave 1 Slave transmit, master reading from slave

Table 1446. IBSR field descriptions (continued)

Field	Description
6 IBIF	I-Bus Interrupt Flag The IBIF bit is set when one of the following conditions occurs: – Arbitration lost (IBAL bit set) – Byte transfer complete (TCF bit set and DMAEN bit not set) – Addressed as slave (IAAS bit set) – NoAck from Slave (MS & Tx bits set) – I^2C Bus going idle (IBB high-low transition and enabled by BIIE) A processor interrupt request will be caused if the IBIE bit is set. This bit must be cleared by software, by writing a one to it. A write of zero has no effect on this bit. In DMA mode (DMAEN set) a byte transfer complete condition will not trigger the setting of IBIF. All other conditions still apply.
7 RXAK	Received Acknowledge This is the value of SDA during the acknowledge bit of a bus cycle. If the received acknowledge bit (RXAK) is low, it indicates an acknowledge signal has been received after the completion of 8 bits data transmission on the bus. If RXAK is high, it means no acknowledge signal is detected at the 9th clock. This bit is valid only after transfer is complete. 0 Acknowledge received 1 No acknowledge received

58.4.7 I^2C Bus Data I/O Register (IBDR)

In master transmit mode, when data is written to the IBDR, a data transfer is initiated. The most significant bit is sent first. In master receive mode, reading this register initiates next byte data receiving. In slave mode, the same functions are available after an address match has occurred.

Note: The IBCR[TXRX] field must correctly reflect the desired direction of transfer in master and slave modes for the transmission to begin. For instance, if the I^2C is configured for master transmit but a master receive is desired, then reading the IBDR will not initiate the receive.

Reading the IBDR will return the most recent byte received while the I^2C is configured in either master receive or slave receive modes. The IBDR does not reflect every byte that is transmitted on the I^2C bus, nor can software verify that a byte has been written to the IBDR correctly by reading it back.

In master transmit mode, the first byte of data written to IBDR following assertion of MSSL is used for the address transfer and should comprise the calling address (in position DATA[7:1]) concatenated with the required R/$\overline{W}$ bit (in position D0).

Note: When the I^2C is configured in master mode and receiving data from a slave that is transmitting data bytes on an irregular basis, the master cannot know whether the data received in the IBDR is the old latched data or the new data received from the slave. To avoid this, 2 consecutive intermittent data bytes from slave should be different.

Access:
Supervisor mode only
(user mode accesses are ignored, and no error response is asserted while accessing this register in user mode)

Offset: 0x0004

	0	1	2	3	4	5	6	7
R W	DATA							
Reset	0	0	0	0	0	0	0	0

Figure 1400. I²C Bus Data I/O Register (IBDR)

Table 1447. IBDR field descriptions

Field	Description
0:7 DATA	Data transmitted or received

58.4.8 I²C Bus Interrupt Config register (IBIC)

Access:
Supervisor mode only
(user mode accesses are ignored, and no error response is asserted while accessing this register in user mode)

Offset: 0x0005

	0	1	2	3	4	5	6	7
R	BIIE	BYTERXIE	0	0	0	0	0	0
W								
Reset	0	0	0	0	0	0	0	0

Figure 1401. I²C Bus Interrupt Config register (IBIC)

Table 1448. IBIC field descriptions

Field	Description
0 BIIE	Bus Idle Interrupt Enable bit This config bit can be used to enable the generation of an interrupt once the I²C bus becomes idle. Once this bit is set, an IBB high-low transition will set the IBIF bit. This feature can be used to signal to the CPU the completion of a STOP on the I²C bus. 0 Bus Idle Interrupts disabled 1 Bus Idle Interrupts enabled
1 BYTERXIE	Byte receive interrupt enable This field is used to generate an interrupt every time the I²C master/slave receives a new byte. This feature can be useful when an I²C master is receiving data from a slave that is transmitting on an irregular basis.

58.4.9 I²C Bus Debug register (IBDBG)

Access:
Supervisor mode only
(user mode accesses are ignored, and no error response is asserted while accessing this register in user mode)

Offset: 0x0006

	0	1	2	3	4	5	6	7
R	0	0	0	0	0	0	IPG_DEBUG_HALTED	IPG_DEBUG_EN
W								
Reset	0	0	0	0	0	0	0	0

Figure 1402. I²C Bus Debug register (IBDBG)

Table 1449. IBDBG field descriptions

Field	Description
6 IPG_DEBUG_HALTED	Debug Halted Bit This is s status bit which can be read back after asserting the debug enable signal so as to know if the IP has entered the debug mode or not. 0 IP is still executing a transaction 1 IP has entered the debug mode
7 IPG_DEBUG_EN	Debug enable bit This bit is used to enable IP enter the debug mode provided the IPG DEBUG signal is high. All the registers, counter values and status bits are frozen and can be accessed by the CPU. 0 Normal operation, Bus Idle Interrupts disabled 1 IP is in debug mode **Note:** If the assertion of this bit along with the IPG DEBUG signal happens in the middle of a transaction, IP enters the debug mode only after successful completion of the current transaction after which no further transaction can take place until the debug mode is exited.

58.5 Functional description

This section presents the following topics:

- *Section 58.5.1: Notes about module operation*
- *Section 58.5.2: Transactions*
- *Section 58.5.3: Arbitration procedure*
- *Section 58.5.4: Clock behavior*
- *Section 58.5.5: Interrupts*
- *Section 58.5.7: IPG DEBUG mode*
- *Section 58.5.8: DMA interface*

58.5.1 Notes about module operation

- The I^2C module always performs as a slave receiver by default, unless explicitly programmed to be a master or slave transmitter.
- When the I^2C module is acting as a master, it must not try to call its own slave address.

58.5.2 Transactions

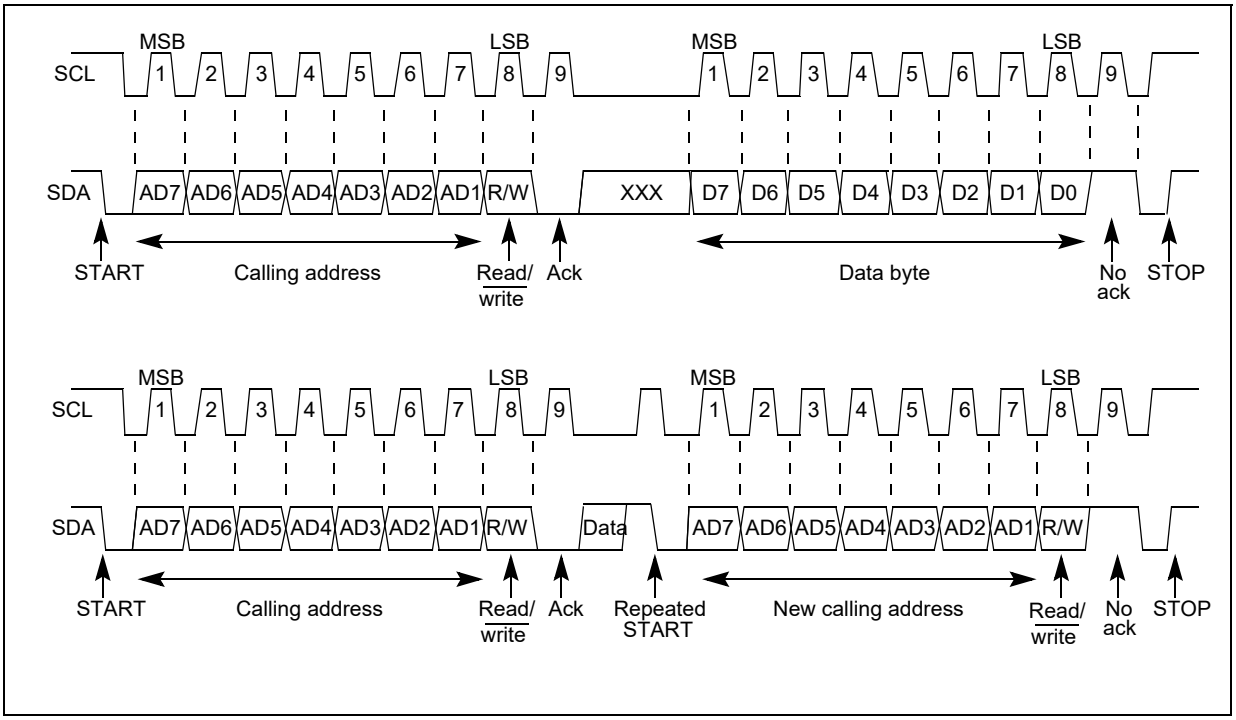
This section presents the following topics:

- *Section 58.5.2.1: Protocol overview*
- *Section 58.5.2.2: Transaction protocol definitions*
- *Section 58.5.2.4: High-level protocol steps*
- *Section 58.5.2.5: START condition*
- *Section 58.5.2.6: Slave address transmission*
- *Section 58.5.2.7: Data transmission*
- *Section 58.5.2.8: STOP condition*
- *Section 58.5.2.9: Repeated START condition*

58.5.2.1 Protocol overview

Figure 1403 shows the behavior of SCL and SDA during a typical I^2C transaction.

Figure 1403. I²C transaction protocol



58.5.2.2 Transaction protocol definitions

This section defines several important terms presented in *Figure 1403*.

Table 1450. I²C definitions

Term	Definition
START	A START condition, as defined in "*Section 58.2.6: Definition: I2C conditions*"
STOP	A STOP condition, as defined in "*Section 58.2.6: Definition: I2C conditions*"
Calling (slave) address	A seven-bit address used to identify a slave on the I²C bus. The requirements for specifying this address are presented in "*Section 58.5.2.3: I2C calling address requirements*."
Read/write (R/$\overline{W}$)	A bit that specifies the direction of the data transfer to the slave as follows: 0 The data is being transferred from the master to the slave ("write"). 1 The data is being transferred from the slave to the master ("read").
Ack	A bit that specifies the acknowledgement of a calling address, indicated by pulling SDA low.

58.5.2.3 I²C calling address requirements

The calling addresses of the devices used on an I²C network are subject to the following requirements:

- Each slave must have a unique calling address.
- A master must not transmit a calling address that is the same as its own slave address.

58.5.2.4 High-level protocol steps

The I^2C protocol conceptually supports two types of transfers, which are illustrated in *Figure 1403*. The significant steps in these transfers are presented in *Table 1451*. Details of each of these steps are presented in subsequent sections.

Table 1451. I^2C high-level protocol steps

Standard transfer	Repeated START transfer
1. START condition 2. Slave target or general call address transmission 3. Acknowledgment from slave 4. Data transfer 5. STOP condition 6. (repeat Steps 1–4)	1. START condition 2. Slave target or general call address transmission 3. Acknowledgment from slave 4. Data transfer 5. Repeated START condition 6. (repeat Steps 2–4 as needed) 7. STOP condition 8. (repeat Steps 1–7)

58.5.2.5 START condition

When the bus is free, that is no master device is engaging the bus (both SCL and SDA lines are at logical high), a master may initiate communication by sending a START condition (refer to *Section 58.2.6*). This signal denotes the beginning of a new data transfer (each data transfer may contain several bytes of data) and brings all slaves out of their idle states.

58.5.2.6 Slave address transmission

The master transmits the slave address on the next clock cycle after the START condition (refer to *Section 58.5.2.5*). The process of slave address transmission is presented in *Table 1452*.

Table 1452. Slave address transmission process

Step	Action
1	The master transmits the seven-bit slave address.
2	The master transmits the R/$\overline{W}$ bit.
3	Each slave examines the transmitted address and compares it to its own. If the addresses match, the slave device returns the acknowledge bit on the ninth SCL clock cycle.
4	The master waits for the acknowledge bit and determines the next step as follows: – The acknowledge bit is set: The master must initiate a data transfer followed by either a STOP condition or a repeated START condition. – The acknowledge bit is cleared: The master must wait for SCL to return to logic zero.

58.5.2.7 Data transmission

A data transfer session has the following characteristics:

- Can transmit one or more bytes of data
- Awakens all slaves
- Proceeds on a byte-by-byte basis in the direction specified by the R/$\overline{W}$ bit sent by the calling master

The transmitted data is subject to the following requirements:

- Each data byte must consist of 8 bits.
- Data bits can be changed only while SCL is low and must be held stable while SCL is high.
- One data bit is transmitted during one SCL clock pulse.
- The most significant bit (msb) must be transmitted first.
- Each data byte must be followed by an acknowledge bit on the ninth SCL clock pulse.

If the slave receiver does not acknowledge the master, the SDA line must be left high by the slave. The master can then generate a stop condition to abort the data transfer or a START condition (repeated START) to begin a new calling.

If the master receiver does not acknowledge the slave transmitter after a byte of transmission, the slave interprets that the end-of-data has been reached. Then the slave releases the SDA line for the master to generate a STOP or a START condition.

58.5.2.8 STOP condition

The master can terminate the communication by generating a STOP condition (refer to *Section 58.2.6*). It can do so even if the slave has generated an acknowledge, at which point the slave must release the bus.

A master is not required to send a STOP condition at the end of every transfer. For more information, refer to *Section 58.5.2.9*.

58.5.2.9 Repeated START condition

The I^2C protocol also supports a repeated START condition, which can be generated without a preceding STOP condition. A master device can use this condition to communicate with another slave or with the same slave in a different mode without releasing the bus. This condition is illustrated in the second timing diagram of *Figure 1403*.

58.5.3 Arbitration procedure

The I^2C bus is a true multi-master bus that allows more than one master to be connected to it. If two or more masters try to control the bus at the same time, a clock synchronization procedure determines the bus clock, for which the low period is equal to the longest clock low period and the high is equal to the shortest one among the masters. The relative priority of the contending masters is determined by a data arbitration procedure. A bus master loses arbitration if it transmits logic “1” while another master transmits logic “0”. The losing masters immediately switch over to slave receive mode and stop driving the SDA output. In this case, the transition from master to slave mode does not generate a STOP condition. Meanwhile, a status bit is set by hardware to indicate loss of arbitration.

58.5.4 Clock behavior

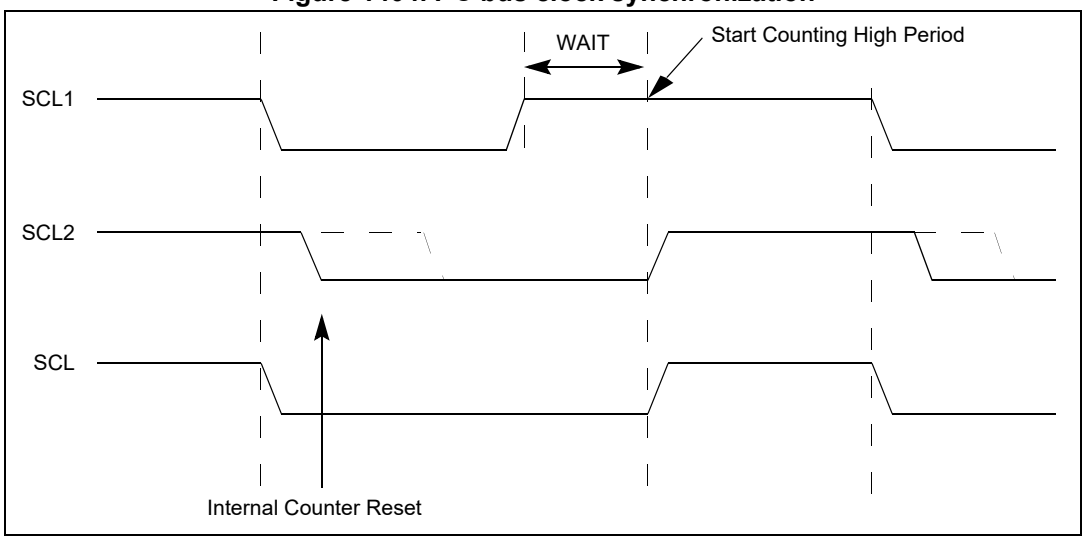
This section presents the following topics:

- *Section 58.5.4.1: Clock synchronization*
- *Section 58.5.4.2: Clock stretching*
- *Section 58.5.4.3: Handshaking*
- *Section 58.5.4.4: Clock rate and IBFD settings*

58.5.4.1 Clock synchronization

Due to the wired-AND logic on the SCL line, a high-to-low transition on the SCL line affects all devices connected on the bus. The devices begin counting their low period when the master drives the SCL line low. After a device has driven SCL low, it holds the SCL line low until the clock high state is reached. However, the change of low-to-high in a device clock may not change the state of the SCL line if another device is still within its low period. Therefore, the synchronized clock signal, SCL, is held low by the device with the longest low period. Devices with shorter low periods enter a high wait state during this time (refer to *Figure 1404*). When all devices concerned have counted off their low period, the synchronized SCL line is released and pulled high. Then there is no difference between the devices' clocks and the state of the SCL line, and all the devices begin counting their high periods. The first device to complete its high period pulls the SCL line low again.

Figure 1404. I²C bus clock synchronization



58.5.4.2 Clock stretching

The clock synchronization mechanism can be used by slaves to slow down the bit rate of a transfer. After the master has driven SCL low, the slave can drive SCL low for the required period and then release it. If the slave SCL low period is greater than the master SCL low period then the resulting SCL bus signal low period is stretched.

58.5.4.3 Handshaking

The clock synchronization mechanism can be used as a handshake in data transfer. Slave devices may hold the SCL low after completion of one byte transfer (9 bits). In such cases, it halts the bus clock and forces the master clock into wait state until the slave releases the SCL line.

58.5.4.4 Clock rate and IBFD settings

The following tables describe the settings of several fields in the IBFD register (refer to *Section 58.4.4: I2C Bus Frequency Divider register (IBFD)*).

Table 1453. I-Bus multiplier factor

IBC[0:1]	MUL
00	01
01	02
10	04
11	Reserved

Table 1454. I-Bus prescaler divider values

IBC[2:4]	scl2start (clocks)	scl2stop (clocks)	scl2tap (clocks)	tap2tap (clocks)
000	2	7	4	1
001	2	7	4	2
010	2	9	6	4
011	6	9	6	8
100	14	17	14	16
101	30	33	30	32
110	62	65	62	64
111	126	129	126	128

Table 1455. I-Bus tap and prescale values

IBC[5:7]	SCL Tap (clocks)	SDA Tap (clocks)
000	5	1
001	6	1
010	7	2
011	8	2
100	9	3
101	10	3
110	12	4
111	15	4

The number of clocks from the falling edge of SCL to the first tap (Tap[1]) is defined by the values shown in the scl2tap column of *Table 1454*. All subsequent tap points are separated by 2^{IBC} as shown in the tap2tap column in *Table 1454*. The SCL Tap is used to generate the SCL period and the SDA Tap is used to determine the delay from the falling edge of SCL to the change of state of SDA that is the SDA Hold time.

Figure 1405. SDA Hold time

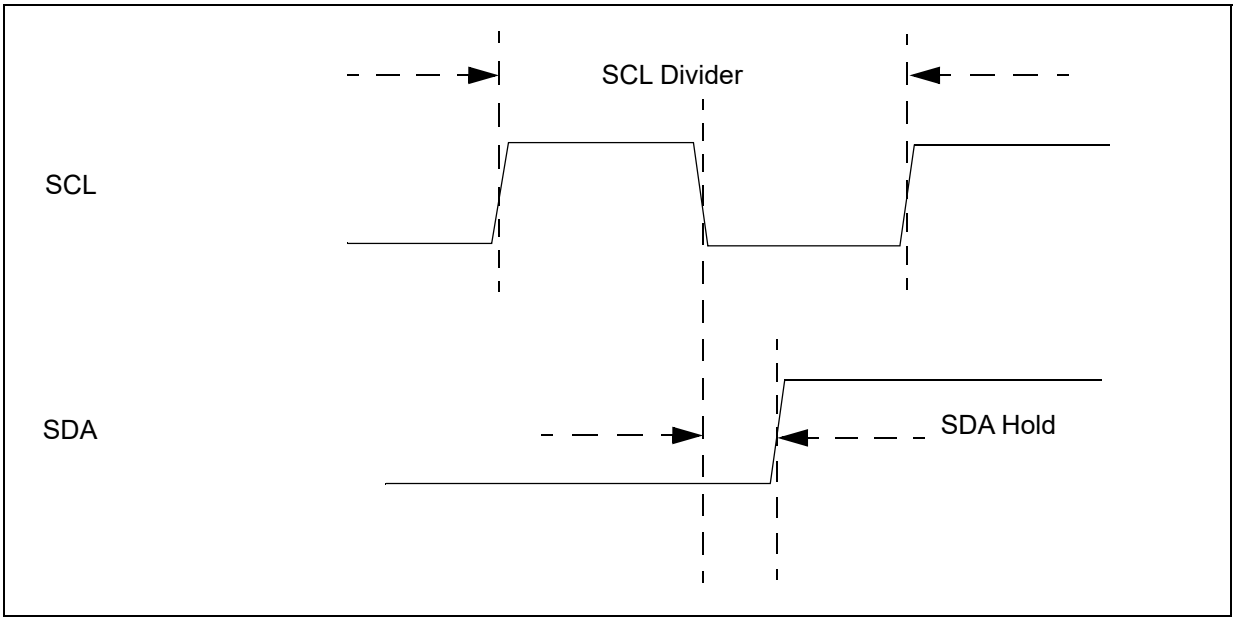
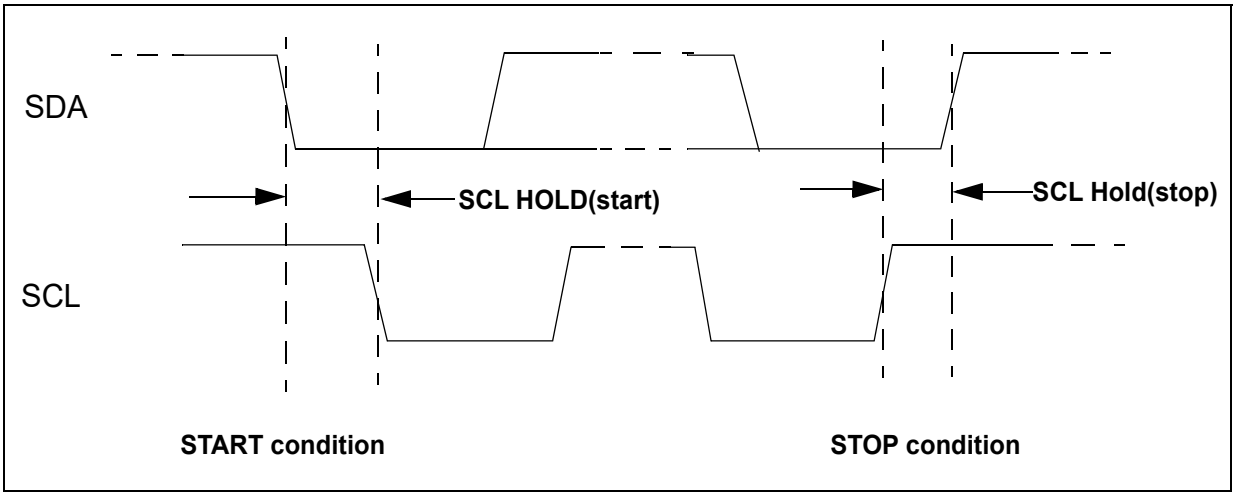


Figure 1406. SCL divider and SDA Hold



The equation used to generate the divider values from the IBFD bits is:

Equation 67 SCL Divider = MUL x {2 x (scl2tap + [(SCL_Tap -1) x tap2tap] + 2)}

The SDA Hold delay is equal to the CPU clock period multiplied by the SDA Hold value shown in *Table 1456*. The equation used to generate the SDA Hold value from the IBFD bits is:

Equation 68 SDA Hold = MUL x {scl2tap + [(SDA_Tap - 1) x tap2tap] + 3}

The equations for SCL Hold values to generate the START and STOP conditions from the IBFD bits are:

Equation 69 SCL Hold(start) = MUL x [scl2start + (SCL_Tap - 1) x tap2tap]

Equation 70 SCL Hold(stop) = MUL x [scl2stop + (SCL_Tap - 1) x tap2tap]

Table 1456. I^2C divider and hold values

MUL	IBC (Hex)	SCL divider (Clocks)	SDA Hold (Clocks)	SCL Hold (Start)	SCL Hold (Stop)
1	0x00	20	7	6	11
1	0x01	22	7	7	12
1	0x02	24	8	8	13
1	0x03	26	8	9	14
1	0x04	28	9	10	15
1	0x05	30	9	11	16
1	0x06	34	10	13	18
1	0x07	40	10	16	21
1	0x08	28	7	10	15
1	0x09	32	7	12	17
1	0x0A	36	9	14	19
1	0x0B	40	9	16	21
1	0x0C	44	11	18	23
1	0x0D	48	11	20	25
1	0x0E	56	13	24	29
1	0x0F	68	13	30	35
1	0x10	48	9	18	25
1	0x11	56	9	22	29
1	0x12	64	13	26	33
1	0x13	72	13	30	37
1	0x14	80	17	34	41
1	0x15	88	17	38	45
1	0x16	104	21	46	53
1	0x17	128	21	58	65
1	0x18	80	9	38	41
1	0x19	96	9	46	49
1	0x1A	112	17	54	57
1	0x1B	128	17	62	65
1	0x1C	144	25	70	73
1	0x1D	160	25	78	81
1	0x1E	192	33	94	97
1	0x1F	240	33	118	121
1	0x20	160	17	78	81

Table 1456. I²C divider and hold values (continued)

MUL	IBC (Hex)	SCL divider (Clocks)	SDA Hold (Clocks)	SCL Hold (Start)	SCL Hold (Stop)
1	0x21	192	17	94	97
1	0x22	224	33	110	113
1	0x23	256	33	126	129
1	0x24	288	49	142	145
1	0x25	320	49	158	161
1	0x26	384	65	190	193
1	0x27	480	65	238	241
1	0x28	320	33	158	161
1	0x29	384	33	190	193
1	0x2A	448	65	222	225
1	0x2B	512	65	254	257
1	0x2C	576	97	286	289
1	0x2D	640	97	318	321
1	0x2E	768	129	382	385
1	0x2F	960	129	478	481
1	0x30	640	65	318	321
1	0x31	768	65	382	385
1	0x32	896	129	446	449
1	0x33	1024	129	510	513
1	0x34	1152	193	574	577
1	0x35	1280	193	638	641
1	0x36	1536	257	766	769
1	0x37	1920	257	958	961
1	0x38	1280	129	638	641
1	0x39	1536	129	766	769
1	0x3A	1792	257	894	897
1	0x3B	2048	257	1022	1025
1	0x3C	2304	385	1150	1153
1	0x3D	2560	385	1278	1281
1	0x3E	3072	513	1534	1537
1	0x3F	3840	513	1918	1921
2	0x40	40	14	12	22
2	0x41	44	14	14	24
2	0x42	48	16	16	26

Table 1456. I²C divider and hold values (continued)

MUL	IBC (Hex)	SCL divider (Clocks)	SDA Hold (Clocks)	SCL Hold (Start)	SCL Hold (Stop)
2	0x43	52	16	18	28
2	0x44	56	18	20	30
2	0x45	60	18	22	32
2	0x46	68	20	26	36
2	0x47	80	20	32	42
2	0x48	56	14	20	30
2	0x49	64	14	24	34
2	0x4A	72	18	28	38
2	0x4B	80	18	32	42
2	0x4C	88	22	36	46
2	0x4D	96	22	40	50
2	0x4E	112	26	48	58
2	0x4F	136	26	60	70
2	0x50	96	18	36	50
2	0x51	112	18	44	58
2	0x52	128	26	52	66
2	0x53	144	26	60	74
2	0x54	160	34	68	82
2	0x55	176	34	76	90
2	0x56	208	42	92	106
2	0x57	256	42	116	130
2	0x58	160	18	76	82
2	0x59	192	18	92	98
2	0x5A	224	34	108	114
2	0x5B	256	34	124	130
2	0x5C	288	50	140	146
2	0x5D	320	50	156	162
2	0x5E	384	66	188	194
2	0x5F	480	66	236	242
2	0x60	320	34	156	162
2	0x61	384	34	188	194
2	0x62	448	66	220	226
2	0x63	512	66	252	258
2	0x64	576	98	284	290

Table 1456. I²C divider and hold values (continued)

MUL	IBC (Hex)	SCL divider (Clocks)	SDA Hold (Clocks)	SCL Hold (Start)	SCL Hold (Stop)
2	0x65	640	98	316	322
2	0x66	768	130	380	386
2	0x67	960	130	476	482
2	0x68	640	66	316	322
2	0x69	768	66	380	386
2	0x6A	896	130	444	450
2	0x6B	1024	130	508	514
2	0x6C	1152	194	572	578
2	0x6D	1280	194	636	642
2	0x6E	1536	258	764	770
2	0x6F	1920	258	956	96
2	0x70	1280	130	636	642
2	0x71	1536	130	764	770
2	0x72	1792	258	892	898
2	0x73	2048	258	1020	1026
2	0x74	2304	386	1148	1154
2	0x75	2560	386	1276	1282
2	0x76	3072	514	1532	1538
2	0x77	3840	514	1916	1922
2	0x78	2560	258	1276	1282
2	0x79	3072	258	1532	1538
2	0x7A	3584	514	1788	1794
2	0x7B	4096	514	2044	2050
2	0x7C	4608	770	2300	2306
2	0x7D	5120	770	2556	2562
2	0x7E	6144	1026	3068	3074
2	0x7F	7680	1026	3836	3842
4	0x80	80	28	24	44
4	0x81	88	28	28	48
4	0x82	96	32	32	52
4	0x83	104	32	36	56
4	0x84	112	36	40	60
4	0x85	120	36	44	64
4	0x86	136	40	52	72

Table 1456. I^2C divider and hold values (continued)

MUL	IBC (Hex)	SCL divider (Clocks)	SDA Hold (Clocks)	SCL Hold (Start)	SCL Hold (Stop)
4	0x87	160	40	64	84
4	0x88	112	28	40	60
4	0x89	128	28	48	68
4	0x8A	144	36	56	76
4	0x8B	160	36	64	84
4	0x8C	176	44	72	92
4	0x8D	192	44	80	100
4	0x8E	224	52	96	116
4	0x8F	272	52	120	140
4	0x90	192	36	72	100
4	0x91	224	36	88	116
4	0x92	256	52	104	132
4	0x93	288	52	120	148
4	0x94	320	68	136	164
4	0x95	352	68	152	180
4	0x96	416	84	184	212
4	0x97	512	84	232	260
4	0x98	320	36	152	164
4	0x99	384	36	184	196
4	0x9A	448	68	216	228
4	0x9B	512	68	248	260
4	0x9C	576	100	280	292
4	0x9D	640	100	312	324
4	0x9E	768	132	376	388
4	0x9F	960	132	472	484
4	0xA0	640	68	312	324
4	0xA1	768	68	376	388
4	0xA2	896	132	440	452
4	0xA3	1024	132	504	516
4	0xA4	1152	196	568	580
4	0xA5	1280	196	632	644
4	0xA6	1536	260	760	772
4	0xA7	1920	260	952	964
4	0xA8	1280	132	632	644

Table 1456. I²C divider and hold values (continued)

MUL	IBC (Hex)	SCL divider (Clocks)	SDA Hold (Clocks)	SCL Hold (Start)	SCL Hold (Stop)
4	0xA9	1536	132	760	772
4	0xAA	1792	260	888	900
4	0xAB	2048	260	1016	1028
4	0xAC	2304	388	1144	1156
4	0xAD	2560	388	1272	1284
4	0xAE	3072	516	1528	1540
4	0xAF	3840	516	1912	1924
4	0x30	2560	260	1272	1284
4	0xB1	3072	260	1528	1540
4	0xB2	3584	516	1784	1796
4	0xB3	4096	516	2040	2052
4	0xB4	4608	772	2296	2308
4	0xB5	5120	772	2552	2564
4	0xB6	6144	1028	3064	3076
4	0xB7	7680	1028	3832	3844
4	0xB8	5120	516	2552	2564
4	0xB9	6144	516	3064	3076
4	0xBA	7168	1028	3576	3588
4	0xBB	8192	1028	4088	4100
4	0xBC	9216	1540	4600	4612
4	0xBD	10240	1540	5112	5124
4	0xBE	12288	2052	6136	6148
4	0xBF	15360	2052	7672	7684

58.5.5 Interrupts

This section presents the following topics:

- *Section 58.5.5.1: Interrupt vector*
- *Section 58.5.5.2: Interrupt description*

58.5.5.1 Interrupt vector

The I²C module uses only one interrupt vector.

Table 1457. Interrupt summary

Interrupt	Offset	Vector	Priority	Source	Description
I^2C Interrupt	—	—	—	IBAL, TCF, IAAS, IBB bits in IBSR register	When any of IBAL, TCF or IAAS bits is set, an interrupt may be caused based on Arbitration lost, Transfer Complete or Address Detect conditions. If enabled by BIIE, the de-assertion of IBB can also cause an interrupt, indicating that the bus is idle.

58.5.5.2 Interrupt description

There are five types of internal interrupts in the I^2C. The interrupt service routine can determine the interrupt type by reading the Status register.

I^2C Interrupt can be generated on the following events:

- Arbitration Lost condition (IBAL bit set)
- Byte Transfer condition (TCF bit set and DMAEN bit not set)
- Address Detect condition (IAAS bit set)
- No Acknowledge from slave received when expected
- Bus Going Idle (IBB bit not set)

The I^2C interrupt is enabled by the IBCR[IBIE] bit. It must be cleared by writing '1' to the IBIF bit in the interrupt service routine. The Bus Going Idle interrupt needs to be additionally enabled by the IBIC[BIIE] bit.

58.5.6 IPG STOP mode

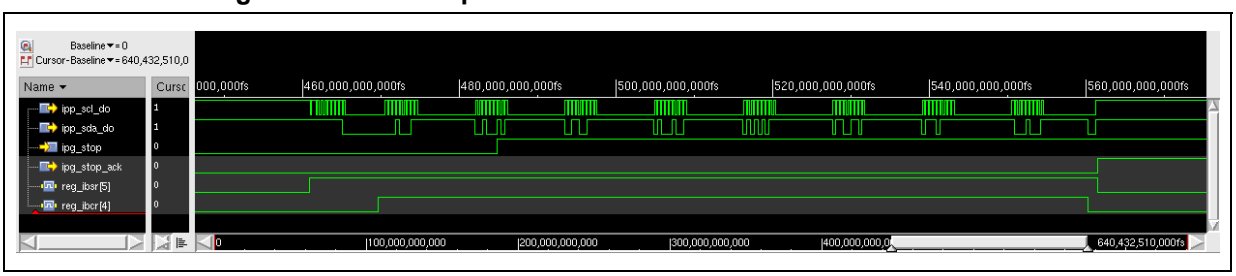
This mode allows the software to put the I^2C module in power-down state. Once the STOP request is asserted, the I^2C module comes to a graceful halt after completing all the ongoing transactions.

As soon as the I^2C module enters IPG STOP mode:

- The I^2C clock is disabled.
- No transaction can take place.
- All registers are inaccessible.

The I^2C module enters this mode only after successfully completing the current ongoing transaction, hence the low-power request is acknowledged once the STOP condition occurs. The user must ensure that the bus is free when STOP is requested. To request STOP for ongoing transmission the user must wait until the transmission is complete, followed by clearing of the IBCR[TXRX] field. Refer to *Figure 1407* for more details.

Figure 1407. I^2C stop mode behavior when master is in Rx and slave is in Tx



58.5.7 IPG DEBUG mode

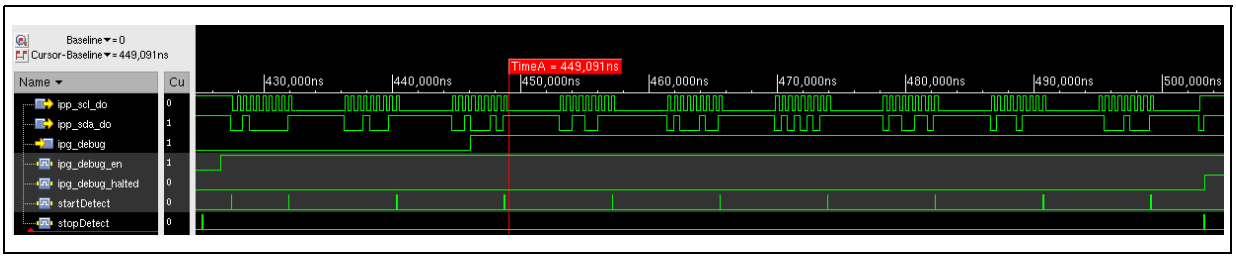
This mode allows CPU to debug the I²C by freezing all the counters, registers and status bits. Once the Debug request is asserted along with IBDBG[IPG_DEBUG_EN] bit, I²C comes to a graceful halt after completing all the ongoing transactions. A Debug halted signal IBDBG[IPG_DEBUG_HALTED] is also asserted to indicate that the debug request has been successfully serviced and is de-asserted when the Debug request is de-asserted.

As soon as the I²C module enters IPG DEBUG mode:

- No transaction can take place.
- All the registers, counters and status bits are frozen. They all can be accessed by the CPU to enable the debugging.

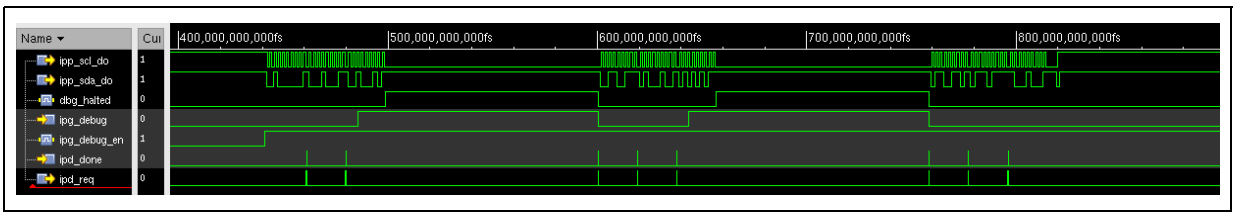
The I²C module enters this mode only after successfully completing the current ongoing transaction. For example, if the current transaction consists of 8 bytes and the debug mode was initiated by user at the time of the second byte, the IPG Debug Halted signal will be asserted after the 8th byte is transmitted/received. Refer to the simulation result in *Figure 1408* for more details.

Figure 1408. Simulation result of IPG Debug Halted in case of frame transmission



If any DMA transaction (transmit or receive) is in progress, and if the debug mode is requested while a byte is being transmitted or received, the I²C module enters IPG DEBUG mode after completing the transmission or reception of the current byte. As soon as the module exits IPG DEBUG mode, transmission or reception of the remaining bytes resumes. Refer to the simulation snapshot shown in *Figure 1409* for details where the I²C is transmitting 8 bytes using DMA and IPG DEBUG mode is requested while a second byte is being transmitted. IPG DEBUG mode is entered after successfully transmitting the second byte (IPG Debug Halted signal asserted). As soon as the module exits IPG DEBUG mode (IPG Debug Halted signal de-asserted), transmission resumes successfully.

Figure 1409. Simulation result of IPG Debug Halted in case of frame transmission using DMA



No more transaction can take place until the debug signal is de-asserted after which the I²C module starts functioning normally. There is a status halted signal IBDBG[IPG_DEBUG_HALTED] to indicate to the user that the I²C module has entered the IPG DEBUG mode. Refer to *Figure 1410* for more details.

Figure 1410. PG DEBUG mode

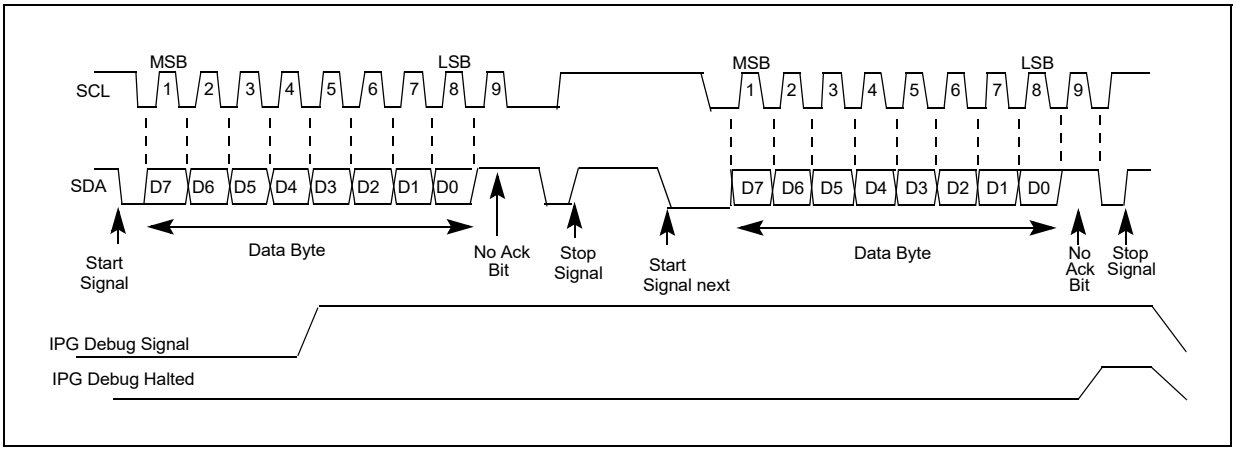
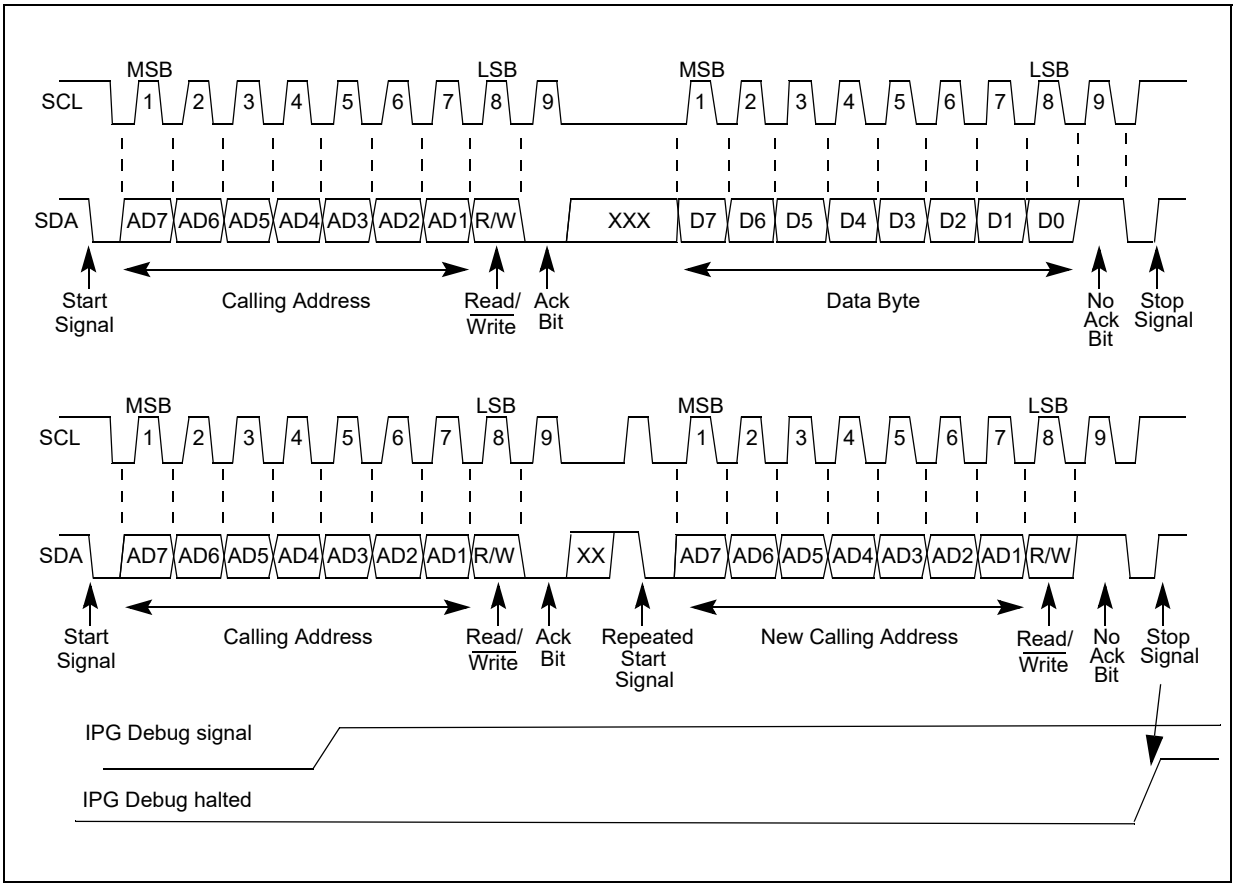


Figure 1411 shows a case of IPG DEBUG mode with repeated START transaction. In this case, if the debug signal is asserted in between the transaction, the entire transaction with multiple repeated start will be completed before the I^2C module enters IPG DEBUG mode.

Figure 1411. IPG debug mode with repeated start



58.5.8 DMA interface

A simple DMA interface is implemented so that the I^2C can request data transfers with minimal support from the CPU (refer to *Section 58.6.5*). DMA mode is enabled by setting bit 1 in the Control Register (IBCR).

The DMA interface is operational when the I^2C module is configured for Master mode.

At least three bytes of data per frame must be transferred from/to the slave when using DMA mode, although in practice it will only be worthwhile using the DMA mode when there is a large number of data bytes to transfer per frame.

Two internal signals, TX request and RX request, are used to signal the DMA controller when the I^2C module requires data to be written or read from the data register.

Further details of the DMA interface can be found in the *Section 58.6*.

58.6 Initialization and application information

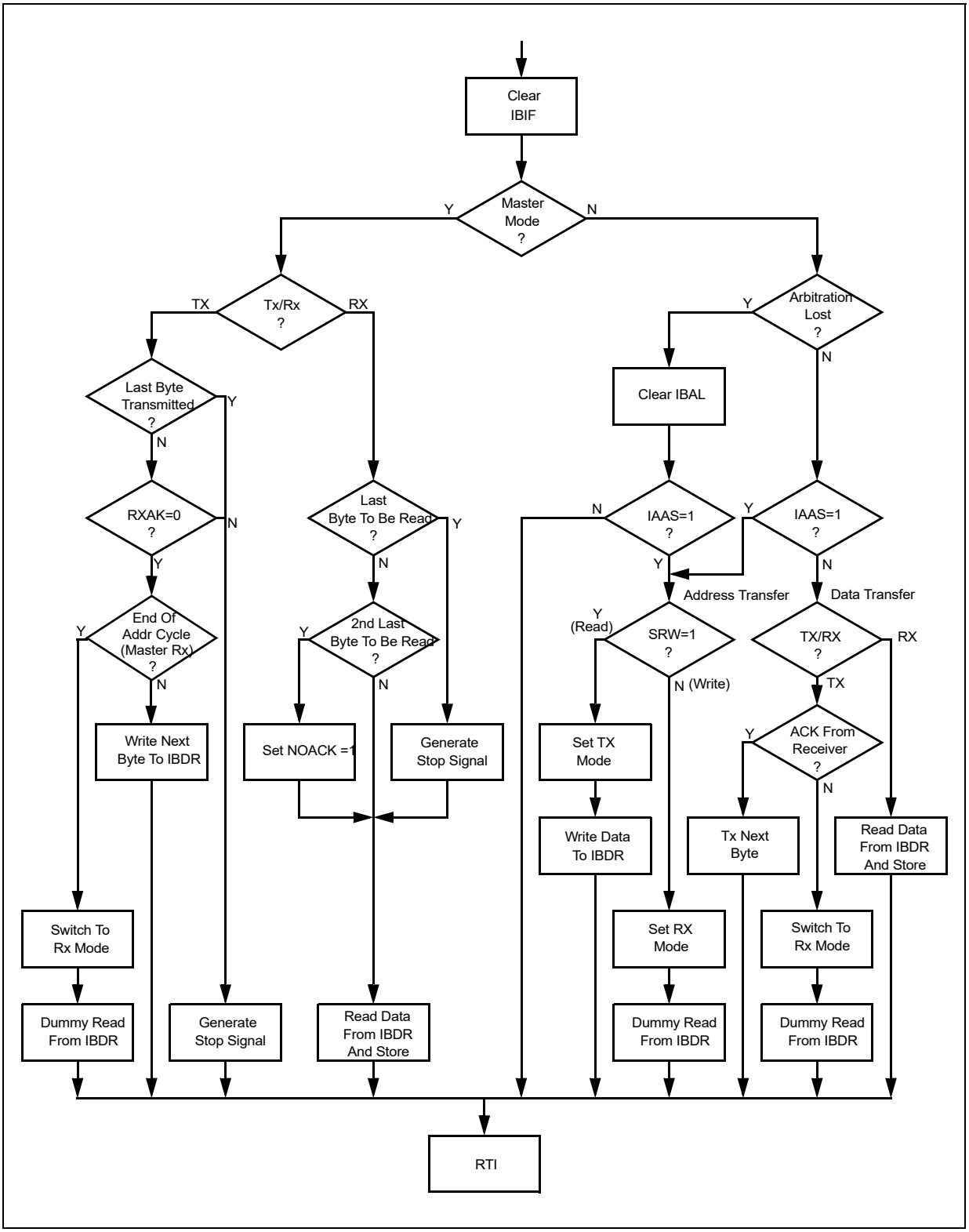
This section presents the following topics:

- *Section 58.6.1: Recommended interrupt service flow*
- *Section 58.6.2: General programming guidelines (for both master and slave mode)*
- *Section 58.6.3: Programming guidelines specific to master mode*
- *Section 58.6.4: Programming guidelines specific to slave mode*
- *Section 58.6.5: DMA application information*

58.6.1 Recommended interrupt service flow

Figure 1412 shows a flowchart for the recommended I^2C interrupt service routine. Deviation from the flowchart may result in unpredictable I^2C bus behavior.

Figure 1412. Recommended I²C interrupt service routine flowchart



58.6.2 General programming guidelines (for both master and slave mode)

This section provides programming guidelines recommended for the I^{2}C module in both master and slave mode. It presents the following topics:

- *Section 58.6.2.1: Initializing the I2C module*
- *Section 58.6.2.2: Software response after a transfer*

58.6.2.1 Initializing the I^{2}C module

The following sequence initializes the I^{2}C module:

1. Use the IBFD register to select the required division ratio to obtain SCL frequency from system clock.
2. Use the IBAD register to define the slave address.
3. Clear the IBCR[IBDIS] bit to enable the I^{2}C interface system.
4. Use the IBCR to select Master/Slave mode, Transmit/Receive mode and interrupt enable or not.
5. (Optional) Use the IBIC register to further refine the interrupt behavior.

58.6.2.2 Software response after a transfer

Transmission or reception of a byte will set the data transferring bit (TCF) to 1, which indicates one byte communication is finished. The I^{2}C Bus interrupt bit (IBIF) is set also; an interrupt will be generated if the interrupt function is enabled during initialization by setting the IBIE bit. The IBIF (interrupt flag) can be cleared by writing one (in the interrupt service routine, if interrupts are used).

The TCF bit will be cleared to indicate data transfer in progress whenever data register is written to in transmit mode, or during reading out from data register in receive mode. The TCF bit should not be used as a data transfer complete flag as the flag timing is dependent on a number of factors including the I^{2}C bus frequency. This bit may not conclusively provide an indication of a transfer complete situation. It is recommended that transfer complete situations are detected using the IBIF flag.

Software may service the I^{2}C I/O in the main program by monitoring the IBIF bit if the interrupt function is disabled. Polling should monitor the IBIF bit rather than the TCF bit since their operation is different when arbitration is lost.

When a “Transfer Complete” interrupt occurs at the end of the address cycle, the master will always be in transmit mode, that is the address is transmitted. If master receive mode is required, indicated by R/W bit sent with slave calling address, then the Tx/Rx bit at Master side should be toggled at this stage. If Master does not receive an ACK from Slave, then transmission must be re-initiated or terminated.

In slave mode, IAAS bit will get set in IBSR if Slave address (IBAD) matches the Master calling address. This is an indication that Master-Slave data communication can now start. During address cycles (IBSR[IAAS]=1), the SRW bit in the status register is read to determine the direction of the subsequent transfer and the Tx/Rx bit is programmed accordingly. For slave mode data cycles (IBSR[IAAS]=0), the SRW bit is not valid. The Tx/Rx bit in the control register should be read to determine the direction of the current transfer.

58.6.3 Programming guidelines specific to master mode

This section presents the following topics:

- *Section 58.6.3.1: Generating START*
- *Section 58.6.3.2: Transmit/receive sequence*
- *Section 58.6.3.3: Generating STOP*
- *Section 58.6.3.4: Generating repeated START*
- *Section 58.6.3.5: Loss of arbitration*

58.6.3.1 Generating START

After completion of the initialization procedure, serial data can be transmitted by selecting the 'master transmitter' mode. If the device is connected to a multi-master bus system, the state of the I^2C Bus Busy bit (IBB) must be tested to check whether the serial bus is free.

If the bus is free (IBB = 0), the start condition and the first byte (the slave address) can be sent. The data written to the data register comprises the slave calling address and the LSB, which is set to indicate the direction of transfer required from the slave.

The bus free time (that is the time between a STOP condition and the following START condition) is built into the hardware that generates the START cycle. Depending on the relative frequencies of the system clock and the SCL period, it may be necessary to wait until the I^2C is busy after writing the calling address to the IBDR before proceeding with the following instructions. This is illustrated in the following example.

An example of the sequence of events which generates the START signal and transmits the first byte of data (slave address) is shown below:

```
while (IBSR[IBB]==1)                  // wait in loop for IBB flag to clear
IBCR[MS/MSL] and IBCR[]Tx/Rx] = 1      // master and transmit mode, that is,
                                      // generate start condition
IBDR = calling_address                // send the calling address to the data
register
while (bit 5, IBSR ==0)               // wait in loop for IBB flag to be set
```

58.6.3.2 Transmit/receive sequence

The following tables present the sequences for:

- Master transmit
- Master receive
- Slave transmit
- Slave receive

Table 1458. Master transmit sequence

Step	Action
a	Use the IBFD register to select the required division ratio to obtain SCL frequency from Platform clock/2.
b	Write 0 to IBCR[IBDIS] to enable the I^2C interface system.
c	Use the IBCR to select Master mode, Transmit mode and interrupt enable.
d	Write 0 to IBSR[IBIF].

Table 1458. Master transmit sequence (continued)

Step	Action
e	Write data to IBDR.
f	Observe changes in IBSR[TCF]: – When IBSR[TCF] becomes 0, the transfer is in progress. – When IBSR[TCF] becomes 1, the transfer is complete.
g	Wait until IBSR[IBIF] = 1.
h	Read the fields in the IBSR to determine what happened: – If TCF = 1, the transfer completed. – If RXAK = 1, a No Acknowledge condition occurred. – If IBB = 0, the bus transitioned from Busy to Idle state. – If IBB = 1, you can ignore check of Arbitration Loss (IBAL = 1). **Note:** Address Detect (IAAS = 1) can be ignored for master mode (it is valid only for slave mode).
i	Examine IBSR[RXAK] for an acknowledgment from the slave.
j	Repeat steps d through i to transfer the next consecutive bytes of data.

Table 1459. Master receive sequence

Step	Action
a	Follow steps a through i in *Table 1458* for address dispatch.
b	Write 0 to IBSR[IBIF].
c	Write 0 to IBCR[TXRX] to select Receive mode.
d	Perform a dummy read of the IBDR to initiate the receive operation.
e	Wait until IBSR[TCF] becomes 1. (This proves that the transfer is complete.)
f	Wait until IBSR[IBIF] = 1.
g	Read the fields in the IBSR to determine what happened: – If TCF = 1, the reception completed. – If IBB = 0, the bus transitioned from Busy to Idle state. – If IBB = 1, you can ignore check of Arbitration Loss (IBAL = 1). **Note:** The No Acknowledge condition (RXAK = 1) can be ignored in receive mode.
h	Read the IBDR to determine the data received from the slave.

Table 1460. Slave transmit sequence

Step	Action
a	Write 0 to IBCR[IBDIS] to enable the I2C interface system.
b	Examine fields in the IBSR as follows: – If IAAS = 1, examine IBSR[SRW]. – If IAAS = 1 and SRW = 1, write 1 to IBCR[TXRX] to select Transmit mode.
c	Write data to IBDR.
d	Wait until IBSR[IBIF] = 1.

Table 1460. Slave transmit sequence (continued)

Step	Action
e	Wait until IBSR[RXAK] = 0.
f	Write 0 to IBSR[IBIF].
g	Repeat steps c through for the next consecutive data transfers.

Table 1461. Slave receive sequence

Step	Action
a	Write 0 to IBCR[IBDIS] to enable the I2C interface system.
b	Examine fields in the IBSR as follows: – If IAAS = 1, examine IBSR[SRW]. – If IAAS = 1 and SRW = 0, write 0 to IBCR[TXRX] to select Receive mode.
c	Write 0 to IBSR[IBIF].
d	Perform a dummy read of the IBDR to initiate the receive operation.
e	Wait until IBSR[TCF] becomes 1. (This proves that the transfer is complete.)
f	Wait until IBSR[IBIF] = 1.
g	Read the fields in the IBSR to determine what happened: – If TCF = 1, the reception completed. – If IBB = 0, the bus transitioned from Busy to Idle state. – If IBB = 1, you can ignore check of Arbitration Loss (IBAL = 1). **Note:** You can ignore the No Acknowledge condition (RXAK = 1) for receive mode.
h	Read the IBDR to determine the data received from the master.

58.6.3.3 Generating STOP

A data transfer ends with a STOP signal generated by the “master” device. A master transmitter can simply generate a STOP signal after all the data has been transmitted. The following is an example showing how a STOP condition is generated by a master transmitter.

```
if (tx_count == 0) or          // check to see if all data bytes have been
transmitted
    (bit 0, IBSR == 1) {       // or if no ACK generated
         clear bit 5, IBCR     // generate stop condition
        }
else {
IBDR = data_to_transmit        // write byte of data to DATA register
         tx_count --           // decrement counter
      }                        // return from interrupt
```

If a master receiver wants to terminate a data transfer, it must inform the slave transmitter by not acknowledging the last byte of data which can be done by setting the NOACK bit in IBCR before reading the 2nd last byte of data. Before reading the last byte of data, a STOP signal must first be generated. The following is an example showing how a STOP signal is generated by a master receiver.

```
rx_count --                  // decrease the rx counter
if (rx_count ==1)            // 2nd last byte to be read ?
   bit 3, IBCR = 1           // disable ACK
if (rx_count == 0)           // last byte to be read ?
  bit 5, IBCR = 0            // generate stop signal
else
data_received = IBDR         // read RX data and store
```

58.6.3.4 Generating repeated START

At the end of data transfer, if the master still wants to communicate on the bus, it can generate another START signal followed by another slave address without first generating a STOP signal. A program example is as shown below.

```
bit 2, IBCR = 1              // generate another start (restart)
IBDR == calling_address      // transmit the calling address
```

58.6.3.5 Loss of arbitration

If several masters try to engage the bus simultaneously, only one master wins and the others lose arbitration. The devices that lost arbitration are immediately switched to slave receive mode by the hardware. Their data output to the SDA line is stopped, but SCL is still generated until the end of the byte during which arbitration was lost. An interrupt occurs at the falling edge of the ninth clock of this transfer with IBAL = 1 and MS/SL = 0. If one master attempts to start transmission, while the bus is being engaged by another master, the hardware will inhibit the transmission, switch the MS/SL bit from 1 to 0 without generating a STOP condition, generate an interrupt to CPU, set the IBAL to indicate that the attempt to engage the bus is failed, and not set the TCF due to the loss of data during arbitration. When considering these cases, the slave service routine should test the IBAL first and the software should clear the IBAL bit if it is set.

58.6.4 Programming guidelines specific to slave mode

In the slave interrupt service routine, the module addressed as slave bit (IAAS) should be tested to check if a calling of its own address has just been received. If IAAS is set, software should set the transmit/receive mode select bit (Tx/Rx bit of IBCR) according to the R/W command bit (SRW). Writing to the IBCR clears IAAS automatically. Note that the only time IAAS is read as set is from the interrupt at the end of the address cycle where an address match occurred. Interrupts resulting from subsequent data transfers will have IAAS cleared. A data transfer may now be initiated by writing information to IBDR for slave transmits or dummy reading from IBDR in slave receive mode. The slave will drive SCL low in-between byte transfers, SCL is released when the IBDR is accessed in the required mode.

In slave transmitter routine, the received acknowledge bit (RXAK) must be tested before transmitting the next byte of data. Setting RXAK means an “end of data” signal from the master receiver, after which it must be switched from transmitter mode to receiver mode by software. A dummy read then releases the SCL line so that the master can generate a STOP signal.

58.6.5 DMA application information

The DMA interface on the I^2C is not completely autonomous and requires intervention from the CPU to start and to terminate the frame transfer. DMA mode is only valid for Master

transmit and Master receive modes. Software must ensure that the DMA enable bit in the control register is not set when the I^2C module is configured in slave mode.

The DMA controller must only transfer one byte of data per Tx/Rx request. This is because there is no FIFO on the I^2C block.

The CPU should also keep the I^2C interrupt enabled during a DMA transfer to detect the arbitration lost condition and take action to recover from this situation. The DMAEN bit in the IBCR register works as a disable for the transfer complete interrupt. This means that during normal transfers (no errors) there will always be either an interrupt or a request to the DMA controller, dependant on the setting of the DMAEN bit. All error conditions will trigger an interrupt and require CPU intervention. The address match condition will not occur in DMA mode as the I^2C should never be configured for slave operation.

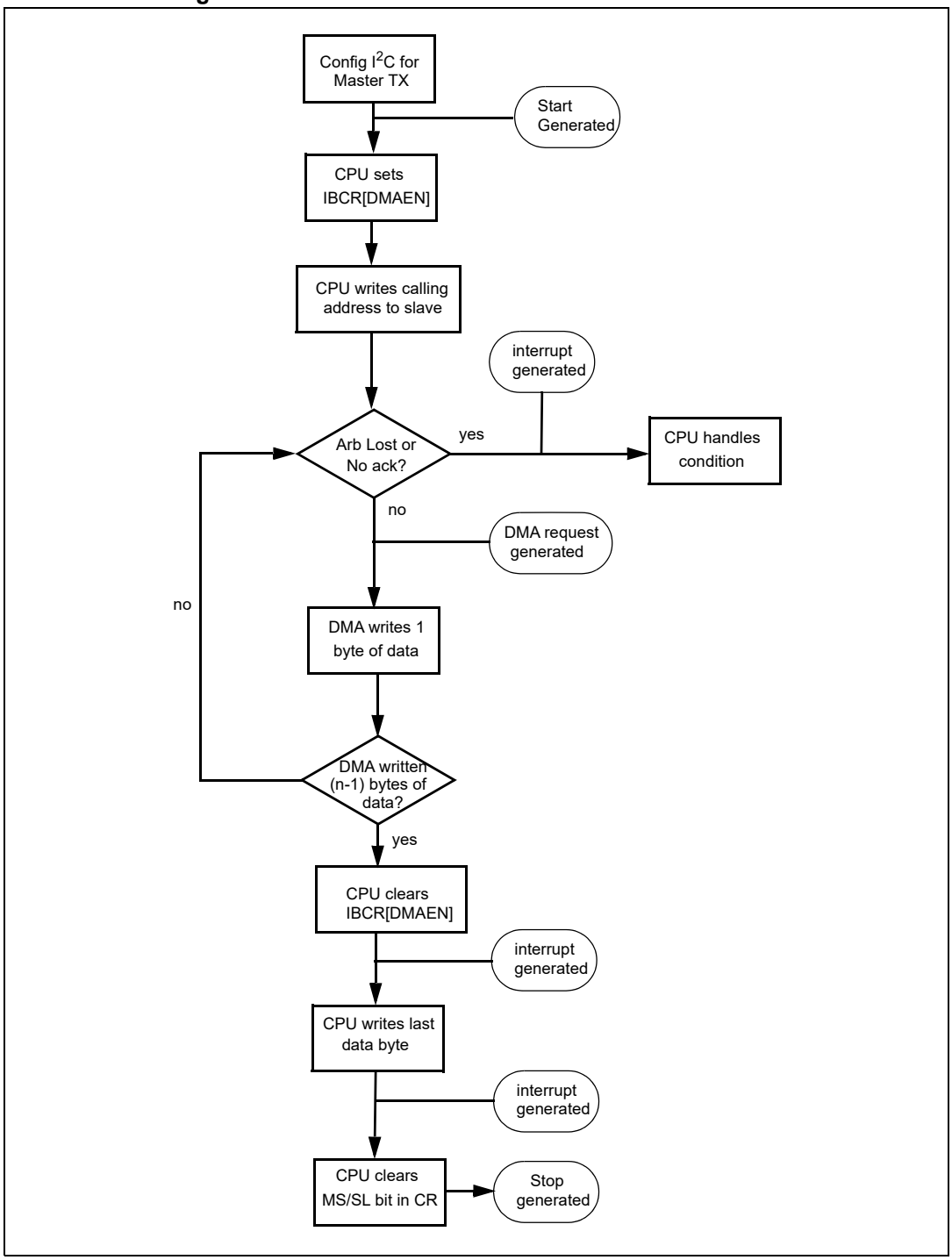
The following sections detail how to set up a DMA transfer and what intervention is required from the CPU. It is assumed that the system DMA controller is capable of generating an interrupt after a certain number of DMA transfers have taken place. The sections present the following topics:

- *Section 58.6.5.1: DMA mode, master transmit*
- *Section 58.6.5.2: DMA mode, master reception*
- *Section 58.6.5.3: Exiting DMA mode, system requirement considerations*

58.6.5.1 DMA mode, master transmit

The following flow diagram details exactly the operation for using a DMA controller to transmit ‘n’ data bytes to a slave. The first byte (the slave calling address) is always transmitted by the CPU. All subsequent data bytes (apart from the last data byte) can be transferred by the DMA controller. The last data byte must be transferred by the CPU.

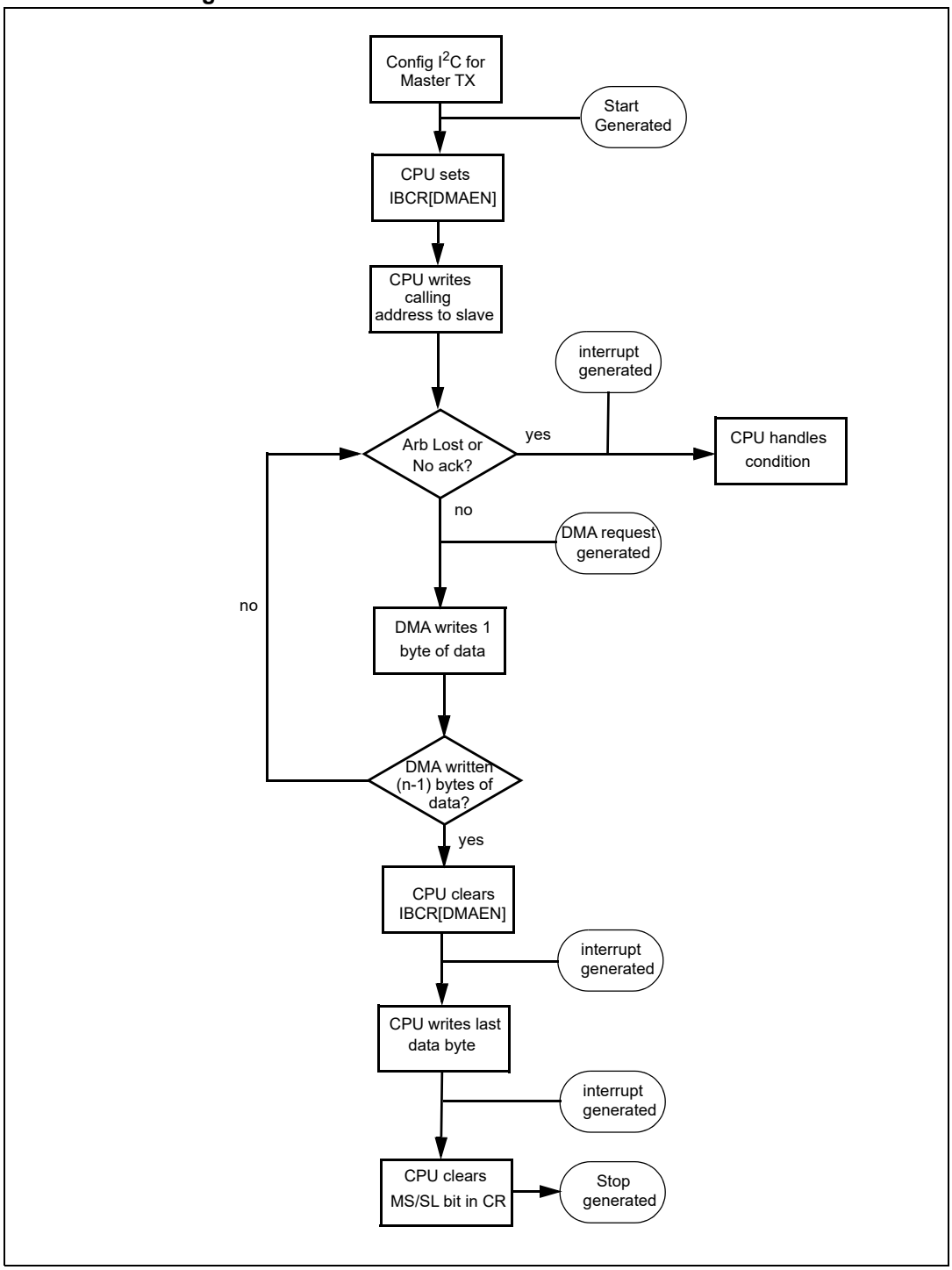
Figure 1413. Flow-Chart of DMA mode master transmit



58.6.5.2 DMA mode, master reception

The following flow diagram details the exact operation for using a DMA controller to receive ‘n’ data bytes from a slave. The first byte (the slave calling address) is always transmitted by the CPU. All subsequent data bytes (apart from the two last data bytes) can be read by the DMA controller. The last two data bytes must be transferred by the CPU.

Figure 1414. Flow-Chart of DMA mode master receive



58.6.5.3 Exiting DMA mode, system requirement considerations

As described above, the final transfers of both TX and RX transfers need to be handled via interrupt by the CPU. To change from DMA to interrupt driven transfers in the I2C module, you have to disable the DMAEN bit in the IBCR register. The trigger to exit the DMA mode is

that the programmed DMA Transfer Control Descriptor (TCD) has completed all its transfers to/from the I^2C module.

After the last DMA write (TX mode) to the I^2C the module will immediately start the next I^2C-bus transfer. The same is true for receive mode. After the DMA read from the IBDR register the module initiates the next I^2C-bus transfer. This results in two possible scenarios in the DMA mode exiting scheme.

1. Fast reaction

   The DMAEN bit is cleared before the next I^2C-bus transfer completes. In this case the module will raise an interrupt request to the CPU which can be serviced normally.

2. Slow reaction

   The DMAEN bit is cleared after the next I^2C-bus transfer has already completed. In this case, the module will not raise an interrupt request to the CPU. Instead the TCF bit can be read to determine that the transfer completed and the module is ready for further transfer.

What is fast/slow reaction?

The reaction time T_R for the system to disable DMAEN after the last DMA controller access to the I^2C is the time required for one byte transfer over the I^2C. For “fast reaction” the disabling has to occur before the 9^{th} bit of the data transfer which is the ACK bit. So the time available is eight times the SCL period.

Equation 71 $T_R = 8 \times T_{SCL}$

In fast mode, with 400 kbit/s, T_{SCL} is 2.5 µs, so T_R is 20 µs.

Depending on the system and DMA controller there are different possibilities for the de-assertion of DMAEN. Three options are:

1. CPU intervention via Interrupt

   The DMA controller is programmed to signal an interrupt to the CPU which is then responsible for the de-assertion of DMAEN. This scheme should be supported by most systems but it can result in a slow reaction time if other higher priority interrupts interfere. Therefore the interrupt handling routine can become complicated as it has to check which of the two cases happened (check TCF bit) and act accordingly. In case of slow reaction you can force an interrupt for the I^2C in the interrupt controller to have the further transfer handled by the normal I^2C interrupt routine.

Note: The use of nested interrupts can still cause potential issues in this scenario, if someone tries to stall the DMA interrupt between the de-assertion and DMAEN bit and checks the TCF bit.

2. DMA channel linking

   If the Transfer control descriptor in the DMA controller that performs the data transfer is linked to another channel that does a write to IBCR to disable the DMAEN field, this might probably be the fastest system solution, but it uses two DMA channels.

Note: Make sure, at system level, that no higher priority DMA requests occur between the two linked TCDs as they could create a slow reaction scenario again.

3. DMA scatter/gather process

   If the Transfer control descriptor in the DMA controller that performs the data transfer has the scatter/gather feature activated, this feature will initiate a reload of another TCD from system RAM after the completion of the first TCD. The new TCD will have its start bit already set and immediately start the required write to the IBCR to disable the DMAEN field. This TCD also has scatter/gather activated and is programmed to reload

the initial TCD upon completion, bringing the system back into a “ready-for-I^2C-transfer” state. The advantage over the two other solutions is that this requires neither CPU intervention nor a second DMA channel. This comes at the cost of 64 bytes RAM (two TCDs), some system bus transfer overhead and a little increase in application code complexity.

Note: Make sure, at system level, that no higher priority DMA requests occur during the scatter/gather process, as they could create a slow reaction scenario again.

Example latencies for a 32 MHz system with a full speed 32-bit AHB bus and an I^2C connected via half speed IPI bus:

- Accessing the I^2C from the DMA controller via IPI bus typically requires four cycles (consecutive accesses to the I^2C could be faster):

Equation 72 $4 \times T_{IPI} = 4 / 16 \text{ MHz} = 250 \text{ ns}$

- Reloading a new TCD (8 x 32-bit) via AHB to the DMA controller (scatter/gather process):

Equation 73 $8 \times T_{AHB} = 8 / 32 \text{ MHz} = 250 \text{ ns}$

Example latencies for a 150 MHz system with a full speed 32-bit AHB bus and an I^2C connected via half speed IPI bus:

- Accessing the I^2C from the DMA controller via IPI bus typically requires four cycles (consecutive accesses to the I2C could be faster):

Equation 74 $4 \times T_{IPI} = 4/150 \text{ MHz} = 26.6 \text{ ns}$

- Reloading a new TCD (4 x 64-bit) via AHB to the DMA controller (scatter/gather process):

Equation 75 $4 \times T_{AHD} = 4/150 \text{ MHz} = 26.6 \text{ ns}$

With the DMA scatter/gather process the required IBCR access can be done in 0.5 µs, leaving a large margin of 19.5 µs for additional system delays. In this way, the slow reaction case can be prevented. The system user needs to decide which usage model best suits his overall requirement.

59 SENT Receiver (SRX)

59.1 Introduction

The Single Edge Nibble Transmission (SENT) Receiver (SRX) module is a multi-channel receiver for receiving serial data frames which are being transmitted by a sensor implementing the SENT encoding scheme and present them to the CPU for further processing.

This module is based on the SAE J2716 information report titled “SENT - Single Edge Nibble Transmission for Automotive Applications” and released on January 27, 2010 (SAE web site), Apr2007 and Feb2008. As per this standard, the SENT protocol is intended for use in applications where high resolution sensor data needs to be communicated from a sensor to an Engine Control Unit (ECU). It is intended as a replacement for the lower resolution methods of 10 bit A/Ds and PWM and as a simpler low-cost alternative to CAN or LIN. The implementation assumes that the sensor is a smart sensor containing a microprocessor or dedicated logic device (ASIC) to create the signal.

The SENT encoding scheme is a unidirectional communications scheme from the sensor / transmitting device to the controller /receiving device which does not include a coordination signal from the controller/receiving device. The sensor signal is transmitted as a series of pulses with data encoded as falling to falling edge periods.

59.1.1 Features

The SENT receiver module has the following features:

- The number of SENT channels supported are device-specific. Refer to the Device Configuration chapter for details.
- Unified message read logic to transfer all messages received to system memory via interrupts or DMA.
  - Programmable option per channel to support reading of messages via DMA or Interrupt.
  - Internal round robin arbitration sequencer to loop across channels to read messages from each channel leaving CPU/DMA transparent to this process.
  - All received messages are stored excluding the synchronization pulse (for Fast messages) and identifiers (for Slow serial messages).
  - Separate buffers for storing Fast and Slow serial messages.
  - Separate DMA and Interrupts generated for Fast and Slow serial messages.
- Supports compensation for variation in SENT Tx Clock up to ±25% and adjusts its measurement for drift in the Transmitter and Receiver clocks per channel.
  - Calibration pulse correction factor per channel is available for use by software.
- Implements for each channel all receiver diagnostics as specified by the SAE specifications.
- Internal 4-bit CRC calculator that can be configured to compute CRC in both Legacy and Recommended method (as given in SAE SENT specification) for both Fast and Short serial message per channel.
- Internal 6-bit CRC calculator for Enhanced serial messages per channel.
- Supports all bit rates by having per channel configurable Rx clock periods from 1.27 µs to 90 µs.
- Support for configurable number of data nibbles per channel.
- Option to disable individual channel via programmable control bit.
- Time stamping on start of each valid message (such as, without errors) across all channels. This time stamp is appended to each message for synchronization of messages and for application to understand how much time has elapsed between successive sensor readings.
- Support for pause pulse per channel. This can be enabled by programming a bit in the control register for that channel.
- Input Programmable Filter on each channel to filter any glitches on input. This filter duration is programmable by the user software.
- 32-bit register interface for programming the module's registers and reading the received messages.
- Works on two clocks, one for accurate receive operations (high frequency receiver clock or protocol clock) and other for message read logic (system bus clock) and the module can work irrespective of the frequency relationship between the two clocks. See *Section 59.5: Clocks and resets* for more details on the clocks.

59.1.2 Modes of operation

59.1.2.1 Debug mode

In debug mode, message reception will be halted and all existing status and messages will be retained. DBG_FRZ bit is required to be set for this mode. On debug exit, a calibration length error (CAL_LEN_ERR) can be reported and the channels will start receiving messages from a valid calibration pulse.

59.1.2.2 Low power modes

SENT Receiver behavior is affected by chip-specific low power modes (for example, STOP or WAIT modes). These low power modes should either gate-off both the system bus clock and the protocol clocks or reduce their frequencies, depending on the application.

On exiting from these low power modes, calibration length error (CAL_LEN_ERR) can get set in channel status registers as the SENT sensor (transmitter) could be in the middle of a message transmission at that point. Thus, software should clear all error status bits after exiting the low power modes.

Modes where both system bus clock and protocol clock are gated:

In this case, both message reception and interrupt / DMA request generation will halt. The internal receiver channels will freeze (since clock is not there) and will not receive any messages. Messages which are completely received before entering the low power mode can be read after the low power mode exit.

Modes where frequency of system bus clock is changed:

For reduced power consumption the system bus clock can be reduced but not below a minimum value (See *Section 59.5.2: System bus clock requirements*) to avoid overflow of received messages. Protocol clock should not go below the minimum value (See *Section 59.5.3: High frequency receiver clock (protocol clock) requirements*) for accurate receive operations.

Note: Gating of clocks in the low power mode is chip-specific, see the chapter that refers how the modules behave in low power modes.

If the SOC requests entry to debug or stop mode, the message being received currently is discarded and the SOC enters debug/stop mode immediately. This does not affect the messages received completely prior to entering debug mode and will still be present on the message buffer and registers until they are read out.

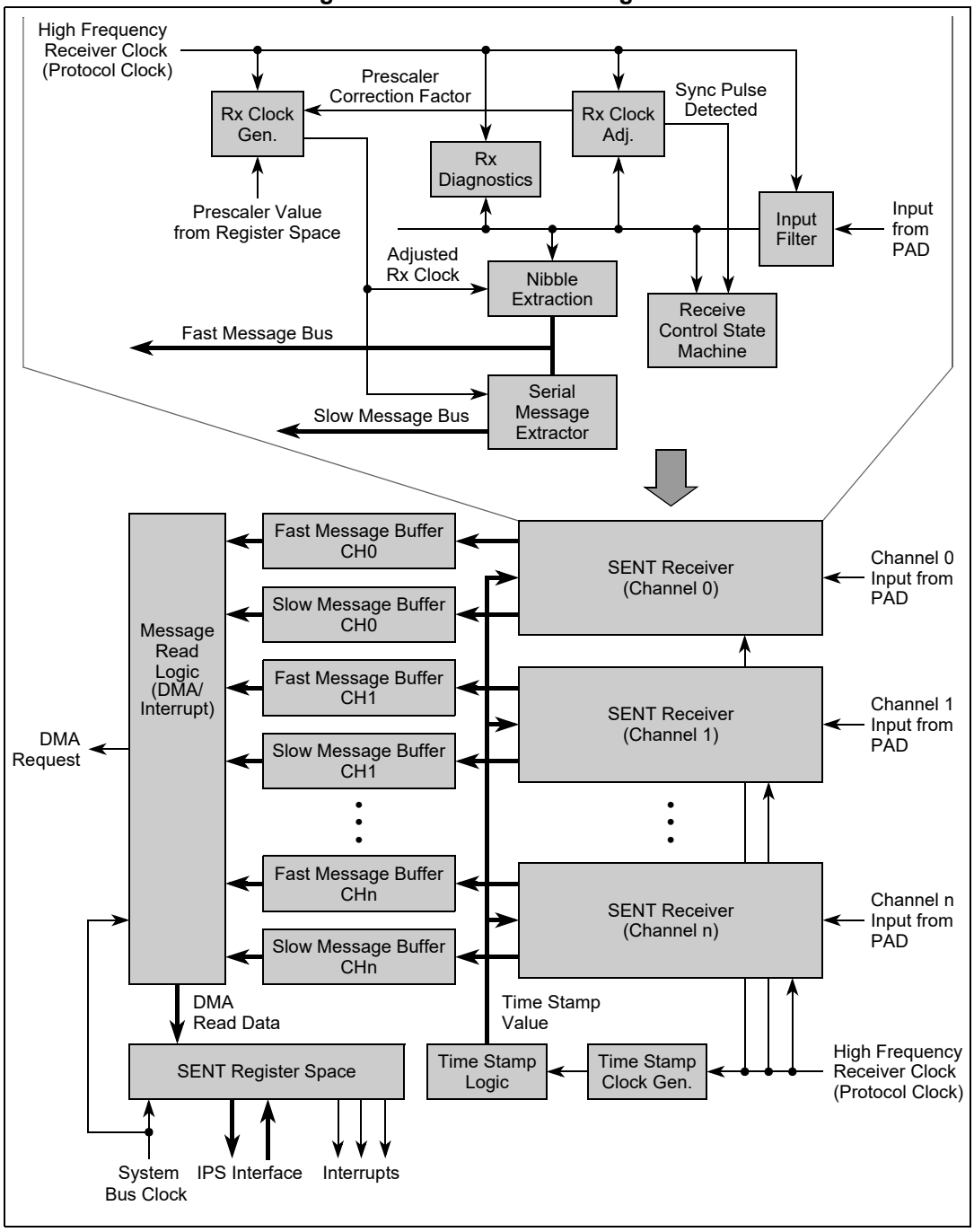
The message read registers [Channel ‘n' Fast Message Data Read Register (n = 0 to (CH-1)) (CHn_FMSG_DATA), Channel ‘n' Fast Message CRC Read Register (n = 0 to (CH-1)) (CHn_FMSG_CRC), Channel ‘n’ Fast Message Time Stamp Read Register (n = 0 to (CH-1)) (CHn_FMSG_TS), Channel ‘n’ Serial Message Read Register (Bit 3) (n = 0 to (CH-1)) (CHn_SMSG_BIT3), Channel ‘n’ Serial Message Read Register (Bit 2) (n = 0 to (CH-1)) (CHn_SMSG_BIT2), Channel ‘n’ Serial Message Time Stamp Read Register (n = 0 to (CH-1)) (CHn_SMSG_TS), DMA Fast Message Data Read Register (DMA_FMSG_DATA), DMA Fast Message CRC Read Register (DMA_FMSG_CRC), DMA Fast Message Time Stamp Read Register (DMA_FMSG_TS), DMA Slow Serial Message Bit3 Read Register (DMA_SMSG_BIT3), DMA Slow Serial Message Bit2 Read Register (DMA_SMSG_BIT2) and DMA Slow Serial Message Time Stamp Read Register (DMA_SMSG_TS)] will appear to lose their contents in the following conditions:

- *The very first message is being received but not yet completely received and the MCU enters debug or freeze mode, the current message reception will get discarded and message read registers (as mentioned above) will read zeros.*
- *Auto clear functionality is enabled (GBL_CTRL[FAST_CLR] = 1). In this case, when first message is read, it will get clear the message read registers (due to auto clear functionality being enabled). On reading again, the message read registers (as mentioned above) might read zeros.*

If the MCU requests entry to debug or stop mode, the message being received currently by SENT Receiver is discarded and the MCU enters debug/stop mode immediately. This does not affect the messages received completely, prior to entering debug mode and these messages will still be present on the message buffer and registers until they are read out. Thus allow one message to be received completely and do not enable “Auto Clear” (GBL_CTRL[FAST_CLR] = 0), to allow messages to be read in debug or stop mode.

59.1.3 Block diagram

Figure 1415. SENT block diagram



59.1.4 Design overview

The SENT Receiver Module is a unified receiver module comprising of a number of receiver channels and a unified DMA and Register Space for accessing the messages from each channel. *Figure 1415* shows the top level block diagram for the SENT Receiver module.

Each channel of the SENT Receiver is composed of a Clock Generation and adjustment block to generate the receiver clock and adjust it for frequency variation in the transmitter clock. This block also detects the synchronization/calibration pulse which is present in every SENT message.

The SENT Receiver State Machine controls the sequencing of decoding the messages and combining the information as nibbles. In parallel a diagnostic block is running that monitors the incoming messages for any errors and also verifies the CRC of the received messages. The fast and serial message extractors collate the message nibbles and add the channel number and time stamp information to the messages. A free running time stamp counter is used to add the time stamp value to each message received. The value of this counter is input to each channel and the channel logic appends the time stamp to each message. These messages are then formatted to be stored in the corresponding message read buffer. Only the messages that pass the diagnostics checks are forwarded to be stored in the buffers.

The SENT Receiver provides two mechanisms for reading the received messages, which are via DMA transfers and via Interrupts. Based on the control bit settings in the control registers, a DMA request or Interrupt is asserted to signal the DMA controller or CPU that messages are available to be read.

The free running time stamp counter, register block and DMA interface are common for all channels configured. The register space contains all the necessary registers for all channels. The register space is 32-bit aligned and accessible via 8-/16-/32-bit accesses. The registers for all un-configured channels are marked as reserved.

59.1.4.1 Functional overview

After reset, the SENT module comes up in the disabled state. All channel enable bits and the global enable bit are set to 0 and the user software must program the SENT module parameters correctly before enabling it. The user software must correctly configure all parameters to avoid any underflow or overflow in the message buffers.

Once enabled, the Receiver Clock Generation and Adjustment block detects the synchronization pulse. Until then, the state machine blocks all subsequent operations of the receiver. Once the synchronization pulse is detected, the receiver clock tick period is compensated for variation in transmitter clock tick period and the state machine then waits for the reception of data nibbles. The adjusted receiver clock is used to sample the data nibbles from the incoming serial message. The state machine waits until the status and communication nibble and configured number of data nibbles are received. The state machine then waits for the pause pulse (if configured to receive a pause pulse). The fast and serial message extractor block provides the state machine with the information on how many nibbles have been received. All diagnostics checks are computed in parallel. Once all required data nibbles are received, the CRC is checked, time stamp and channel number is added to the message. The time stamp for a Fast Message is sampled at the falling edge that comes at the end of the calibration pulse (which is also the start of the status and communication nibble pulse) and is stored in the message read buffer along with the message. This final message is stored in the message buffer as described in the subsequent sections. Any error detected during the receive process causes the status bits in the corresponding channel’s status register to be set.

The above process repeats across each enabled channel in parallel and keeps pushing error free messages into the buffers on high frequency receiver clock. The module’s message read logic synchronizes all these messages on the bus clock and pops out these messages on any DMA or CPU read access. Any DMA underflow errors are stored in the

global status register while the overflow errors are stored in the corresponding channel's status register. In case of an overflow in Fast messages, the newer messages received will overwrite the previous message.

For detailed description on the complete functionality of each block, see *Section 59.4: Functional description*.

59.2 External signal description

This section lists all inputs to the SENT block.

59.2.1 SENT_RX [(CH-1):0]

These pins serve as the serial data input for each of the SENT channel. There is one bit per channel supported by the device. The SENT sensor input pads on the device using this module should have pull-up enabled to ensure a 1'b1 is seen at the module input when SENT sensor is not driving any data.

Note: The value of the CH is device-specific. Refer to the Device Configuration chapter for details.

59.3 Memory map and register definition

This section provides a detailed description of all registers accessible in the SENT block.

59.3.1 Memory map

Table 1462 gives an overview on all SENT registers. The memory map comprises of 32-bit aligned registers which can be accessed via 8-bit, 16-bit, or 32-bit accesses. Write access to reserved locations will generate transfer error. Read access to reserved locations will also generate transfer error and the read data bus will show all 0s. The Memory Map and complete module are in Big Endian Format.

The module will not check for correctness of programmed values in Register Space and software must ensure that correct values are being written.

Note: Register Address = Base Address + Address Offset, where the Base Address is defined at the MCU level and the Address Offset is defined at the module level. The number of registers are device specific. Refer to the Device Configuration chapter for details.

The value of the CH is device-specific. Refer to the Device Configuration chapter for details.

Table 1462. SENT memory map

Offset	Register name	Access	Reset	Location
0x0000	Global Control Register (GBL_CTRL)	RW	0x0000_0010	*Section 59.3.2.1*
0x0004	Channel Enable Register (CHNL_EN)	RW	0x0000_0000	*Section 59.3.2.2*
0x0008	Global Status Register (GBL_STATUS)	RWC1	0x0000_0000	*Section 59.3.2.3*
0x000C	Fast Message Ready Register (FMSG_RDY)	RW(1)	0x0000_0000	*Section 59.3.2.4*
0x0010	Slow Serial Message Ready Register (SMSG_RDY)	RW(1)	0x0000_0000	*Section 59.3.2.5*

Table 1462. SENT memory map (continued)

Offset	Register name	Access	Reset	Location
0x0014–0x0017	Reserved			
0x0018	Data Control register 1 (DATA_CTRL1)	RW	0x1111_1111	*Section 59.3.2.6*
0x001C	Data Control register 2 (DATA_CTRL2)	RW	0x1111_1111	*Section 59.3.2.7*
0x0020–0x0027	Reserved			
0x0028	Fast Message DMA Control register (FDMA_CTRL)	RW	0x0000_0000	*Section 59.3.2.8*
0x002C	Slow Serial Message DMA Control register (SDMA_CTRL)	RW	0x0000_0000	*Section 59.3.2.9*
0x0034	Fast Message Ready Interrupt Control register (FRDY_IE)	RW	0x0000_0000	*Section 59.3.2.10*
0x0038	Slow Serial Message Ready Interrupt Control register (SRDY_IE)	RW	0x0000_0000	*Section 59.3.2.11*
0x0040	DMA Fast Message Data read register (DMA_FMSG_DATA)	R	0x0000_0000	*Section 59.3.2.12*
0x0044	DMA Fast Message CRC read register (DMA_FMSG_CRC)	R	0x0000_0000	*Section 59.3.2.13*
0x0048	DMA Fast Message Time Stamp read register (DMA_FMSG_TS)	R	0x0000_0000	*Section 59.3.2.14*
0x004C–0x004F	Reserved			
0x0050	DMA Serial Message read register (Bit 3) (DMA_SMSG_BIT3)	R	0x0000_0000	*Section 59.3.2.15*
0x0054	DMA Serial Message read register (Bit 2) (DMA_SMSG_BIT2)	R	0x0000_0000	*Section 59.3.2.16*
0x0058	DMA Serial Message Time Stamp Read register (DMA_SMSG_TS)	R	0x0000_0000	*Section 59.3.2.17*
0x005C–0x005F	Reserved			
CH registers (n = 0 to (CH-1))				
0x0060 + n*0x10	Channel n Clock Control register (CHn_CLK_CTRL)	RW	0x0000_8000	*Section 59.3.2.18*
0x0064 + n*0x10	Channel n Status register (CHn_STATUS)	RWC1	0x0000_0000	*Section 59.3.2.19*
0x0068 + n*0x10	Channel n Configuration register (CHn_CONFIG)	RW	0x0000_0104	*Section 59.3.2.20*
(0x006C + n*0x10)–(0x006F + n*0x10)	Reserved			
CH_MSG registers (n = 0 to (CH-1))				
0x0160 + n*0x18	Channel n Fast Message Data read register (CHn_FMSG_DATA)	R	0x0000_0000	*Section 59.3.2.21*
0x0164 + n*0x18	Channel n Fast Message CRC read register (CHn_FMSG_CRC)	R	0x0000_0000	*Section 59.3.2.22*

Table 1462. SENT memory map (continued)

Offset	Register name	Access	Reset	Location
0x0168 + n*0x18	Channel n Fast Message Time Stamp read register (CHn_FMSG_TS)	R	0x0000_0000	*Section 59.3.2.23*
0x016C + n*0x18	Channel n Slow Serial Message Bit3 read register (CHn_SMSG_BIT3)	R	0x0000_0000	*Section 59.3.2.24*
0x0170 + n*0x18	Channel n Slow Serial Message Bit2 read register (CHn_SMSG_BIT2)	R	0x0000_0000	*Section 59.3.2.25*
0x0174 + n*0x18	Channel n Slow Serial Message Time Stamp read register (CHn_SMSG_TS)	R	0x0000_0000	*Section 59.3.2.26*
(0x0160 + CH*0x18) –0x2E3	Reserved			
(0x02E4 + CH*0x04) –0x323	Reserved			
(0x0324 + CH*0x04) –0x364	Reserved			

1. This bit is either “write 1 to clear” (w1c) or self-clearing depending on the value of the FAST_CLR bit in the *Global Control register (GBL_CTRL)*

59.3.2 Register descriptions

This section describes in address order all the SENT registers and their individual bits.

59.3.2.1 Global Control register (GBL_CTRL)

This register provides the global control and setting for the SENT Receiver Module. The SENT_EN bit must be programmed after all other modules settings and control bits have been written to.

Offset: 0x0000 Access: Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	TSPRSC								0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	NIB_LEN_VAR_LIMIT	0	Reserved	0	0	Reserved	FMDUIE	SMDUIE	0	0	0	FAST_CLR	0	DBG_FRZ	0	SENT_EN
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0

Figure 1416. Global Control register (GBL_CTRL)

Table 1463. GBL_CTRL field descriptions

Field	Description
0:7 TSPRSC	Time Stamp Prescaler Value Value in number of clock cycles of high frequency receiver clock set in order to obtain a clock frequency for the time stamp counter block. (0 to 255) Time Stamp Clock = (High Frequency Clock)/(Value + 1). Value set by user software should be such that the clock period is at least 1 µs. If the high frequency clock input is 75 MHz, then the value set in this field would be 0x4A (or 74) to obtain a 1µs clock. **Note:** Since the time stamp clock is common for all channels, the timestamp clock frequency must be at least twice the fastest receiver channel in the application. This is to ensure that the time stamp clock is fast enough to show change in timestamp from message to message.
16 NIB_LEN_VAR_LIMIT	Decides the allowable jitter/clock drift limit for nibble length measurement. 0 Allows a jitter/clock drift of approx up to 12.5% of transmitter clock tick period. This actual allowed jitter/drift value is dependent on the transmitter clock tick period and HF receiver clock period. 1 Allows a jitter/clock drift as per SAE spec (50%). This bit should be set to 1 for proper functioning of SRX.
18	Reserved. Read returns zero.
21	Reserved. Read returns zero.
22 FMDUIE	Fast Message DMA Underflow Interrupt Enable Enables interrupt assertion when there is an underflow condition on Fast Message DMA registers. Default value is 0. 0 Interrupt is disabled. 1Interrupt is enabled.
23 SMDUIE	Slow Serial Message DMA Underflow Interrupt Enable Enables interrupt assertion when there is an underflow condition on Slow message DMA registers. Default value is 0. 0 Interrupt is disabled. 1Interrupt is enabled.
27 FAST_CLR	Fast Clearing Enable bit Enables clearing of ready status (for both fast and slow) bit automatically when corresponding message is read. Default value is 1. See *Fast Message Ready status register (FMSG_RDY)* and *Slow Serial Message Ready status register (SMSG_RDY)* for details on the ready status bits. 0 Fast Clearing is disabled. Bits will be cleared by writing 1. 1 Fast Clearing is enabled.

Table 1463. GBL_CTRL field descriptions (continued)

Field	Description
29 DBG_FRZ	Debug Freeze This bit will enable the debug mode support. SENT Module will freeze in debug mode if this bit is set. Default is 0. 0 No effect in debug mode 1 Freeze in debug mode
31 SENT_EN	SENT Receiver Global Enable This bit enables or disables the complete receiver module irrespective of the setting of individual channel enable bits. User software must program all channel parameters before setting this bit. Default is 0. 0 Entire module is disabled. 1 Module is enabled.

59.3.2.2 Channel Enable register (CHNL_EN)

This register is used to enable individual SENT Receiver channels in the module. The user software must ensure all channel parameters and control settings have been programmed before writing to this bit. In case these parameters are changed while the channel is enabled, normal operation is not guaranteed.

Note: The following register, figure and table show the maximum possible configuration but the exact number of valid register bits depends on supported SENT channels on the device. Refer to the Device Configuration chapter for number of support SENT channels. Reads of bits beyond the supported number of channels should be ignored and these bits must be written to 0.

Offset: 0x0004 Access: Read/Write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	EN_CH15	EN_CH14	EN_CH13	EN_CH12	EN_CH11	EN_CH10	EN_CH9	EN_CH8	EN_CH7	EN_CH6	EN_CH5	EN_CH4	EN_CH3	EN_CH2	EN_CH1	EN_CH0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1417. Channel Enable register (CHNL_EN)

Table 1464. CHNL_EN field descriptions

Field	Description
(32-CH):31 EN_CH[(CH-1):0]	Enable bits for Channels 0 to (CH-1) All channels are disabled after reset. User software must program all channel parameters before setting this bit. Once set, other channel parameters in registers should not be changed until this bit is cleared. Only Interrupt Enable and Status bits can be written to. 0 Channel is disabled. 1 Channel is enabled. Channel parameter registers are locked for writing.

59.3.2.3 Global Status register (GBL_STATUS)

This register provides the status of various conditions in the module that are common across all channels.

Offset: 0x0008 Access: Read/Write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	FMDU	SMDU	0	0	0	0	0	0	0	0
W							w1c	w1c								
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1418. Global Status register (GBL_STATUS)

Table 1465. GBL_STATUS field descriptions

Field	Description
22 FMDU	Fast Message DMA Underflow Interrupt Status bit that indicates when there is an underflow in Fast Message buffers in any channel which has DMA enabled. This bit is set when there is no message in the DMA buffers for read and the software still reads the Fast Message DMA register. Default value is 0. 0 No underflow 1 Underflow occurred
23 SMDU	Slow Serial Message DMA Underflow Interrupt status bit that indicates when there is an Underflow in Slow serial message DMA buffers on channel which has DMA enabled. The bit is set when there is no message in the DMA buffers for read and the software still reads the Slow serial message registers. Default value is 0. 0 No underflow 1 Underflow occurred

59.3.2.4 Fast Message Ready status register (FMSG_RDY)

The bits of this register are asserted when a Fast Message arrives on a corresponding channel that is not configured to be read via DMA. If a channel is not configured for DMA (via *Fast Message DMA Control register (FDMA_CTRL)*), it is assumed to be read via Interrupt or Polling this status bit. Even if the Interrupt Enable bit for that channel is not set (via *Fast Message Ready Interrupt Control register (FRDY_IE)*) and the channel is enabled, the message ready status bit in this register will get asserted on Fast Message reception. Software must ensure that if a channel is enabled; either DMA or Interrupt must be enabled for that channel. Proper flow cannot be guaranteed if neither is selected.

These bits can be configured to be cleared via “Write 1 to Clear” or “Fast Clearing Mechanism”. The configuration is done by writing to the FAST_CLR bit in the *Global Control register (GBL_CTRL)*.

Note: Reads of bits beyond those for the supported number of channels should be ignored. Writes to these bits are un-advisable and may cause unexpected behavior.

Note: Under certain conditions, the Single Edge Nibble Transmission (SENT) Receiver (SRX) stalls and the Fast Message Data Ready bit for the SENT channel (FMSG_RDY[F_RDYn]) will no longer get set to indicate that a fast message is available. Reads of any of the fast message registers by the MCU core will stall and not complete. The registers affected are: DMA_FMSG_DATA (Direct Memory Access (DMA) Fast Message Data Read Register), DMA_FMSG_CRC (DMA Fast Message Cyclic Redundancy Check Register), DMA_FMSG_TS (DMA Fast Message Time-stamp Register), CHn_FMSG_DATA (Channel Fast Message Data Read Register), CHn_FMSG_CRC (Channel Cyclic Redundancy Check Register), CHn_FMSG_TS (Channel Fast Message Time-stamp Register).

This condition occurs when the SENT Receiver is configured as follows:

- *SENT high frequency clock is set higher than the Peripheral Bridge clock.*
- *Polling of ready status bit (FMSG_RDY[F_RDYn]) is used to detect a new message reception instead of interrupt or DMA request.*
- *The fast clear mechanism is used (GBL_CTRL[FAST_CLR] = 1) or the Software Clear (GBL_CTRL[FAST_CLR] = 0) is used for clearing the ready status bits (FMSG_RDY[F_RDYn]).*
- *An overflow status condition is detected in the SENT Receiver Channel Status register (CHn_STATUS[FMSG_OFLW] = 1).*

The overflow occurs when two messages are allowed to queue in the internal buffers of the SENT Receiver.

Program the clock dividers in MCU for peripheral bridge clock and SENT clock such that SENT clock is always less than or equal to the peripheral bridge clock. In this case, the stall does not occur and MCU transactions to SENT registers will complete.

Offset: 0x000C Access: Read/Write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	F_RDY15	F_RDY14	F_RDY13	F_RDY12	F_RDY11	F_RDY10	F_RDY9	F_RDY8	F_RDY7	F_RDY6	F_RDY5	F_RDY4	F_RDY3	F_RDY2	F_RDY1	F_RDY0
W	*(1)	*(1)	*(1)	*(1)	*(1)	*(1)	*(1)	*(1)	*(1)	*(1)	*(1)	*(1)	*(1)	*(1)	*(1)	*(1)
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. This bit is either “write 1 to clear” (w1c) or self-clearing depending on the value of the FAST_CLR bit in the *Global Control register (GBL_CTRL)*.

Figure 1419. Fast Message Ready status register (FMSG_RDY)

Table 1466. FMSG_RDY field descriptions

Field	Description
(32-CH):31 F_RDY[(CH-1):0]	Fast Message Data Ready Status Bit Indicates that a Fast Message is ready to be read by CPU from that specific channel. This bit is asserted only when **DMA is NOT** enabled for that channel (from the DMA Control Register) and an error free message is received, irrespective of the value of corresponding enable bit in Message Ready Interrupt Control Register. Reading the message from the corresponding memory-mapped Fast message read registers would automatically clear this bit (Fast Clearing Mechanism). **Note:** These bits are either “write 1 to clear” (w1c) or self-clearing depending on the value of the FAST_CLR BIT in the Global Control Register (GBL_CTRL). 0 No Fast Message is available to be read out via Interrupt. 1 Fast Message is available to be read out via Interrupt. The clearing mechanism is selected by user by programming the FAST_CLR bit in the Global Control register. See Section *Global Control register (GBL_CTRL)* for details on FAST_CLR bit. **Note:** When using the fast clearing option, these bits will not clear till the complete message is read out which requires the CPU to read all 3 fast message registers via 32-bit read accesses. For 8/16-bit accesses fast clearing mechanism should not be used as it is not supported. When the fast clearing option is not configured, the bits will be cleared only if 1 is written to those bits.

59.3.2.5 Slow Serial Message Ready status register (SMSG_RDY)

The bits of this register are asserted when a Slow serial message arrives on a corresponding channel that is not configured to be read via DMA. If a channel is not configured for DMA (via *Fast Message DMA Control register (FDMA_CTRL)* and *Slow Serial Message DMA Control register (SDMA_CTRL)*), it is assumed to be read via Interrupt or Polling this status bit. Even if the Interrupt Enable bit for that channel is not set (via *Fast Message Ready Interrupt Control register (FRDY_IE)* and *Slow Serial Message Ready*

Interrupt Enable register (SRDY_IE)) and the channel is enabled, the message ready status bit in this register will get asserted on Slow message reception. The software must ensure that if a channel is enabled either the DMA or the interrupt must be enabled for that channel. Erroneous flow can happen if neither of them is selected.

These bits can be configured to be cleared via “Write 1 to Clear” or “Fast Clearing Mechanism”. The configuration is done by writing to the FAST_CLR bit in the *Global Status register (GBL_STATUS)*.

Note: Reads of bits beyond those for the supported number of channels should be ignored. Writes to these bits are un-advisable and may cause unexpected behavior.

Offset: 0x0010 Access: Read/Write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	S_RDY15	S_RDY14	S_RDY13	S_RDY12	S_RDY11	S_RDY10	S_RDY9	S_RDY8	S_RDY7	S_RDY6	S_RDY5	S_RDY4	S_RDY3	S_RDY2	S_RDY1	S_RDY0
W	*(1)	*(1)	*(1)	*(1)	*(1)	*(1)	*(1)	*(1)	*(1)	*(1)	*(1)	*(1)	*(1)	*(1)	*(1)	*(1)
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. This bit is either “write 1 to clear” (w1c) or self-clearing depending on the value of the FAST_CLR bit in the *Global Control register (GBL_CTRL)*.

Figure 1420. Slow Serial Message Ready status register (SMSG_RDY)

Table 1467. SMSG_RDY field descriptions

Field	Description
(32-CH):31 S_RDY[(CH-1):0]	Slow Serial Message Data Ready Status Bit Indicates that a Slow serial message is ready to be read by CPU from that specific channel. This bit is asserted only when **DMA is NOT** enabled for that channel (from the DMA Control Register) and an error free message is received, irrespective of the value of corresponding enable bit in Message Ready Interrupt Control Register. Reading the message from the corresponding memory-mapped serial message Read Registers would automatically clear this bit (Fast Clearing Mechanism). 0 No Slow serial message is available to be read out via Interrupt 1 Slow serial message is available to be read out via Interrupt The clearing mechanism is selected by user by programming the FAST_CLR bit in the Global Control register. See Section *Global Control register (GBL_CTRL)* for details on FAST_CLR bit. **Note:** When using the fast clearing option, these bits will not clear till the complete message is read out which requires the CPU to read all 3 Fast Message registers via 32-bit read accesses. For 8/16-bit accesses fast clearing mechanism should not be used as it is not supported. When the fast clearing option is not configured, the bits will be cleared only if 1 is written to those bits.

59.3.2.6 Data Control register 1 (DATA_CTRL1)

The Data Control Register 1 (DATA_CTRL1) is described below.

Note: Reset value changes depending on the number of channels that are enabled on the device.

Offset: 0x0018 Access: Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	NIBBCH0			0	NIBBCH1			0	NIBBCH2			0	NIBBCH3		
W																
Reset	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0	1

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	NIBBCH4			0	NIBBCH5			0	NIBBCH6			0	NIBBCH7		
W																
Reset	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0	1

Figure 1421. Data Control register 1 (DATA_CTRL1)

Table 1468. DATA_CTRL1 field descriptions

Field	Description
1:3 NIBBCH0	Number of Data Nibbles supported in Channel 0 000 Invalid value. Must not be used 001 Number of data nibbles to be supported by respective channel 010 Number of data nibbles to be supported by respective channel 011 Number of data nibbles to be supported by respective channel 100 Number of data nibbles to be supported by respective channel 101 Number of data nibbles to be supported by respective channel 110 Number of data nibbles to be supported by respective channel 111 Invalid value. Must not be used
5:7 NIBBCH1	Number of Data Nibbles supported in Channel 1 000 Invalid value. Must not be used 001 Number of data nibbles to be supported by respective channel 010 Number of data nibbles to be supported by respective channel 011 Number of data nibbles to be supported by respective channel 100 Number of data nibbles to be supported by respective channel 101 Number of data nibbles to be supported by respective channel 110 Number of data nibbles to be supported by respective channel 111 Invalid value. Must not be used
9:11 NIBBCH2	Number of Data Nibbles supported in Channel 2 000 Invalid value. Must not be used 001 Number of data nibbles to be supported by respective channel 010 Number of data nibbles to be supported by respective channel 011 Number of data nibbles to be supported by respective channel 100 Number of data nibbles to be supported by respective channel 101 Number of data nibbles to be supported by respective channel 110 Number of data nibbles to be supported by respective channel 111 Invalid value. Must not be used

Table 1468. DATA_CTRL1 field descriptions (continued)

Field	Description
13:15 NIBBCH3	Number of Data Nibbles supported in Channel 3 000 Invalid value. Must not be used 001 Number of data nibbles to be supported by respective channel 010 Number of data nibbles to be supported by respective channel 011 Number of data nibbles to be supported by respective channel 100 Number of data nibbles to be supported by respective channel 101 Number of data nibbles to be supported by respective channel 110 Number of data nibbles to be supported by respective channel 111 Invalid value. Must not be used
17:19 NIBBCH4	Number of Data Nibbles supported in Channel 4 000 Invalid value. Must not be used 001 Number of data nibbles to be supported by respective channel 010 Number of data nibbles to be supported by respective channel 011 Number of data nibbles to be supported by respective channel 100 Number of data nibbles to be supported by respective channel 101 Number of data nibbles to be supported by respective channel 110 Number of data nibbles to be supported by respective channel 111 Invalid value. Must not be used
21:23 NIBBCH5	Number of Data Nibbles supported in Channel 5 000 Invalid value. Must not be used 001 Number of data nibbles to be supported by respective channel 010 Number of data nibbles to be supported by respective channel 011 Number of data nibbles to be supported by respective channel 100 Number of data nibbles to be supported by respective channel 101 Number of data nibbles to be supported by respective channel 110 Number of data nibbles to be supported by respective channel 111 Invalid value. Must not be used
25:27 NIBBCH6	Number of Data Nibbles supported in Channel 6 000 Invalid value. Must not be used 001 Number of data nibbles to be supported by respective channel 010 Number of data nibbles to be supported by respective channel 011 Number of data nibbles to be supported by respective channel 100 Number of data nibbles to be supported by respective channel 101 Number of data nibbles to be supported by respective channel 110 Number of data nibbles to be supported by respective channel 111 Invalid value. Must not be used
29:31 NIBBCH7	Number of Data Nibbles supported in Channel 7 000 Invalid value. Must not be used 001 Number of data nibbles to be supported by respective channel 010 Number of data nibbles to be supported by respective channel 011 Number of data nibbles to be supported by respective channel 100 Number of data nibbles to be supported by respective channel 101 Number of data nibbles to be supported by respective channel 110 Number of data nibbles to be supported by respective channel 111 Invalid value. Must not be used

59.3.2.7 Data Control register 2 (DATA_CTRL2)

Note: Reset value changes depending on the number of channels that are enabled on the device.

Offset: 0x001C Access: Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	NIBBCH8			0	NIBBCH9			0	NIBBCH10			0	NIBBCH11		
W																
Reset	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0	1

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	NIBBCH12			0	NIBBCH13			0	NIBBCH14			0	NIBBCH15		
W																
Reset	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0	1

Figure 1422. Data Control register 2 (DATA_CTRL2)

Table 1469. DATA_CTRL2 field descriptions

Field	Description
1:3 NIBBCH8	Number of Data Nibbles supported in Channel 8 000 Invalid value. Must not be used 001 Number of data nibbles to be supported by respective channel 010 Number of data nibbles to be supported by respective channel 011 Number of data nibbles to be supported by respective channel 100 Number of data nibbles to be supported by respective channel 101 Number of data nibbles to be supported by respective channel 110 Number of data nibbles to be supported by respective channel 111 Invalid value. Must not be used
5:7 NIBBCH9	Number of Data Nibbles supported in Channel 9 000 Invalid value. Must not be used 001 Number of data nibbles to be supported by respective channel 010 Number of data nibbles to be supported by respective channel 011 Number of data nibbles to be supported by respective channel 100 Number of data nibbles to be supported by respective channel 101 Number of data nibbles to be supported by respective channel 110 Number of data nibbles to be supported by respective channel 111 Invalid value. Must not be used
9:11 NIBBCH10	Number of Data Nibbles supported in Channel 10 000 Invalid value. Must not be used 001 Number of data nibbles to be supported by respective channel 010 Number of data nibbles to be supported by respective channel 011 Number of data nibbles to be supported by respective channel 100 Number of data nibbles to be supported by respective channel 101 Number of data nibbles to be supported by respective channel 110 Number of data nibbles to be supported by respective channel 111 Invalid value. Must not be used

Table 1469. DATA_CTRL2 field descriptions (continued)

Field	Description
13:15 NIBBCH11	Number of Data Nibbles supported in Channel 11 000 Invalid value. Must not be used 001 Number of data nibbles to be supported by respective channel 010 Number of data nibbles to be supported by respective channel 011 Number of data nibbles to be supported by respective channel 100 Number of data nibbles to be supported by respective channel 101 Number of data nibbles to be supported by respective channel 110 Number of data nibbles to be supported by respective channel 111 Invalid value. Must not be used
17:19 NIBBCH12	Number of Data Nibbles supported in Channel 12 000 Invalid value. Must not be used 001 Number of data nibbles to be supported by respective channel 010 Number of data nibbles to be supported by respective channel 011 Number of data nibbles to be supported by respective channel 100 Number of data nibbles to be supported by respective channel 101 Number of data nibbles to be supported by respective channel 110 Number of data nibbles to be supported by respective channel 111 Invalid value. Must not be used
21:23 NIBBCH13	Number of Data Nibbles supported in Channel 13 000 Invalid value. Must not be used 001 Number of data nibbles to be supported by respective channel 010 Number of data nibbles to be supported by respective channel 011 Number of data nibbles to be supported by respective channel 100 Number of data nibbles to be supported by respective channel 101 Number of data nibbles to be supported by respective channel 110 Number of data nibbles to be supported by respective channel 111 Invalid value. Must not be used
25:27 NIBBCH14	Number of Data Nibbles supported in Channel 14 000 Invalid value. Must not be used 001 Number of data nibbles to be supported by respective channel 010 Number of data nibbles to be supported by respective channel 011 Number of data nibbles to be supported by respective channel 100 Number of data nibbles to be supported by respective channel 101 Number of data nibbles to be supported by respective channel 110 Number of data nibbles to be supported by respective channel 111 Invalid value. Must not be used
29:31 NIBBCH15	Number of Data Nibbles supported in Channel 15 000 Invalid value. Must not be used 001 Number of data nibbles to be supported by respective channel 010 Number of data nibbles to be supported by respective channel 011 Number of data nibbles to be supported by respective channel 100 Number of data nibbles to be supported by respective channel 101 Number of data nibbles to be supported by respective channel 110 Number of data nibbles to be supported by respective channel 111 Invalid value. Must not be used

59.3.2.8 Fast Message DMA Control register (FDMA_CTRL)

Note: The following register figure and table show the maximum possible configuration but the exact number of valid register bits depends on supported SENT channels on the device.

Refer to the Device Configuration chapter for number of support SENT channels. Reads of bits beyond the supported number of channels should be ignored and writes to these bits are not advisable and may cause unexpected behavior.

Offset: 0x0028 Access: Read/Write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	FDMA_EN15	FDMA_EN14	FDMA_EN13	FDMA_EN12	FDMA_EN11	FDMA_EN10	FDMA_EN9	FDMA_EN8	FDMA_EN7	FDMA_EN6	FDMA_EN5	FDMA_EN4	FDMA_EN3	FDMA_EN2	FDMA_EN1	FDMA_EN0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1423. Fast Message DMA Control register (FDMA_CTRL)

Table 1470. FDMA_CTRL field descriptions

Field	Description
(32-CH):31 FDMA_EN[(CH-1):0]	Enable DMA for Fast messages on Channels 0 to (CH-1) These bits are writable when corresponding Fast Message Ready Interrupt Enable bits are set to 0. 0 DMA for Fast messages is disabled. 1 DMA for Fast messages is enabled.

59.3.2.9 Slow Serial Message DMA Control register (SDMA_CTRL)

Note: The following register, figure and table show the maximum possible configuration but the exact number of valid register bits depends on supported SENT channels on the device. Refer to the Device Configuration chapter for number of support SENT channels. Reads of bits beyond the supported number of channels should be ignored and writes to these bits are not advisable and may cause unexpected behaviors.

Offset: 0x002C Access: Read/Write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	SDMA_EN15	SDMA_EN14	SDMA_EN13	SDMA_EN12	SDMA_EN11	SDMA_EN10	SDMA_EN9	SDMA_EN8	SDMA_EN7	SDMA_EN6	SDMA_EN5	SDMA_EN4	SDMA_EN3	SDMA_EN2	SDMA_EN1	SDMA_EN0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1424. Slow Serial Message DMA Control register (SDMA_CTRL)

Table 1471. SDMA_CTRL field descriptions

Field	Description
(32-CH):31 SDMA_EN[(CH-1):0]	Enable DMA for Slow Serial Messages on Channels 0 to (CH-1) These bits are writable when corresponding Slow Serial Message Ready Interrupt Enable bits are set to 0. 0 DMA for Slow serial messages is disabled. 1 DMA for Slow serial messages is enabled.

59.3.2.10 Fast Message Ready Interrupt Control register (FRDY_IE)

Note: The following register, figure and table show the maximum possible configuration but the exact number of valid register bits depends on supported SENT channels on the device. Refer to the Device Configuration chapter for number of support SENT channels. Reads of bits beyond the supported number of channels should be ignored and writes to these bits are not advisable and may cause unexpected behavior.

Offset: 0x0034 Access: Read/Write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	FRDY_IE15	FRDY_IE14	FRDY_IE13	FRDY_IE12	FRDY_IE11	FRDY_IE10	FRDY_IE9	FRDY_IE8	FRDY_IE7	FRDY_IE6	FRDY_IE5	FRDY_IE4	FRDY_IE3	FRDY_IE2	FRDY_IE1	FRDY_IE0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1425. Fast Message Ready Interrupt Enable Register (FRDY_IE)

Table 1472. FRDY_IE field descriptions

Field	Description
(32-CH):31 FRDY_IE[(CH-1):0]	Enable for Fast Message Ready Interrupt on Channels 0 to (CH-1) These bits are writable when corresponding DMA enable bits are set to 0 in the Fast Message DMA Control Register. The availability of message is indicated via assertion of corresponding bit in Fast Message Ready Status Register irrespective of the value set in corresponding interrupt enable bit and DMA is not enabled. When a bit is asserted in this register, an interrupt for corresponding channel is generated. 0 Interrupt on reception of Fast messages is disabled. 1 Interrupt on reception of Fast messages is enabled.

59.3.2.11 Slow Serial Message Ready Interrupt Enable register (SRDY_IE)

Note: The following register, figure and table show the maximum possible configuration but the exact number of valid register bits depends on supported SENT channels on the device. Refer to the Device Configuration chapter for number of support SENT channels. Reads of bits beyond the supported number of channels should be ignored and writes to these bits are not advisable and may cause unexpected behavior.

Offset: 0x0038 Access: Read/Write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	SRDY_IE15	SRDY_IE14	SRDY_IE13	SRDY_IE12	SRDY_IE11	SRDY_IE10	SRDY_IE9	SRDY_IE8	SRDY_IE7	SRDY_IE6	SRDY_IE5	SRDY_IE4	SRDY_IE3	SRDY_IE2	SRDY_IE1	SRDY_IE0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1426. Slow Serial Message Ready Interrupt Enable register (SRDY_IE)

Table 1473. SRDY_IE field descriptions

Field	Description
(32-CH):31 SRDY_IE[(CH-1):0]	Enable for Slow Serial Message Ready Interrupt on Channels 0 to (CH-1) These bits are writable when corresponding DMA enable bits are set to 0 in the Slow Serial DMA Control Register. The availability of message is indicated via assertion of corresponding bit in Slow Serial Message Ready Status Register irrespective of the value set in corresponding interrupt enable bit and DMA is not enabled. When a bit in this register is asserted, an interrupt for corresponding channel is generated. 0 Interrupt on reception of Slow serial messages is disabled. 1 Interrupt on reception of Slow serial messages is enabled.

59.3.2.12 DMA Fast Message Data read register (DMA_FMSG_DATA)

DMA Source Size must be set to 32 bits when reading this register (DMA_FMSG_DATA) via DMA.

Offset: 0x0040 Access: User read-only

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	CHNUM				SCNIB				DNIB1				DNIB2			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	DNIB3				DNIB4				DNIB5				DNIB6			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1427. DMA Fast Message Data read register (DMA_FMSG_DATA)

Table 1474. DMA_FMSG_DATA field descriptions

Field	Description
0:3 CHNUM	Channel Number Indicates the channel number of the received message being read by DMA. Valid values are 0 to ‘15’
4:7 SCNIB	Status and Communication Nibble of message
8:11 DNIB1	Data Nibble 1 Always supported.
12:15 DNIB2	Data Nibble 2 Should be ignored if number of data nibbles supported by channel is 1.
16:19 DNIB3	Data Nibble 3 Should be ignored if number of data nibbles supported by channel is 2 or less.
20:23 DNIB4	Data Nibble 4 Should be ignored if number of data nibbles supported by channel is 3 or less.
24:27 DNIB5	Data Nibble 5 Should be ignored if number of data nibbles supported by channel is 4 or less.
28:31 DNIB6	Data Nibble 6 Should be ignored if number of data nibbles supported by channel is 5 or less.

59.3.2.13 DMA Fast Message CRC read register (DMA_FMSG_CRC)

DMA Source Size must be set to 32 bits when reading this register (DMA_FMSG_CRC) via DMA.

Offset: 0x0044 Access: User read-only

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	CRC4b			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

Figure 1428. DMA Fast Message CRC Read Register (DMA_FMSG_CRC)

Table 1475. DMA_FMSG_CRC field descriptions

Field	Description
12:15 CRC4b	4-bit CRC value of the message

59.3.2.14 DMA Fast Message Time Stamp read register (DMA_FMSG_TS)

The resolution of the time stamp is set by programming the Time Stamp Prescaler Value in the *Section 59.3.2.1: Global Control register (GBL_CTRL)* register. DMA Source Size must be set to 32 bits when reading this register (DMA_FMSG_TS) via DMA.

Offset: 0x0048 Access: User read-only

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	TS															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	TS															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1429. DMA Fast Message Time Stamp read register (DMA_FMSG_TS)

Table 1476. DMA_FMSG_TS field descriptions

Field	Description
0:31 TS	Time Stamp for received message Indicates the relative time when the falling edge that comes at the end of the calibration pulse (which is also the start of the status and communication nibble pulse) was detected.

59.3.2.15 DMA Slow Serial Message Bit3 read register (DMA_SMSG_BIT3)

The DMA_SMSG_BIT3 is common for all types of slow serial messages. Bit C defines the type of enhanced packet and bit TYPE defines whether it is short serial or enhanced serial message. Since this register is used to read both Short and Enhanced serial messages, the register fields have been placed in such a way to match the bit positions in actual message defined by SAE specification as shown in *Figure 1445*, *Figure 1446* and *Figure 1447*. *Figure 1445* and *Figure 1446* show the enhanced serial messages format to be used when reading this register for these two types of messages. *Figure 1447* shows the short serial message.

DMA Source Size must be set to 32 bits when reading this register via DMA.

Offset: 0x0050 Access: User read-only

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	CHNUM				TYPE	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	CFG	ID7_6/ ID3_2		ID5_4/ ID1_0		0	ID3_0/ DATA15_12				0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1430. DMA Slow Serial Message Bit3 read register (DMA_SMSG_BIT3)

Table 1477. DMA_SMSG_BIT3 Field Descriptions

Field	Description
0:3 CHNUM	Channel Number Indicates the channel number of the received message being read by DMA. Valid values are 0 to ‘15’.
4 TYPE	Serial Message Type Indicates the type of serial message received. 0 Short serial message 1 Enhanced serial message
20	Placed as 0 by specification.
21 CFG	Configuration bit ‘C’ Indicates the type of Enhanced Serial message. 0 Enhanced serial message with 8-bit ID field 1 Enhanced serial message with 4-bit ID field
22:23 ID7_6/ID3_2	ID Field Value depends on setting of ‘C’ bit. If ‘C’ bit is 0, this field is ID[7:6] and if the ‘C’ bit is 1, this field is ID[3:2].
24:25 ID5_4/ID1_0	ID Field Value depends on setting of ‘C’ bit. If ‘C’ bit is 0, this field is ID[5:4] and if the ‘C’ bit is 1, this field is ID[1:0].

Table 1477. DMA_SMSG_BIT3 Field Descriptions (continued)

Field	Description
26	0 as per SAE specification
27:30 ID3_0/DATA15_12	ID or Data Field Value depends on setting of 'C' bit. If 'C' bit is 0, this field is ID[3:0] and if the 'C' bit is 1, this field is Data[15:12].
31	0 As per SAE specification.

Note: If the TYPE bit in the register reads '0', then remaining bits (for CFG, ID & DATA) will be redundant and read zeros. The short serial message will be located in DMA_SMSG_BIT2 register fields. If TYPE bit is '1', then the message in register is Enhanced serial message and using CFG bit, software can decode the ID & DATA fields.

59.3.2.16 DMA Slow Serial Message Bit2 read register (DMA_SMSG_BIT2)

This register (DMA_SMSG_BIT2) is common for all types of slow serial messages. Hence, the register fields have been placed in such a way to match the bit positions in actual message defined by SAE specification as shown in *Figure 1445*, *Figure 1446* and *Figure 1447*. *Figure 1445* and *Figure 1446* show the enhanced serial messages format to be used when reading this register for these two types of messages. *Figure 1447* shows the short serial message.

DMA Source Size must be set to 32 bits when reading this register via DMA.

Offset: 0x0054 Access: User read-only

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	SMCRC					
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	ID3_0/ DATA11_8				DATA7_0							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1431. DMA Slow Serial Message Bit2 read register (DMA_SMSG_BIT2)

Table 1478. DMA_SMSG_BIT2 field descriptions

Field	Description
10:15 SMCRC	6-bit CRC value of the message When TYPE bit in *DMA Slow Serial Message Bit3 read register (DMA_SMSG_BIT3)* is set to 0, this CRC is of 4-bits for Short serial messages and when TYPE bit is set to 1, this CRC is of 6-bits for Enhanced serial messages.
20:23 ID3_0/DATA11_8	ID or Data Field When TYPE bit in is set to 0, this field is ID [3:0] for Short serial messages and when TYPE bit is set to 1, this field is DATA [11:8] for Enhanced serial messages.
24:31 DATA7_0	Data Field Data field [7:0] for short serial messages and enhanced serial messages.

59.3.2.17 DMA Slow Serial Message Time Stamp read register (DMA_SMSG_TS)

The resolution of the time stamp is set by programming the Time Stamp Prescaler Value in the register.

DMA Source Size must be set to 32 bits when reading this register (DMA_SMSG_TS) via DMA.

Offset: 0x0058 Access: User read-only

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	TS															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	TS															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1432. DMA Slow Serial Message Time Stamp read register (DMA_SMSG_TS)

Table 1479. DMA_SMSG_TS field descriptions

Field	Description
0:31 TS	Time Stamp for the received message Indicates the relative time of detection of the falling edge that comes at the end of the calibration pulse (which is also the start of the status and communication nibble pulse) of the last Fast Message that completed the serial message. Time Stamp for Slow serial messages is not latched when the Slow serial message starts but when the Slow serial message is completely received.

59.3.2.18 Channel n Clock Control register (CHn_CLK_CTRL)

Note: The value of the CH is device-specific. Refer to the Device Configuration chapter for details.

The SENT module generates the Receiver Sensor Clock (or simply Receiver Clock) internally to sample the incoming filtered sensor data. The user software must program the prescaler value (for Receiver Clock) to match the ideal Sensor Tx Clock period. The generated Receiver Clock is compensated for variations, clock drift and jitter that might occur in the Sensor Tx Clock. See *Section 59.4.5: Adjustment for variation in sensor (Tx) clock* for more details.

This register configures the generation and subsequent compensation logic of the Receiver Clock. User must program the contents of this register before enabling the channel by writing to the corresponding channel enable bit in the Channel Enable Register (CHNL_EN).

Offset: 0x0060 + n*0x10 (n = 0 to (CH-1)) Access: Read/Write

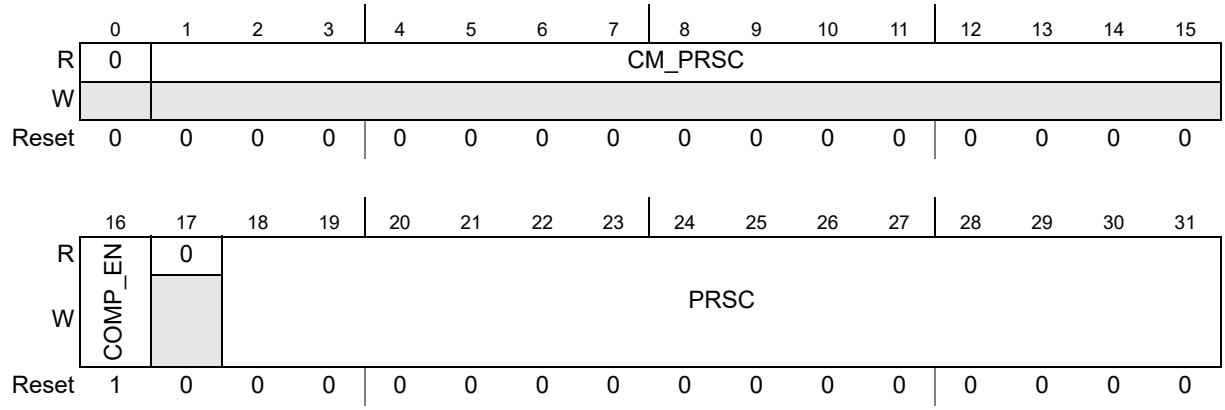


Figure 1433. Channel n Clock Control register (CHn_CLK_CTRL))

Table 1480. CH*n*_CLK_CTRL field descriptions

Field	Description
1:15 CM_PRSC	Compensated Prescaler Value This is the compensated prescaler value calculated by Clock Compensation logic and this value is used to compensate for sensor Tx clock variation. This value is indicated as a read only field for use by user software, if needed. This is a 15-bit value. '= Rx Prescaler Value' – No variation, hence no compensation or Compensation is disabled. '< Rx Prescaler Value' – Tx Clock is SLOWER than the programmed receive clock. '> Rx Prescaler Value' – Tx Clock is FASTER than the programmed receive clock. Thus, the new Receiver Clock Tick Period can be computed by software as Compensated Prescaler Value × High Frequency Receiver Clock Period **Note:** The value provided in this field is ceiling value of actual tick period as measured by the Receiver.

Table 1480. CH*n*_CLK_CTRL field descriptions (continued)

Field	Description
16 COMP_EN	Compensation Enable To enable compensation logic to adjust the receiver clock against variation in Tx clock on this channel. User must set the Prescaler Value before enabling this bit. 0 Compensation is disabled. 1 Compensation is enabled.
18:31 PRSC	Rx Prescaler Value This factor is required to generate required Receiver Clock tick from the High Frequency Receiver Clock. This value can only be written when Channel Enable bit is Zero. Hence this value must be programmed before enabling the channel. If required receiver clock tick period (T_{rx_clk}) is not divisible by the high frequency receiver clock (protocol clock) period (T_{hf_clk}), the required prescaler value will have a fractional part which cannot be programmed. In this case then user is advised to program the prescaler value to the lower integer value. 0 Prescaler is bypassed. Should not be used. Non-Zero Value Enables Prescaler. The required value is computed by software as follows: **Equation 76 Prescaler_value = floor (Trx_clk / Thf_clk)** (Both clock periods taken in µs). See *Section 59.5.3: High frequency receiver clock (protocol clock) requirements* before setting this value. For example, if the required receiver clock tick period is 43 µs and the High Frequency Receiver clock is selected to be 62.5 MHz (or 0.016 µs) then: Prescaler Value = 43/0.016 = 2687.5. Hence user must program 2687 in this register.

59.3.2.19 Channel *n* Status register (CH*n*_STATUS)

This registers stores information about certain events and error conditions that occur in the channel during the message reception process. The assertion of these bit is not linked to any message but it is a general alert to the CPU that a particular condition (error or normal event) happened in the recent past. Messages are automatically discarded when an error is detected.

User software can clear these bit at any time by writing 1 to them. This register can be updated at any time irrespective of the setting in the bit for the corresponding channel.

Note: The value of the CH is device-specific. Refer to the Device Configuration chapter for details.

Offset: 0x0064 + n*0x10 (n = 0 to (CH-1)) Access: Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	BUS_IDLE	0	0	0	CAL_RESYNC	CAL_20_25	SMSG_OFLW	FMSG_OFLW	0	PP_DIAG_ERR	CAL_LEN_ERR	CAL_DIAG_ERR	NIB_VAL_ERR	SMSG_CRC_ERR	FMSG_CRC_ERR	NUM_EDGES_ERR
W	w1c				w1c	w1c	w1c	w1c		w1c	w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

Figure 1434. Channel n Status register (CHn_STATUS)

Table 1481. CHn_STATUS field descriptions

Field	Description
0 BUS_IDLE	Bus Idle Status This bit indicates that the sensor interface has been idle for more than the period defined by the programmed Bus Idle Count value (in *Channel n Configuration register (CHn_CONFIG)*). CPU should write 1 to this clear this bit once it is read. 0 Bus is not idle. 1 Channel has been idle for more than the allowed value.
4 CAL_RESYNC	Successive Calibration Check (Option 2 of SAE SENT Spec) Resynchronized This bit indicates that Successive Calibration Check (Option 2 of SAE Spec) has failed three times and has resynchronized to assume 3rd message to be correct and will be received. The successive calibration check error will also be asserted during re-synchronization (to indicate the 3rd error being detected). 0 No interrupt. 1 Interrupt Status condition has occurred.
5 CAL_20_25	Calibration Variation 20–25% Interrupt Status This bit indicates that the calibration pulse received on this channel has variation in between ±20% to ±25% from 56 ticks. 0 No Interrupt. 1 Interrupt Status condition has occurred.
6 SMSG_OFLW	Slow Serial Message Overflow Status This bit indicates overflow occurred on this channel. Overflow will cause the older messages (which are not read out) to be overwritten with newly received messages. However, any message that is currently being read out by DMA or CPU will not get over-written. 0 No Interrupt. 1 Interrupt Status condition has occurred.

Table 1481. CHn_STATUS field descriptions (continued)

Field	Description
7 FMSG_OFLW	Fast Message Overflow Status This bit indicates overflow occurred on this channel. Overflow happens when effective data read rate by CPU is slower than data receive rate on channel. On overflow, new incoming messages will overwrite the previous messages (that are not read out) stored in the receiver buffers. However, any message that is currently being read out by DMA or CPU will not get over-written. 0 No Interrupt. 1 Interrupt Status condition has occurred.
9 PP_DIAG_ERR	This diagnostic checks status bit indicates that the ratio of calibration pulse length to overall message length (with pause pulse) is more than ±1.5625% between two messages. This check is valid only for messages with pause pulse. This bit indicates whether this diagnostic has failed or passed on this channel, and is cleared by writing a 1 to it. 0 Error Check has passed. 1 Error Check has failed.
10 CAL_LEN_ERR	This diagnostic checks status bit indicates that Calibration pulse is more than 56 ticks ±25%. This bit indicates whether this diagnostic has failed or passed on this channel, and is cleared by writing a 1 to it. 0 Error Check has passed. 1 Error Check has failed.
11 CAL_DIAG_ERR	Successive Calibration pulses differ by ±1.56%. SAE spec defines it be 1.5625% but the accuracy of this check depends on the frequency of protocol clock (see *Section 59.5.3: High frequency receiver clock (protocol clock) requirements*). This diagnostic checks status bit indicates whether this diagnostic has failed or passed on this channel, and is cleared by writing a 1 to it. 0 Error Check has passed. 1 Error Check has failed.
12 NIB_VAL_ERR	Any nibble data value <0 or >15. This diagnostic checks status bit indicates whether this diagnostic has failed or passed on this channel, and is cleared by writing a 1 to it. 0 Error Check has passed. 1 Error Check has failed.
13 SMSG_CRC_ERR	Checksum error in Slow Serial Message This diagnostic checks status bit indicates whether this diagnostic has failed or passed on this channel, and is cleared by writing a 1 to it. 0 Error Check has passed. 1 Error Check has failed.

Table 1481. CHn_STATUS field descriptions (continued)

Field	Description
14 FMSG_CRC_ERR	Checksum error in Fast Message This diagnostic checks status bit indicates whether this diagnostic has failed or passed on this channel, and is cleared by writing a 1 to it. 0 Error Check has passed. 1 Error Check has failed.
15 NUM_EDGES_ERR	Not the expected number of negative edges between calibration pulse. This diagnostic checks status bit indicates whether this diagnostic has failed or passed on this channel. This bit will not set in case of “Option 2” for successive calibration pulse check is selected. This error will only set when "Option 1" for successive calibration check is selected (SUCC_CAL_CHK). This bit is cleared by writing a 1 to it. 0 Error Check has passed. 1 Error Check has failed.

59.3.2.20 Channel n Configuration register (CHn_CONFIG)

This register (CHn_CONFIG) provides the configurability for interrupt assertion for various events and error as explained in *Channel n Status register (CHn_STATUS)* and control bits for selection of various receive options that might be required by the user software. This register also configures the width of noise glitches (on the sensor input) to be filtered out by the input programmable filter.

User must program the contents of this register before enabling the channel by writing to the corresponding channel enable bit in the Channel Enable Register (CHNL_EN).

Note: The value of the CH is device-specific. Refer to the Device Configuration chapter for details.

Offset: 0x0068 + n*0x10 (n = 0 to (CH-1)) Access: Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	BUS_IDLE_CNT				IE_CAL_RESYNC	IE_CAL_20_25	IE_SMSG_OFLW	IE_FMSG_OFLW	FCRC_CHK_OFF	IE_PP_DIAG_ERR	IE_CAL_LEN_ERR	IE_CAL_DIAG_ERR	IE_NIB_VAL_ERR	IE_SMSG_CRC_ERR	IE_FMSG_CRC_ERR	IE_NUM_EDGES_ERR
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	DCHNG_INT	CAL_RNG	PP_CHKSEL	FCRC_TYPE	FCRC_SC_EN	SCRC_TYPE	PAUSE_EN	SUCC_CAL_CHK	FIL_CNT							
Reset	0	0	0	0	0	0	0	1	0	0	0	0	0	1	0	0

Figure 1435. Channel n Configuration register (CHn_CONFIG)

Table 1482. CHn_CONFIG field descriptions

Field	Description
0:3 BUS_IDLE_CNT	Bus Idle Count This register value defines the maximum allowable idle period on the sensor interface of this channel. The value is defined as follows: 0000 Disabled 0001 127*2 Receiver Clock Tick Counts 0010 127*4 Receiver Clock Tick Counts 0100 127*8 Receiver Clock Tick Counts 1000 127*16 Receiver Clock Tick Counts
4 IE_CAL_RESYNC	Successive Calibration Check Resynchronized Interrupt Enable This bit enables interrupt assertion when Successive Calibration diagnosis has failed three times in case of “Option 2” being selected for Successive Calibration Check Method and the check resynchronizes the take the third message to be correct. 0 Interrupt is disabled. 1 Interrupt is enabled.
5 IE_CAL_20_25	Calibration Variation 20 - 25% Interrupt Enable This bit enables interrupt assertion when corresponding status bit is set. 0 Interrupt is disabled. 1 Interrupt is enabled.
6 IE_SMSG_OFLW	Slow Serial Message Overflow Interrupt Enable This bit enables interrupt assertion when overflow occurs on reception of Slow serial messages in corresponding channel 0 Interrupt is disabled. 1 Interrupt is enabled.
7 IE_FMSG_OFLW	Fast Message Overflow Interrupt Enable This bit enables interrupt assertion when overflow occurs on reception of Fast messages in corresponding channel. 0 Interrupt is disabled. 1 Interrupt is enabled.
8 FCRC_CHK_OFF	Fast Message CRC Check Off This bit can be used to switch off CRC check in Fast Message 0 Check is enabled. 1 Check is disabled/off.
9 IE_PP_DIAG_ERR	Ratio of calibration pulse length to message length varies by more than ±1.5625% between two frames. Valid for messages with pause pulse. This bit enables interrupt assertion when its respective diagnostic check fails on the corresponding channel. 0 Interrupt is disabled. 1 Interrupt is enabled.
10 IE_CAL_LEN_ERR	Calibration pulse is wider than 56 ticks ±25%. This bit enables interrupt assertion when its respective diagnostic check fails on the corresponding channel. 0 Interrupt is disabled. 1 Interrupt is enabled.

Table 1482. CHn_CONFIG field descriptions (continued)

Field	Description
11 IE_CAL_DIAG_ERR	Successive Calibration pulses differ by more than ±1.56%. SAE spec defines it be 1.5625% but the accuracy of this check depends on the protocol clock. For instance for 62.5 MHz, this check will fail for 1.56% and passes for 1.55%. This bit enables interrupt assertion when its respective diagnostic check fails on the corresponding channel. 0 Interrupt is disabled. 1 Interrupt is enabled.
12 IE_NIB_VAL_ERR	Any nibble data value <0 or >15 This bit enables interrupt assertion when its respective diagnostic check fails on the corresponding channel. 0 Interrupt is disabled. 1 Interrupt is enabled.
13 IE_SMSG_CRC_ERR	Checksum error in Slow Serial Message This bit enables interrupt assertion when its respective diagnostic check fails on the corresponding channel. 0 Interrupt is disabled. 1 Interrupt is enabled.
14 IE_FMSG_CRC_ERR	Checksum error in Fast Message This bit enables interrupt assertion when its respective diagnostic check fails on the corresponding channel. 0 Interrupt is disabled. 1 Interrupt is enabled.
15 IE_NUM_EDGES_ERR	Not the expected number of negative edges between calibration pulse. This bit enables interrupt assertion when its respective diagnostic check fails on the corresponding channel. 0 Interrupt is disabled. 1 Interrupt is enabled.
16 DCHNG_INT	Enable for Interrupt on Reception of Fast Message with Changed Data This bit is used to enable storing and generating corresponding DMA or Message Ready Interrupt Request for Fast Message only when any of the data nibbles after different from previously received Fast Message on that channel. If a new Fast Message is received that has same values for all data nibbles as that of previous message then the current message will be discarded and will not be stored in the buffer, and neither Fast Message Ready Interrupt nor DMA request will be generated. 0 All Fast messages will be received even if data nibbles are same. 1 Only Fast messages with differing values of data nibbles will be received.
17 CAL_RNG	Valid Calibration Pulse Range Selection This bit is used to control whether 20% variation or 25% variation in Calibration Pulse will be tolerated when detecting a Calibration Pulse 0 20% variation is acceptable. 1 25% variation is acceptable.

Table 1482. CHn_CONFIG field descriptions (continued)

Field	Description
18 PP_CHKSEL	Pause Pulse Diagnostic Check Selection This bit controls which diagnostic checks to run when messages are enabled to be received with pause pulses. 0 Both Successive Calibration Pulse Check and Pause Pulse Diagnostic are run. 1 Only Pause Pulse Diagnostic is run. Successive Calibration Pulse Check is disabled.
19 FCRC_TYPE	Fast Message CRC Type This indicates the type of CRC method to be used for CRC Diagnostic Check on the received fast messages 0 XOR based implementation (Recommended by SAE specification). 1 Legacy LUT based implementation.
20 FCRC_SC_EN	Fast Message CRC Status and Communication Nibble Enable This bit enables the inclusion of Status and Communication Nibble when CRC is calculated for Fast messages. 0 Status and Communication Nibble not included in CRC calculation. 1 Status and Communication Nibble is included in CRC calculation.
21 SCRC_TYPE	Slow Serial Message CRC Type This indicates the type of CRC method to be used for CRC Diagnostic Check on the received slow serial messages. 0 XOR based implementation (Recommended by SAE specification). 1 Legacy LUT based implementation.
22 PAUSE_EN	Pause Pulse Enable Enables the channel receiver to detect a pause pulse. 0 Detection of Pause Pulse is disabled. 1 Detection of Pause Pulse enabled.
23 SUCC_CAL_CHK	Successive Calibration Pulse Check Method This bit indicates the method to be used for performing the successive calibration pulse check. Default value is 1. 0 Option 2 such as Low Latency Option as per SAE specification. 1 Option 1 such as Preferred but High Latency Option as per SAE specification.
24:31 FIL_CNT	Input Filter Sample Count This indicates twice the number of Protocol Clock (of High Frequency Receiver Clock) cycles required for channel input from device's pad to remain stable before it is sampled as 0 or 1. This field defines the width of glitches to be filtered out by the input programmable filter on that channel. Only one bit should be set in this field and the output of filter will be offset by twice the same number of clocks plus additional 3 clocks due to synchronization. Default value is '4'. 0x00: No filtering. Input from device's pad is only synchronized. Non-Zero: Filtering is enabled.

59.3.2.21 Channel n Fast Message Data read register (CHn_FMSG_DATA)

Offset: 0x0160 + n*0x18 (*n* = 0 to (CH(1)-1)) Access: User read-only

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	CHNUM				SCNIB				DNIB1				DNIB2			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	DNIB3				DNIB4				DNIB5				DNIB6			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. The value of the CH is device-specific. Refer to the Device Configuration chapter for details.

Figure 1436. Channel n Fast Message Data read register (CHn_FMSG_DATA)

Table 1483. CHn_FMSG_DATA field descriptions

Field	Description
0:3 CHNUM	Channel Number Indicates the channel number of the received message being read by DMA. Valid values are 0 to ‘15’.
4:7 SCNIB	Status and Communication Nibble of message
8:11 DNIB1	Data Nibble 1 Always supported.
12:15 DNIB2	Data Nibble 2 Should be ignored if number of data nibbles supported by channel is 1.
16:19 DNIB3	Data Nibble 3 Should be ignored if number of data nibbles supported by channel is 2 or less.
20:23 DNIB4	Data Nibble 4 Should be ignored if number of data nibbles supported by channel is 3 or less.
24:27 DNIB5	Data Nibble 5 Should be ignored if number of data nibbles supported by channel is 4 or less.
28:31 DNIB6	Data Nibble 6 Should be ignored if number of data nibbles supported by channel is 5 or less.

59.3.2.22 Channel n Fast Message CRC read register (CHn_FMSG_CRC)

Offset: 0x0164 + n*0x18 (n = 0 to (CH[1]-1)) Access: User read only

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	CRC4b			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

1. The value of the CH is device-specific. Refer to the Device Configuration chapter for details.

Figure 1437. Channel n Fast Message CRC read register (CHn_FMSG_CRC)

Table 1484. CHn_FMSG_CRC field descriptions

Field	Description
12:15 CRC4b	4-bit CRC value of the message

59.3.2.23 Channel n Fast Message Time Stamp read register (CHn_FMSG_TS)

Offset: 0x0168 + n*0x18 (n = 0 to (CH[1]-1)) Access: User read-only

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	TS															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	TS															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. The value of the CH is device-specific. Refer to the Device Configuration chapter for details.

Figure 1438. Channel n Fast Message Time Stamp read register (CHn_FMSG_TS)

Table 1485. CH*n*_FMSG_TS field descriptions

Field	Description
0:31 TS	Time Stamp for received message Indicates the relative time when the falling edge that comes at the end of the calibration pulse (which is also the start of the status and communication nibble pulse) was detected.

59.3.2.24 Channel n Slow Serial Message Bit3 read register (CHn_SMSG_BIT3)

Offset: 0x016C + n*0x18 (n = 0 to (CH[1]-1)) Access: User read-only

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	CHNUM				TYPE	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	CFG	ID7_6/ ID3_2		ID5_4/ ID1_0		0	ID3_0/ DATA15_12				0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. The value of the CH is device-specific. Refer to the Device Configuration chapter for details.

Figure 1439. Channel n Slow Serial Message Bit3 read register (CHn_SMSG_BIT3)

Table 1486. CH*n*_SMSG_BIT3 field descriptions

Field	Description
0:3 CHNUM	Channel Number Indicates the channel number of the received message being read by DMA. Valid values are 0 to 15.
4 TYPE	Serial Message Type Indicates the type of serial message received. 0 Short serial message 1 Enhanced serial message
20	Placed as 0 by specification.
21 CFG	Configuration bit ‘C’ Indicates the type of Enhanced Serial message. 0 Enhanced serial message with 8-bit ID field 1 Enhanced serial message with 4-bit ID field
22:23 ID7_6/ID3_2	ID Field Value depends on setting of ‘C’ bit. If ‘C’ bit is 0, this field is ID[7:6] and if the ‘C’ bit is 1, this field is ID[3:2].
24:25 ID5_4/ID1_0	ID Field Value depends on setting of ‘C’ bit. If ‘C’ bit is 0, this field is ID[5:4] and if the ‘C’ bit is 1, this field is ID[1:0].
26	0 As per SAE specification.
27:30 ID3_0/DATA15_12	ID or Data Field Value depends on setting of ‘C’ bit. If ‘C’ bit is 0, this field is ID[3:0] and if the ‘C’ bit is 1, this field is Data[15:12].
31	0 As per SAE specification.

59.3.2.25 Channel n Serial Message Bit2 read register (CHn_SMSG_BIT2)

Offset: 0x0170 + n*0x18 (n = 0 to (CH(1)-1)) Access: User read-only

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	SMCRC					
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	ID3_0/ DATA11_8				DATA7_0							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. The value of the CH is device-specific. Refer to the Device Configuration chapter for details.

Figure 1440. Channel *n* Slow Serial Message Bit2 read register (CH*n*_SMSG_BIT2)

Table 1487. CH*n*_SMSG_BIT2 field descriptions

Field	Description
10:15 SMCRC	6-bit CRC value of the message When TYPE bit in *Channel n Slow Serial Message Bit3 read register (CHn_SMSG_BIT3)* is set to 0, this CRC is of 4-bits for Short serial messages and when TYPE bit is set to 1, this CRC is of 6-bits for Enhanced serial messages.
20:23 ID3_0/DATA11_8	ID or Data Field When TYPE bit in *Channel n Slow Serial Message Bit3 read register (CHn_SMSG_BIT3)* is set to 0, this field is ID [3:0] for Short serial messages and when TYPE bit is set to 1, this field is DATA [11:8] for Enhanced serial messages.
24:31 DATA7_0	Data Field Data field [7:0] for short serial messages and enhanced serial messages.

59.3.2.26 Channel n Serial Message Time Stamp read register (CH*n*_SMSG_TS)

Offset: 0x0174 + n*0x18 (n = 0 to (CH(1)-1)) Access: User read-only

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	TS															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	TS															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. The value of the CH is device-specific. Refer to the Device Configuration chapter for details.

Figure 1441. Channel *n* Slow Serial Message Time Stamp read register (CH*n*_SMSG_TS)

Table 1488. CHn_SMSG_TS field descriptions

Field	Description
0:31 TS	Time Stamp for the received message Indicates the relative time of detection of the falling edge that comes at the end of the calibration pulse (which is also the start of the status and communication nibble pulse) of the last Fast Message that completed the serial message. Time Stamp for Slow serial messages is not latched when the Slow serial message starts but when the Slow serial message is completely received.

59.3.3 Register classification for safety

The registers of this module are classified as per Safety Requirements as follows:

Safety Critical: There are no safety critical registers in this module.

Safety Relevant: The register bits/fields in this category are safety relevant as any random error in these bits will lead to unwanted interference to CPU/DMA. The registers containing these bits must be protected by periodic CRC or equivalent check implemented at SoC Level. The following register bits/fields are classified in this category:

- *Channel Enable register (CHNL_EN)*
- *Fast Message Ready status register (FMSG_RDY)*
- *Slow Serial Message Ready status register (SMSG_RDY)*
- *Data Control register 1 (DATA_CTRL1)* and *Data Control register 2 (DATA_CTRL2)*
- *Fast Message DMA Control register (FDMA_CTRL)*
- *Slow Serial Message DMA Control register (SDMA_CTRL)*
- *Fast Message Ready Interrupt Control register (FRDY_IE)*
- *Slow Serial Message Ready Interrupt Enable register (SRDY_IE)*
- Fast Message DMA read register set and Slow message DMA read register set

Safety Latent: The register bits/fields in this category should be protected by LBIST at SoC level. The following register bits/fields are classified in this category:

- *Global Status register (GBL_STATUS)*
- *Channel n Status register (CHn_STATUS)*
- *Channel n Configuration register (CHn_CONFIG)*

Non-Safety: The following register bits are classified in this category:

- All other registers and bits fall in this category.

59.4 Functional description

59.4.1 Initialization sequence

All the channels are required to be configured before enabling them by writing into corresponding channel enable bit in the Channel Enable Register (CHNL_EN). There are some registers which must be configured and some others which are to be changed if reset value is not as required for that channel. After enabling the channel, no configuration should be changed and if required to be changed, the user needs to first disable the channel and then reconfigure the channel parameters. The following must be configured before enabling the corresponding channel:

- Time Stamp Prescaler Value in to generate the time stamping clock.
- Rx Prescaler Value in *Channel n Clock Control register (CHn_CLK_CTRL)* to generate the channel's receiver clock.
- Compensation enable in *Channel n Clock Control register (CHn_CLK_CTRL)*.
- Channel Configuration settings in the *Channel n Configuration register (CHn_CONFIG)*.
- Number of Data Nibbles in Channel in the *Data Control register 1 (DATA_CTRL1)* and *Data Control register 2 (DATA_CTRL2)*.
- Control bits that control whether messages received in channel are to read by interrupt or DMA in *Fast Message DMA Control register (FDMA_CTRL)*, *Slow Serial Message DMA Control register (SDMA_CTRL)*, *Fast Message Ready Interrupt Control register (FRDY_IE)* and *Slow Serial Message DMA Control register (SDMA_CTRL)*.

Note: When configuring the Single Edge Nibble Transmission (SENT) Receiver (SRX) to receive message with the Option 1 of the successive calibration pulse check method (CHn_CONFIG[SUCC_CAL_CHK] = 1), the number of expected edges error (CHn_STATUS[NUM[EDGES_ERR]) gets randomly asserted. Option 2 is not affected as the number of expected edges are not checked in this mode.

The error occurs randomly when the channel input (on the MCU pin) goes from idle to toggling of the calibration pulse.

Note: The Successive Calibration Pulse Check Method Option 1 and Option 2 are defined as follows:

- *Option 1: Preferred but High Latency Option per SAE specification.*
- *Option 2: Low Latency Option per SAE specification.*

To avoid getting the error, the sensor should be enabled first (by the MCU software) and when it starts sending messages, the SENT module should be enabled in the SENT Global Control register (by making GBL_CTRL[SENT_EN] = 1). The delay in start of the two can be controlled by counting a fixed delay in software between enabling the sensor and enabling

the SENT module. The first message will not be received but subsequent messages will get received and there will be no false assertions of the number of expected edges error status bit (CHn_STATUS[NUM[EDGES_ERR]).

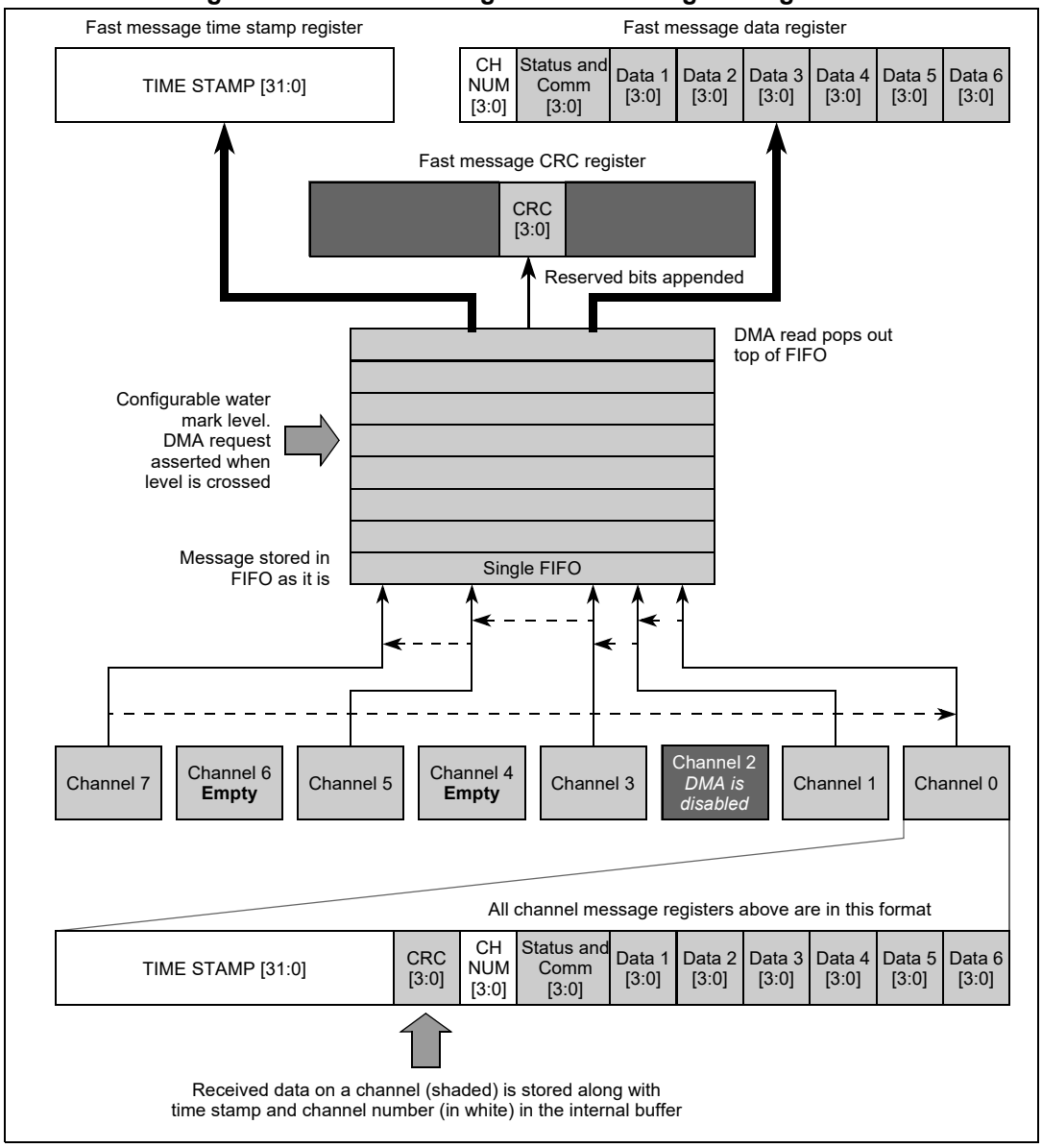
Alternatively, software can count the period from SENT enable (GBL_CTRL[SENT_EN] = 1) to the first expected calibration pulse. If the number of expected edges error status bit (CHn_STATUS[NUM[EDGES_ERR]) is asserted, software can simply clear it as there have no messages which have been completely received.

Alternatively, the software can clear this bit at the start and move ahead. When pause pulse is enabled, then NUM_EDGES will not assert spuriously for subsequent messages which do not have errors in them or cause overflows.

59.4.2 DMA read logic

The SENT module generates separate DMA requests for all Fast messages and Slow serial messages received. The DMA requests are common for all channels. Each packet stored in the channels' buffers are appended with a time stamp taken from a free running time keeping counter and the channel number on which it was received. Details on this time stamp counter can be found in *Section 59.4.8: Time stamp logic*.

Figure 1442. Fast message DMA Read Logic using FIFO



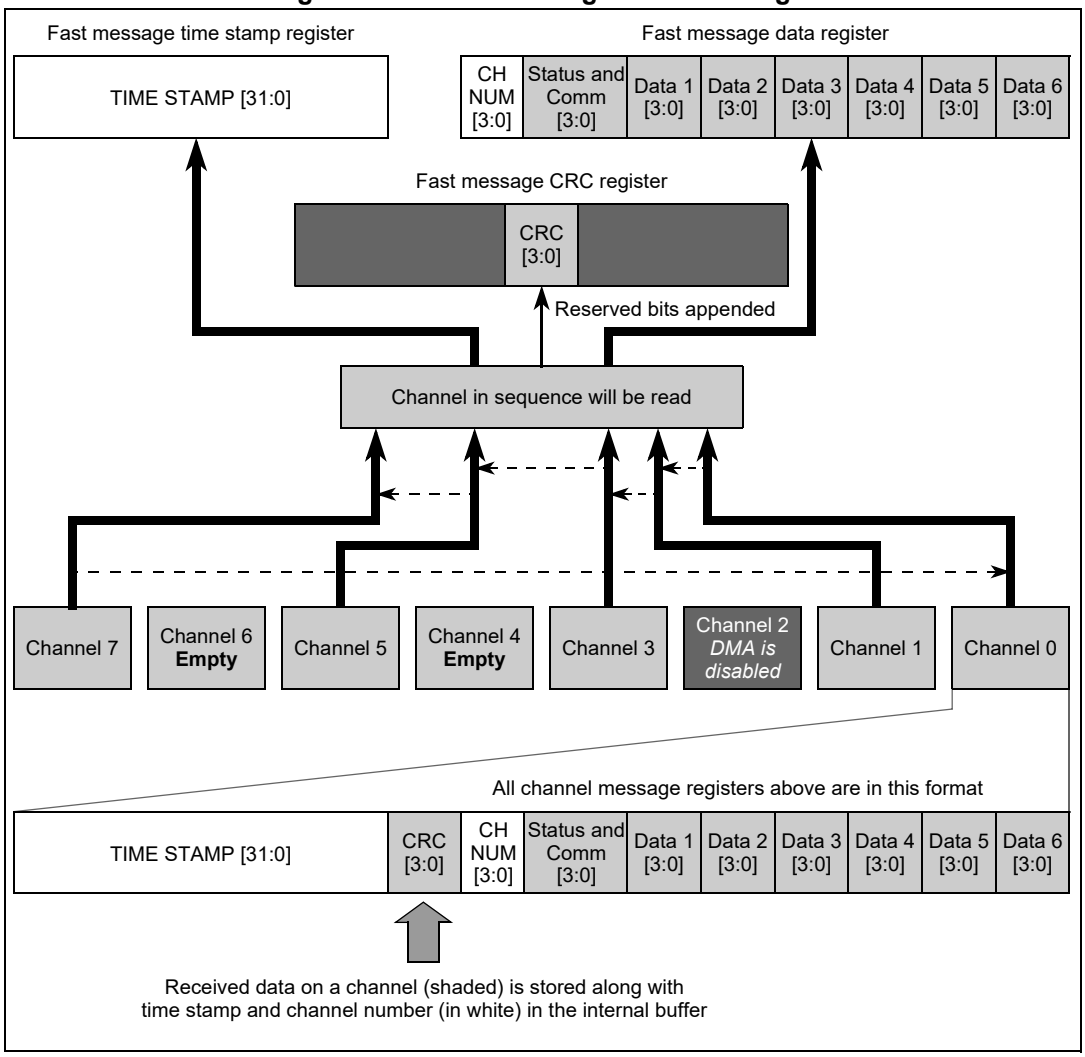
59.4.2.1 DMA request for Fast message reading

Fast messages received are stored in a single system bus clock domain buffer to be read out by the DMA controller via the DMA Fast message read register set. Note that the DMA controller must read all 3 DMA read registers to obtain the complete message.

The data flow will happen as follows:

- A DMA request will be asserted when any of the DMA enabled channels has successfully received one complete Fast message. The request is kept asserted till all messages (across DMA enabled channels) have been read out.
- DMA controller should read the complete message by accessing each register of the DMA Fast message read register set, the registers being:
  - *DMA Fast Message Data read register (DMA_FMSG_DATA)*
  - *DMA Fast Message CRC read register (DMA_FMSG_CRC)*
  - *DMA Fast Message Time Stamp read register (DMA_FMSG_TS)*
- A fixed round robin sequencing runs, as shown in *Figure 1443*, and this logic checks for messages in sequence 0→1→3 →5→7 and back to channel 0 and so on. Channel 2 is skipped as it is not DMA enabled) and channel 4 and 6 are skipped as message is not received completely. This sequence continues like this.
- The sequence in which messages are read through DMA interface is not deterministic and depends on the sequence in which they are completely received.
- When the sequencing logic detects a message ready to be read, it will update the DMA Fast message read register set for DMA read and move to the next channel when this message is read out completely (such as, 3 read accesses to the above mentioned registers).
- The DMA controller should be programmed to read only one message in one DMA transfer (such as,12 bytes). If it is programmed to read more than one message in one DMA transfer, underflow in buffers might occur.
- User software should ensure there is no overflow in channels. In case overflow occurs, the message in the channel buffer will get overwritten by the newly received message. However, when a DMA read is in progress, the buffers will not be overwritten if overflow occurs to maintain data integrity.
- The Round Robin sequencing logic loops through each DMA enabled channel and halts at any channel having data till it is read out by the DMA controller.

Figure 1443. Fast message DMA read logic



Note: The data buffer related to the SENT data transferred by the DMA in the system memory is in the following order DMA_FMSG_TS, then DMA_FMSG_CRC and then DMA_FMSG_DATA.

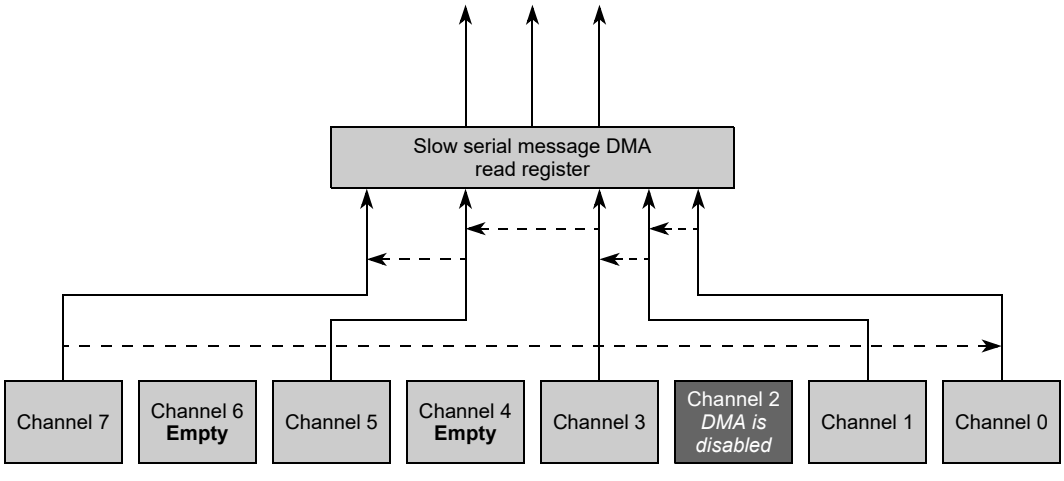
59.4.2.2 DMA request for Slow serial message reading

A separate DMA request (different from DMA request for Fast message) is generated for all slow serial messages received on all channels. The data flow will happen as follows:

- Request will be asserted when any of the DMA enabled channels has successfully received one complete serial message (short or enhanced) and kept asserted till all messages have been read from their respective memory-mapped registers.
- Similar to DMA read for Fast messages, data for slow serial messages will be read from DMA Slow serial message status register set in memory map, the register set comprising of:
  - *DMA Slow Serial Message Bit3 read register (DMA_SMSG_BIT3)*
  - *DMA Slow Serial Message Bit2 read register (DMA_SMSG_BIT2)*
  - *DMA Slow Serial Message Time Stamp read register (DMA_SMSG_TS)*
- A fixed round robin sequencing logic will be running and it will update the DMA Slow serial message status register set for DMA read and move to the next channel when this message is read out completely (such as, 3 DMA read accesses to the above mentioned registers).
- The channel number and time stamp will also be appended for every message.
- A message type bit (TYPE) will also be appended to distinguish between short and enhanced serial messages.
- The DMA controller should be programmed to read only one message in one DMA transfer (such as, 12 bytes). If it is programmed to read more than one message in one DMA transfer, underflow in buffers might occur.
- Since serial messages receive rate is very slow, user software should ensure there is no overflow.
- The fixed round robin sequence logic for DMA read is the same as that used for Fast messages (see *DMA request for Fast message reading*).

Figure 1444 shows the sequencing of channels for serial messages to be read when DMA read is done on DMA Slow serial message status register set.

Figure 1444. Slow serial message DMA read logic



Note: The arrows in the figure indicate next DMA transfer. One DMA Read Register set (3 registers) will be read in one transfer.

The formats in which the 3 types of serial messages will be stored in the buffers are shown in the figures below.

Figure 1445. Enhanced serial message with 8-bit ID field

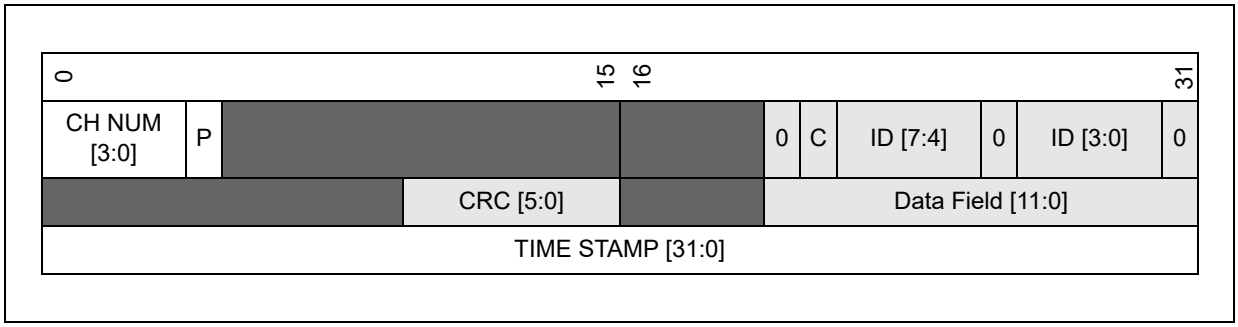


Figure 1446. Enhanced serial message with 4-bit ID field

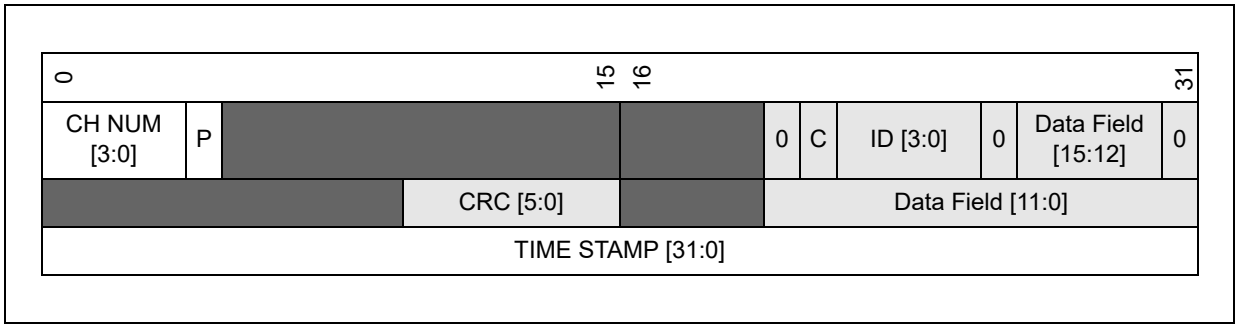
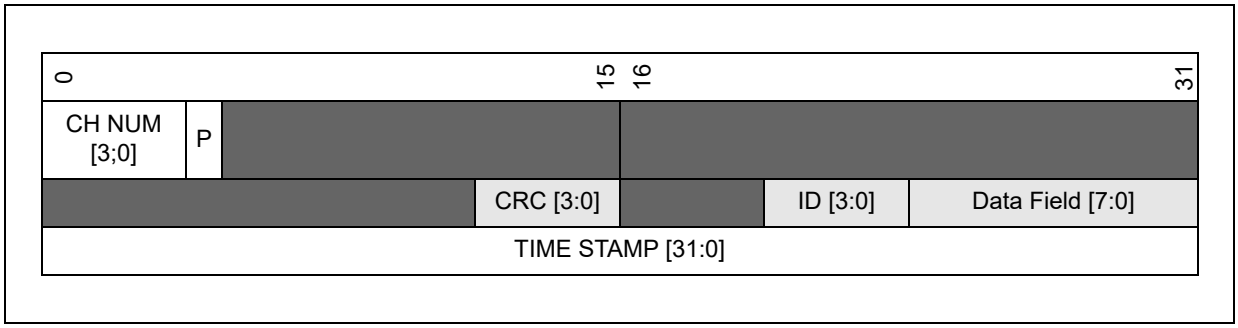


Figure 1445 and *Figure 1446* show the enhanced serial messages. Bit C (shown in figure) or bit CFG (as shown in *DMA Slow Serial Message Bit3 read register (DMA_SMSG_BIT3)* description) defines the type of enhanced packet and bit P (shown in figure) or bit TYPE (shown in *DMA Slow Serial Message Bit3 read register (DMA_SMSG_BIT3)* description) defines whether it is short serial or enhanced serial message. CRC, ID and Data fields are stored in format as received and shown in SAE specification with just one exception that CRC is shifted to align with 16-bit boundary.

Figure 1447 shows the short serial message. Field locations have been placed in such a way to allow a common register to read all the slow serial messages and also match the bit positions in actual message defined by SAE specification.

Figure 1447. Short serial message



Three 32-bit accesses to be made by DMA to get one complete message with time stamp.

59.4.3 Message reading via interrupts

There is one interrupt request per channel for a Fast message received and one for a serial message received. Each channel is individually configurable to enable or disable interrupt on reception of Fast or Slow Serial messages separately. In case of interrupt request of Fast message, an interrupt will be asserted when any of the channels has received a complete Fast message without errors and that channel is not configured for DMA Reads. The interrupt routine should be coded in such a way to take the least amount of overhead (in CPU cycles) while entering the routine and start reading the available messages. The routine should be capable of accessing all channels that are enabled to give out messages via interrupts through the register interface.

In response to these interrupts, the CPU is expected to read interrupt status register for fast or slow messages (depending on which interrupt was asserted) and store it in system memory. Using this status register value, the interrupt routine can decide from which channels the message is to be read. For instance if bit corresponding to channel 0 is set then routine should read from the address location corresponding to Channel 0 Fast message read registers and move onto the next asserted bit. The interrupt routine should scan through all the bits of this stored interrupt status register value and read corresponding channel's fast or Slow message registers. In this manner all available messages can be read. After scanning through all the bits of status register stored in system memory, routine can read again the message ready interrupt status register to find out if any other channel has received in the mean time. This will save another interrupt routine time mainly from stacking/unstacking delay.

It is assumed that the interrupt controller will be able to respond fast enough to avoid any overflow as only one message per channel will be stored and the system bus clock frequency is sufficient to avoid this.

59.4.3.1 Suggested software interrupt service routine steps

These are the suggested steps to have an interrupt routine that can read all messages quickly and avoid any underflow/overflow. Actual implementations can vary.

When separate Fast message and Slow serial message interrupts are used:

The Fast message Interrupt should have higher priority than Slow serial message interrupt. The suggested ISR steps for a Fast message interrupts can be:

- Read the Fast message ready register and store it in a variable.
- Shift out bits starting from MSB from this variable.
- If bit is 1, perform read operation on corresponding channel's memory-mapped Fast message read register set (3 registers). Three read accesses will fetch complete message. Shift out new bit. Repeat step.
- Else if bit is 0, shift out new bit.
- When all bits are shifted out, read the Fast message status registers again to ensure no new message has arrived during the ISR.
- If there are no bits set, exit the ISR, else repeat this process.

The ISR steps for Slow serial message interrupt can be the same as for Fast message ISR.

When common Fast message and Slow serial message interrupts are used:

The suggested ISR steps can be:

- Read the Fast message status register and store it in a variable.
- Shift out bits starting from MSB from this variable.
- If bit is 1, perform read operation on corresponding channel's memory-mapped Fast message read register. Threeread accesses will fetch complete message . Shift out new bit. Repeat step.
- Else if bit is 0, shift out new bit.
- When all bits are shifted out, read the Slow serial message status register into a new variable.
- Shift out bits starting from MSB from this new variable.
- If bit is 1, perform read operation on corresponding channel's memory-mapped Slow serial message status register. Three read accesses will fetch complete message. Shift out new bit. Repeat step.
- Else if bit is 0, shift out new bit.
- When all bits are shifted out, software can again check both Fast and Slow serial message Ready register to ready any message that might have been received while the routine was being serviced, or can choose to exit the routine.

59.4.4 Overflow behavior

This section describes the overflow behavior.

59.4.4.1 Fast messages buffers

Overflow in Fast message buffers can occur when the CPU or DMA does not read messages from that channel for a long period of time. The receiver channel stores two messages in its buffers and indicates an overflow when these buffers are full.

When an overflow event occurs, the newly received message overwrite the previously stored message and the overflow status bit is set. However, if overflow asserts while CPU/DMA is reading that channel (such as, all **3** read registers have not yet been read), then that message is not overwritten. While messages are being overwritten, the slow message reception is not hampered.

59.4.4.2 Slow message buffers

Overflow in Slow message buffers can occur when the CPU or DMA does not read messages from that channel for a long period of time. The receiver channel stores two messages in its buffers and indicates an overflow when these buffers are full.

When an overflow event occurs, further reception of slow messages is halted until the buffers are read out by CPU or DMA. Since it is less likely that CPU or DMA do not read the read registers for 32 or 36 fast message length duration, an overflow in slow messages would not be as frequent as in fast messages.

59.4.5 Adjustment for variation in sensor (Tx) clock

The clock used by the Channel Receiver (referred to as Receiver Clock, or Rx_CLK) to sample the message nibbles is generated from a High Frequency Receiver Clock or Protocol Clock. This high frequency receiver clock is divided using a prescaler counter

which generates a tick when it rollovers on reaching a count equal to the Prescaler Value programmed in the channel's clock control register. The receive clock is generated separately for each channel. By varying this Prescaler Factor, we can control the period of the receive clock and make it close to the Transmitter clock used by the sensor for the particular channel.

After reset and once channel is enabled, the compensated prescaler value is set to Zero and the user software programs the Prescaler Value to obtain the desired receiver clock frequency for the particular channel. Thus, the receiver starts working by assuming that the transmitter is sending messages on the frequency that is programmed in its channel clock control register. The channel's input after filtering is continuously measured by the Prescaler Counter. The compensation logic checks each pulse length and determines the calibration pulse from them. When a calibration pulse is detected, the compensated prescaler value is computed and stored in the Channel's Clock Control Register (*Channel n Clock Control register (CHn_CLK_CTRL)*). Software can use this value to determine the variation in Tx Clock.

Note: A pause pulse may incidentally be detected as a synchronization/calibration pulse during start up after reset. The correction value would be computed on the pause pulse too but since the next synchronization pulse would follow, the logic would again detect the falling edge on the synchronization pulse and the correction value would now be correctly computed on the correct synchronization pulse.

The Clock Compensation or Correction happens for every messages that is received and the compensated prescaler value remains constant for that message.

Note: At ambient temperatures above 114 degree C, Single Edge Nibble Transmission (SENT) Receiver (SRX) generates CAL_LEN_ERR and NUM_EDGES error when calibration pulse length from sensor is 25% higher than normal length. As temperature is increased, the frequency of occurrence of these errors also increases.

By configuring the SENT high frequency clock in the SRX module to run at a slower frequency (40 MHz) will help to reduce the occurrence of these errors.

59.4.5.1 Adjustment for nibble length variation

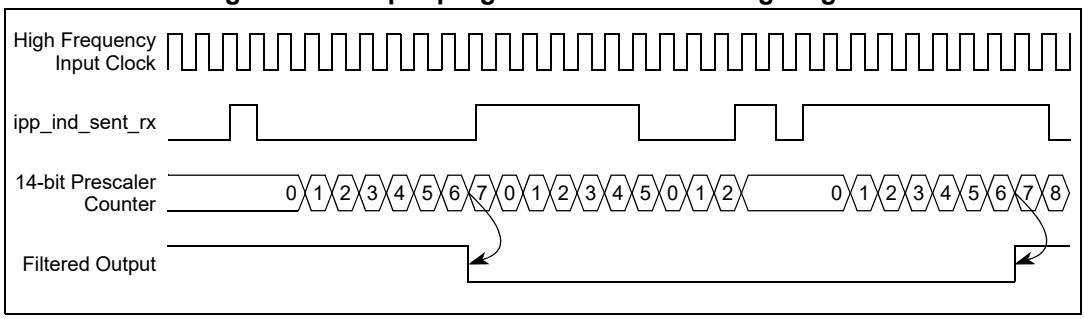
Section 6.2 of SAE specifications specifies the maximum allowable limits for clock drift and jitter for the transmitter and receiver separately. This can cause the length of nibbles to vary and can cause incorrect sampling of data nibbles. This variation is adjusted by the SENT Receiver when nibbles are being sampled.

59.4.6 Input programmable filter

The Input Programmable Filter ensures that only valid input pin transitions are received by the SENT Receiver module. The input programmable filter is an 8-bit programmable up counter that increments on the high frequency receiver clock. The sensor input signal from device's pad is synchronized by high frequency receiver clock. When a state change occurs in the sensor input signal, the 8-bit counter resets and starts counting up on the high frequency receiver clock. As long as the new state is stable on the pin, the counter remains incrementing. If a counter overflows occurs, the new pin value (such as, sensor input) is validated and latched in the filtered output flop (also running on high frequency receiver clock). If the opposite edge appears on the sensor input before validation (such as, overflow), the counter is reset and the input is not passed onto the filtered output. At the next pin transition, the counter starts counting again. Any pulse that is shorter than a full range of the masked counter is regarded as a glitch and it is not passed on to the filter

output. The number of samples (or the match value of the up counter) is programmable by the user in the channel's configuration register (*Channel n Configuration register (CHn_CONFIG)*). When zero is programmed into the registers, it bypasses the filter and the synchronized input is output as the filtered output. A non-zero value delays the filtered output by the number of clocks proportional to the programmed value in the Channel Configuration Register (*Channel n Configuration register (CHn_CONFIG)*). A timing diagram of the input filter when FIL_CNT is 4, is shown in *Figure 1448*.

Figure 1448. Input programmable filter timing diagram



59.4.7 Receiver diagnostics

Most of the receiver diagnostics are controlled by the Fast Message State Machine. The following checks are conducted by the receiver in each channel.

- Calibration pulse length < 56 clock ticks – 25% or > 56 clock ticks + 25%.
- Not the expected number of falling edges between calibration pulses (Message length is pre-defined by each sensor device and programmed by user for each channel).
- Checksum error. Two 4-bit CRC checks and one 6-bit CRC check according to the message type. User has a programmable option to select the method of CRC to use such as, Legacy or XOR-based.
- Any nibble data values measured as < 0 or > 15.
- Successive calibration pulses differ by > +1.5625% (1/64) or < -1.5625%. The accuracy of this check will depend on the frequency of the protocol clock used.
- For messages with pause pulse, an additional diagnostic is done; however, this diagnostic will not fail any message (to reduce latency in message reception) but the failure of this diagnostic will be indicated in the channel's status register.

On detection of any of the above errors, the receiver rejects the current packet until a new calibration pulse is detected. Occurrence of an error is flagged in the respective channel's status register (*Channel n Status register (CHn_STATUS)*). If an error interrupt is enabled, an error interrupt will be generated. There is a single error interrupt generated for all errors, so the CPU must read the status register to find out which error occurred. Once error condition is flagged, subsequent errors (except NUM_EDGES_ERR) are automatically masked out until a valid calibration pulse is detected, after which all checks are automatically enabled. This is done to avoid multiple assertions of the interrupt to the CPU in case the channel becomes unstable and sends wrong data continuously.

59.4.7.1 Calibration pulse length check

The calibration pulse length check happens in conjunction with Clock Compensation Logic since current calibration pulse is detected in this block. The Clock Compensation Logic

detects a calibration pulse of length between 42 and 70 tick counts. This check will assert the CAL_LEN_ERR status bit on failure.

59.4.7.2 Successive calibration pulse check

The successive calibration pulse check checks if successive calibration pulses differ by > +1.5625% (1/64) or < -1.5625%. The accuracy of this check will depend on the frequency of the protocol clock used. Specifically, the higher the frequency, the closer the check will be to 1.5625%.

So if protocol clock is T_{HF_CLK} and channel tick period is T_{TX_CLK} then the check will be of 1.5625% ± {100 × (4 × T_{HF_CLK}) / (42 × T_{TX_CLK})}.

So it will not allow variance in clock of a channel by 1.5625% - {100 × (4 × T_{HF_CLK}) / (42 × T_{TX_CLK})}.

This check will assert the CAL_DIAG_ERR status bit on failure.

When Option 2 of successive calibration pulse check is enabled, the CAL_RESYNC bit will be asserted on every third successive error detected by this check.

59.4.7.3 Not the expected number of edges check

The number of negative edges between two successive calibration pulses is counted and compared with expected number of edges in case of preferred option of successive calibration pulse check. The expected number of nibbles is the sum of following:

- Status and Communication nibble (1 No.)
- Data nibbles (Programmable as per user software)
- CRC nibble (1 No.)
- Pause Pulse, if configured (1 No.)

This check (“Not the expected number of edges check”) will assert the NUM_EDGES_ERR status bit on failure. The check does not run when Option 2 (low latency) of successive calibration pulse check is enabled. This check is not masked after any other error is detected so it could be redundant if it is flagged along with CAL_DIAG_ERR and CAL_LEN_ERR. However, this is true only when CPU is clearing error status bits as and when they are asserted. If errors are accumulated, the above cannot be deduced.

59.4.7.4 Nibble value check

As part of this check, the lengths of the following nibbles are checked to remain in between 0 and 15 ticks:

- Status and Communication nibble
- all Data nibbles
- CRC nibble

The nibble value check will assert the NIB_VAL_ERR status bit on failure. This error can be flagged along with NUM_EDGES_ERR in which case the former can be ignored as the number of pulses in the message is incorrect. However, this is true only when CPU is clearing error status bits as and when they are asserted. In case, errors are accumulated, the above cannot be deduced.

59.4.7.5 CRC check

The CRC check will compute both 4-bit and 6-bit CRC as requested by State Machine control logic. Two 4-bit CRCs and one 6-bit CRC are supported to compute the CRC on Fast/Slow serial message (4-bit CRC) and Enhanced serial message (6-bit CRC) respectively. The SENT Receiver Module allows the user to calculate the 4-bit CRC using either the legacy CRC implementation method or the new XOR-based CRC implementation via a control bit in the channel's corresponding control register. Refer to SAE SENT specification for details on CRC algorithm.

Asserting the CHn_CONFIG_REG[11] bit, causes the Status and Communication nibble to be included in the 4-bit CRC calculation for Fast serial messages. De-asserting this bit enables normal operation as described above.

This check will assert the FMSG_CRC_ERR or SMSG_CRC_ERR status bit on failure, depending on whether the check is run for Fast message or Slow message.

The FMSG_CRC_ERR can be flagged along with the NUM_EDGES_ERR, in which case the CRC error can be ignored as the number of pulses in the message was incorrect. However, this is true only when CPU is clearing error status bits as and when they are asserted. If errors are accumulated, the above cannot be deduced.

Note: The Single Edge Nibble Transmitter (SENT) Receiver does not properly round off incoming data to nearest nibble. For example, if the data nibble input to the SENT receiver is 3.5 to 3.87, it is rounded off to 3 instead of 4 as the jitter tolerance is 12.5% instead of 50%. As a result, the Channel n Fast Message Data Read Register (CHn_FMSG_DATA), or Channel n Fast Message Cyclic Redundancy Check (CRC) register (CHn_FMSG_CRC) register values may not match the expected values or the Message is not received at all because of the CRC mismatch.

Disable the CRC check by setting (to 0b1) the Fast Message CRC Check Off bit in the Channel Configuration register (CHn_CONFIG) and perform the CRC check of messages in software.

59.4.7.6 Pause pulse diagnostic

The pause pulse diagnostic checks whether the ratio of calibration pulse length to full message length (including pause pulse) varies from one message to another by less than 1.5625% (as per specification). In order to have reduced latency of operation, message reception is not gated due to this check and no messages are discarded. However, when this check fails, a status bit is asserted to let the software know of this condition. The successive calibration pulse check can be configured to operate along with this check. This configurability is added to avoid conflicting results from the two checks.

There can be some inaccuracy in measuring pause pulse length as SAE specification specifies maximum clock drift/jitter up to maximum nibble length only and the frequency of high frequency receiver clock (protocol clock) used, allows clock compensation logic to ensure accuracy up to maximum nibble length. Hence there could be some inaccuracy in this check when sampling pause pulses that are larger than the maximum nibble pulse length.

This check will assert the PP_DIAG_ERR status bit on failure. This check does not run when reception of pause pulse is disabled.

When the Single Edge Nibble Transmission (SENT) Receiver (SRX) is configured to receive a pause pulse (Channel 'n' Configuration Register - CHn_CONFIG[PAUSE_EN] = 1) and the length of the pause pulse from the sensor is in range of the valid calibration pulse length,

the NUM_EDGES error can get asserted spuriously (Channel ‘n' Status Register - CHn_STATUS(NUM_EDGES_ERR] = 1) when there is any diagnostic error (other than number of expected edges error) or overflow in the incoming messages from the sensor.

Software can distinguish a spurious NUM_EDGES_ERR error from a real one by monitoring other error bits. The following tables will help distinguish between a false and real assertion of NUM_EDGES_ERR error and other errors. Software should handle the first error detected as per application needs and other bits can be evaluated based on these tables. *Table 1489* contains information due to erratum behavior and *Table 1490* below contains clarification of normal NUM_EDGES_ERR behavior.

Table 1489. Behavior of NUM_EDGES_ERR

First error detected	Other error bits asserted	Cause for extra error bits getting asserted	Action
NIB_VAL_ERR	NUM_EDGES_ERR asserted twice	Upon detection of the first error, the state machine goes into a state where it waits for a calibration pulse, the first NUM_EDGES_ERR error is for the current message as the state machine does not detect an end of message. The second error comes when both the Pause pulse and the Calibration pulse are seen as back to back calibration pulses and no edges in between.	Ignore both NUM_EDGES_ERR error
FMSG_CRC_ERR	NUM_EDGES_ERR asserted twice	Upon detection of the first error, the state machine goes into a state where it waits for a calibration pulse, the first NUM_EDGES_ERR error is for the current message as the state machine does not detect an end of message. The second error comes when both the Pause pulse and the Calibration pulse are seen as back to back calibration pulses and no edges in between.	Ignore both NUM_EDGES_ERR errors
CAL_LEN_ERR	NUM_EDGES_ERR asserted once	Since the calibration pulse is not detected as a valid calibration pulse, the internal edges counter does not detect the end of one message and start of bad message (which has CAL_LEN_ERR); hence the NUM_EDGES_ERR gets asserted.	Ignore NUM_EDGES_ERR error
FMSG_OFLW	NUM_EDGES_ERR asserted once (random occurrence)	A message buffer overflow may lead the state machine to enter a state where it waits for a calibration pulse (behavior also seen in ERR007404). When in this state, the state machine can detect both a Pause pulse and a Calibration pulse as back to back calibration pulses and no edges in between. Then, the NUM_EDGES_ERR can get asserted. Since entry into this state is random, the error can be seen occasionally.	Ignore NUM_EDGES_ERR error

Table 1490. Expected behavior, clarification of NUM_EDGES_ERR cases

First error detected	Other error bits asserted	Cause for extra error bits getting asserted	Action
NUM_EDGES_ERR (when edges are less than expected)	NIB_VAL_ERR is asserted	When the actual number of edges in the message are less than expected, then a pause pulse gets detected as a nibble since the state machine expects nibbles when actually there is a pause pulse present. This generates NIB_VAL_ERR.	Ignore the NIB_VAL_ERR
NUM_EDGES_ERR (when edges are more than expected)	NIB_VAL_ERR and PP_DIAG_ERR are asserted	When the actual number of edges in a message are more than expected, then after receiving the programmed number of data nibbles, the state machine expects a pause pulse. However, the pause pulse comes later and gets detected as a nibble and hence NIB_VAL_ERR is asserted. Since the message length is not correct, PP_DIAG_ERR is also asserted.	Ignore NIB_VAL_ERR and PP_DIAG_ERR

59.4.8 Time stamp logic

A single counter maintains the time stamp for all channels which can be used by the software to determine the relative time of arrival of the messages for each channel. The time stamp counter is a 32-bit counter.

The clock for the counter is derived from the high frequency receiver clock using a Time Stamp Prescaler Counter that divides the high frequency receiver clock to generate the clock of required resolution. The user must program the time stamp prescaler value in order to generate the time stamp clock. The period of the time stamp clock must be less than that of the fastest receiver channel being used, or it can be set at some fixed value determined by application, such as 1 µs. For example, if the high frequency receiver clock is 60 MHz, the required prescaler value to generate a time stamp clock of 1 µs resolution is 59.

Note: The rollover of time stamp values should be determined in user software, by comparing the time stamp values in previous and current messages that are read out. User software should adjust the time stamp values in these messages accordingly, after reading them.

The time stamp for a Fast message is sampled at the falling edge that comes at the end of the calibration pulse (which is also the start of the status and communication nibble pulse) and is stored in the message read buffer. However, if the current message is found to be in error, the message buffer is discarded. Thus, the time stamp will be available to user software for only those messages which do not have any error.

The time stamp for a serial message is sampled at the end of the message such as when the serial message is completely received and when the message is found to have no errors. the time stamp is stored in the message read buffer to be read by the user software. Again, for messages with error the message and time stamp are discarded.

59.4.8.1 Limitation

Since the same counter is used for all channels, it may happen that more than 1 message is received within the same time stamp across different channels. Hence the software would not be able to determine the sequence of arrival of messages which have the same time stamp value.

59.4.9 Bus-idle diagnostic

This diagnostic checks the connected sensors for inactivity. SENT module can indicate if the idle period on a particular channel has crossed the period defined by the programmed value in Bus Idle Count (in *Channel n Configuration register (CHn_CONFIG)*). Status bit (bus_idle in *Channel n Status register (CHn_STATUS)*) will be set when idle period crosses the allowed value. CPU should write 1 to this clear this bit once it is read.

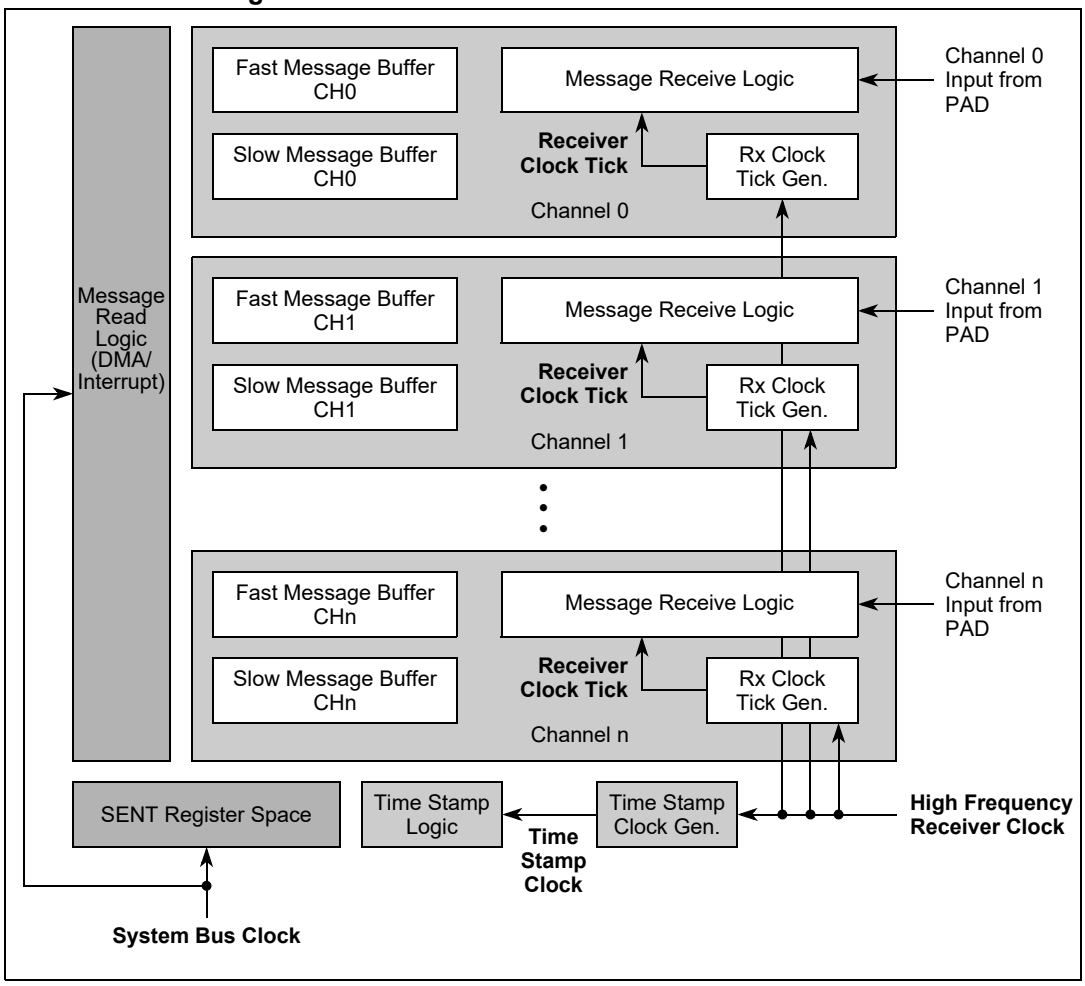
59.5 Clocks and resets

The SENT Receiver works on two clocks. First is the System Bus Clock that is used to program the registers and read messages via DMA or Interrupt from the module's register interface. Second is the High Frequency Receiver Clock or Protocol Clock that is used for accurate message receiving operations. There is a single asynchronous reset which is the system asynchronous reset and one internal reset for each of the channel logic.

59.5.1 Clocking strategy

Figure 1449 shows the clock domain in which each module is functioning.

Figure 1449. SENT receiver module clock domains



59.5.2 System bus clock requirements

Since all the received messages are read on the System Bus Clock it is necessary that both the interrupt service routine and the DMA controller must be able to access all channels with messages ready to be read within a time to prevent overflow in any channel. The minimum time available to the CPU or DMA is the case when the smallest Fast message is received back to back on all channels while working on the smallest receive clock tick period.

Now, the smallest Fast message size with valid checksum as per SAE specifications is 154 ticks. At 3 µs receive clock tick with a -20% variation in the transmitter clock tick, the effective receiver clock tick is 2.4 µs (after compensation). This gives us the time in which the minimum size Fast message is received in any channel as 154 $\times$ 2.4 = 369.6 µs. Now since messages are arriving in parallel, 369.6 µs is the time available for CPU or DMA to access all channels to prevent an overflow in the channel that received the very first message.

Thus, if assume that 'N_{CHNL}' channels are being used and all channels have messages ready to be read together from either DMA or CPU then the time available to read a message in one channel will be 369.6/N_{CHNL} µs or ~370/N_{CHNL} µs. Since the CPU or the DMA needs to make three 32-bit accesses in order to read a complete message and taking

the time required for one 32-bit read access as 'T_{RD}' (assuming it to be same for DMA and CPU); we can determine the minimum frequency for the System Bus Clock as:

For CPU,

Equation 77 $$3 \times T_{RD} + T_{ISR} + T_{WAKUP} < 370/N_{CHNL}$$

where T_{ISR} is the overhead in initiating an ISR and T_{WAKUP} is the time to wakeup from low power mode.

For DMA,

Equation 78 $$3 \times T_{RD} + T_{WAKUP} < 370/N_{CHNL}$$

Replacing $T_{RD} = N_{RD} \times T_{SYS_CLK}$ and $T_{ISR} = N_{ISR} \times T_{SYS_CLK}$, where N_{RD} is the number of System Bus Clock cycles in one 32-bit read access, N_{ISR} is the number of System Bus Clock Cycles in initiating an interrupt service routine and T_{SYS_CLK} is the period of the same clock, we get:

For CPU,

Equation 79 $$(3 \times N_{RD} + N_{ISR}) \times T_{SYS_CLK} + T_{WAKUP} < 370/N_{CHNL}$$

(all units is µs).

For DMA,

Equation 80 $$3 \times N_{RD} \times T_{SYS_CLK} + T_{WAKUP} < 370/N_{CHNL}$$

(all units is µs).

59.5.3 High frequency receiver clock (protocol clock) requirements

For accurate receive operations, the protocol is required to satisfy a minimum frequency defined as per the formula below.

Equation 81 $$F_{HF_CLK} > 332 / (3 \times T_{Tx_CLK})$$

where F_{HF_CLK} is the High Frequency Receiver Clock or Protocol Clock in MHz and T_{Tx_CLK} is the Transmitter or Sensor Clock Frequency in µs.

Note: T_{TX_CLK} in the above equation should be chosen based on the fastest sensor clock among all channels in the device.

60 Peripheral Sensor Interface (PSI5)

60.1 Introduction

The PSI5 interface offers power supply and data transfer between the application and multiple sensors over a two-wire interface. Data transmission from the application to the sensor or sensors is achieved through modulation of the power supply voltage (sync pulse). Data reception from the sensor or sensors to the application is achieved through modulation of the current drawn by a sensor using Manchester encoding.

At the system level, the PSI5 interface consists of mechanisms integrated on the microcontroller and an external physical layer that provides the high-voltage supply and converts the current drawn by the sensor into a digital Manchester-encoded data stream.

60.1.1 Overview

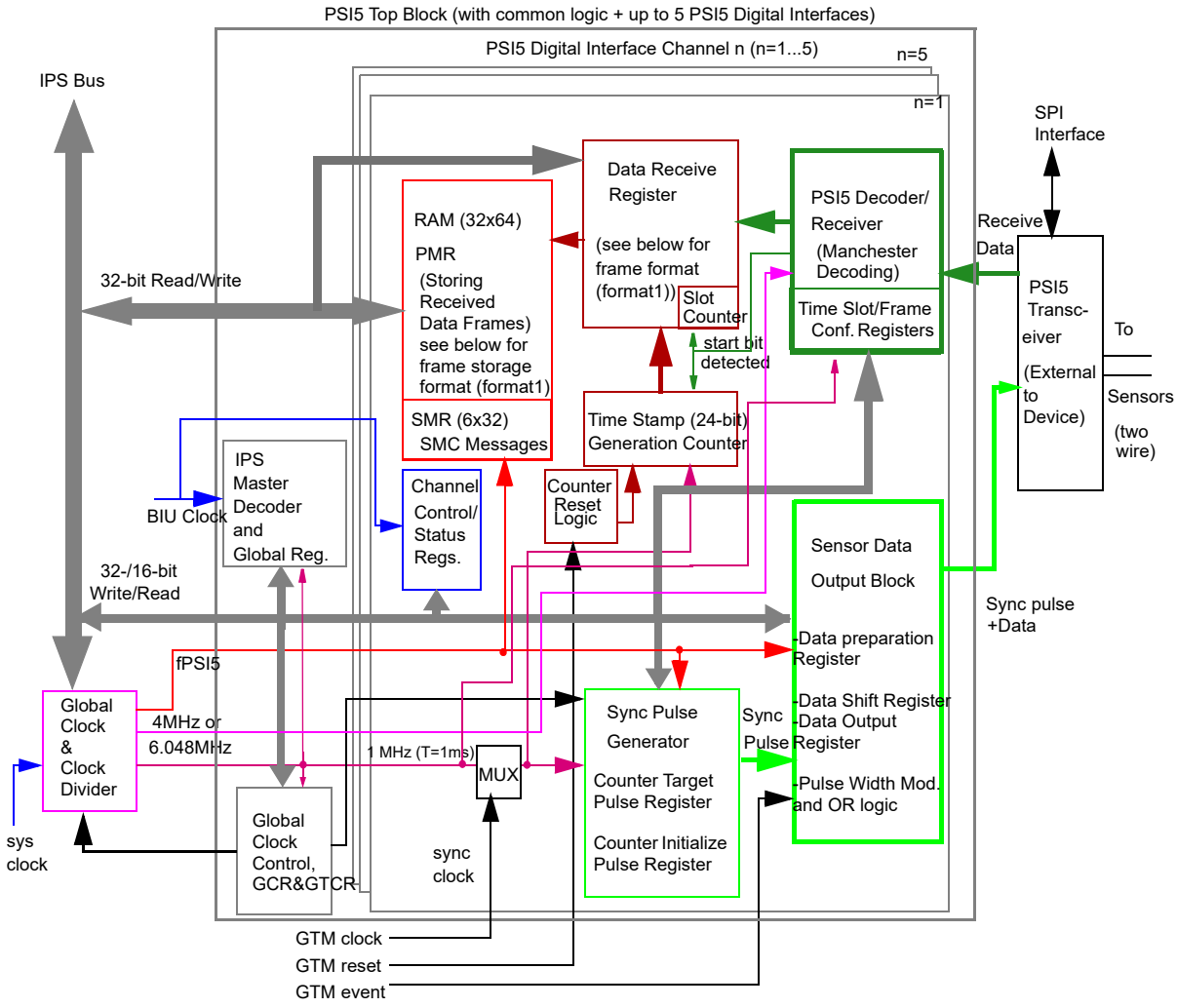
The PSI5 is an interface for automotive sensor applications. PSI5 is an open standard based on existing sensor interfaces for peripheral airbag sensors, already proven in millions of safety airbag systems. The technical characteristics, the low implementation overhead, as well as the attractive cost make the PSI5 also suitable for many other automotive sensor applications. The development goal of the PSI5 is a flexible, reliable communication standard for automotive sensor applications that can be used and implemented free of charge.

The main features of the PSI5 are high-speed and high-reliability data transfer at the lowest possible implementation overhead cost, offering a universal and flexible solution for multiple sensor applications. The main features of PSI5 standard are:

- Two-wire current interface
- Manchester-coded digital data transmission from sensor to ECU
- High data transmission speed of 125 Kbit/s or optional 189 Kbit/s
- High EMC robustness and low emission
- Wide range of sensor supply current
- Configurable data word length (8 to 28 bits)
- Asynchronous or synchronous operation
- Bidirectional communication

In short, PSI5 standardizes the low-level communication between peripheral sensors and electronic control units. Note that the physical layer for the IP (transceiver) is to be implemented separately outside the IP.

Figure 1450. PSI5 digital interface block diagram



Format1: Stored in RAM Registers

O	F	EM	E	T	Slot Counter	Time Stamp	Header+ Sensor Data(i)	CRC or Parity bits	C

1. **"C"** is set to 1 when CRC or Parity check identifies Error.
2. **"T"** is set to 1 at timing errors of the incoming frame.
3. **"E&EM"** is set to 1 at electric errors at any bit PSI5/SMC
4. **"F"** is set to 1 if no frame received in the slot.
5. **"O"** is set to 1 if any new frame overwrites previous Unread.Fr.
6. **"Slot Counter"** Gives the slot number in which the start bits of the frame were detected.
7. **"Sensor Data"** are stored in RAM automatically after all data bits are received in data receive register.
8. **"Time Stamp"** is appended to received frame in once start bits are received or rise edge of sync pulse.

60.1.2 Features

- IPS interface for register read/write (big endian)
- Supports bidirectional full duplex operations, but with differing baud rate—low, ECU-to-sensor, fast sensor to ECU
- Manchester decoder operating at both 125 Kbit/s and 189 Kbit/s bit rates with ×32 sample clock rate
- One 64-bit sensor data receive register to which the decoded received sensor data—including header, data, timestamp, frame counter, status bits (E, EM, T, C, F) and CRC, but not including start bits—are written
- A 24-bit time stamp register with a 1 µs (or the GTM clock) resolution. When first rising edge of start bits (S0) of a message or the rising edge of the SYNC pulse is recognized, the value in the time stamp register will be written to the data receive register
- Provision of resetting the Time Stamp Counter through the reset from the GTM module
- Single 32 × 64-bit RAM buffer for storing (moving) content of the data receive register once end of message is recognized
  - Related to each RAM buffer are two diagnostic registers, N and EI. Register N is used to identify the occurrence of new messages in the corresponding RAM buffer, while register EI identifies the error status of these messages. The errors that can set the EI register bits are the F (no frame), E (electrical error), EM (electrical error for SMC bits embedded in PSI5 messages), T (Timing bit error) and C (CRC error). Which of these errors actually set the EI register bits is configurable.
- DMA support for transferring RAM data (including diagnostic data) to system memory
- DMA transfer through IPS bus
- Support for optional serial messaging channel (SMC) and additional logic to extract SMC frames from PSI5 frames
- Operating modes supported:
  - Asynchronous mode (sensor periodically sends data, no sync pulse, one sensor per interface)
  - Synchronous mode (sync pulse generated, multiple sensor per interface)
- Sync pulse generation for synchronous modes:
  - The start of each time slot (1 to 6) (see *Figure 1497*) on the bus (after the end of the sync pulse) can be defined with a 1 µs resolution
  - The end of the last time slot on the bus can be defined with a 1 µs resolution, up to a maximum sync period time of 16 ms
  - Trigger logic to generate sync pulses

    16-bit Channel Trigger Counter (CTC) used to generate the sync pulses by comparing the values with CIPR and CTPR

    Counter Initialize Pulse Register (CIPR) to allow configuration of initialization (offset) period

    Channel Trigger Period (CTP) register to allow configuration of sync period. Value in CTP is the value to which the CTC is updated after each sync pulse generation

    Possibility to have a common 1 MHz clock source
- An upstream register bit chain consisting of Preparation Register, Buffer Register, Shift register (output data) and pulse width modulation logic, is implemented. Using this, it is

possible to send a data word to the sensors through use of data sync pulses: logic 1 with programmable data sync pulse width W1, logic 0 with programmable data sync pulse width W0. The data sync pulse timing can be configured to be dependent on the periodic or event driven timing SYNC pulse. Further, there is no difference between timing sync pulse and data sync pulse during transmission (ECU-to-sensor communication).

60.1.3 Modes of operation

60.1.3.1 Disable mode

Disable mode in the PSI5 reduces power consumption. This mode is entered by setting GCR[GLOBAL_DISABLE_REQ]. All the channels enter Disable mode at once when this bit is set, irrespective of the setting of PCCR[PSI5_CH_EN] and PCCR[PSI5_CH_CONFIG]. In this mode the RX, TX and common time base counters are disabled for any operation. This is the default state after module reset is released. PSI5 can exit Disable mode by clearing the GLOBAL_DISABLE_REQ bit and enabling individual PSI5 channels (setting PCCR[PSI5_CH_EN]). Note that when entering Disable mode from any other mode, the register contents are retained and are not changed. The time stamp counter AND the internal sync pulse generator are, however, reset when entering Disable mode.

There is no software reset bit in the PSI5 module. The same has to be managed outside the IP at the system level. Channels will enter “Configuration/Normal mode” based on the value of PCCR[PSI5_CH_CONFIG].

60.1.3.2 Initialization/Configuration mode

This is the configuration phase, in which all PSI5 global, channel specific configuration/control registers can be written. The receiver and transmitter sub-blocks are disabled in this mode. The message receive registers/buffers can be accessed in this mode. This mode can be entered by clearing GCR[GLOBAL_DISABLE_REQ] and setting PCCR[PSI5_CH_CONFIG] and PCCR[PSI5_CH_EN]. All the channel specific control registers can be programmed only in the CONFIG mode. This prevents the software from accidentally causing a change in any critical register.

60.1.3.3 Normal mode (NORMAL mode)

In Normal mode, the receiver and the transmitter blocks become active, error handling is enabled and all the PSI5 protocol functions are also enabled. In this mode, all the PSI5 global and channel-specific configuration registers are locked for writing, although these registers can still be read. Data Preparation Register, Shift Register and data output registers are writable in this mode. Apart from them, there are some register bits that can be written in this mode also. Refer to the detailed register description. This mode can be entered clearing GCR[GLOBAL_DISABLE_REQ], clearing PCCR[PSI5_CH_CONFIG], and setting PCCR[PSI5_CH_EN].

60.1.3.3.1 PSI5 topology

The different PSI5 user operation modes define topology and parameters of the communication between ECU and sensors such as communication mode, number of data bits, error detection, cycle time, number of time slots per cycle, and bit rate.

These modes as generally denoted as

PSI5-(A / P / U / D)-dd-(P / CRC)-tttt / n(L / H)

Table 1491 provides a key for these notations.

Table 1491. Normal operating states

Communication modes		Description
A	Asynchronous mode	PSI5-A describes a point-to-point connection for unidirectional, asynchronous data transmission. Each sensor is connected to the ECU by two wires. After switching on the power supply, the sensor starts transmitting data to the ECU periodically. Timing and repetition rate of the data transmission are controlled by the sensor. No Sync pulse is send from ECU side.
P	Synchronous Parallel Bus mode	The synchronous operation modes work according to the TDMA method (Time Division Multiple Access). Each sensor is connected to the ECU by two wires. The sensor data transmission is synchronized by the ECU using voltage modulation. Synchronization (sync pulses) can be optionally used for point-to-point configurations; it is mandatory for bus modes. The sensors can be arranged as parallel bus, universal bus and daisy chain topology based on sensor addressing and interchangeability of sensors.
U	Synchronous Universal Bus mode	
D	Synchronous Daisy Chain Bus mode	
Data Bits		
dd	No. of data bits	—
Error Detection		
P	1 parity bit	—
CRC	3-bit Cyclic Redundancy Check (CRC)	—
Cycle Time		
tttt	Cycle Time in µs	—
n	Number of time slots per cycle	—
Bit Rate		
L	125 Kbit/s	—
H	189 Kbit/s	—

60.1.3.4 Stop mode

When the Stop mode request is asserted, and if there is no active PSI5 transaction ongoing, then the acknowledgment to stop the clocks is given immediately.If start bits of any PSI5 packet have been detected (successful detection of both S0 and S1) then the concerned PSI5 packet is successfully stored and then the Stop mode acknowledgment is generated by the IP. Note that for generating the acknowledgment, the IP does not wait for the complete reception of any ongoing SMC message, but only waits for the complete reception of any ongoing PSI5 packet transaction.

If the PSI5 transceiver is asynchronous to the PSI5 IP receiver then loss of data can result when the STOP mode is deasserted. Hence the following guidelines should be followed :

- If the IP is in synchronous mode of operation, then the IP gets resynchronized on the next available sync pulse which occurs after STOP mode deassertion. Hence the PSI5 packets occurring between the STOP mode deassertion event and the subsequent synch pulse would not be reliable and should be ignored by the application.
- If the IP is configured in asynchronous mode, then it should be ensured that the idle time gap between PSI5 packets is greater than or equal to the length of one PSI5 message. In this case only one PSI5 message will get corrupted, but the IP will sample the subsequent PSI5 messages correctly.

60.1.3.5 Debug mode

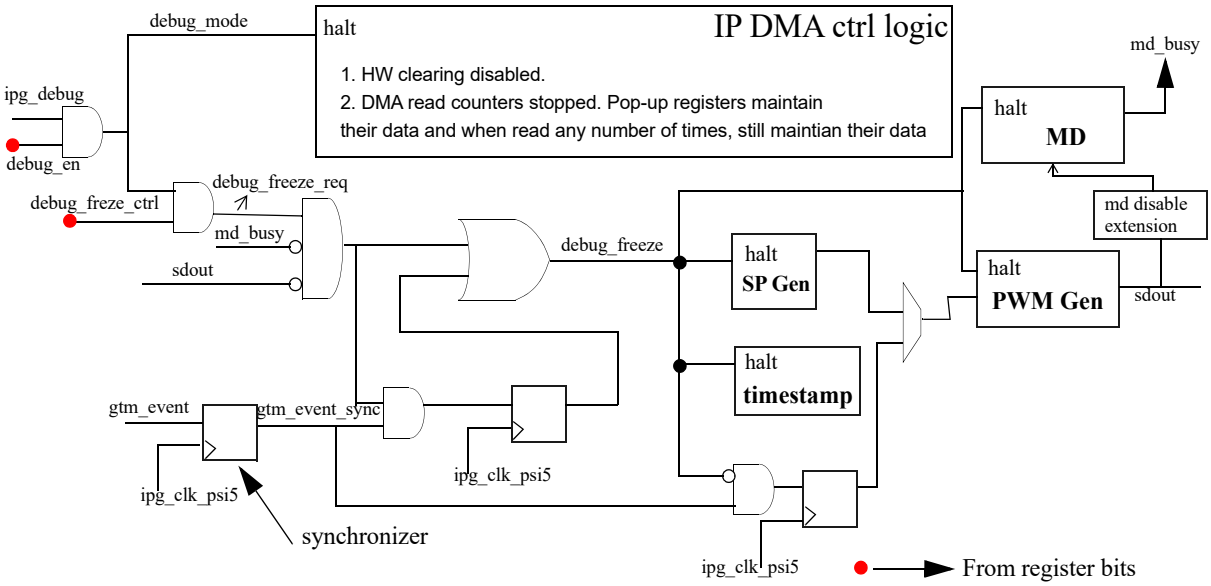
In order to enter this mode the debugger has to be connected and it has to encounter a breakpoint. In addition, the DEBUG_EN bit has to be written to 1. The DEBUG_EN bit can be written by either the CPU or by the debugger but only in Configuration mode. Once the above conditions are satisfied the IP enters the Debug Mode. There are two debugging modes available. For simplicity these are being referred to as the “Freeze” and “Open” modes. The IP enters one of the above debugging modes, based on the configuration of a register bit DEBUG_FREEZE _CTRL mentioned in the *Section 60.3.2.2: PSI5 Channel Control Register (PSI5_CH0_PCCR)*.

In “Open” mode, the core of the IP keeps on working normally and all the registers are updated from the hardware; just like they were being done in the functional mode (when the IP was out of the debug mode). Thus, even a long term entry to the Debug Mode, wont affect the normal operation of the various registers that are to be updated by the hardware. Down side is that the IP can keep on changing the internal status bits and the registers as the time progresses.

In “Freeze” mode, the core of the IP, the output and the input paths are all stopped coherently. Once the input and the output paths are stopped coherently, then all the internal status bits and registers maintain their status and value irrespective of the time progression. Coherent halting of the input path would mean that there is no active reception ongoing in the Manchester decoder. Thus, if the Manchester decoder has already detected valid start bits(S0 and S1) and then there is a request to enter the “Freeze” mode, then the current reception is completed before the IP enters the Freeze mode. Coherent halting of the output path would mean that the “sdout” output pin of the IP should be at a logic 0. Once the above conditions are satisfied, only then can the IP enter the “Freeze” mode. Once the IP enters the Freeze mode, the time stamp counter is also halted.

In both the above modes, the various interface registers would be completely visible to the debugger, but debugger action would have no impact on the various status bits and counters that normally get affected by a CPU/DMA read. In this mode, the read pointers of the FIFO do not get incremented, when being read through the Pop-up registers. Thus in Debug mode, reading the DMA PSI5 Message Register (PSI5_CH0_DPMR),DMA SMC Frame Register (PSI5_CH0_DSFR) or the DMA Diagnostic Status Register (PSI5_CH0_DDSR) would keep on returning the same data. Hence the interrupts related to DMA completion operation would not get generated. Further the hardware clearing related to FAST_CLR_SMC and FAST_CLR_PSI5 bits would not have any effect and the hardware clearing would remain disabled. This is also highlighted in *Figure 1451*.

Figure 1451. Block Diagram of Debug Mode



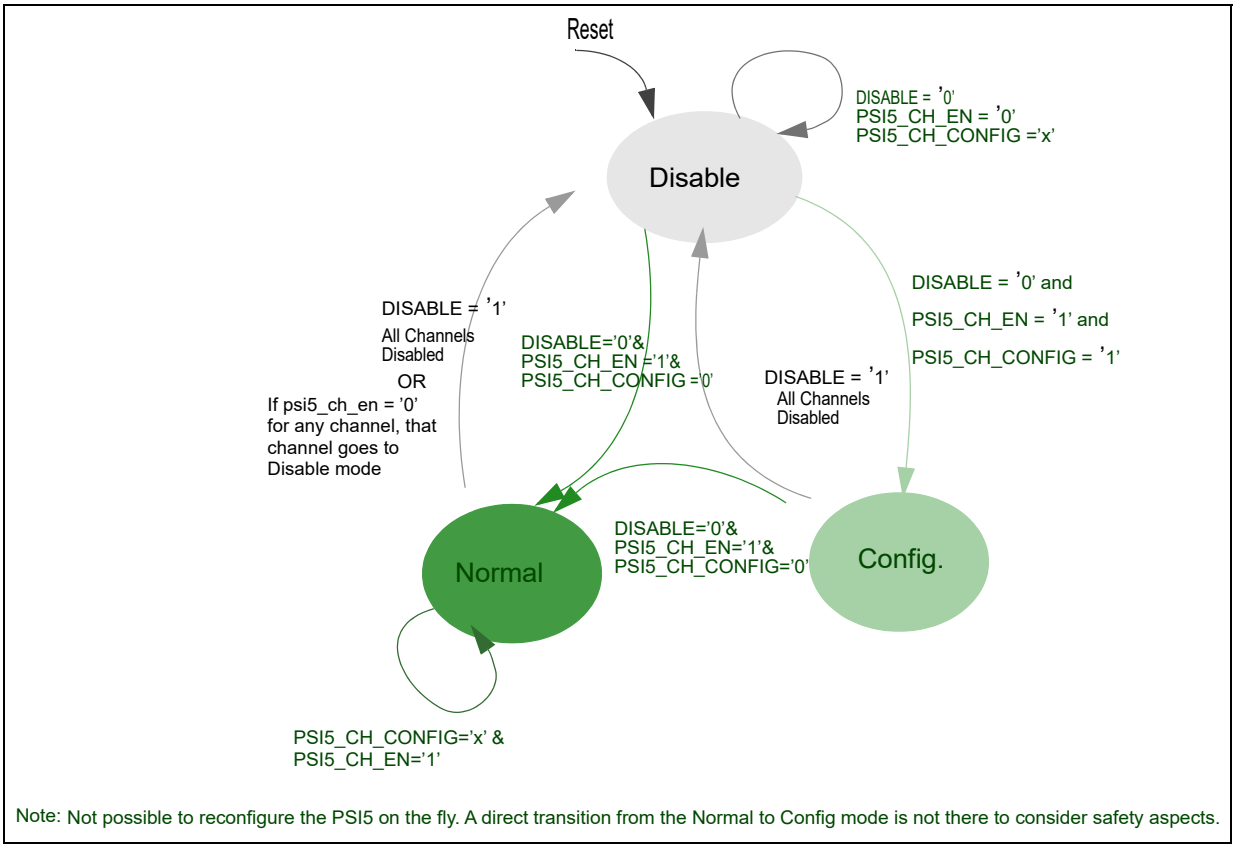
1. “halt” in the above block means, that when “1”, then the specified block will stall its operation.
2. When “debug_mode” == 1, then the IP enters debug mode.
3. When “md_busy” is “1”, it means that MD is busy in reception of some data.
4. “ipg_debug” when “1” is an indication from the Debugger Logic that a breakpoint has been encountered and IP has to stop its operation.
5. Once the “debug_freeze_req” is asserted, then the “debug_freeze” is asserted ONLY when no pending reception is there in the MD and the “sdout” has a low level. Once the “debug_freeze” == 1, then the blocks are halted as shown above.
6. During deassertion of the “debug_freeze_req”, if the “gtm_event_sync” is already high, then the “debug_freeze” is not de-asserted until the “gtm_event” goes low. This is to prevent an “untimed” sync pulse from going to the output path.
7. “debug_en” bit allows selective disabling of the debug mode of the IP.

The debugger can write to all the registers just like the CPU can. However, no special privilege is given to the debugger as compared to the CPU. The read/write behavior of the debugger for various registers in Normal/Configuration mode is identical to the behavior of the CPU. Note that the various status bit and counter protections become active ONLY in Debug mode.

60.1.3.6 Operating modes

Figure 1452 shows the operating modes and *Table 1492* shows the state transition.

Figure 1452. Operating modes



Note: Not possible to reconfigure the PSI5 on the fly. A direct transition from the Normal to Config mode is not there to consider safety aspects.

Table 1492. State transition table

GCR [GLOBAL_DISABLE_REQ]	PCCR		State	
	[PSI5_CH_CONFIG]	[PSI5_CH_EN]	Present	Next
1	x	x	x	All channel Disable
0	x	0	x	Specific channel Disable
0	1	1	Disable	Config
0	0	1	Config	Normal
0	x	1	Normal	Normal
0	0	1	Disable	Normal

60.2 External signal description

Table 1493. PSI5 port list

Name	Function	I/O
Signals to/from pads		
SDINn_PSI5_m(1)	Digital sensor data input pin of PSI5 module	I
SDOUTn_PSI5_m (1)	Digital sensor data output pin of PSI5 module (sync pulse + data)	O

1. “n” is the Channel Number and “m” is the Module Instance Number.

60.3 Memory map and register description

60.3.1 Memory map

Table 1494 shows the memory map.

Table 1494. PSI5 memory map(1)

Address offset (hex)	Register	Access(2)			Reset value(3) (hex)	Location (section)
		D	C	N		
0x0000	Global Control Register (PSI5_GCR(4))	R/W	R/W	R/W	— (5)	*Section 60.3.2.1*
0x0002–0x0007	Reserved					
PSI5 Channel 0						
0x0008	PSI5 Channel Control Register (PSI5_CH0_PCCR)	R/W(6)	R/W	R/W(6)	0x001F_0000	*Section 60.3.2.2*
0x000C	DMA Control Register (PSI5_CH0_DCR)	R	R/W	R/W(6)	0x0000_0000	*Section 60.3.2.3*
0x0010	DMA Status Register (PSI5_CH0_DSR)	R	R/W1c	R/W1c	0x0000_0000	*Section 60.3.2.4*
0x0014	General Interrupt Control Register (PSI5_CH0_GICR)	R	R/W	R/W	0x0000_0000	*Section 60.3.2.5*
0x0018	New Data Interrupt Control Register (PSI5_CH0_NDICR)	R	R/W	R/W	0x0000_0000	*Section 60.3.2.6*
0x001C	Overwrite Interrupt Control Register (PSI5_CH0_OWICR)	R	R/W	R/W	0x0000_0000	*Section 60.3.2.7*

Table 1494. PSI5 memory map[1] (continued)

Address offset (hex)	Register	Access[2]			Reset value[3] (hex)	Location (section)
		D	C	N		
0x0020	Error Interrupt Control Register (PSI5_CH0_EICR)	R	R/W	R/W	0x0000_0000	*Section 60.3.2.8*
0x0024	General Interrupt Status Register (PSI5_CH0_GISR)	R	R/W1c/ HC[7]	R/W1c/ HC[7]	0x0007_0000	*Section 60.3.2.9*
0x0028	DMA PSI5 Message Register (PSI5_CH0_DPMR)	R	R	R	0x0000_0000	*Section 60.3.2.10*
0x002C	DMA SMC Frame Register (PSI5_CH0_DSFR)	R	R	R	0x0000_0000	*Section 60.3.2.11*
0x0030	DMA Diagnostic Status Register (PSI5_CH0_DDSR)	R	R	R	0x0000_0000	*Section 60.3.2.12*
0x0034	PSI5 Message Receive Register Low (PSI5_CH0_PMRRL)	R	R	R	0x0000_0000	*Section 60.3.2.13*
0x0038	PSI5 Message Receive Register High (PSI5_CH0_PMRRH)	R	R	R	0x0000_0000	*Section 60.3.2.14*
0x003C	PSI5 Message Register 0 Low Word of Data Frame (PSI5_CH0_PMRL0)	R	R/W[8]	R	0x0000_0000	*Section 60.3.2.15*
0x0040	PSI5 Message Register 0 High Word of Data Frame (PSI5_CH0_PMRH0)	R	R/W[8]	R	0x0000_0000	*Section 60.3.2.16*
0x0044	PSI5 Message Register 1 Low Word of Data Frame (PSI5_CH0_PMRL1)	R	R/W[8]	R	0x0000_0000	*Section 60.3.2.15*
0x0048	PSI5 Message Register 1 High Word of Data Frame (PSI5_CH0_PMRH1)	R	R/W[8]	R	0x0000_0000	*Section 60.3.2.16*
0x004C	PSI5 Message Register 2 Low Word of Data Frame (PSI5_CH0_PMRL2)	R	R/W[8]	R	0x0000_0000	*Section 60.3.2.15*
0x0050	PSI5 Message Register 2 High Word of Data Frame (PSI5_CH0_PMRH2)	R	R/W[8]	R	0x0000_0000	*Section 60.3.2.16*
0x0054	PSI5 Message Register 3 Low Word of Data Frame (PSI5_CH0_PMRL3)	R	R/W[8]	R	0x0000_0000	*Section 60.3.2.15*

Table 1494. PSI5 memory map[1] (continued)

Address offset (hex)	Register	Access[2]			Reset value[3] (hex)	Location (section)
		D	C	N		
0x0058	PSI5 Message Register 3High Word of Data Frame (PSI5_CH0_PMRH3)	R	R/W[8]	R	0x0000_0000	*Section 60.3.2.16*
0x005C	PSI5 Message Register 4 Low Word of Data Frame (PSI5_CH0_PMRL4)	R	R/W[8]	R	0x0000_0000	*Section 60.3.2.15*
0x0060	PSI5 Message Register 4 High Word of Data Frame (PSI5_CH0_PMRH4)	R	R/W[8]	R	0x0000_0000	*Section 60.3.2.16*
0x0064	PSI5 Message Register 5 Low Word of Data Frame (PSI5_CH0_PMRL5)	R	R/W[8]	R	0x0000_0000	*Section 60.3.2.15*
0x0068	PSI5 Message Register 5 High Word of Data Frame (PSI5_CH0_PMRH5)	R	R/W[8]	R	0x0000_0000	*Section 60.3.2.16*
0x006C	PSI5 Message Register 6 Low Word of Data Frame (PSI5_CH0_PMRL6)	R	R/W[8]	R	0x0000_0000	*Section 60.3.2.15*
0x0070	PSI5 Message Register 6 High Word of Data Frame (PSI5_CH0_PMRH6)	R	R/W[8]	R	0x0000_0000	*Section 60.3.2.16*
0x0074	PSI5 Message Register 7 Low Word of Data Frame (PSI5_CH0_PMRL7)	R	R/W[8]	R	0x0000_0000	*Section 60.3.2.15*
0x0078	PSI5 Message Register 7 High Word of Data Frame (PSI5_CH0_PMRH7)	R	R/W[8]	R	0x0000_0000	*Section 60.3.2.16*
0x007C	PSI5 Message Register 8 Low Word of Data Frame (PSI5_CH0_PMRL8)	R	R/W[8]	R	0x0000_0000	*Section 60.3.2.15*
0x0080	PSI5 Message Register 8 High Word of Data Frame (PSI5_CH0_PMRH8)	R	R/W[8]	R	0x0000_0000	*Section 60.3.2.16*
0x0084	PSI5 Message Register 9 Low Word of Data Frame (PSI5_CH0_PMRL9)	R	R/W[8]	R	0x0000_0000	*Section 60.3.2.15*

Table 1494. PSI5 memory map[1] (continued)

Address offset (hex)	Register	Access[2]			Reset value[3] (hex)	Location (section)
		D	C	N		
0x0088	PSI5 Message Register 9 High Word of Data Frame (PSI5_CH0_PMRH9)	R	R/W[8]	R	0x0000_0000	*Section 60.3.2.16*
0x008C	PSI5 Message Register 10 Low Word of Data Frame (PSI5_CH0_PMRL10)	R	R/W[8]	R	0x0000_0000	*Section 60.3.2.15*
0x0090	PSI5 Message Register 10 High Word of Data Frame (PSI5_CH0_PMRH10)	R	R/W[8]	R	0x0000_0000	*Section 60.3.2.16*
0x0094	PSI5 Message Register 11 Low Word of Data Frame (PSI5_CH0_PMRL11)	R	R/W[8]	R	0x0000_0000	*Section 60.3.2.15*
0x0098	PSI5 Message Register 11 High Word of Data Frame (PSI5_CH0_PMRH11)	R	R/W[8]	R	0x0000_0000	*Section 60.3.2.16*
0x009C	PSI5 Message Register 12 Low Word of Data Frame (PSI5_CH0_PMRL12)	R	R/W[8]	R	0x0000_0000	*Section 60.3.2.15*
0x00A0	PSI5 Message Register 12 High Word of Data Frame (PSI5_CH0_PMRH12)	R	R/W[8]	R	0x0000_0000	*Section 60.3.2.16*
0x00A4	PSI5 Message Register 13 Low Word of Data Frame (PSI5_CH0_PMRL13)	R	R/W[8]	R	0x0000_0000	*Section 60.3.2.15*
0x00A8	PSI5 Message Register 13 High Word of Data Frame (PSI5_CH0_PMRH13)	R	R/W[8]	R	0x0000_0000	*Section 60.3.2.16*
0x00AC	PSI5 Message Register 14 Low Word of Data Frame (PSI5_CH0_PMRL14)	R	R/W[8]	R	0x0000_0000	*Section 60.3.2.15*
0x00B0	PSI5 Message Register 14 High Word of Data Frame (PSI5_CH0_PMRH14)	R	R/W[8]	R	0x0000_0000	*Section 60.3.2.16*
0x00B4	PSI5 Message Register 15 Low Word of Data Frame (PSI5_CH0_PMRL15)	R	R/W[8]	R	0x0000_0000	*Section 60.3.2.15*

Table 1494. PSI5 memory map[1] (continued)

Address offset (hex)	Register	Access[2]			Reset value[3] (hex)	Location (section)
		D	C	N		
0x00B8	PSI5 Message Register 15 High Word of Data Frame (PSI5_CH0_PMRH15)	R	R/W[8]	R	0x0000_0000	*Section 60.3.2.16*
0x00BC	PSI5 Message Register 16 Low Word of Data Frame (PSI5_CH0_PMRL16)	R	R/W[8]	R	0x0000_0000	*Section 60.3.2.15*
0x00C0	PSI5 Message Register 16 High Word of Data Frame (PSI5_CH0_PMRH16)	R	R/W[8]	R	0x0000_0000	*Section 60.3.2.16*
0x00C4	PSI5 Message Register 17 Low Word of Data Frame (PSI5_CH0_PMRL17)	R	R/W[8]	R	0x0000_0000	*Section 60.3.2.15*
0x00C8	PSI5 Message Register 17 High Word of Data Frame (PSI5_CH0_PMRH17)	R	R/W[8]	R	0x0000_0000	*Section 60.3.2.16*
0x00CC	PSI5 Message Register 18 Low Word of Data Frame (PSI5_CH0_PMRL18)	R	R/W[8]	R	0x0000_0000	*Section 60.3.2.15*
0x00D0	PSI5 Message Register 18 High Word of Data Frame (PSI5_CH0_PMRH18)	R	R/W[8]	R	0x0000_0000	*Section 60.3.2.16*
0x00D4	PSI5 Message Register 19 Low Word of Data Frame (PSI5_CH0_PMRL19)	R	R/W[8]	R	0x0000_0000	*Section 60.3.2.15*
0x00D8	PSI5 Message Register 19 High Word of Data Frame (PSI5_CH0_PMRH19)	R	R/W[8]	R	0x0000_0000	*Section 60.3.2.16*
0x00DC	PSI5 Message Register 20 Low Word of Data Frame (PSI5_CH0_PMRL20)	R	R/W[8]	R	0x0000_0000	*Section 60.3.2.15*
0x00E0	PSI5 Message Register 20 High Word of Data Frame (PSI5_CH0_PMRH20)	R	R/W[8]	R	0x0000_0000	*Section 60.3.2.16*
0x00E4	PSI5 Message Register 21 Low Word of Data Frame (PSI5_CH0_PMRL21)	R	R/W[8]	R	0x0000_0000	*Section 60.3.2.15*

Table 1494. PSI5 memory map[1] (continued)

Address offset (hex)	Register	Access[2]			Reset value[3] (hex)	Location (section)
		D	C	N		
0x00E8	PSI5 Message Register 21 High Word of Data Frame (PSI5_CH0_PMRH21)	R	R/W[8]	R	0x0000_0000	*Section 60.3.2.16*
0x00EC	PSI5 Message Register 22 Low Word of Data Frame (PSI5_CH0_PMRL22)	R	R/W[8]	R	0x0000_0000	*Section 60.3.2.15*
0x00F0	PSI5 Message Register 22 High Word of Data Frame (PSI5_CH0_PMRH22)	R	R/W[8]	R	0x0000_0000	*Section 60.3.2.16*
0x00F4	PSI5 Message Register 23 Low Word of Data Frame (PSI5_CH0_PMRL23)	R	R/W[8]	R	0x0000_0000	*Section 60.3.2.15*
0x00F8	PSI5 Message Register 23 High Word of Data Frame (PSI5_CH0_PMRH23)	R	R/W[8]	R	0x0000_0000	*Section 60.3.2.16*
0x00FC	PSI5 Message Register 24 Low Word of Data Frame (PSI5_CH0_PMRL24)	R	R/W[8]	R	0x0000_0000	*Section 60.3.2.15*
0x0100	PSI5 Message Register 24 High Word of Data Frame (PSI5_CH0_PMRH24)	R	R/W[8]	R	0x0000_0000	*Section 60.3.2.16*
0x0104	PSI5 Message Register 25 Low Word of Data Frame (PSI5_CH0_PMRL25)	R	R/W[8]	R	0x0000_0000	*Section 60.3.2.15*
0x0108	PSI5 Message Register 25 High Word of Data Frame (PSI5_CH0_PMRH25)	R	R/W[8]	R	0x0000_0000	*Section 60.3.2.16*
0x010C	PSI5 Message Register 26 Low Word of Data Frame (PSI5_CH0_PMRL26)	R	R/W[8]	R	0x0000_0000	*Section 60.3.2.15*
0x0110	PSI5 Message Register 26 High Word of Data Frame (PSI5_CH0_PMRH26)	R	R/W[8]	R	0x0000_0000	*Section 60.3.2.16*
0x0114	PSI5 Message Register 27 Low Word of Data Frame (PSI5_CH0_PMRL27)	R	R/W[8]	R	0x0000_0000	*Section 60.3.2.15*

Table 1494. PSI5 memory map[1] (continued)

Address offset (hex)	Register	Access[2]			Reset value[3] (hex)	Location (section)
		D	C	N		
0x0118	PSI5 Message Register 27 High Word of Data Frame (PSI5_CH0_PMRH27)	R	R/W[8]	R	0x0000_0000	*Section 60.3.2.16*
0x011C	PSI5 Message Register 28 Low Word of Data Frame (PSI5_CH0_PMRL28)	R	R/W[8]	R	0x0000_0000	*Section 60.3.2.15*
0x0120	PSI5 Message Register 28 High Word of Data Frame (PSI5_CH0_PMRH28)	R	R/W[8]	R	0x0000_0000	*Section 60.3.2.16*
0x0124	PSI5 Message Register 29 Low Word of Data Frame (PSI5_CH0_PMRL29)	R	R/W[8]	R	0x0000_0000	*Section 60.3.2.15*
0x0128	PSI5 Message Register 29 High Word of Data Frame (PSI5_CH0_PMRH29)	R	R/W[8]	R	0x0000_0000	*Section 60.3.2.16*
0x012C	PSI5 Message Register 30 Low Word of Data Frame (PSI5_CH0_PMRL30)	R	R/W[8]	R	0x0000_0000	*Section 60.3.2.15*
0x0130	PSI5 Message Register 30 High Word of Data Frame (PSI5_CH0_PMRH30)	R	R/W[8]	R	0x0000_0000	*Section 60.3.2.16*
0x0134	PSI5 Message Register 31 Low Word of Data Frame (PSI5_CH0_PMRL31)	R	R/W[8]	R	0x0000_0000	*Section 60.3.2.15*
0x0138	PSI5 Message Register 31 High Word of Data Frame (PSI5_CH0_PMRH31)	R	R/W[8]	R	0x0000_0000	*Section 60.3.2.16*
0x013C	SMC Frame RX Register 1 (PSI5_CH0_SFR1)	R	R/W[8]	R	0x0000_0000	*Section 60.3.2.17*
0x0140	SMC Frame RX Register 2 (PSI5_CH0_SFR2)	R	R/W[8]	R	0x0000_0000	*Section 60.3.2.17*
0x0144	SMC Frame RX Register 3 (PSI5_CH0_SFR3)	R	R/W[8]	R	0x0000_0000	*Section 60.3.2.17*
0x0148	SMC Frame RX Register 4 (PSI5_CH0_SFR4)	R	R/W[8]	R	0x0000_0000	*Section 60.3.2.17*
0x014C	SMC Frame RX Register 5 (PSI5_CH0_SFR5)	R	R/W[8]	R	0x0000_0000	*Section 60.3.2.17*

Table 1494. PSI5 memory map[1] (continued)

Address offset (hex)	Register	Access[2]			Reset value[3] (hex)	Location (section)
		D	C	N		
0x0150	SMC Frame RX Register 6 (PSI5_CH0_SFR6)	R	R/W[8]	R	0x0000_0000	*Section 60.3.2.17*
0x0154	New Data Status Register (PSI5_CH0_NDSR)	R	R/W1C/ HC[9]	R/W1C/ HC[9]	0x0000_0000	*Section 60.3.2.18*
0x0158	Over Write Status Register (PSI5_CH0_OWSR)	R	R/W1C/ HC[9]	R/W1C/ HC[9]	0x0000_0000	*Section 60.3.2.19*
0x015C	Error Status Register (PSI5_CH0_EISR)	R	R/W1C/ HC[9]	R/W1C/ HC[9]	0x0000_0000	*Section 60.3.2.20*
0x0160	Set New Data Status Register (PSI5_CH0_SNDSR)	R	R/W1S	R/W1S	0x0000_0000	*Section 60.3.2.21*
0x0164	Set Overwrite Status Register (PSI5_CH0_SOWSR)	R	R/W1S	R/W1S	0x0000_0000	*Section 60.3.2.22*
0x0168	Set Error Status Register (PSI5_CH0_SEISR)	R	R/W1S	R/W1S	0x0000_0000	*Section 60.3.2.23*
0x016C	Set SMC Error Status Register (PSI5_CH0_SSESR)	R	R/W1S	R/W1S	0x0000_0000	*Section 60.3.2.24*
0x0170	Sync Pulse Start Time Stamp Read Register (PSI5_CH0_STSRR)	R	R	R	0x0000_0000	*Section 60.3.2.25*
0x0174	Data Reception Time Stamp Read Register (PSI5_CH0_DTSRR)	R	R	R	0x0000_0000	*Section 60.3.2.26*
0x0178	Slot 1 Frame Configuration Register (PSI5_CH0_S1FCR)	R	R/W	R	0x0000_0010	*Section 60.3.2.27*
0x017C	Slot 2 Frame Configuration Register (PSI5_CH0_S2FCR)	R	R/W	R	0x0000_0010	*Section 60.3.2.27*
0x0180	Slot 3 Frame Configuration Register (PSI5_CH0_S3FCR)	R	R/W	R	0x0000_0010	*Section 60.3.2.27*
0x0184	Slot 4 Frame Configuration Register (PSI5_CH0_S4FCR)	R	R/W	R	0x0000_0010	*Section 60.3.2.27*
0x0188	Slot 5 Frame Configuration Register (PSI5_CH0_S5FCR)	R	R/W	R	0x0000_0010	*Section 60.3.2.27*
0x018C	Slot 6 Frame Configuration Register (PSI5_CH0_S6FCR)	R	R/W	R	0x0000_0010	*Section 60.3.2.27*
0x0190	Slot 1 Start Boundary Register (PSI5_CH0_S1SBR)	R	R/W	R	0x0000	*Section 60.3.2.28*

Table 1494. PSI5 memory map[1] (continued)

Address offset (hex)	Register	Access[2]			Reset value[3] (hex)	Location (section)
		D	C	N		
0x0192	Slot 2 Start Boundary Register (PSI5_CH0_S2SBR)	R	R/W	R	0x0000	*Section 60.3.2.28*
0x0194	Slot 3 Start Boundary Register (PSI5_CH0_S3SBR)	R	R/W	R	0x0000	*Section 60.3.2.28*
0x0196	Slot 4 Start Boundary Register (PSI5_CH0_S4SBR)	R	R/W	R	0x0000	*Section 60.3.2.28*
0x0198	Slot 5 Start Boundary Register (PSI5_CH0_S5SBR)	R	R/W	R	0x0000	*Section 60.3.2.28*
0x019A	Slot 6 Start Boundary Register (PSI5_CH0_S6SBR)	R	R/W	R	0x0000	*Section 60.3.2.28*
0x019C	Slot *n* End Boundary Register (PSI5_CH0_SNEBR)	R	R/W	R	0x0000_0000	*Section 60.3.2.29*
0x01A0	Data Output Block Configuration Register (PSI5_CH0_DOBCR)	R	R/W/ W1S[10]	R/ W1S[10]	0x0000	*Section 60.3.2.30*
0x01A2	Manchestor Decoder Disable Offset (PSI5_CH0_MDDIS_OFF)	R	R/W	R	0x0000	*Section 60.3.2.31*
0x01A4	Pulse Width for Data Bit Value 0 (PSI5_CH0_PW0D)	R	R/W	R	0x0000	*Section 60.3.2.32*
0x01A6	Pulse Width for Data Bit Value 1 (PSI5_CH0_PW1D)	R	R/W	R	0x0000	*Section 60.3.2.33*
0x01A8	Counter target Pulse Register (PSI5_CH0_CTPR)	R	R/W	R	0x0000	*Section 60.3.2.34*
0x01AA	Counter Initialize Pulse Register (PSI5_CH0_CIPR)	R	R/W	R	0x0000	*Section 60.3.2.35*
0x01AC	Data Preparation Register Low Word (PSI5_CH0_DPRL)	R	R	R/W[11]	0x0000_0000	*Section 60.3.2.36*
0x01B0	Data Preparation Register High Word (PSI5_CH0_DPRH)	R	R	R	0x0000_0000	*Section 60.3.2.37*
0x01B4	Data Buffer Register Low Word (PSI5_CH0_DBRL)	R	R	R/W[11]	0x0000_0000	*Section 60.3.2.38*
0x01B8	Data Buffer Register (High Word (PSI5_CH0_DBRH)	R	R	R/W[11]	0x0000_0000	*Section 60.3.2.39*
0x01BC	Data Shift Register Low Word (PSI5_CH0_DSRL)	R	R	R/W[11]	0x0000_0000	*Section 60.3.2.40*

Table 1494. PSI5 memory map[1] (continued)

Address offset (hex)	Register	Access[2]			Reset value[3] (hex)	Location (section)
		D	C	N		
0x01C0	Data Shift Register High Word (PSI5_CH0_DSRH)	R	R	R/W[11]	0x0000_0000	*Section 60.3.2.41*
__[12]	Reserved					

1. Refer to the Device Configuration Chapter for instance specific PSI5 channels.
2. In this column, R/W = Read/Write, R = Read-only, W = Write-only, W1C = write-1-to-clear and W1S = “write-to-1-shot, always read as 0,” D = Disable mode, C = Config mode, N = Normal mode. All 32-bit registers are word/half word/byte addressable for write. Read is always word addressable. All 16-bit registers are half word/byte addressable for write. Read is always word addressable. Trying to access any register space of PSI5, other than those mentioned in the memory map, results in the generation of transfer error. Note that the generation of transfer error is confined to this scenario only.
3. In this column, the format of the reset value indicates the register width. Thus, reset values of the formats 0xHH, 0xHHHH, and 0xHHHH_HHHH, where “H” is a hexadecimal digit, indicate 8-bit, 16-bit, and 32-bit registers, respectively.
4. This register is not present if the specific instance is configured as a slave . Master or slave configuration of the specific PSI5 instance decides if the operating mode (DISABLE/CONFIG/NORMAL) as well as the staggering of the channels can be controlled independently. If the instance is configured as a master then it controls its own operating mode, staggering of all the channels in that instance and the operating mode and the staggering of any associated slave instance. If the instance is configured as a slave then its operating mode and staggering can only be controlled by another master instance. Refer to the Device Configuration Chapter for instance specific details.
5. The reset value is dependent on the parameter PSI5_PDIS_RST:
When PSI5_PDIS_RST = 0 then the reset value is “0x0000”(PSI5 module is enabled after reset).
When PSI5_PDIS_RST = 1 then the reset value is “0x0001” (PSI5 module is disabled after reset).
For the value of parameter PSI5_PDIS_RST refer to the Device Configuration File.
6. Not all the register bits are writable in this mode. Refer to the Register Description section for details.
7. The bits can be cleared by hardware or by software using w1c. This is controlled by the “fast_clr_xxx” bit in the PSI5_PCCR. However not all bits of this register have this behavior. Refer to the Register Description section for details.
8. Writable ONLY in Config mode for test purposes. Otherwise updated by hardware.
9. The bits are set by hardware. Alternatively the software can also set these bits by writing to the corresponding SET registers. This provision should be used only for test purpose. Clearing of these bits is either by w1c or by hardware depending on the setting of the “fast_clr_xxx” bit in the PSI5_PCCR.
10. Few bits are writable to One Shot. Refer to the Register Description section for details.
11. The bits of this register are writable ONLY when the corresponding DxR_RDY bit in the PSI5_GISR is equal to 1.
12. Refer to the Device Configuration Chapter for instance specific PSI5 reserved locations.

60.3.2 Register description

Unless specifically stated, the register bits are readable in all modes, but writable only in Config mode.

Note that, an access (read/write) to any register in the unimplemented register space will result in the generation of transfer error. All word locations are byte/half word/word accessible for write. Read is always a word access. Trying to do a word operation on 16-bit registers will NOT result in the generation of any transfer error. Trying to write to “read only” bits of implemented register space, will be ignored without the generation of any error. Trying to write to any register outside its intended mode of write (Normal/config) will be ignored without the generation of any transfer error.

For a summary of the register bit accesses in various device modes, refer to *Section 60.3.2.42: Device modes and Register bit accesses*

60.3.2.1 Global Control Register (PSI5_GCR)

The Global Control Register (PSI5_GCR) provides control functions for all the PSI5 channels simultaneously.

Address: 0x0000 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	CTC_GED	GLOBAL_DISABLE_REQ
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	—(1)

Figure 1453. Global Control Register (PSI5_GCR)

1. The reset value is dependent on the parameter PSI5_PDIS_RST. When PSI5_PDIS_RST = 0 then the reset value is “0” (PSI5 module is enabled after reset). When PSI5_PDIS_RST = 1 then the reset value is “1” (PSI5 module is disabled after reset). For the value of parameter PSI5_PDIS_RST refer to the Device Configuration Chapter.

Table 1495. PSI5_GCR(1) field descriptions

Field	Description
14 CTC_GED	Channel Target Counter (CTC) Global Enable/Disable. 0 All the CTC counters having PCCR[CTC_GED_SEL] = 1 are reset. 1 Simultaneously starts the CTC of all the channels that have PCCR[CTC_GED_SEL] = 1. **Note:** This bit is writable in all modes.
15 GLOBAL_DISABLE_REQ	PSI5 Receive Global disable request. When set, all PSI5 channels enter Disable mode (from individual Normal or Config modes) regardless of the settings of the PCCR[PSI5_CH_CONFIG] and PCCR[PSI5_CH_EN] bits. When cleared, each PSI5 channel enters Config mode or Normal mode based on the setting of the PCCR[PSI5_CH_CONFIG] bit if PCCR[PSI5_CH_CONFIG] is set for that channel; otherwise, that particular channel continues to stay in Disable mode. The reset value is dependent on the parameter PSI5_PDIS_RST. When PSI5_PDIS_RST = 0 then the reset value is “0”(PSI5 module is enabled after reset). When PSI5_PDIS_RST = 1then the reset value is “1”(PSI5 module is disabled after reset). For the value of parameter PSI5_PDIS_RST refer to the Device Configuration Chapter. This bit is writable in all the modes. 0 PSI5 channel(s) enters Normal or Config mode, depending on the value of the PCCR[PSI5_CH_CONFIG] bit. 1 All PSI5 Channels are in Disable mode. **Note:** When GLOBAL_DISABLE_REQ == 1 then all the clock enables (shown in *Figure 1518*) from the IP go to a 0 (inactive) state.

1. This register is not present if the specific instance is configured as a slave. Master or slave configuration of the specific PSI5 instance decides if the operating mode (DISABLE/CONFIG/NORMAL) as well as the staggering of the channels can be controlled independently. If the instance is configured as a master then it controls its own operating mode, staggering of all the channels in that instance and the operating mode and the staggering of any associated slave instance. If the instance is configured as a slave then its operating mode and staggering can only be controlled by another master instance. Refer to the Device Configuration Chapter for instance specific details.

60.3.2.2 PSI5 Channel Control Register (PSI5_CH0_PCCR)

Figure 1454 shows the PSI5 Channel Control Register.

Address: 0x0008 + *i**0x1C0 (*i*=0) Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	CTC_GED_SEL	CTC_ED	0	MEM_DEPTH					0	0	0	ERROR_SELECT4_	ERROR_SELECT3_	ERROR_SELECT2_	ERROR_SELECT1_	ERROR_SELECT0_
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0(1)	GTM_RESET_ASYNC_EN	0	0	0	DEBUG_EN	DEBUG_FREEZE_CTRL	SP_TS_CLK_SEL	0	0	FAST_CLR_SMC	FAST_CLR_PSI5	BIT_RATE	MODE	PSI5_CH_CONFIG	PSI5_CH_EN
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1454. PSI5 Channel Control Register (PSI5_CH0_PCCR)

1. This is a redundant bit and RW will not have any effect.

Table 1496. PSI5_CH0_PCCR field descriptions

Field	Description
0 CTC_GED_SEL	Channel Target Counter, Global Enable/Disable Select This bit selects whether the CTC should be enabled/disabled by CTC_GED bit (in GCR) or the CTC_ED bit. Enabling by the CTC_GED bit is used when staggering of the channels is required. 0 CTC enabled disabled by CTC_ED. 1 CTC enabled/disabled by CTC_GED.
1 CTC_ED	Channel Target Counter Enable/Disable This bit is used to control the CTC locally from the channel in case the CTC_GED_SEL = 0. 0 The CTC counters is disabled and reset. 1 The CTC counter is enabled and start counting. **Note:** This bit is writable in Normal mode and the Config mode.

Table 1496. PSI5_CH0_PCCR field descriptions (continued)

Field	Description
3:7 MEM_DEPTH	Can be programmed from 0 to 31 and denotes the size of the memory that should be used for storing the PSI5 messages. Area above the MEM_DEPTH is treated as being unavailable for message storage. MEM_DEPTH also denotes the number of RAM registers available for the CPU access. Note that the same memory can be accessed sequentially (though the DMA) or randomly anytime by the CPU. 0x00 Indicates that only 1 location of the memory is available for storage. 0x1F (31) Indicates that all the 32 locations of the memory are available for storage. **Note:** Depending on the configuration of the DMA_PM_DS_CONFIG bits in the DCR, the MEM_DEPTH memory size may be accessed by CPU only (interrupts are generated) or CPU and DMA (through the dedicated DMA popup registers). DMA_PM_DS_CONFIG = conf1 then DMA requests are disabled and only the interrupts can be used for data transfer. DMA_PM_DS_CONFIG = conf2,conf3,conf4 then DMA requests are enabled. The interrupts should be disabled by writing a 0 in the corresponding IE bits. For more details refer to description of DMA_PM_DS_CONFIG in the DCRs. The data in the memory area defined by the MEM_DEPTH parameter is always filled in a sequential fashion as in a FIFO, but can be read sequentially or randomly. There can be no slot-memory location correspondence due to the T bit errors in the Manchester decoder. The memory area above the configured MEM_DEPTH size will always be read as 0. Irrespective of the configuration, the data can always be randomly read by the CPU from the PMRL/PMRH register.
11 ERROR_SELECT4	The ERROR_SELECT bitfield indicates which of the C, E, EM, T or F error conditions generate an interrupt when the EICR Interrupt Enable bit is set. The individual bits are mapped as {C,E,EM,T,F}. The corresponding error is also latched in the PSI5_EISR. Default: ERROR_SELECT = 11111: Any of C,E,EM,T or F generate an interrupt. C error CRC Error (C) Bit 20: ERROR_SELECT[4] or C_INT_SEL 0 The C bit does not generate an interrupt. 1 The C bit generates an interrupt.
12 ERROR_SELECT3	Error_Select[3] or E_INT_SEL E error Electrical Error (E) 0 The E bit does not generate an interrupt. 1 The E bit generates an interrupt.
13 ERROR_SELECT2	Error_Select[2] or EM_INT_SEL 0 The EM bit does not generate an interrupt. 1 The EM bit generates an interrupt.
14 ERROR_SELECT1	Error_Select[1] or T_INT_SEL 0 The T bit does not generate an interrupt. 1 The T bit generates an interrupt.

Table 1496. PSI5_CH0_PCCR field descriptions (continued)

Field	Description
15 ERROR_SELECT0	Error_Select[0] or F_INT_SEL 0 The F bit does not generate an interrupt. 1 The F bit generates an interrupt.
17 GTM_RESET _ASYNC_EN	GTM_RESET_ASYNC_EN 0 Both the assertion and the deassertion of the GTM reset are treated synchronously. Inside the IP the GTM reset is synchronized on the "ipg_clk_psi5". This can delay the impact of the GTM reset by 1 to 2 system clock(ipg_clk_psi5). Refer to *Figure 1518* for the definition of "ipg_clk_psi5". 1 Assertion of the GTM reset is treated asynchronously but deassertion is synchronized on the ipg_clk_psi5.
21 DEBUG_EN	This bit allows/prevents the IP from entering the debug mode, whenever the debugger is connected and a breakpoint is encountered by the debugger: 0 The IP never enters the debug mode, even if the debugger is connected and a breakpoint is encountered. It keeps on working normally, as in the functional mode. 1 Whenever the debugger is connected and a breakpoint is encountered, the IP enters the debug mode. In the Debug mode, the DMA read pointers and other status registers that normally get affected by the READ operation remain unaffected.
22 DEBUG_FREEZE _CTRL	0 When the IP enters the Debug mode then it goes into the "Open" mode, that is, the core of the IP keeps on working normally. In this case, there can be a difference between the value of a particular status bit, which is there at the time Debug mode is entered and the time at which the debugger reads it, since some upcoming events might change the status. The hardware clearing, of course, remains disabled. **Note:** In the above configuration, the status of registers read through the debugger may be different when entering debug mode and when actually accessing registers at some later point of time. 1 When the IP enters the "Debug mode" then it goes into the "Freeze Mode". Note that whenever there is a request to enter the freeze mode then the IP enters this mode, ONLY when the Manchester decoder and the Sync Pulse Generator are stopped coherently. This means that before entering the "freeze mode", if there is any pending reception that is ongoing, then the same is completed first. Simultaneously the "sdout" line of the IP should be at a low level. Once these two conditions are satisfied, the IP enters the Debug Freeze mode. The Time Stamp counters are also halted. In the above configuration, incoming packets could get lost while the Debug mode is active.
23 SP_TS_CLK_SEL	This bit controls which clock goes to the Sync Pulse and Time Stamp Generator Unit. 0 The 1 MHz clock generated from the CGM goes for clocking the Time Stamp Unit and sync pulse generator. 1 The clock generated from the GTM goes for clocking the Time Stamp Unit and Sync Pulse Generator. **Note:** The above selection affects ONLY the clock for the Time Stamp Unit and Sync Pulse Generator. The 1 MHz clock to the rest of the logic like the Manchester decoder or the Slot Boundary Counters is still the 1 MHz from the CGM. The clock selected using this bit is called SP_TS_CLK.

Table 1496. PSI5_CH0_PCCR field descriptions (continued)

Field	Description
26 FAST_CLR_SMC	This bit controls the clearing mechanism of the IS_NVSM[x], IS_CESM[x], and IS_OWSM[x] bits in the GISR. 0 Fast clearing is disabled (clear when written to 1). 1 Fast clearing is enabled. **Note:** This bit is effective only when DCR[DMA_EN_SF] = 1. When FAST_CLR_SMC = 1 then the IS_NVSM[x], IS_CESM[x], and IS_OWSM[x] bits are automatically cleared upon reading the DSFR message register, as per the slot[x] message read in from the DSFR. When reading directly from the SFR registers, this bit does not affect auto clearing. If FAST_CLR_SMC = 0 then the above bits have to be cleared by a w1c.

Table 1496. PSI5_CH0_PCCR field descriptions (continued)

Field	Description
27 FAST_CLR_PSI5	This bit controls the clearing mechanism of bits in NDSR (in conf2, conf3 and conf4 modes), EISR (in conf2, conf3 and conf4 modes), and OWSR (in conf2 and conf3 modes). 1 Fast Clearing Enabled 0 Fast Clearing Disables (clear when written to 1) **Note:** This bit is effective only when DMA_PM_DS_CONFIG = conf2, conf3, or conf4 else this bit has no effect and only software clearing is available. Further, though the OWSR is not directly read by the DMA as a single register, it is cleared in specific modes (when FAST_CLR_PSI5 == 1) as can be seen below. Following are the details in the various modes of the DCR[DMA_PM_DS_CONFIG] bits: – In conf1 mode: This bit has no effect. Reading the DDSR and the DPMR returns a 0. – In conf2 mode: In this case the diagnostic bits (EISR, NDSR) are cleared ONLY when the diagnostic registers appended with the corresponding PSI5 messages are completely read using the DPMR. During the intermediate operation (when the PSI5 message has been read but the diagnostic bits are still pending), the PSI5 message read from the DPMR will not clear these bits. Both of these registers are simultaneously cleared when the corresponding DMA operation is over. However, only those bits are cleared, the corresponding messages of which form the part of the DMA request being serviced. Note that the OWSR bits are cleared upon the corresponding PSI5 message read from the DPMR. Reading the DDSR returns a 0 in this mode. – In conf3 mode: The PSI5 diagnostic register bits (EISR,NDSR) and OWSR are automatically cleared when the corresponding PSI5 message is read from the DPMR. The specific register bits of EISR,NDSR and OWSR, corresponding to each message, are cleared sequentially one by one as the corresponding message is being read by the DMA. Note that the three bits (each of EISR, NDSR, and OWSR) corresponding to a specific location are always cleared simultaneously. Reading the DDSR returns a 0. When reading from the PMRH/PMRL registers, this bit does not have any effect on fast clearing. – In conf4 mode: In this case the diagnostic bits (EISR, NDSR) are cleared as soon as these registers are read through the DDSR. These registers are cleared simultaneously in one go, as soon as the corresponding DMA request has been serviced. Only those bits of these registers are cleared, the messages of which form a part of the DMA request. Note that in this mode the reading of the PSI5 messages from the DMA is not available as the request is being used for reading the diagnostic registers. Reading the DPMR returns a 0. The PSI5 messages can be read only from the PMRH/PMRL register. Hence the OWSR has to be cleared only by w1c. – In all the above modes, when reading from the PMRH/PMRL registers, the FAST_CLR_PSI5 does not have any effect on fast clearing. – Further, In all the above cases if FAST_CLR_PSI5 = 0 then the specific diagnostic bits and the OWSR have to be cleared by w1c.

Table 1496. PSI5_CH0_PCCR field descriptions (continued)

Field	Description
28 BIT_RATE	This bit selects the receive message bit rate (T bit) for this particular PSI5 Channel, that is, it selects one of the two driven clocks (4 MHz or 6.048 MHz) in CGM. 0 125 Kbit/s bit rate selected (4 MHz clock). 1 189 Kbit/s bit rate selected (6.048 MHz clock).
29 MODE	This bit selects the operating modes. 0 Asynchronous operating mode (Integrated sync pulse generator module is switched off, only RX active). Note that in this mode the sync pulses from the internal sync pulse generator or the GTM are not allowed to be propagated to the output path, which always remains at logic 0. There is no T-bit error in the Asynchronous mode. 1 Synchronous operating modes (Integrated sync pulse generator module is switched on, both TX and RX subblocks active).
30 PSI5_CH_CONFIG	PSI5 Channel Config mode request If this bit is set (and PCCR[PSI5_CH_EN] = '1') this particular psi5 channel enters Configuration mode from Disable mode. Default value is 0. User software must program all PSI5 channel parameters before clearing this bit. Once cleared, channel parameters in registers are locked for writing. The values set cannot be changed until this bit is set again after entering the Disable mode 1 (If PCCR[PSI5_CH_EN] = '1' and GCR[GLOBAL_DISABLE_REQ] = '0') - PSI5 Channel enters Config mode. Various configuration registers can be programmed in this mode. The only registers NOT writable in the config mode but writable in the Normal mode are the output registers PSI5_DBR and the PSI5_DSR. 0 (If PCCR[PSI5_CH_EN] = '1' and GCR[GLOBAL_DISABLE_REQ] = '0') PSI5 Channel enters Normal mode. Channel parameter registers are locked for writing. This bit can be written in all the modes but writing a "0" in the Normal mode will not change the operation mode of the device. **Note:** This bit can be written in all modes, but writing a 0 in the Normal mode does not change the operation mode of the device. For more details refer to *Figure 1452* and to *Table 1492*. Do note that when going from Disable to Config mode, the programming of other bits of this register CANNOT be done in the same cycle, as the one in which this bit is being written.
31 PSI5_CH_EN	PSI5 Channel Enable. If this bit is cleared this particular psi5 channel continues to stay in Disable mode even if GCR[GLOBAL_DISABLE_REQ] = 0. If set, it enters Configuration mode from Disable mode (based on PCCR[PSI5_CH_CONFIG]). Default value is 0. All state machines are disabled. This bit can be written in all modes. 0 PSI5 channel continues in Disable mode. 1 PSI5 channel enabled to enter Config/Normal mode. **Note:** When PSI5_CH_EN == 0 then the clock enable for that particular channel that goes from the IP, goes to "0" (inactive state). ""The enable for the GTM Clock is not controlled by the IP and should be controlled externally. Do note that when going from Disable to Config mode, the programming of other bits of this register CANNOT be done in the same cycle, as the one in which this bit is being written.

60.3.2.3 DMA Control Register (PSI5_CH0_DCR)

Figure 1455 shows the DMA Control Register.

Address: 0x000C + *i**0x1C0 (*i*=0) Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	DMA_PM_DS_WM					0	0	0	0	0	IE_DMA_TF_SF	IE_DMA_TF_PM_DS	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	IE_DMA_PM_DS_FIFO_FULL	IE_DMA_SFUF	0	IE_DMA_PM_DS_UF	0	0	0	0	0	DMA_EN_SF	DMA_PM_DS_CONFIG	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1455. DMA Control Register (PSI5_CH0_DCR)

Table 1497. PSI5_CH0_DCR field descriptions

Field	Description
3:7 DMA_PM _DS_WM	0–31 Valid Water Mark levels Value fixed by user software. Since each location has two 32-bit registers hence a total of sixty-four 32-bit words is present. The default and minimum value of dma_pm_ds_wm is '0', that is, 1 message/Diagnostic bit of 1 location. The maximum value of dma_pm_ds_wm is 31, that is, the actual watermark = 32 RAM locations. **Note:** These bits are ineffective if DMA_PM_DS_CONFIG = conf1. These bits indicates the number of PSI5 message to be stored in FIFO / number of unread diagnostic bits in the diagnostic registers with a correspondence to the messages in the FIFO, before the corresponding DMA request is asserted. The watermark definition changes depending on the configuration of the DMA_PM_DS_CONFIG bits in the DCR as mentioned below. When DMA_PM_DS_CONFIG = conf2 or conf3 then the watermark refers to the number of new unread PSI5 messages that are stored in the FIFO before a DMA request is asserted. In conf2 mode of DMA_PM_DS_CONFIG, the number of words to be transferred, set in the DMA controllers TCD must be set equal to the number of words corresponding to the set water mark value + 2 words (for the two 32-bit diagnostic registers). In conf3, the number of words to be transferred set in the DMA controllers TCD must be set equal to the number of words corresponding to the set water mark value. When DMA_PM_DS_CONFIG = conf4 then the watermark refers to the number of new unread diagnostic bits in the diagnostic register. In this configuration the PSI5 messages can only be read by the CPU, since the DMA request would cater to reading the diagnostic bits only. In conf4, the number of words transferred in each DMA request = 2 x 32-bit words (for the two 32-bit diagnostic registers). The water mark however refers to the number of new unread diagnostic bits pertaining to each PSI5 message in the FIFO.
13 IE_DMA_TF_SF	Enable for interrupt, which is generated when DMA transfer finishes for DMA SMC Frame register. 0 Interrupt is disabled. 1 Interrupt is enabled. **Note:** This bit is writable in Config and Normal modes.
14 IE_DMA_TF _PM_DS	Enable for Interrupt, which is generated when DMA transfer finishes for PSI5 messages/DMA Diagnostic Status register depending on the DMA_PM_DS_CONFIG bits of the DCR. 0 Interrupt is disabled. 1 Interrupt is enabled. **Note:** This bit is writable in Config and Normal modes.
20 IE_DMA_PM_DS_ FIFO_FULL	This bit is effective only when the DMA_PM_DS_CONFIG = conf2,conf3 or conf4. It enables the FIFO FULL condition generation interrupt. For the cases that generate these FIFO FULL conditions, refer to the description of IS_DMA_PM_DS_FIFO_FULL register bit of the DSR. **Note:** This bit is writable in Config and Normal modes.

Table 1497. PSI5_CH0_DCR field descriptions (continued)

Field	Description
21 IE_DMA_SFUF	Enables interrupt when there is underflow in DMA SMC Frame register when DMA is enabled. It is set when the DSFR is read without a valid DMA request, that is, it is empty. Default value is 0. 0 Interrupt is disabled. 1 Interrupt is enabled. **Note:** This bit is writable in Config and Normal modes.
23 IE_DMA_PM _DS_UF	Enables interrupt when there is underflow in DMA PSI5 message register when DMA is enabled. Default value is 0. For the details as to when this underflow occurs, refer to the details of IS_DMA_PM_DS_UF bit details of the DSR. 0 Interrupt is disabled. 1 Interrupt is enabled. **Note:** This bit is writable in Config and Normal modes.

Table 1497. PSI5_CH0_DCR field descriptions (continued)

Field	Description
29 DMA_EN_SF	Enable DMA request for SMC Frame Data 0 DMA for SMC frame is disabled. The six dedicated 1-1 slot corresponding SMC registers can only be read by the CPU through the SFR. Reading the DSFRs returns 0. 1 DMA for SMC frame is enabled. The six dedicated slot corresponding SMC registers can be read by reading the DSFR successively. In the DMA mode the six registers are read in a round robin fashion as explained in the DSFR details. Reading directly through the SFR registers is still available.
30:31 DMA_PM_DS _CONFIG	These bits define how the PSI5 messages are stored in the MEM_DEPTH memory area and the associated diagnostic bits (EISR, NDSR) are transferred. 00 **c**onf1 The DMA request is disabled. In this mode, only the interrupts can be used for indicating that data transfer is required. 01 conf2 After transferring the “dma_pm_ds_wm” number of PSI5 messages, there will be two additional 32-bit transfers in which the NDSR and the EISR registers are sequentially transferred through the DPMR registers by using DMA. 10 conf3 “dma_pm_ds_wm” number of PSI5 messages is transferred in each DMA request. The DMA read is through the DPMR registers. 11 conf4 Only the diagnostic bits are transferred through the DDSR. The DMA read is through the DDSR registers. The DPMR, DDSR, and DSFR registers should be used for DMA access and the PMRH/PMRL, NDSR, EISR, and OWSR registers should be used for CPU access. **Note:** For more details refer to the description of the DCR[DMA_PM_DS_WM] bits. When DMA_PM_DS_CONFIG bits = conf2 or conf3 and if the CPU and the DMA are simultaneously used for reading the PSI5 messages, then in the course of reading the messages it is possible that there can be an overflow, though some intermediate locations, which have been read by the CPU, are empty. This will occur because the DMA access is made independent of the CPU access. The DMA logic would never know that the CPU has already read a message that was scheduled to be read by the DMA. Whenever the DMA is used for reading the PSI5 message (conf2/conf3) modes then though the CPU can read the messages in parallel through the PMRH/PMRL registers, still for the purpose of setting the Overwrite/Overflow bits only a read from the DMA through the popup registers (DPMR) would be treated as valid read. This will also be true for the SMC messages when accessed simultaneously by the CPU/DMA.

60.3.2.4 DMA Status Register (PSI5_CH0_DSR)

Figure 1456 shows the DMA Status Register.

Address: 0x0010 + *i**0x1C0 (*i*=0) Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	IS_DMA_TF_SF	IS_DMA_TF_PM_DS	0
W														w1c	w1c	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	IS_DMA_PM_DS_FIFO_FULL	IS_DMA_SFUF	0	IS_DMA_PM_DS_UF	0	0	0	0	0	0	0	0
W					w1c	w1c		w1c								
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1456. DMA Status Register (PSI5_CH0_DSR)

Table 1498. PSI5_CH0_DSR field descriptions

Field	Description
13 IS_DMA_TF_SF	This flag is set when DMA transfer finishes for DMA SMC Frame register.
14 IS_DMA_TF_PM_DS	This flag is set when DMA transfer finishes. For the various configurations of this request refer to the descriptions of the DMA_PM_DS_CONFIG bits in the DCR. This flag is cleared by w1c.

Table 1498. PSI5_CH0_DSR field descriptions (continued)

Field	Description
20 IS_DMA_PM_DS _FIFO_FULL	Interrupt Status when there is FIFO FULL corresponding to the DMA request ipd_psi5_dma_req_pm_ds. The FIFO FULL scenarios are detailed below with respect to the various settings of the DMA_PM_DS_CONFIG bit. conf1: FIFO FULL disabled conf2 and conf3: FIFO FULL occurs when all the memory locations become full in the PSI5 message FIFO. It indicates that without any read, any new upcoming message will now overwrite the existing messages and set the overwrite bits. This bit is cleared either by w1c or upon a PSI5 message read depending on the FAST_CLR_PSI5 bit. conf4: FIFO FULL occurs when there is a “REGISTER FULL” in the Diagnostic registers. FIFO FULL here means that the diagnostic registers contain 32 unread diagnostic bits. A new message starts changing the status of the diagnostic bits. In this configuration it is necessary to read the diagnostic bits through the DMA using the DDSR in order to prevent FIFO FULL. Of course the diagnostic registers NDSR/EISR can be read any time by the CPU, but to prevent FIFO FULL they HAVE to be read through the DDSR registers only. Also in this mode the PSI5 messages DONT contribute to the FIFO FULL. The PSI5 messages have to be read by the CPU using the PMRH/PMRL registers in order to prevent the overwrite bits from being set in the OWSR. In this mode the OWSR bits are NOT automatically cleared upon the PSI5 message read since the PSI5 message has to be read by the PMRH/PMRL register and NOT though the DPMR. The OWSR bits have to be cleared by a w1c. Reading the DPMR returns a 0 in this mode. This bit is cleared by a w1c. 0 No FIFO full. 1 FIFO full has occurred
21 IS_DMA_SFUF	SMC Frame DMA underflow: This happens when the DSFR has been read without a proper DMA request being asserted. The DSFR is empty and it is read. This bit is cleared by a w1c. 0 No underflow has occurred. 1 Underflow has occurred.
23 IS_DMA_PM_ DS_UF	Depending on the DMA_PM_DS_CONFIG bits following is the underflow conditions: Default value is 0. conf1: underflow disabled conf2 and conf3: Underflow happens when the software reads the PSI5 message FIFO though the DPMR, beyond the available messages (empty area). conf4: Underflow happens when the diagnostic registers are read through the DDSR without a valid DMA request and the DDSR contains no new data. 0 No underflow has occurred. 1 Underflow has occurred. The bits are cleared by a w1c.

60.3.2.5 General Interrupt Control Register (PSI5_CH0_GICR)

The General Interrupt Control Register (PSI5_GICR) contains the bits for controlling the SMC message related interrupts, interrupts related to ECU-to-sensor communication, and interrupts to the Time Stamp.

Address: 0x0014 + *i**0x1C0 (*i*=0) Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	IE_CESM[6:1]						IE_STS	IE_DTS	IE_DSROW	IE_BROW	IE_PROW	IE_DSRR	IE_BRR	IE_PRR
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	IE_OWSM[6:1]						0	0	IE_NVSM[6:1]					
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1457. General Interrupt Control Register (PSI5_CH0_GICR)

Table 1499. PSI5_CH0_GICR field descriptions

Field	Description
2:7 IE_CESM[6:1]	Interrupt request when received SMC frame in corresponding slot has CRC failure (CRC recalculation on SMC). 0 Interrupt is disabled. 1 Interrupt is enabled. **Note:** These bits are writable in Config and Normal modes.
8 IE_STS	Interrupt request enabled when the Sync Pulse Triggered Time Stamp value is refreshed on STSRR. 0 Interrupt is disabled. 1 Interrupt is enabled. **Note:** This bit is writable in Config and Normal modes.
9 IE_DTS	Interrupt request enabled when the Data Start sequence Triggered Time Stamp value is refreshed on DTSRR. 0 Interrupt is disabled. 1 Interrupt is enabled. **Note:** This bit is writable in Config and Normal modes.
10 IE_DSROW	Interrupt request enabled when system tries to overwrite on Data Shift register when it is not ready to accept new data. 0 Interrupt is disabled. 1 Interrupt is enabled. **Note:** This bit is writable in Config and Normal modes.
11 IE_BROW	Interrupt request enabled when system tries to overwrite on buffer register when it is not ready to accept new data. 0 Interrupt is disabled. 1 Interrupt is enabled. **Note:** This bit is writable in Config and Normal modes.

Table 1499. PSI5_CH0_GICR field descriptions (continued)

Field	Description
12 IE_PROW	Interrupt request enabled when system tries to overwrite on Preparation register when it is not ready to accept new data. 0 Interrupt is disabled. 1 Interrupt is enabled. **Note:** This bit is writable in Config and Normal modes.
13 IE_DSRR	Interrupt request enabled when Data Shift Register is ready to accept new data. 0 Interrupt is disabled. 1 Interrupt is enabled. **Note:** This bit is writable in Config and Normal modes.
14 IE_BRR	Interrupt request enabled when buffer Register is ready to accept new data. 0 Interrupt is disabled. 1 Interrupt is enabled. **Note:** This bit is writable in Config and Normal modes.
15 IE_PRR	Interrupt request enabled when Preparation Register ready to accept new data. 0 Interrupt is disabled. 1 Interrupt is enabled. **Note:** This bit is writable in Config and Normal modes.
18:23 IE_OWSM[6:1]	Interrupt request enable for the SMC message Overwrite bits. The interrupt for SMC message overwrite occurs when any unread SMC message in a particular SMC slot register gets overwritten by a new SMC message of that particular slot. Even if the SMC message has a CRC error, still if it is unread and is overwritten the overwrite bit gets set. **Note:** These bits are writable in Config and Normal modes.
26:31 IE_NVSM[6:1]	Interrupt request enabled when any new valid SMC message (fault free) is received in SFR[i]{i:0 to 5}, that is, during slot1 to slot6. This interrupt is generated only when corresponding IS_NVSMS[5:0] is set. For each bit position, the corresponding request is enabled as follows: 0 Interrupt is disabled. 1 Interrupt is enabled. **Note:** These bits are writable in Config and Normal modes.

60.3.2.6 New Data Interrupt Control Register (PSI5_CH0_NDICR)

The New Data Interrupt Control Register (PSI5_NDICR) contains the interrupt enable bits related to a new PSI5 message arrival, in the corresponding PSI5 message buffer location.

Address: 0x0018 + *i**0x1C0 (*i*=0) Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	IE_ND															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	IE_ND															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1458. New Data Interrupt Control Register (PSI5_CH0_NDICR)

Table 1500. PSI5_CH0_NDICR field descriptions

Field	Description
0:31 IE_ND	Interrupt request enabled when any new message (fault-free/with fault) is received in the RAM buffer Register 'x' location [x: 0 to 31], This interrupt is generated only when corresponding RAM buffer Ready with new data. 0 Interrupt is disabled. 1 Interrupt is enabled. **Note:** These bits are writable in Config and Normal modes.

Note: Depending on the value of MEM_DEPTH in the PCCR, only the corresponding diagnostic bits are set in this register.

60.3.2.7 Overwrite Interrupt Control Register (PSI5_CH0_OWICR)

The Overwrite Interrupt Control Register (PSI5_OWICR) is related to the generation of the overwrite interrupts. The overwrite interrupt is generated when the unread PSI5 message in a particular RAM location is overwritten by another PSI5 message.

Address: 0x001C + *i**0x1C0 (*i*=0) Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	IE_OW															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	IE_OW															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1459. Overwrite Interrupt Control Register (PSI5_CH0_OWICR)

Table 1501. PSI5_CH0_OWICR field descriptions

Field	Description
0:31 IE_OW	Interrupt request enabled when any new message overwrites the old unread PSI5 message in the RAM buffer Register 'x' location [x: 0 to 31], This interrupt generated only when corresponding RAM buffer Ready with new data. 0 Interrupt is disabled 1 Interrupt is enabled **Note:** These bits are writable in Config and Normal modes.

Note: Depending on the value of MEM_DEPTH in the PCCR, only the corresponding diagnostic bits are set in this register.

60.3.2.8 Error Interrupt Control Register (PSI5_CH0_EICR)

Figure 1460 shows the Error Interrupt Control Register.

Address: 0x0020 + *i**0x1C0 (*i*=0) Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	IE_ERROR															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	IE_ERROR															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1460. Error Interrupt Control Register (PSI5_CH0_EICR)

Table 1502. PSI5_CH0_EICR field descriptions

Field	Description
0:31 IE_ERROR	Interrupt request enabled when any/all of the error conditions C, E, EM, T, or F is observed in a PSI5 message in the RAM buffer Register 'x' location [x: 0 to 31]. This interrupt is generated only when the corresponding RAM buffer is Ready with new data. Which error condition (any of C, E, EM, T, or F) generates the interrupt is selectable by the PCCR[ERROR_SELECT] field. If all are selected, then which of C,E,EM,T or F is set can be isolated by reading the corresponding received message. 0 Interrupt is disabled. 1 Interrupt is enabled. **Note:** These bits are writable in Config and Normal modes.

Note: Depending on the value of MEM_DEPTH in the PCCR, only the corresponding diagnostic bits are set in this register.

60.3.2.9 General Interrupt Status Register (PSI5_CH0_GISR)

The General Interrupt Status Register (PSI5_GISR) contains the status bits related to SMC messages, ECU-to-sensor communication, and the Time Stamp.

Address: 0x0024 + *i**0x1C0 (*i*=0) Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	IS_DEBUG_FREEZE	0	IS_CESM6	IS_CESM5	IS_CESM4	IS_CESM3	IS_CESM2	IS_CESM1	IS_STS	IS_DTS	IS_DSROW	IS_BROW	IS_PROW	DSR_RDY	DBR_RDY	DPR_RDY
W			w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	IS_OWSM6	IS_OWSM5	IS_OWSM4	IS_OWSM3	IS_OWSM2	IS_OWSM1	0	0	IS_NVSM6	IS_NVSM5	IS_NVSM4	IS_NVSM3	IS_NVSM2	IS_NVSM1
W			w1c	w1c	w1c	w1c	w1c	w1c			w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1461. General Interrupt Status Register (PSI5_CH0_GISR)

Table 1503. PSI5_CH0_GISR field descriptions

Field	Description
0 IS_DEBUG_FREEZE	This flag is set to “1” when the IP enters the debug freeze mode. Refer to *Section 60.1.3.5: Debug mode* for details about the debug mode. This bit is auto cleared by the hardware when the IP exits the debug freeze mode. 0 IP not in debug freeze mode 1 IP in debug freeze mode
2:7 IS_CESM[6:1]	These flags are set when the received SMC frame in the corresponding slot has a CRC failure (CRC recalculation on SMC). These bits are automatically cleared upon message read, when the FAST_CLR_SMC =1 and the DMA_EN_SF = 1 or by w1c.
8 IS_STS	This Interrupt flag is set when the Sync Pulse Triggered Time Stamp value is refreshed on STSRR.
9 IS_DTS	This Interrupt flag is set when the Data Start sequence Triggered Time Stamp value is refreshed on DTSRR.
10 IS_DSROW	This flag is set when the system tries to overwrite on Data Shift register when it is not ready to accept new data.
11 IS_BROW	This flag is set when the system tries to overwrite on buffer register when it is not ready to accept new data.
12 IS_PROW	This flag is set when the system tries to overwrite on Preparation register when it is not ready to accept new data.

Table 1503. PSI5_CH0_GISR field descriptions (continued)

Field	Description
13 DSR_RDY(1)	This bit acts both as a status and a control bit. Status bit Action: When “1” it indicates that the Data Shift register is ready to receive new data from the system bus side or the Data Buffer register. When “0” it indicates that there is a ongoing command transmission of the data in DSR, or the CPU has done a “w1c” to this bit leading to the new command transmission. In this case any write by the CPU to this register will be rejected and the PSI5_GISR[IS_DSROW] bit gets set. A write to PSI5_DOBCR[DSR_RST] will abort the current command transmission and make the DSR_RDY as “1”. Control Bit Action: When a “w1c” is done to this register bit, and PSI5_DOBCR[SW_READY] == 1, then it indicates to the IP that the data in DSR is valid and is to be used for command shifting. This procedure is used when the data is written by the CPU to DSR directly skipping the DBR. For detailed action of this bit refer to *Section 60.4.3.4: Data transmission*.
14 DBR_RDY(1)	This bit acts both as a status and a control bit. Status bit Action: When “1” it indicates that the Data Buffer register is ready to receive new data from the system bus side or the Data Preparation register. When “0” it indicates that there is a ongoing shift of the data from the DPR to the DBR, or the CPU has done a “w1c” to this bit. In this case any write by the CPU to this register will be rejected and the PSI5_GISR[IS_BROW] bit gets set. A write to PSI5_DOBCR[DBR_RST] will abort the current data transaction in DBR and make the DBR_RDY as “1”. Control Bit Action: When a “w1c” is done to this register bit, then it indicates to the IP that the data in DBR is valid and is to be used for shifting to DSR. This procedure is used when the data is written by the CPU to DBR directly skipping the DPR. For detailed action of this bit refer to *Section 60.4.3.4: Data transmission*.
15 DPR_RDY(1)	This bit acts both as a status and a control bit. Status bit Action: When “1” it indicates that the Data Preparation register is ready to receive new data from the system bus side. When “0” it indicates that there is a ongoing processing of the data in DPR or there is a data shift ongoing from the DPR to the DBR, or the CPU has done a “w1c” to this bit. In this case any write by the CPU to this register will be rejected and the PSI5_GISR[IS_PROW] bit gets set. When PSI5_DOBCR[CMD_TYPE] == “7” then this bit remains at value “0”. Control Bit Action: When a “w1c” is done to this register bit, then it indicates to the IP that the data in DPR is valid and is to be used for processing and shifting to DBR. For detailed action of this bit refer to *Section 60.4.3.4: Data transmission*.
18:23 IS_OWSM[6:1]	Status for the SMC message overwrites bits. Each of the 6 locations corresponds to the SMC message pertaining to each SFR[x] register, x = 1 to 6. These bits are cleared automatically upon message read when the FAST_CLR_SMC = 1 and the DMA_EN_SF = 1 or by w1c.
26:31 IS_NVSM[6:1]	These flags are set when corresponding SFR[i] receives and is ready (to read) with new valid SMC message. These bits are cleared automatically upon message read, when the FAST_CLR_SMC = 1 and the DMA_EN_SF = 1 or by w1c.

1. For these register bits individual “bit setting” commands should not be used. Rather, the normal register read/write commands should be used.

60.3.2.10 DMA PSI5 Message Register (PSI5_CH0_DPMR)

Figure 1462 shows the DMA PSI5 Message Register.

Address: 0x0028 + *i**0x1C0 (*i*=0) Access: User read-only

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	PSI5_RXDATA															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	PSI5_RXDATA															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1462. DMA PSI5 Message Register (PSI5_CH0_DPMR)

Table 1504. PSI5_CH0_DPMR field descriptions

Field	Description
0:31 PSI5_RXDATA	PSI5_RXDATA[31:0] This popup register contains the PMRL[31:0] data followed by the PMRH[31:0] data for each of the FIFO locations, when the FIFO is read by the DMA. It would require two 32-bit IPS access to read one PSI5 Message. For each of the locations, the DATA is pushed sequentially until all the data corresponding to the programmed watermark is complete, after which the DMA request is deasserted. The depth of the FIFO is 32 × 64. This popup register can be used differently based on the configuration of the DMA_PM_DS_CONFIG bits in the DCR. When DMA_PM_DS_CONFIG = conf2 then it pops the PSI5 message, followed by NDSR, followed by EISR. When DMA_PM_DS_CONFIG = conf3, it pops only the PSI5 messages. When DMA_PM_DS_CONFIG = conf4, it pops the NDSR followed by EISR. This register is a reflection of the location, pointed to by the DMA read pointer. For more details refer to the *Section 60.3.2.2: PSI5 Channel Control Register (PSI5_CH0_PCCR)*, *Section 60.3.2.3: DMA Control Register (PSI5_CH0_DCR)*, and *Section 60.3.2.4: DMA Status Register (PSI5_CH0_DSR)*.

60.3.2.11 DMA SMC Frame Register (PSI5_CH0_DSFR)

Figure 1463 shows the DMA SMC Frame Register.

Address: 0x002C + *i**0x1C0 (*i*=0) Access: User read-only

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	SMC_RXDATA															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	SMC_RXDATA															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1463. DMA SMC Frame Register (PSI5_CH0_DSFR)

Table 1505. PSI5_CH0_DSFR field descriptions

Field	Description
0:31 SMC_RXDATA	When the DMA_EN_SF = 1, then the six SFR registers are searched in a round robin fashion for the reception of the complete SMC data. The DMA request is asserted as soon as the first encountered SFR has a complete SMC frame. This request remains asserted until all the registers that have pending data, have transferred the data through the DMA. Each other SFR is a 32-bit register. This register is a reflection of the location pointed to by the DMA read pointer.

60.3.2.12 DMA Diagnostic Status Register (PSI5_CH0_DDSR)

Figure 1464 shows the DMA Diagnostic Status Register.

Address: 0x0030 + *i**0x1C0 (*i*=0) Access: User read-only

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	DDS															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	DDS															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1464. DMA Diagnostic Status Register (PSI5_CH0_DDSR)

Table 1506. PSI5_CH0_DDSR field descriptions

Field	Description
0:31 DDS	This register maps each of the individual Diagnostic registers (in the order NDSR, EISR) depending on the configuration of the DMA_PM_DS_CONFIG bits in the DCR. This register is a reflection of the location, pointed to by the DMA read pointer. For more details refer to the *Section 60.3.2.2: PSI5 Channel Control Register (PSI5_CH0_PCCR)*,*Section 60.3.2.3: DMA Control Register (PSI5_CH0_DCR)* and *Section 60.3.2.4: DMA Status Register (PSI5_CH0_DSR)*.

Note: No FIFO is implemented for diagnostic registers. They are two 32-bit registers addressable by either the CPU (reading NDSR or EISR directly) or the DMA (reading the DDSR successively).

60.3.2.13 PSI5 Message Receive Register Low (PSI5_CH0_PMRRL)

Figure 1465 shows the PSI5 Message Receive Register Low.

Address: 0x0034 + *i**0x1C0 (*i*=0) Access: User read-only

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	DATA_REGION															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	DATA_REGION												CRCP			C
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1465. PSI5 Message Receive Register Low (PSI5_CH0_PMRRL)

Table 1507. PSI5_CH0_PMRRL field description

Field	Description
0:27 DATA_REGION	These are application-specific optional bits + data payload of received message. See *Section 60.4.2.1: Reception of data frames* for more details. The data is left-aligned, that is, Data Region[0]/D0 corresponds to first received bit of the PSI5 message after Start sequence detection and Data Region[n]/D'n' corresponds to successive bits up to expected number of bits (n: 8 <= n <= 28). If n < 28 remaining bit positions are filled with '0's, that is, between last received bit and CRC field.
28:30 CRCP	This field represents CRC/Parity value of the data. When parity configuration is selected in the S[1-6]FCR registers, the CRC[2] bits denotes the parity.
31 C	This bit will be set if CRC/P recalculation return an Error.

60.3.2.14 PSI5 Message Receive Register High (PSI5_CH0_PMRRH)

Figure 1466 shows the PSI5 Message Receive Register High.

Address: 0x0038 + *i**0x1C0 (*i*=0) Access: User read-only

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	F	EM	E	T	SlotCounter			TimeStampValue							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	TimeStampValue															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1466. PSI5 Message Receive Register High (PSI5_CH0_PMRRH)

Table 1508. PSI5_CH0_PMRRH field descriptions

Field	Description
1 F	This represents that NO frame was received in the corresponding configured slot. 0 Frame was received in the corresponding configured slot. 1 No frame was received (F bit error).
2 EM	This bit indicates electrical error in the PSI5 Message at bits corresponding to Serial Messaging Channel (M0,M1), that is, at least one bit is absent during the T bit period. 0 No electrical error in M0, M1 fields. 1 Electrical error in M0, M1 fields.
3 E	This bit indicates electrical error, that is, at least one bit is absent during the Tbit period (except M0 and M1 bits and the start bits). 0 No electrical error in fields other than M0, M1, and start fields. 1 Electrical error in fields other than M0, M1, and start fields.
4 T	This bit indicates timing error, that is, frame has started in an unconfigured slot, spread across two slots, started before the first configured slot. 0 No timing error in the corresponding slot. 1 Timing error in the corresponding slot.
5:7 SlotCounter	This value indicates the slot number in which this frame was received. More specifically, it indicates in which slot the start bits of the frame have been detected. If the start bits get detected before the start boundary of Slot1, then the slot counter has a value of “0”. For asynchronous modes the slot counter has a fixed value of “1”. The six slots are numbered as Slot1, Slot2,....Slot6.
8:31 TimeStampValue	This is 24-bit Time Stamp value appended to Received message as soon as start bits are detected. This time stamp is captured at rising edge of S0 and stored at the rising edge of S1. The specific value that is appended can be programmed by setting the TS_CAPT bit in the *Section 60.3.2.27: Slot n Frame Configuration Register (PSI5_CH0_SnFCR)*. In other words these bits contain either the start bit(S0) captured time stamp or the SYNC pulse captured time stamp, dependent on TS_CAPT bit. Note that in messages where “F” bit has been set; the value of the Time Stamp appended to the message is “0”. It is possible to reset the time stamp counter by asserting to “1” the “gtm_reset” signal from the GTM.

Note: Any new PSI5 message is always stored in the PSI5 receive register (PMRRL/PMRRH). In parallel it is also stored in the RAM registers in a sequential manner like in a Ring Buffer. When a new message comes it would always overwrite the data in the PSI5 receive register. But the original message would still be available in the RAM register provided all locations in the RAM registers are not full, and an overwrite has not occurred.

60.3.2.15 PSI5 Message Register Low *n* (PSI5_CH0_PMRL*n*)

Figure 1467 shows the PSI5 Message Register Low *n*.

Address: 0x003C + *i**0x1C0 (*i*=0) + *n**0x8 (*n*=0 to 31) Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	DATA_REGION															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	DATA_REGION												CRCP			C
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1467. PSI5 Message Register Low *n* (PSI5_CH0_PMRL*n*)

Table 1509. PSI5_CH0_PMRL*n* field descriptions

Field	Description
0:27 DATA_REGION	These are application-specific optional bits + data payload of received message. See *Section 60.4.2.1: Reception of data frames*for more details. The data is left-aligned, that is, Data Region[0]/D0 corresponds to first received bit of the PSI5 message after Start sequence detection and Data Region[n]/D'n' corresponds to successive bits up to expected number of bits (n: 8 <= n <= 28). If n < 28 remaining bit positions are filled with '0's, that is, between last received bit and CRC field. **Note:** In the normal mode, it is auto updated by hardware and not writable by any other sources. It is writable in configuration mode (for test purposes).
28:30 CRCP	This field represents CRC/Parity value of the data. When parity configuration is selected in the S[1-6]FCR registers, the CRCP[2] bits denotes the parity. **Note:** In the normal mode, it is auto updated by hardware and not writable by any other sources. It is writable in configuration mode (for test purposes).
31 C	This bit will be set if CRC/Parity recalculation return an Error.

60.3.2.16 PSI5 Message Register High *n* (PSI5_CH0_PMRH*n*)

Figure 1468 shows the PSI5 Message Register High *n*.

Address: 0x0040 + *i**0x1C0 (*i*=0) + *n**0x8 (*n*=0 to 31) Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	O	F	EM	E	T	Slot_Counter			Time_Stamp_Value							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	Time_Stamp_Value															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1468. PSI5 Message Register High *n* (PSI5_CH0_PMRH*n*)

Table 1510. PSI5_CH0_PMRH*n* field descriptions

Field	Description
0 O	This bit carries the same information as OWSR, but appended here for each message separately. This bit is set when the current message (which is being read) has overwritten some previous unread message. Note that, if the corresponding location in the OWSR has been cleared or has been set by the SOWSR, then the same will be reflected in the message stored in the corresponding RAM location as well.
1 F	This represents that NO frame was received in corresponding configured Slot. 0 Frame was received in the corresponding configured slot. 1 No frame was received (F bit error). In the normal mode, it is auto updated by hardware and not writable by any other sources. It is writable in configuration mode (for test purposes).
2 EM	This bit indicates electrical error in the PSI5 Message at bits corresponding to Serial Messaging Channel (M0, M1), that is, at least one bit is absent during the Tbit period. 0 No electrical error in M0, M1 fields. 1 Electrical error in M0, M1 fields. In the normal mode, it is auto updated by hardware and not writable by any other sources. It is writable in configuration mode (for test purposes).
3 E	This Bit Indicates Electrical Error, that is, at least one bit is absent during the Tbit period (except M0 and M1 bits and the start bits). 0 No electrical error in fields other than M0, M1 and start fields. 1 Electrical error in fields other than M0, M1 and start fields. In the normal mode, it is auto updated by hardware and not writable by any other sources. It is writable in configuration mode (for test purposes).
4 T	This bit indicates timing error, that is, frame has started in an unconfigured slot or it has spread across two slots or it has started before the first configured slot. 0 No timing error in the corresponding slot. 1 Timing error in the corresponding slot. In the normal mode, it is auto updated by hardware and not writable by any other sources. It is writable in configuration mode (for test purposes).

Table 1510. PSI5_CH0_PMRH*n* field descriptions (continued)

Field	Description
5:7 Slot_Counter	This value indicates the slot number in which this frame was received. More specifically, it indicates in which slot the start bits of the frame have been detected. If the start bits get detected before the start boundary of Slot1, then the slot counter has a value of “0”. The slot counter is automatically incremented at each slot boundary irrespective of whether a message is received or not. For asynchronous modes the slot counter has a fixed value of “1”. **Note:** The six slots are numbered as Slot1, Slot2,...., Slot6.In the normal mode, it is auto updated by hardware and not writable by any other sources. It is writable in configuration mode (for test purposes).
8:31 Time_Stamp _Value	This is 24-bit Time Stamp value appended to Received message as soon as start bits are detected. This time stamp is captured at rising edge of S0 and stored at the rising edge of S1. The specific value that is appended can be programmed by setting the TS_CAPT bit in the *Section 60.3.2.27: Slot n Frame Configuration Register (PSI5_CH0_SnFCR)*. In other words these bits contain either the start bit (S0) captured time stamp or the SYNC pulse captured time stamp, dependent on TS_CAPT bit. Note that in messages where “F” bit has been set; the value of the Time Stamp appended to the message is “0”. It is possible to reset the time stamp counter by asserting to “1” the “gtm_reset” signal from the GTM.In the normal mode, it is auto updated by hardware and not writable by any other sources. It is writable in configuration mode (for test purposes).

60.3.2.17 SMC Frame Register *n* (PSI5_CH0_SFR*n*)

Figure 1469 shows the SMC Frame Register *n*.

Address: 0x013C + *i**0x1C0 (*i*=0) + (*n*-1)*0x4 (*n*=1 to 6) Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	SLOT_NO			CER	OW	CRC						C	ID			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	IDDATA				DATA											
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1469. SMC Frame Register *n* (PSI5_CH0_SFR*n*)

Table 1511. PSI5_CH0_SFR*n* field descriptions

Field	Description
0:2 SLOT_NO	This indicates in which slot this SMC frame was received. Note that here the slot number refers to the slot in which all the re-arranged PSI5 messages (from which the M0,M1 bits have been extracted) are supposed to have occurred after the correct PSI5 message placement and Timing corrections have been performed. For SFR[1], slot_no = 1; For SFR[2], slot_no = 2; For SFR[3], slot_no = 3; For SFR[4], slot_no = 4; For SFR[5], slot_no = 5; For SFR[6], slot_no = 6; **Note:** In the normal mode, it is auto updated by hardware and not writable by any other sources. It is writable in configuration mode (for test purposes).
3 CER	CRC Error 0 The present data has no CRC error. 1 The present data has a CRC error.
4 OW	Overwrite status 0 The present data was already read when new message arrived. 1 New message has overwritten the previous unread message. **Note:** If the corresponding SOWSM bits in the SSESR have been set or the OWSM bits in the GISR have been cleared, then the same will be reflected in the message stored in the corresponding RAM location as well. The overwrite occurs when any unread SMC message in a particular SMC slot register gets overwritten by a new SMC message of that particular slot. Even if the SMC message has a CRC error, still if it is unread and is overwritten, the overwrite bit gets set.
5:10 CRC	6-bit CRC for slow serial message **Note:** In the normal mode, it is auto updated by hardware and not writable by any other sources. It is writable in configuration mode (for test purposes).
11 C	Configuration bit 0 12-bit data and 8-bit message ID. 1 16-bit data and 4-bit message ID. **Note:** In the normal mode, it is auto updated by hardware and not writable by any other sources. It is writable in configuration mode (for test purposes).
12:15 ID	Message ID: If C = ’0’ indicates ID[7:4], if C = ’1’ indicates ID[3:0]. **Note:** In the normal mode, it is auto updated by hardware and not writable by any other sources. It is writable in configuration mode (for test purposes).
16:19 IDDATA	Message ID/DATA: If C = ’0’ indicates ID[3:0], if C = ’1’ indicates DATA[15:12]. **Note:** In the normal mode, it is auto updated by hardware and not writable by any other sources. It is writable in configuration mode (for test purposes).
20:31 DATA[11:0]	DATA payload **Note:** In the normal mode, it is auto updated by hardware and not writable by any other sources. It is writable in configuration mode (for test purposes).

Note: Six dedicated 32-bit SFR[x] registers for SMC are available with a 1-1 slot correspondence. These can be read by either the CPU or the DMA (when DMA_EN_SF = 1).

No FIFO for SMC frame is available. Six 32-bit registers are available in the address space, which can be accessed by either the DMA (using the DSFR) or the CPU (using the six SFR registers), depending on the setting of the DMA_EN_SMC bit.

60.3.2.18 New Data Status Register (PSI5_CH0_NDSR)

Figure 1470 shows the New Data Status Register.

Address: 0x0154 + *i**0x1C0 (*i*=0) Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	NDS															
W	w1c															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	NDS															
W	w1c															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1470. New Data Status Register (PSI5_CH0_NDSR)

Table 1512. PSI5_CH0_NDSR field descriptions

Field	Description
0:31 NDS	New Data Status flags for PSI5 Messages corresponding to each PSI5 MB locations. These bits are set when a new Message arrives, that is, faulty or non faulty in corresponding PSI5 MB location. These bits can be cleared by w1c or the fast clearing. Refer to PCCR, DCR and DSR registers for more information. Also, these bits can be set independently by writing in SNDR.

60.3.2.19 Overwrite Status Register (PSI5_CH0_OWSR)

Address: 0x0158 + *i**0x1C0 (*i*=0) Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	OWS															
W	w1c															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	OWS															
W	w1c															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1471. Overwrite Status Register (CH0_OWSR)

Table 1513. CH0_OWSR field descriptions

Field	Description
0:31 OWS	Over Write Status flags for PSI5 Messages corresponding to each PSI5 location in the RAM registers. These bits are set when any unread receive PSI5 message is overwritten by a newly arrived message. These bits can be cleared by w1c or the fast clearing. Refer to PCCR, DCR and DSR registers for more information. Also, these bits can be set independently by writing in SOWSR.

60.3.2.20 Error Indication Status Register (PSI5_CH0_EISR)

Figure 1472 shows the Error Indication Status Register.

Address: 0x015C + *i**0x1C0 (*i*=0) Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	ERROR															
W	w1c															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	ERROR															
W	w1c															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1472. Error Indication Status Register (PSI5_CH0_EISR)

Table 1514. PSI5_CH0_EISR field descriptions

Field	Description
0:31 ERROR	Error Status flags for PSI5 Messages corresponding to each PSI5 MB locations. These bits are set when any of the five selectable (by ERROR_SELECT) bits (C, E, EM, T or F) is set for the corresponding PSI5 Message. These bits can be cleared by w1c or the fast clearing. Refer to PCCR, DCR and DSR registers for more information. Also, these bits can be set independently by writing in SEISR.

60.3.2.21 Set New Data Status Register (PSI5_CH0_SNDSR)

Figure 1473 shows the Set New Data Status Register.

Address: 0x0160 + *i**0x1C0 (*i*=0) Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0															
W	SNDS															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0															
W	SNDS															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1473. Set New Data Status Register (PSI5_CH0_SNDSR)

Table 1515. PSI5_CH0_SNDSR field descriptions

Field	Description
0:31 SNDS	Sets New Data Status flags for PSI5 Messages corresponding to each PSI5 MB locations. The corresponding NDS Flags can be set by writing to corresponding bits in this registers. This register is required for IP testing purposes (testing of Error Conditions through software). These bits can SET the status flags in parallel to the hardware events and are used for test purpose. These registers are one-shot and would always be read as '0'. **Note:** This field is writable in Config and Normal modes. This field is w1s.

60.3.2.22 Set Overwrite Status Register (PSI5_CH0_SOWSR)

Figure 1474 shows the Set Overwrite Status Register.

Address: 0x0164 + *i**0x1C0 (*i*=0) Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0															
W	SOWS															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0															
W	SOWS															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1474. Set Overwrite Status Register (PSI5_CH0_SOWSR)

Table 1516. PSI5_CH0_SOWSR field descriptions

Field	Description
0:31 SOWS	Sets overwrite status flags for PSI5 messages corresponding to each PSI5 MB locations. The corresponding OWS flags can be set by writing to corresponding bits in this registers. This register is required for IP testing purposes (testing of Error Conditions through software). These bits can set the status flags in parallel to the hardware events and are used for test purpose. These registers are one-shot and would always be read as 0. **Note:** These bits are writable in Config and Normal modes. This field is w1s.

60.3.2.23 Set Error Status Register (PSI5_CH0_SEISR)

Figure 1475 shows the Set Error Status Register.

Address: 0x0168 + *i**0x1C0 (*i*=0) Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0															
W	SERROR															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0															
W	SERROR															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1475. Set Error Status Register (PSI5_CH0_SEISR)

Table 1517. PSI5_CH0_SEISR field descriptions

Field	Description
0:31 SERROR	Sets Error Status flags for PSI5 messages corresponding to each PSI5 MB locations. The corresponding ERROR flags can be set by writing to corresponding bits in this registers. This register is required for IP testing purposes (testing of Error Conditions through software). These bits can set the status flags in parallel to the hardware events and are used for test purpose. These registers are one-shot and would always be read as 0. **Note:** These bits are writable in Config and Normal modes. This field is w1s.

60.3.2.24 Set SMC Error Status Register (PSI5_CH0_SSESR)

Figure 1476 shows the Set SMC Error Status Register.

Address: 0x016C + *i**0x1C0 (*i*=0) Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0						0	0	0	0	0	0	0	0
W			SCESM[6:1]													
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0						0	0	0					
W			SOWSM[6:1]								SNVSM[6:1]					
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1476. Set SMC Error Status Register (PSI5_CH0_SSESR)

Table 1518. PSI5_CH0_SSESR field descriptions

Field	Description
2:7 SCESM[6:1]	Sets IS_CESM Status flags for SMC Messages corresponding to each SMC MB locations. The corresponding IS_CESM Flags can be set by writing to corresponding bits in this registers. This register is required for IP testing purposes (testing of Error Conditions through software). These bits can SET the status flags in parallel to the hardware events and are used for test purpose. These registers are one-shot and would always be read as ’0’ (w1s). **Note:** These bits are writable in Config and Normal modes.
18:23 SOWSM[6:1]	Sets IS_OWSM Status flags for SMC Messages corresponding to each SMC MB locations. The corresponding IS_OWSM Flags can be set by writing to corresponding bits in this registers. This register is required for IP testing purposes (testing of Error Conditions through software). These bits can SET the status flags in parallel to the hardware events and are used for test purpose. These registers are one-shot and would always be read as ’0’ (w1s). **Note:** These bits are writable in Config and Normal modes.
26:31 SNVSM[6:1]	Sets IS_NVSM Status flags for SMC Messages corresponding to each SMC MB locations. The corresponding IS_NVSM Flags can be set by writing to corresponding bits in this registers. This register is required for IP testing purposes (testing of Error Conditions through software). These bits can SET the status flags in parallel to the hardware events and are used for test purpose. These registers are one-shot and would always be read as ’0’ (w1s). **Note:** This bit is writable in Config and Normal modes.

60.3.2.25 Sync Time Stamp Read Register (PSI5_CH0_STSRR)

Figure 1477 shows the Sync Time Stamp Read Register.

Address: 0x0170 + *i**0x1C0 (*i*=0) Access: User read-only

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	STSV							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	STSV															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1477. Sync Time Stamp Read Register (PSI5_CH0_STSRR)

Table 1519. PSI5_CH0_STSRR field descriptions

Field	Description
8:31 STSV	Sync Time Stamp Value[23:0] This bitfield provides the value of the free-running 24-bit time stamp counter triggered (captured) at last sync pulse.

60.3.2.26 Data Time Stamp Read Register (PSI5_CH0_DTSRR)

Figure 1478 shows the Data Time Stamp Read Register.

Address: 0x0174 + *i**0x1C0 (*i*=0) Access: User read-only

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	SLOT_COUNTER			DTSV							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	DTSV															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1478. Data Time Stamp Read Register (PSI5_CH0_DTSRR)

Table 1520. PSI5_CH0_DTSRR field descriptions

Field	Description
5:7 SLOT_COUNTER	3-bit Slot Counter, which corresponds to the slot for which the DTSRR contains the Time Stamp. The slot_counter contains the slot in which the start bits of this frame was captured (after successful detection of S1). If the start bits get detected before the start boundary of Slot1, then the slot counter has a value of “0”. The slot counter is automatically incremented at each slot boundary irrespective of whether a message is received or not. In case of asynchronous mode, the slot counter has a fixed value of “1”.
8:31 DTSV	Data Time Stamp Value[23:0] Whenever a valid sequence of the start bits (S0,S1) is detected, the value of the Time Stamp counter captured at the rising edge of the S0 is stored in this register at the rising edge of S1.

Note: If S0 and S1 are detected in different slots then the Time Stamp value and the Slot Counter, both will correspond to the first slot (S0)

60.3.2.27 Slot *n* Frame Configuration Register (PSI5_CH0_S*n*FCR)

Figure 1479 shows the Slot n Frame Configuration Register.

Address: 0x0178 + *i**0x1C0 (*i*=0) + (*n*-1)*0x4 (*n*= 1 to 6) Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	SLOT_EN	TS_CAPT	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	SMCL	0	0	0	0	0	0	0	0	0	DRL					CRCP
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0

Figure 1479. Slot *n* Frame Configuration Register (PSI5_CH0_S*n*FCR)

Table 1521. PSI5_CH0_S*n*FCR field descriptions

Field	Description
13 SLOT_EN	0 This particular slot is disabled, that is, not in use and corresponding configuration bits (as defined by other bits of this PSI5_SnFCR) have no effect. 1 This particular slot is enabled, that is, in use and corresponding configuration bits (as defined by other bits of this PSI5_SnFCR) are applicable.
14 TS_CAPT	0 Time stamp value captured at rising edge of S0 and stored Start sequence first edge. 1 Time stamp value captured at corresponding Tsync pulse (posedge) and stored with corresponding slot PSI5 message.

Table 1521. PSI5_CH0_S*n*FCR field descriptions (continued)

Field	Description
16 SMCL	Serial Messaging Channel field 0 Serial Messaging Channel (optional) not present on Rx Message during Time Slot 'n' 1 2-bit{M1,M0} Serial Messaging Channel present on Rx Message during Time Slot 'n' This field is used to enable the detection of start sequence of SMC frame on particular slot. Detection is not done if this field is '0'.
26:30 DRL	Data Region Length[4:0] This field represents Length of Data region (including Frame Control, Status, Data payload1&2 regions and optional messaging bits M0, M1) in Rx Message during Time Slot 'n' 00000 invalid ... 00111 Invalid 01000 8 bit long data region 01001 9 bit long data region .... 11011 27 bit long data region 11100 28 bit long data region 11101 invalid 11111 Invalid
31 CRCP	0 3 bit CRC[2:0] present on Rx Message during Time Slot 'n' 1 1 bit Parity field present on Rx Message during Time Slot 'n'

Note: In Asynchronous mode, the PSI5_S1FCR register has to be programmed for controlling the various parameters pertaining to the message received. The other registers PSI5_S[2-6]FCR have no effect in the asynchronous mode.

60.3.2.28 Slot *n* Start Boundary Register (PSI5_CH0_S*n*SBR)

Figure 1480 shows the Slot *n* Start Boundary Register (PSI5_S*n*SBR).

Total of six registers for each of the six slots. These are S1SBR, S2SBR, S3SBR, S4SBR, S5SBR, and S6SBR. Here *n* = 1 to 6 corresponding to each of the six slots.

Address: 0x0190 + *i**0x1C0 (*i*=0) (PSI5_CH0_S1SBR)
0x0192 + *i**0x1C0 (*i*=0) (PSI5_CH0_S2SBR)
0x0194 + *i**0x1C0 (*i*=0) (PSI5_CH0_S3SBR)
0x0196 + *i**0x1C0 (*i*=0) (PSI5_CH0_S4SBR)
0x0198 + *i**0x1C0 (*i*=0) (PSI5_CH0_S5SBR)
0x019A + *i**0x1C0 (*i*=0) (PSI5_CH0_S6SBR)
Access: User read-only

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	S*n*SBT														
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1480. Slot *n* Start Boundary Register (PSI5_CH0_S*n*SBR)

Table 1522. PSI5_CH0_S*n*SBR field descriptions

Field	Description
1:15 S*n*SBT	Slot *n* Start Boundary Time This field specifies the time in intervals of 1 µs, occurring after the *rising edge* of the timing sync pulse, at which the slot *n* should start. n = 1 to 6 dedicated for each of the six slots.

60.3.2.29 Slot *n* End Boundary Register (PSI5_CH0_SNEBR)

Figure 1480 shows the Slot *n* Start Boundary Register (PSI5_S*n*SBR).

Slot n End Boundary register is a single register, which can be programmed to define the “end” of the nth slot. The n can be defined in the SLOT_NO field of this register.

There are seven boundary registers for the six slots. Six of these registers (S1SBR, S2SBR, S3SBR+, S4SBR, S5SBR, and S6SBR) are used to define the start of each of the six slots. The seventh register SNEBR (Slot n End Boundary register) defines the end of the nth slot. The n is programmed in the slot_no field of this register.

The number of slots would automatically be available from the SNEBR. The SNEBR contains the slot number of the last slot and the end boundary of the same. Each slot can separately be enabled or disabled by using the slot_en bit in the S*n*FCR. When the start slot boundaries are properly defined in the SnSBR (Slot *n* Start Boundary Register) register but the slot_en bit = 0 in the corresponding S*n*FCR (Slot *n* Frame Configuration register) register, then the corresponding slot is treated as “available” but “unconfigured.” For such a slot the data/diagnostic bits all remain 0*. The “T” bit will, however, get set if a message that was supposed to come in a configured slot actually arrives in an unconfigured slot. Note that the various parameters like the Data length/CRC configuration and so on are valid ONLY for the configured slots.* Even when sampling a message in an unconfigured slot, these parameters would be taken from the previous/next configured slot only. It is assumed that no data would be sent in an unconfigured slot. If a data is received in this unconfigured slot it would never be treated as belonging to the unconfigured slot. As a special case if “S0” and “S1” are detected in different slots then a T-bit error is set.

Address: 0x019C + *i**0x1C0 (*i*=0) Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	SLOT_NO		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	SNEBT														
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1481. Slot *n* End Boundary Register (PSI5_CH0_SNEBR)

Table 1523. PSI5_CH0_SNEBR field descriptions

Field	Description
13:15 SLOT_NO	The field indicates the Slot number{1–6} for which Slot End Boundary Time is defined. Any value other than 1–6 is defaulted to 0. A value 0 here would mean that the Manchester decoder would not sample any slots. This slot number also defines the number of slots that the IP assumes to exist between 2 sync pulses.
17:31 SNEBT	Slot *n* End Boundary Time This field specifies the time in intervals of 1 µs, occurring after the rising edge of timing sync pulse, at which the slot *n* should end.

Note: The start slot boundaries in the SnSBR should be correctly programmed. If this is not done, the correct operation cannot be guaranteed. Slots occurring after the slot_no value programmed in the SNEBR are treated as being unavailable (that is, the data in all the slots occurring after the SNEBR is rejected by the Manchester decoder).

Following are the examples:

- If only three slots are required between two sync pulses, program S1SBR, S2SBR, and S3SBR with the correct value of the start boundaries of the three slots and program slot_no = 3 in the SNEBR and define the end boundary of the third slot in SNEBR. The design ignores slots 4, 5, and 6.
- For six slots, the required programming would be: program the start of each of the six slots in S1SBR, S2SBR, S3SBR, S4SBR, S5SBR, and S6SBR. Then program the end of the sixth slot in the SNEBR by putting slot_no = 6 and define the end boundary of the sixth slot in SNEBR.

60.3.2.30 Data Output Block Configuration Register (PSI5_CH0_DOBCR)

Figure 1482 shows the Data Output Block Configuration Register.

Address: 0x01A0 + *i**0x1C0 (*i*=0) Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	DATA_LENGTH						0	0	CMD_TYPE			DEFAULT_SYNC	GTM_TRIG_SEL	SP_PULSE_SEL	OP_SEL	SW_READY
W							DBR_RST	DSR_RST								
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1482. Data Output Block Configuration Register (PSI5_CH0_DOBCR)

Table 1524. PSI5_CH0_DOBCR field descriptions

Field	Description
0-5 DATA_LENGTH	Can take on values from 0 to 63 corresponding to 1bit to 64bit non standard length commands. When PSI5_DOBCR[CMD_TYPE] == 7(Non Standard), then this field controls the length of command that can be written to the DBR and the DSR registers, else the value in this field is ignored. This when DATA_LENGTH = 0 then the length of the command that can be written to DSR and the DBR is 1, when DATA_LENGTH=1 then the length of the command that can be written to DSR and the DBR is 2 and so on until a maximum of 64-bit length command when DATA_LENGTH = 63. Note that the length of the command that is programmed by using this field, is also equal to the length of the command that shifts out of the DSR register. In other words, it is equal to the number of bits, that shift out of DSR register once the DSR_RDY goes low. After these many shifts have happened, then the value corresponding to “DEFAULT_SYNC” starts getting shifted out and DSR_RDY goes as “1”.
6 DBR_RST	This is to reset and reject current content to Data Buffer Register. When this bit is written as “1” then the contents of the DBR are reset to all “0s” (if DEFAULT_SYNC= = 0) or all “1s” (if DEFAULT_SYNC == 1). As soon as content is reset the DBR would be ready for new data. Reading this bit would always return a “0” (w1s). Writable to 1 in one shot. This bit is writable in the CONFIG and the Normal modes.
7 DSR_RST	This is to reset and reject current content to Data Shift Register.When this bit is written as “1” then the contents of the DSR are reset to all “0s” (if DEFAULT_SYNC= = 0) or all “1s” (if DEFAULT_SYNC == 1). As soon as content is reset the DSR would be ready for new data.Reading this bit would always return a “0” (w1s). Writable to 1 in one shot. This bit is writable in the CONFIG and the Normal modes.
8-10 CMD_TYPE	These 3 bits indicate the type of command that needs to be transmitted during the ECU to sensor communication. *Table 1525* is a brief description of the same.
11 DEFAULT_SYNC	When this bit is set to “0” then the default value of DSR and DBR registers are all “0s”; when this bit is set to 1 then the default value of DBR and DSR registers are all “1s”. This value is used as the value of the default sync pulses shifted at the output path when the DSR_RDY == 1.
12 GTM_TRIG_SEL	GTM event triggered/internal sync pulse generator selection as shift clock for the DSR. 1 GTM event shift triggered 0 Internal sync pulse generator shift triggered
13 SP_PULSE_SEL	Selects the source for the short pulse PWM module: 1 SP module directly gets input from the GTM_event/internal sync pulse generator 0 SP module gets the data from the DSR

Table 1524. PSI5_CH0_DOBCR field descriptions (continued)

Field	Description
14 OP_SEL	This bit selects the driving source of the "ipp_do_psi5_sdout" port. 0 The sync pulse generator select as per the PSI5_DOBCR[GTM_TRIG_SEL] 1 PWM output
15 SW_READY	When this bit is written to 1 the transfer from DBR to DSR automatically happens as soon as DSR_RDY becomes 1. When this bit is kept to 0 then the transfer from DBR to DSR will remain pending. Once the software makes this bit as "1" and DSR_RDY also goes as "1" the transfer to DSR takes place. However, note that when this bit is 0, and the DSR_RDY becomes 1, still the transfer to the DSR will NOT happen. Only when this bit is a 1, will the transfer happen. This bit can be written in CONFIG and the Normal modes.

Table 1525. CMD_TYPE details

CMD_TYPE	PSI5 Frame	Effect on registers	Writable Register lengths
"000" - 0	Short Frame with 31 "1s" as the start condition(1)	Command can be written to DPR for auto calculation of stuff/start/CRC bits. Can always be overwritten in DSR/DBR registers provided DSR_RDY == 1 or DBR_RDY == 1 respectively	6 bits in DPR 15 bits(2) in DBR/DSR.
"001" - 1	Short Frame with 5 "0s" as the start condition(1)	-do-	-do-
"010" - 2	Long Frame with 31 "1s" as the start condition(1)	-do-	16 bits in DPR. 29 bits(2) in DBR/DSR.
"011" - 3	Long Frame with 5 "0s" as the start condition(1)	-do-	-do-
"100" - 4	X-Long Frame with 31 "1s" as the start condition(1)	-do-	22 bits in DPR. 37 bits(2) in DBR/DSR
"101" - 5	X-Long Frame with 5 "0s" as the start condition(1).	-do-	-do-
"110" - 6	XX-Long	-do-	24 bits in DPR. 43 bits(2) in DBR/DSR
"111" - 7	Non Standard Length.	Command can only be written to DSR and DBR registers provided DSR_RDY == 1 or DBR_RDY == 1 respectively	Programmable from 1 to 64 bits in DBR/DSR, depending on the value of PSI5_DOBCR[DATA_LENGTH] field.

1. The start condition (31 "1s" or 5 "0s") is automatically taken care by the hardware and should NOT be written to the registers as part of the command; neither is this visible in any register.
2. This is also the length of the command, that will be shifted out during the ECU to Sensor communication, irrespective of how many "actual" number of bits are written to DBR/DSR.

Table 1526. DOBCR bit configurations and the related output states

Output path states	GTM_TRIG_SEL	SP_PULSE_SEL	OP_SEL
State1(1)	0	1	1
State2(1)	0	0	1
State3(1)	1	1	1
State4(1)	1	0	1
State5(1)	1	x	0

1. These states are shown in *Section 60.8.2: Valid states for integrated sync pulse generator*. In State5 the pulse width and period is to be controlled directly from the GTM. In this state, the PW0/1D registers have no effect on the Pulse Width Modulation, when observed on the "sdout" pin.

Note: Refer to Figure 1516 for the details about the above bits.

60.3.2.31 Manchester Decoder Disable Offset Register (PSI5_CH0_MDDIS_OFF)

The Manchester decoder Disable Offset register (PSI5_MDDIS_OFF) defines the time in intervals of 1 µs, for which the Manchester decoder remains disabled after the falling edge of the sync pulse. Using this register, this time can be programmed from 0 µs to 128 µs. This works on the 1 µs clock from the CGM.

Address: 0x01A2 + *i**0x1C0 (*i*=0) Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	MDDIS_OFF						
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1483. Manchester Decoder Disable Offset Register (PSI5_CH0_MDDIS_OFF)

Table 1527. PSI5_CH0_MDDIS_OFF field descriptions

Field	Description
9:15 MDDIS_OFF	These 7 bits can be used to program the time for which the Manchester decoder remains disabled. This offset is added AFTER the falling edge of the sync pulse. Thus the total time for which the Manchester decoder remains disabled = TsyncH + MDDIS_OFF where TsyncH is the high time for the sync pulse. Thus TsyncH = Pulse_Width0 or Pulse_Width1 as the case maybe, depending on whether a "0" is being pulse modulated or a "1". Note that the Manchester decoder will ALWAYS remain disabled during the high time of the Sync Pulse. Using this register it is possible to disable the Manchester decoder BEYOND the Sync Pulse High Time as well. *Figure 1521* shows the "MDDIS_OFF" definition.

60.3.2.32 Pulse Width for Data Bit 0 Register (PSI5_CH0_PW0D)

Figure 1484 shows the Pulse Width for Data Bit 0 Register.

Address: 0x01A4 + *i**0x1C0 (*i*=0) Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	Pulse_Width0						
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1484. PSI5_Pulse Width for Data Bit 0 Register (PSI5_CH0_PW0D)

Table 1528. PSI5_CH0_PW0D field descriptions

Field	Description
9:15 Pulse_Width0	This defines the width (in µs) of data value ’0’ to be sent from Data Output Register. It is the number of clock cycles (1µs clock) counted up to width “Pulse_Width0”; starting as soon as trigger appears from the ISPG (Internal Sync Pulse Generator) or the GTM. Can take max value of 127. When, intending to use the” tooth gap method” of PSI5 command transmission, the Pulse_Width0 should be configured to 0. *Figure 1521* shows the definition of “Pulse_Width0”.

60.3.2.33 Pulse Width for Data Bit 1 Register (PSI5_CH0_PW1D)

Figure 1485 shows the Pulse Width for Data Bit 1 Register.

Address: 0x01A6 + *i**0x1C0 (*i*=0) Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	Pulse_Width1						
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1485. PSI5_Pulse Width for Data Bit 1 Register (PSI5_CH0_PW1D)

Table 1529. PSI5_CH0_PW1D field descriptions

Field	Description
9:15 Pulse_Width1	This defines the width (in µs) of data value ’1’ to be sent from Data Output Register. It is the number of clock cycles (1µs clock) counted up to width “Pulse_Width1”; starting as soon as trigger appears from the ISPG(Internal Sync Pulse Generator) or the GTM. Can take max value of 127. *Figure 1521* shows the definition of “Pulse_Width1”.

Note: When using the “tooth gap method” of PSI5 command transmission, the pulse width is NOT disabled. In this case, PSI5_PW0D has to be programmed as 0. PSI5_PW1D has to have the value of the length of the pulse desired when a 1 has to be transferred.

60.3.2.34 Counter Target Pulse Register (PSI5_CH0_CTPR)

Figure 1486 shows the Counter Target Pulse Register.

Address: 0x01A8 + *i**0x1C0 (*i*=0) Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	CTPR															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1486. Counter Target Pulse Register (PSI5_CH0_CTPR)

Table 1530. PSI5_CH0_CH0_CTPR field descriptions

Field	Description
0:15 CTPR	Counter Target Pulse Register This is the target counter value. Once the CTC (channel target counter) reaches the value of CTPR this CTC is reset. CTC is a 16-bit free running counter that starts/stops depending on the CTC_GED (in GCR) bit or the CTC_ED bit (in PCCR). It gets reset initially when its value reaches CIPR and subsequently when its value reaches CTPR. The fact that which out of CTC_GED or the CTC_ED bit enables/disables it, is dependent on the CTC_GED_SEL bit in the PCCR. When CTC_GED/CTC_ED == 0 then the CTC is reset. The minimum value of CTPR is 6. Values less than 6 will be defaulted to 0 and the CTPR counter will not start. The value in CTPR indicates the time period of the internally generated sync pulses. The un-modulated (No PWM) internally generated sync pulses have a high time of 4-sp_ts_clk cycles, that is, the duty cycle of the un-modulated internally generated sync pulses is (4/|CTPR| *100). The 4 sp_ts_clk cycles is just the nascent value of the pulse that, actually triggers the PWD counters Note that this gives one more possible state, in addition to the states shown in *Section 60.8.2: Valid states for integrated sync pulse generator*. This additional state allows the output of the internal sync pulse generator to be made directly available at the output, without being pulse modulated. The CTPR describes ONLY the SYNC pulse period and not the sync pulse length, which is governed by the PWD registers, except in State5 where in the GTM governs these parameters.

Note: Refer to Section 60.8.1 for details about the Sync Pulse generation and coordination.

60.3.2.35 Counter Initialize Pulse Register (PSI5_CH0_CIPR)

Figure 1487 shows the Counter Initialize Pulse Register.

Address: 0x01AA + *i**0x1C0 (*i*=0) Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	CIPR															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1487. Counter Initialize Pulse Register (PSI5_CH0_CIPR)

Table 1531. PSI5_CH0_CIPR field descriptions

Field	Description
0:15 CIPR	Counter initialization pulse register This is the value at which the CTC gets reset the first time, after which it is enabled using CTC_GED (in GCR) or the CTC_ED bit (in PCCR). All subsequent resets are done when the CTC value reaches CTPR. The CIPR is used to set the offset time between the start of sync pulses between two different PSI5 channels, when the CTC of both of these channels are started simultaneously using CTC_GED. The offset between any two PSI5 channels would be |CIPR_channel1 - CIPR_channel2| sp_ts_clk cycles. Thus if channel 1 has a CIPR = 0 while channel2 has CIPR = 10, then the offset would be 10 sp_ts_clk cycles.

Note: The CIPR resolution is in terms of the time period of the "sp_ts_clk" ($T_{sp_ts_clk}$). Therefore the total "CIPR" time is "CIPR$T_{sp_ts_clk}$", beginning from when the CIPR counters are enabled using the sync pulse trigger bits(PSI5_PCCR[CTC_ED] or PSI5_GCR[CTC_GED]). Due to internal Flop synchronization, this time would not exactly be CIPR*$T_{sp_ts_clk}$ but would have an inaccuracy of 1$T_{sp_ts_clk}$. A further factor of $T_{ipg_clk_psi5}$ is also added to this inaccuracy.*
The CTPR resolution is also in terms of the "sp_ts_clk". Therefore the total "CTPR" time is "CTPR$T_{sp_ts_clk}$". However when measured on the "sdout" pin of the IP this time would not exactly be "CTPR*$T_{sp_ts_clk}$". Due to internal Flop synchronization on the 1Mhz clock, an in accuracy of "1μs" would be added. A further factor of $T_{ipg_clk_psi5}$ is also added to this inaccuracy. Hence the "CTPR" period measured on the "sdout" pin, can range from (CTPR*$T_{sp_ts_clk}$) -1μs -$T_{ipg_clk_psi5}$ to (CTPR*$T_{sp_ts_clk}$) +1μs + $T_{ipg_clk_psi5}$. Refer to Figure 1521 for more details about this inaccuracy.*

Refer to Section 60.8.1 for details about the Sync Pulse generation and coordination.

60.3.2.36 Data Preparation Register Low (PSI5_CH0_DPRL)

Figure 1488 shows the Data Preparation Register Low.

Address: 0x01AC + *i**0x1C0 (*i*=0) Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	DPR							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	DPR															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1488. Data Preparation Register Low (PSI5_CH0_DPRL)

Table 1532. PSI5_CH0_DPRL field descriptions

Field	Description
8:31 DPR	DPR[23:0] are the 24-bits of the DPR register used for writing the variable length, standard ECU-to-Sensor commands, comprising of the address, data and other fields. Note that the IP is transparent to the arrangement of the address, data and other fields and it treats the bits in these fields identically for the purpose of CRC calculation and transmission. This register is writable only when using the standard length PSI5 frames. For details about CRC calculation refer to *Section 60.4.3.4: Data transmission*. The writable bits in this register are governed by the PSI5_DOBCR[CMD_TYPE] bit fields as per *Table 1533*. Note that the unwritable bits in this register field are always read as “0s”. This register has to be written by the CPU.These bits can be written only in the Normal mode.

Table 1533. DPR details

PSI5_DOBCR[CMD_TYPE]	Standard PSI5 FRAME Types	Writable bits in PSI5_DPRL
“0” or “1”	Short Frame	6(DPR[5:0])
“2” or “3”	Long Frame	16(DPR[15:0])
“4” or “5”	X-Long Frame	22(DPR[21:0])
“6”	XX-Long Frame	24(DPR[23:0])
“7”	Not Available for write	Not Available for write

Note: This register is the lower register used for writing the standard data and the address commands and is to be used for writing standard length ECU-to-sensor commands. The CRC/stuff/start bits are automatically appended by the hardware before the command is transferred to DBR. The commands to this register are accepted ONLY when PSI5_GISR[DPR_RDY] == 1. Whenever PSI5_GISR[DPR_RDY] == 1, then each bit of this register defaults to 0. Writing to this register when PSI5_GISR[DPR_RDY] == 0 will cause the PSI5_GISR[IS_PROW] status bit to be set and the generation of an error interrupt, if it is enabled. Further, trying to write to this register when PSI5_DOBCR[CMD_TYPE] = “7” results in the setting of the PSI5_GISR[IS_PROW] bit and the resulting write is rejected.

60.3.2.37 Data Preparation Register High (PSI5_CH0_DPRH)

Address: 0x01B0 + *i**0x1C0 (*i*=0) Access: User read/write

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

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

Figure 1489. Data Preparation Register High (PSI5_CH0_DPRH)

Table 1534. PSI5_CH0_DPRH field descriptions

Field	Description
0:31	This field is reserved This register is always read as 0s. Writing to this register does not have any effect.

60.3.2.38 Data Buffer Register Low (PSI5_CH0_DBRL)

Figure 1490 shows the Data Buffer Register Low.

Address: 0x01B4 + *i**0x1C0 (*i*=0) Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	DBR															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	DBR															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1490. Data Buffer Register Low (PSI5_CH0_DBRL)

Table 1535. PSI5_CH0_DBRL field descriptions

Field	Description
0:31 DBR	This register contains the Lower 32 bits(DBR[31:0]) of the max-64 bit length command (DBR[63:0]). The higher bits, DBR[63:32] are contained in the DBRH register. These fields can automatically be updated by the hardware (when the CRC/stuff/start bits are appended to the data in DPRL) or these fields can be written by the CPU (when PSI5_GISR[DBR_RDY] == 1 in the Normal mode). Hardware update occurs when PSI5_GISR[DPR_RDY]== 0 and PSI5_GISR[DBR_RDY] == 1. It cannot be written in Config mode. Depending on the PSI5_DOBCR[CMD_TYPE] value this register can have different data assignments as describe in *Table 1536*. The tables (except when PSI5_DOBCR[CMD_TYPE] = “7”) show the bit assignment of the DBR[31:0] when these bits are automatically updated by the hardware. When PSI5_DOBCR[CMD_TYPE] == “7”), then only the CPU can write to these registers.

Table 1536. DBRL[DBR] details

PSI5_DOBCR [CMD_TYPE]	DBR bits	Bit Assignment
“7” (Non Standard Length)	DBR[31:0]]	Variable length of writable bits, as per the length programmed in the PSI5_DOBCR[DATA_LENGTH]
“6”(XX Long)	DBR[31:30]	DPR[19:18]
	DBR[29]	1’b0(stuff)
	DBR[28:23]	DPR[17:12]
	DBR[22]	1’b0(stuff)
	DBR[21:16	DPR[11:6]
	DBR[15]	1’b0(stuff)
	DBR[14:9]	DPR[5:0
	DBR[8:0]	“011111110”
“4” or “5” (X-Long)	DBR[31]	1’b1(stuff)
	DBR[30:28]	DPR[20:18]
	DBR[27]	1’b1(stuff)
	DBR[26:24]	DPR[17:15]
	DBR[23]	1’b1(stuff)
	DBR[22:20]	DPR[14:12]
	DBR[19]	1’b1(stuff)
	DBR[18:16]	DPR[11:9]
	DBR[15]	1’b1(stuff)
	DBR[14:12]	DPR[8:6]
	DBR[11]	1’b1(stuff)
	DBR[10:8]	DPR[5:3]
	DBR[7]	1’b1(stuff)
	DBR[6:4]	DPR[2:0]
	DBR[3]	1’b1(stuff)
	DBR[2:0]	“010”(start seq)

Table 1536. DBRL[DBR] details (continued)

PSI5_DOBCR [CMD_TYPE]	DBR bits	Bit Assignment
“2” or “3” (Long)	DBR[31:29]	“000”
	DBR[28]	CRC[0]
	DBR[27]	1’b1(stuff)
	DBR[26:24]	{CRC[1],CRC[2],DPR[15]}
	DBR[23]	1’b1(stuff)
	DBR[22:20]	DPR[14:12]
	DBR[19]	1’b1(stuff)
	DBR[18:16]	DPR[11:9]
	DBR[15]	1’b1(stuff)
	DBR[14:12]	DPR[8:6]
	DBR[11]	1’b1(stuff)
	DBR[10:8]	DPR[5:3]
	DBR[7]	1’b1(stuff)
	DBR[6:4]	DPR[2:0]
	DBR[3]	1’b1(stuff)
	DBR[2:0]	“010”(start seq)
“0” or “1” (Short)	DBR[31:15]	0
	DBR[14:12]	{CRC[0],CRC[1],CRC[2]}[
	DBR[11]	1’b1(stuff)
	DBR[10:8]	DPR[5:3]
	DBR[7]	1’b1(stuff)
	DBR[6:4]	DPR[2:0]
	DBR[3]	1’b1(stuff)
	DBR[2:0]	“010”(start seq)

This register is CPU writable only in Normal mode.

Note: When DEFAULT_SYNC == 0 then default value of this register is 00000000. When DEFAULT_SYNC== 1 then the default value of this register is FFFFFFFF.Note that the default value is different from the reset value. The reset value of this register is always “00000000”. The default value is loaded once the IP enters the NORMAL mode.

Further, the commands to this register are accepted ONLY when PSI5_GISR[DBR_RDY] == 1. Whenever PSI5_GISR[DBR_RDY] == 1 then each bit of this register defaults to DEFAULT_SYNC value. Writing to this register when PSI5_GISR[DBR_RDY] == 0 will set the status bit PSI5_GISR[IS_BROW] and generate an error interrupt, if it is enabled.

60.3.2.39 Data Buffer Register High (PSI5_CH0_DBRH)

Figure 1491 shows the Data Buffer Register High.

Address: 0x01B8 + *i**0x1C0 (*i*=0) Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	DBR															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	DBR															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1491. Data Buffer Register High (PSI5_CH0_DBRH)

Table 1537. PSI5_CH0_DBRH field descriptions

Field	Description
0:31 DBR	This register contains the Upper 32 bits(DBR[63:32]) of the max-64 bit length command (DBR[63:0]). The lower bits, DBR[31:0] are contained in the DBRL register. These fields can automatically be updated by the hardware (when the CRC/stuff/start bits are appended to the data in DPRL) or these fields can be written by the CPU (when PSI5_GISR[DBR_RDY] == 1 in the Normal mode). Hardware update occurs when PSI5_GISR[DPR_RDY]== 0 and PSI5_GISR[DBR_RDY] == 1. It cannot be written in Config mode. Depending on the PSI5_DOBCR[CMD_TYPE] value this register can have different data assignments as described in the below tables. Note that when PSI5_DOBCR[CMD_TYPE] == “7”), then only the CPU can write to these registers.

Table 1538. DBRH[DBR] details

PSI5_DOBCR [CMD_TYPE]	DBR bits	Bit Assignment
“7” - Non Standard Frame lengths	DBR[63:32]]	Variable length of writable bits, as per the length programmed in the PSI5_DOBCR[DATA_LENGTH]
“6” - XX Long	DBR[63:43]	All 0s
	DBR[42:37]	{CRC[5],CRC[4],CRC[3],CRC[2],CRC[1],CRC[0]}
	DBR[36]	1’b0(stuff)
	DBR[32:35]	DPR[23:20]

Table 1538. DBRH[DBR] details (continued)

PSI5_DOBCR [CMD_TYPE]	DBR bits	Bit Assignment
"4" or "5" -X Long	DBR[63:37]	All 0s
	DBR[36]	CRC[0]
	DBR[35]	1'b1((stuff)
	DBR[34:32]	{CRC[1],CRC[2],DPR[21]}
"3" or "2"(Long) OR "1" or "0" (Short)	DBR[63:32]	All 0s

This register is CPU writable only in Normal mode.

Note: When DEFAULT_SYNC == 0 then the default value of this register is 00000000. When DEFAULT_SYNC == 1 then the default value of this register is FFFFFFFF. Note that the default value is different from the reset value. The reset value of this register is always "00000000". The default value is loaded once the IP enters the NORMAL mode.

Further, the commands to this register are accepted ONLY when PSI5_GISR[DBR_RDY]== 1. Whenever PSI5_GISR[DBR_RDY] == 1 then each bit of this register defaults to DEFAULT_SYNC value. Writing to this register when PSI5_GISR[DBR_RDY] == 0 will set the status bit PSI5_GISR[IS_BROW] and generate an error interrupt, if it is enabled.

60.3.2.40 Data Shift Register Low (PSI5_CH0_DSRL)

Figure 1492 shows the Data Shift Register Low. This register is CPU writable only in Normal mode.

Address: 0x01BC + *i**0x1C0 (*i*=0) Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	DSR															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	DSR															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1492. Data Shift Register Low (PSI5_CH0_DSRL)

Table 1539. PSI5_CH0_DSRL field descriptions

Field	Description
0:31 DSR	These bits can be updated by the hardware or can be written by the CPU. The number of accessible bits in DSR are always equal to the number of accessible bits in DBR, which in turn depend on the value of PSI5_DOBCR[*CMD_TYPE*] register bits.When these bits are updated by the hardware then there is a one to one correspondence between the bit positions in this register and the bit positions in the DBRL. Thus when updated by the hardware, then DSR[31:0] = DBR[31:0]. Hardware update occurs when PSI5_GISR[DSR_RDY]== 1 and PSI5_GISR[DBR_RDY] == 0 and PSI5_DOBCR[SW_READY] == 1. Note that if PSI5_DOBCR[SW_READY] == 0 then the update of the data from the DBR to DSR will NOT happen even though PSI5_GISR[DSR_RDY] == 1. These bits can be written by the CPU in the Normal mode, when PSI5_GISR[DSR_RDY] == 1'b1. It cannot be written in Config mode.

This register is CPU writable only in Normal mode.

Note: When DEFAULT_SYNC == 0 then default value of this register is 00000000. When DEFAULT_SYNC == 1 then the default value of this register is FFFFFFFF. Note that the default value is different from the reset value. The reset value of this register is always "00000000". The default value is loaded once the IP enters the NORMAL mode.

Further, the commands to this register are accepted ONLY when PSI5_GISR[DSR_RDY] == 1. Whenever PSI5_GISR[DBR_RDY] == 1 then each bit of this register defaults to DEFAULT_SYNC value. Writing to this register when PSI5_GISR[DSR_RDY] == 0 will set the status bit GISR[IS_DSROW] and generate an error interrupt, if it is enabled. Note that for the hardware update of DSR with DBR contents, additional condition PSI5_DOBCR[SW_READY] == 1, should also be satisfied in addition to PSI5_GISR[DSR_RDY] == 1.

60.3.2.41 Data Shift Register High (PSI5_CH0_DSRH)

Figure 1493 shows the Data Shift Register High.

Address: 0x01C0 + *i**0x1C0 (*i*=0) Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	DSR															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	DSR															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1493. Data Shift Register High (PSI5_CH0_DSRH)

Table 1540. PSI5_CH0_DSRH field descriptions

Field	Description
0:31 DSR	These bits can be updated by the hardware or can be written by the CPU. The number of accessible bits in DSR are always equal to the number of accessible bits in DBR, which in turn depend on the value of PSI5_DOBCR[*CMD_TYPE*] register bits. When these bits are updated by the hardware then there is a one to one correspondence between the bit positions in this register and the bit positions in the DBRH. Thus when updated by the hardware, then DSR[63:32]] = DBR[63:32]. Hardware update occurs when PSI5_GISR[DSR_RDY]== 1 and PSI5_GISR[DBR_RDY] == 0 and PSI5_DOBCR[SW_READY] == 1. Note that if PSI5_DOBCR[SW_READY] == 0 then the update of the data from the DBR to DSR will NOT happen even though PSI5_GISR[DSR_RDY] == 1. These bits can be written by the CPU in the Normal mode, when PSI5_GISR[DSR_RDY] == 1'b1. It cannot be written in Config mode.

This register is writable only in Normal mode.

Note: When DEFAULT_SYNC == 0 then the default value of this register is 00000000. When DEFAULT_SYNC == 1 then the default value of this register is FFFFFFFF. Note that the default value is different from the reset value. The reset value of this register is always "00000000". The default value is loaded once the IP enters the NORMAL mode.

Further, the commands to this register are accepted ONLY when PSI5_GISR[DSR_RDY] == 1. Whenever PSI5_GISR[DBR_RDY] == 1 then each bit of this register defaults to DEFAULT_SYNC value. Writing to this register when PSI5_GISR[DSR_RDY] == 0 will set the status bit GISR[IS_DSROW] and generate an error interrupt, if it is enabled. Note that for the hardware update of DSR with DBR contents, additional condition PSI5_DOBCR[SW_READY] == 1, should also be satisfied in addition to PSI5_GISR[DSR_RDY] == 1.

60.3.2.42 Device modes and Register bit accesses

1. The following bits are writable in all modes (Normal, Config, and Disable).
   - GCR[GLOBAL_DISABLE_REQ]
   - PCCR[PSI5_CH_CONFIG]
   - PCCR[PSI5_CH_EN]
   - Global Counter Enable/Disable (GCR[CTC_GED]). This is to allow the Sync Pulse Generator to be enabled/disabled independently among those channels that are used for staggering, irrespective of the state of the channels NOT being used for staggering.
2. The following register is writable only in Normal mode and is initialized to all 0s.
   - DPRL
3. The following registers are writable ONLY in Normal mode. Initial is at ion in Config mode depends on the value of PSI5_DOBCR[DEFAULT_SYNC] bit.When PSI5_DOBCR[DEFAULT_SYNC] == 1 in Config mode, then the below registers are initialized to all 1s in Normal mode. When PSI5_DOBCR[DEFAULT_SYNC] = 0 in Config mode, then the below registers are initialized to all 0s in Normal mode. Once in

Normal mode, these registers can then be programmed for any command that needs to be sent.

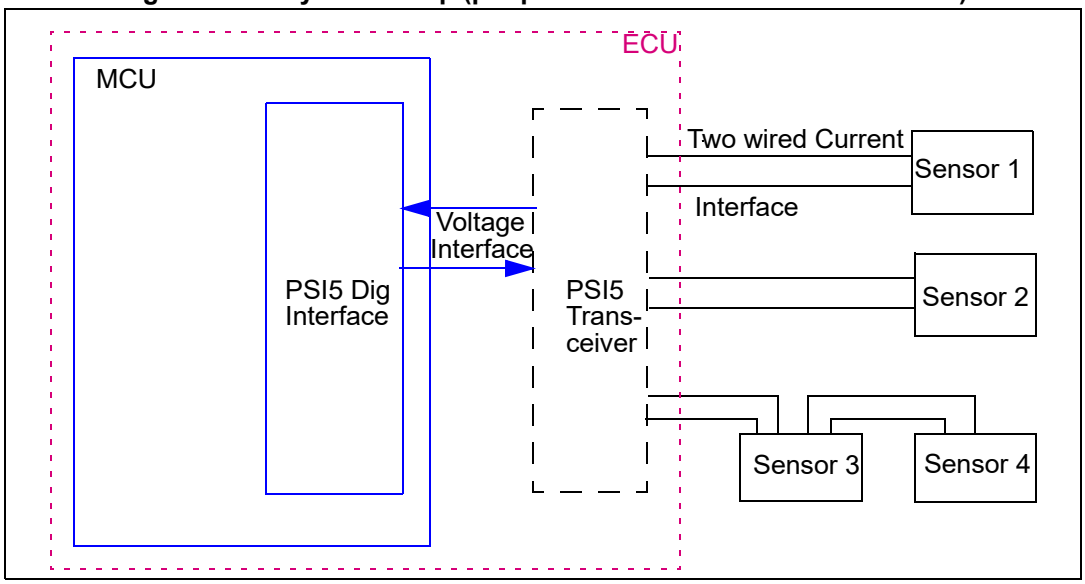
- DBRH/DBRL
- DSRH/DSRL

4. The following registers are writable in Config and Normal modes only:
   - All interrupt control registers
   - All SET registers (the one shot registers that are used to set the error flags for software drivers)
   - Channel Enable/Disable (PCCR[CTC_ED]). Using this it is possible to allow the Sync Pulse Generator to be enabled/disabled in Normal mode. The counter is reset when CTC_ED == 0
   - All w1c bits
5. Apart from these, all other register bits can be written only in Config mode, including the Data registers (PMRH/PMRL/SFRn), that, in Normal mode, are updated by the hardware only.
6. Debug mode: If the device is in Normal mode and the debug mode is enabled, then ONLY the registers in *Section 1.*,*Section 2.*,*Section 3.*,*Section 4.* of *Section 60.3.2.42: Device modes and Register bit accesses*: can be written by the debugger. Register in *Section 5.* of *Section 60.3.2.42: Device modes and Register bit accesses* will not be available for write through the debugger. In case the debugger has to change any register, then the device has to be brought in the Disable → Config mode. The debugger can, however, read any register in any mode.

60.4 Functional description

The PSI5 is a communication protocol for peripheral inertial, pressure, temperature, and position sensors. These sensors are connected to the ECU by just two wires, using the same lines for power supply and data transmission. The transceiver (present externally) provides a pre-regulated voltage to the sensors and reads in the transmitted sensor data. *Figure 1494* shows a point-to-point connection for sensors 1 and 2 and bus configuration for sensors 3 and 4.

Figure 1494. System setup (peripheral sensor connected to device)



ECU (Electronic Control Unit) is comprised of PSI5 digital interface and the PSI5 transceiver (present externally).

60.4.1 Sensor-to-ECU communication

60.4.1.1 Physical layer

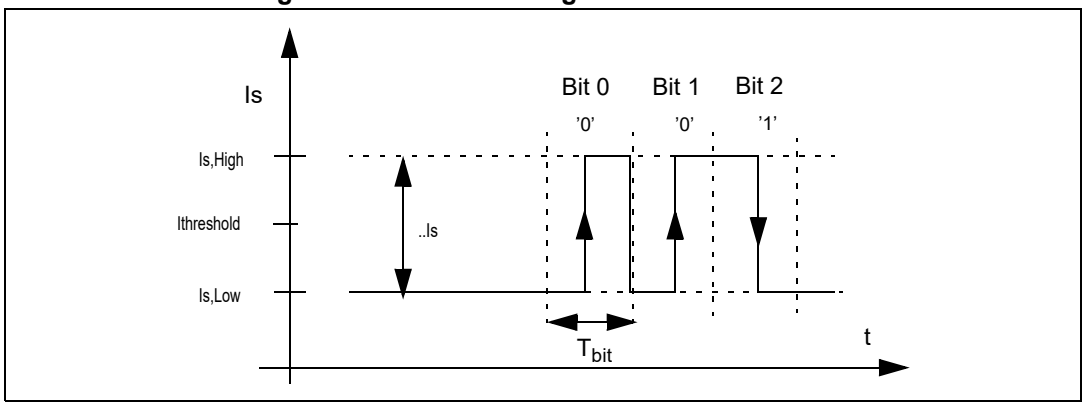
The PSI5 uses two wires to provide both power supply and data transmission to the sensors. The ECU provides a pre-regulated voltage to the sensors. Data transmission from the sensor to the ECU is done by current modulation on the power supply lines. Current oscillations are damped by the ECU.

60.4.1.2 Bit encoding

A “low” level ($I_{S,Low}$) is represented by the Normal (quiescent) current consumption of the sensor(s). A “high” level ($I_{S,High}$) is generated by an increased current sink of the sensor ($I_{S,Low}$ + ...I_S). The current modulation is detected within the PSI5 transceiver.

Manchester coding is used for data transmission. A logic 0 is represented by a rising slope and a logic 1 by a falling slope of the current in the middle of T bit.

Figure 1495. Bit encoding of data from sensor



60.4.1.3 Synchronization

For synchronized operation and bus mode, a sync pulse is generated by the PSI5 receiver. The sync pulse is output from the integrated PSI5 receiver to an external PSI5 transceiver that processes these pulses to a pre-regulated voltage level on supply lines to sensors. The voltage change is detected within the sensors.

60.4.2 Data Link Layer

The PSI5 message is received as serial Manchester-coded bits. The bit stream is passed through a Manchester decoder stage that converts it to normal NRZ format bit by bit and stores it serially in the receive register (64 bits). This block clearly identifies start (00) and end of receive frame using preprogrammed frame configuration registers.

- The Manchester-coded bits can be received at the rate of 125 Kbit/s or 189 Kbit/s (selectable by programming frame configuration registers).
- The Manchester decoder is designed to consider data with duty cycle of 30–70%.
- The bit decoding is done using an oversampling and counting algorithm with majority voting and edge recognition. (125/189 kHz) × 32, that is, (4/6.048 MHz also referred as 32× clock) clocks are supplied from a common clock generator (to be implemented in CGM) module. This clock is used for oversampling of the receive bit and thereby decoding it as normal 1 or 0 (NRZ).
- The Manchester decoder is masked (switched off) for the duration when the sync pulse is transmitted. In addition to this, it can be configured to be off for an additional time, after the falling edge of the sync pulse. The duration of this time is configurable in the MDDIS_OFF register. It uses a 2oo3 voter to recognize the encoded half bits.
- There is a 3oo5 majority voting glitch filter at the data input of the Manchester decoder.

60.4.2.1 Reception of data frames

The extended PSI5 (Power Train Applications) receive data frame consists of *p* bits containing:

- Two start bits (S1 and S2) {S1=0, S2=0}.
- One parity bit (P) with even parity (or 3 CRC bits C0, C1, and C2).
- A data region (D0.....D[*k* – 1]) with *k* = 8 to 28 bits with 1-bit granularity.
- The total length of a PSI5 frame is p = *k* + 3 data bits (in case of frames with parity bit) or p = *k* + 5 data bits (in case of frames with CRC).
- Data bits are received LSB first. The parity or CRC check bits cover the bits of the entire data region.

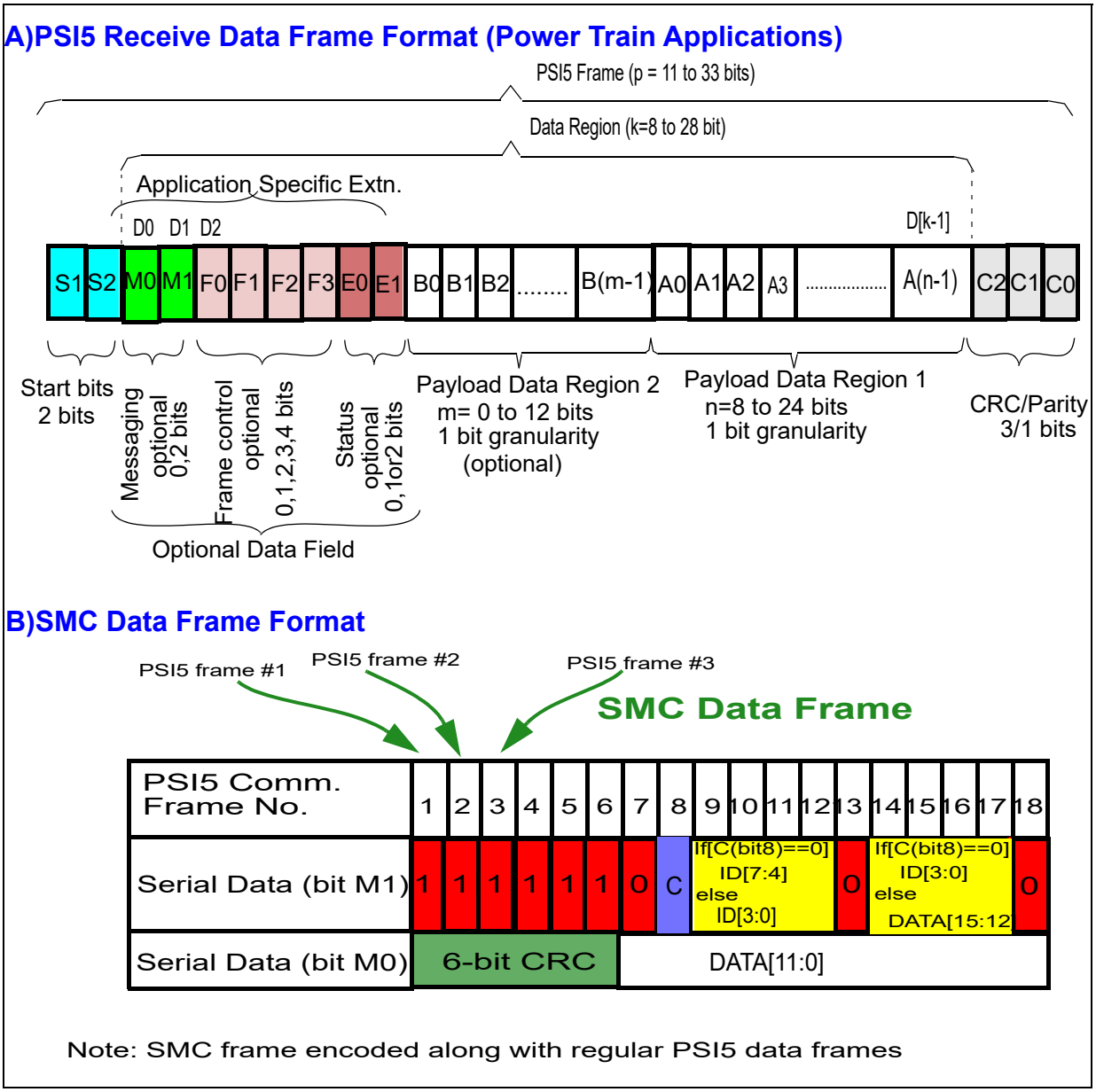
The length of the data region can vary between *k* = 8 to 28 bits (with 1-bit granularity). The data region can be split into the following fields and regions:

- The bits D0 and D1 can optionally represent Serial Messaging Channel M0, M1.
  - This Serial Messaging Channel (SMC) is used to receive initialization data during fast startup phase or slow acknowledgment/response messages (to command transmitted from ECU requesting data or diagnostic information from the sensor).

Note that the maximum length of the entire data region is 28 bits. Thus the signal payload region and the optional data fields cannot concurrently be used at their maximum length in some cases.

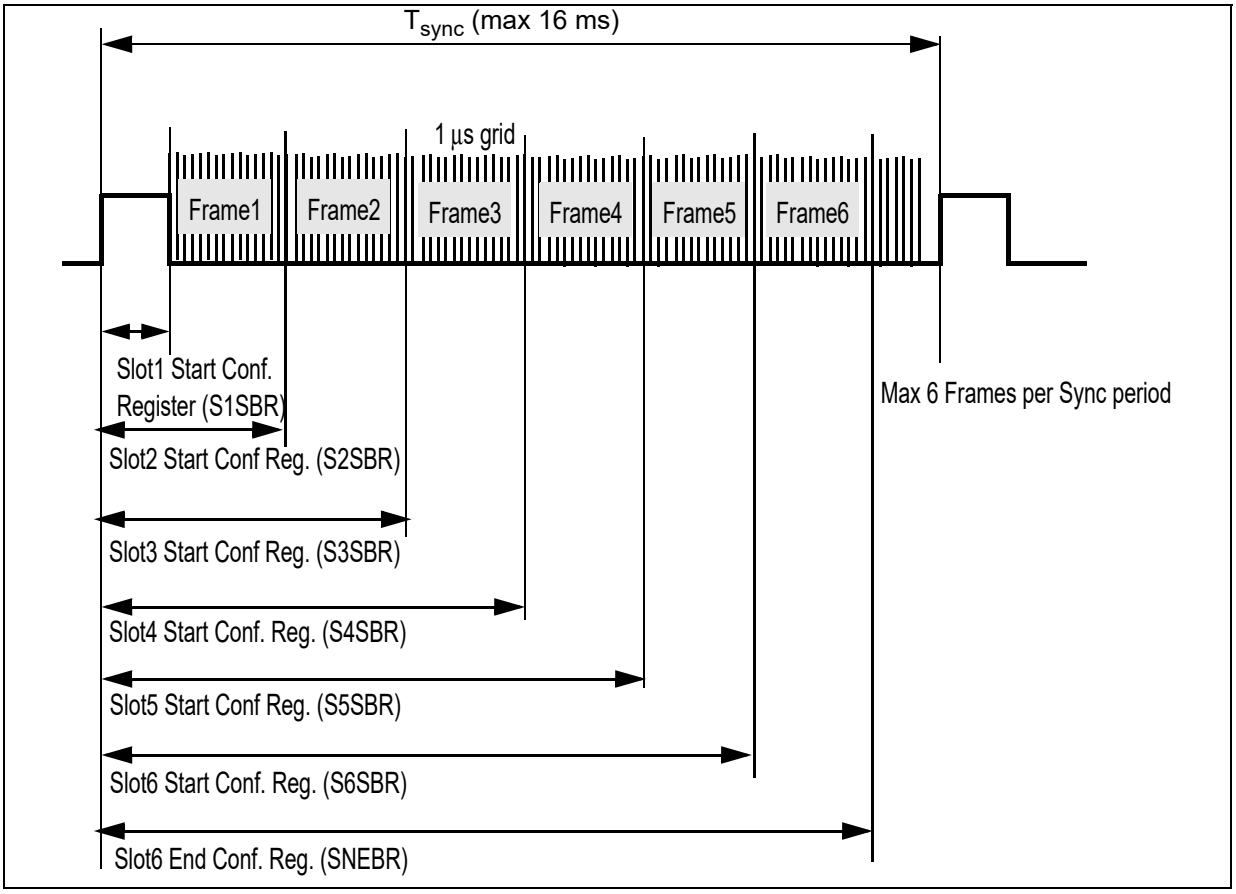
As the PSI5 module is targeted for powertrain applications, the data frame format differs from standard PSI5 frames for safety applications specified in version 1.3 of the PSI5 specifications. More specifically for supporting the version 2.0 powertrain substandard. The Powertrain data frame format and optional Serial Messaging Channel (SMC) frame format are shown in *Figure 1496*.

Figure 1496. Receive data frame format



60.4.2.2 Data Frame Configuration Control

Figure 1497. Data frame configuration and sampling



At the receive path, as many as six frame slots (reception windows) can be programmed with a start time of consecutive slots (slot1 to slot6) and end time or last slot, which can be configured through the Slot *n* Start/End Configuration Registers (*Section 60.3.2.28: Slot n Start Boundary Register (PSI5_CH0_SnSBR)* and *Section 60.3.2.29: Slot n End Boundary Register (PSI5_CH0_SNEBR)*) there are seven registers for six slots. The slot time value is with respect to timing sync pulse.

The following range is always maintained for T_{sync} (during programming of the GCR and CTPR registers) when timing (sync) pulses are generated through the Sync Pulse Generator.

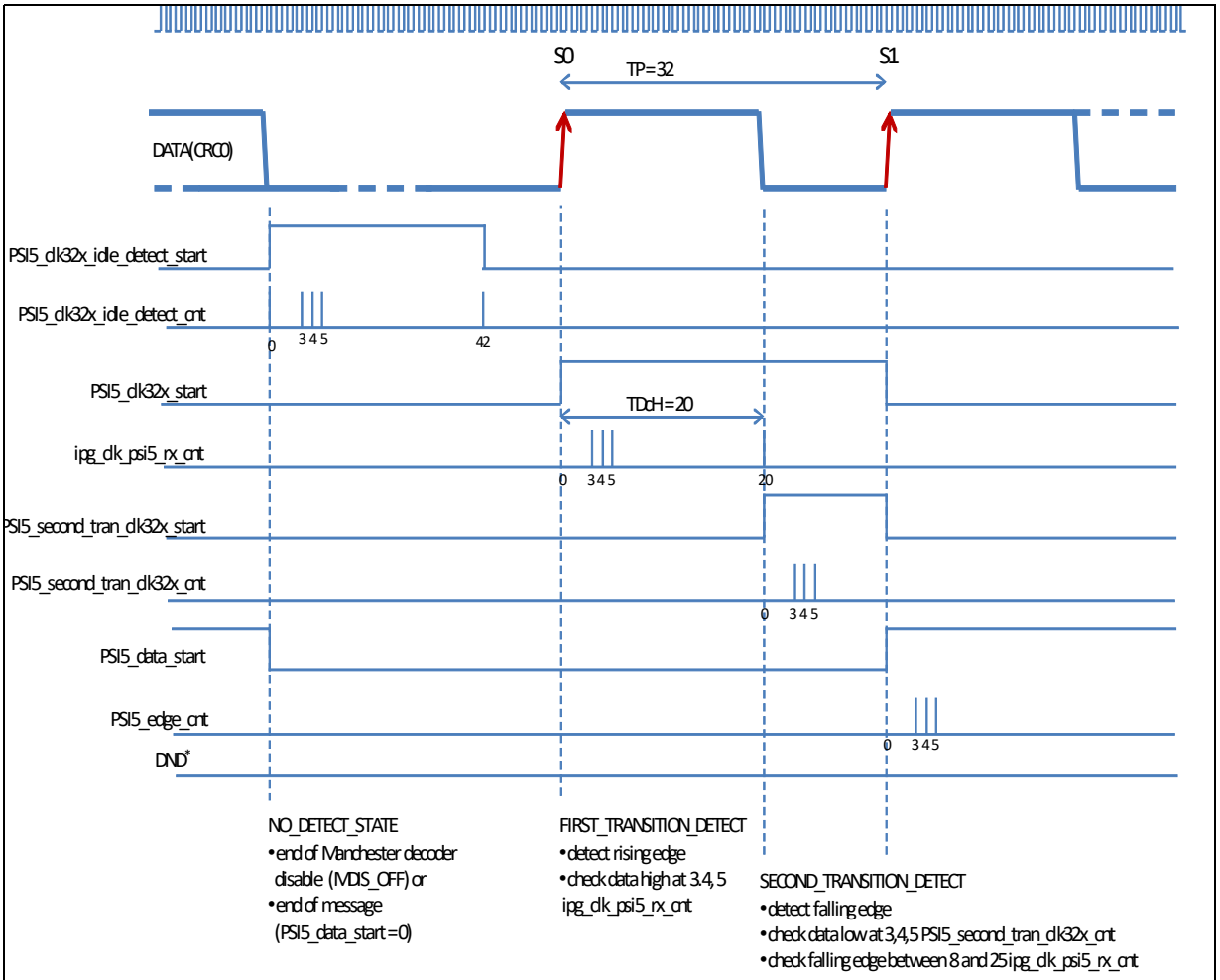
Equation 82 T_{sync} < 16 ms

For the maximum defined six frame slots, six frame configuration registers (S1FCR, S2FCR, S3FCR, S4FCR, S5FCR, and S6FCR) are also available. In these registers it is possible to configure the payload region of the expected frames, in addition to other channel-specific parameters.

60.4.2.3 Data frame handling

Figure 1498 shows the detection of the start bits and the various parameters for the same.

Figure 1498. Start Bit Detection

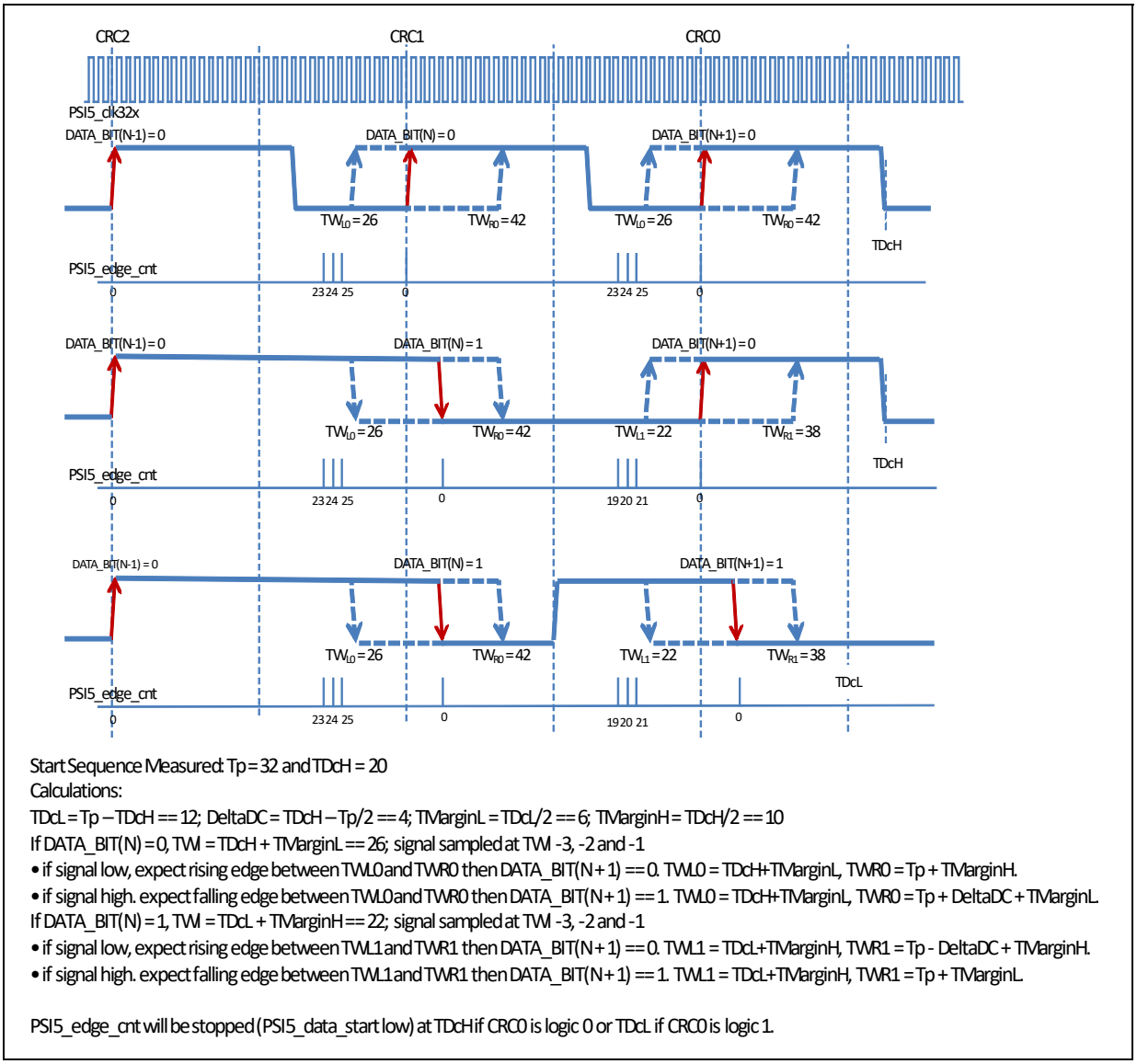


(*) If DND is set to '1' because a message has overlapped with the Manchester decoder disable phase, then the Manchester decoder disable phase will be extended till the end of the first configured slot. DND will be cleared to '0' at the end of this first slot and NO_DETECT_STATE will be entered immediately. No idle detection will be carried out (that is PSI5_clk32x_idle_detect_cnt will not be started) to ensure that a message occurring within the first bit time of slot2 will be captured.

If DND is set to '1' because a first message (MSGn) has overlapped with a second message (MSGn+1), then MSGn+1 will be ignored, that is the Manchester decoder will be disabled until the end of the slot in which MSGn+1 was expected. This will be the case whether the first transition detection of MSGn+1 occurs in the expected slot of MSGn or MSGn+1. At the end of the slot in which MSGn+1 was expected, DND will be cleared to '0' and NO_DETECT_STATE will be entered immediately. No idle detection will be carried out (that is PSI5_clk32x_idle_detect_cnt will not be started) to ensure that a message occurring within the first bit time of the slot in which the next message (MSGn+2) is expected will be captured.

Figure 1499 shows how the bits are differentiated into a “1” or a “0” based on the various margins and sample points.

Figure 1499. Bit Extraction



Start Sequence Measured: Tp = 32 and TDcH = 20
Calculations:
TDcL = Tp – TDcH == 12; DeltaDC = TDcH – Tp/2 == 4; TMarginL = TDcL/2 == 6; TMarginH = TDcH/2 == 10
If DATA_BIT(N) = 0, TWI = TDcH + TMarginL == 26; signal sampled at TWI -3, -2 and -1
- if signal low, expect rising edge between TWL0 and TWR0 then DATA_BIT(N+1) == 0. TWL0 = TDcH+TMarginL, TWR0 = Tp + TMarginH.
- if signal high. expect falling edge between TWL0 and TWR0 then DATA_BIT(N+1) == 1. TWL0 = TDcH+TMarginL, TWR0 = Tp + DeltaDC + TMarginL.

If DATA_BIT(N) = 1, TWI = TDcL + TMarginH == 22; signal sampled at TWI -3, -2 and -1
- if signal low, expect rising edge between TWL1 and TWR1 then DATA_BIT(N+1) == 0. TWL1 = TDcL+TMarginH, TWR1 = Tp - DeltaDC + TMarginH.
- if signal high. expect falling edge between TWL1 and TWR1 then DATA_BIT(N+1) == 1. TWL1 = TDcL+TMarginH, TWR1 = Tp + TMarginL.

PSI5_edge_cnt will be stopped (PSI5_data_start low) at TDcH if CRC0 is logic 0 or TDcL if CRC0 is logic 1.

60.4.2.3.1 Start of frame recognition

The start of frame recognition is as follows:

- With the Manchester decoder in NO_DETECT state, the first rising edge of the incoming data signal is captured and assumed to be the midpoint transition of the first start bit S0.
- As soon as the first rising edge (S0) on the bus is recognized, the 24-bit time stamp value is captured internally. On the rising edge of the S1 this value is stored in the data

receive register and available for processing. If a valid S1 edge is NOT recognized then this time stamp value is discarded and not stored in any register.

- The middle transition from the first start bit S0 to the second start bit S1 is assumed to be the bit duration for the rest of the bits.
- The time of the high period is measured using the 32x sample clock to enable analysis of the data signal duty cycle. The measured value is used in the data state to calculate the appropriate sample time of the first half-bit of the following data bits
- The second rising edge of the incoming data (S1) is captured and assumed to be the mid-point transition from the second start bit S1 if it appears in the allowed range of samples of the 32x sample clock.
- The time between the first and second rising edge is measured using the 32x sample clock to enable analysis of the sensor clock tolerance. The measured value is used together with the calculated duty cycle to set the window for capturing the mid-point bit transition the following data bits.
- *Figure 1498* and *Figure 1499* show the start bit and the data bit extraction mechanisms respectively, in the Manchester decoder.

60.4.2.3.2 Data state

In the data state, the Manchester decoder samples the number of data bits corresponding to the configured length of the data payload region and 3-bit CRC or 1-bit parity.

60.4.2.3.3 Message diagnostic checks

During the data state, electrical correctness of each bit in regard of Manchester encoding is checked. If an incorrect bit is detected, a 0 is written into the data payload region. Following completion of reception, correctness of the entire message is checked with the CRC and timing of the message reception.

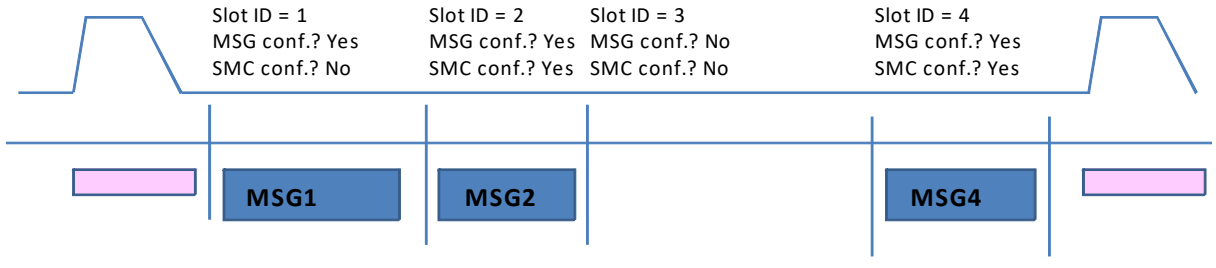
- CRC is recalculated for the data and compared with received CRC bits. The CRC comparison result is stored as a status bit, denoted as the C field in the receive register and the RAM register. Irrespective of the value of the “C” bit the, the frame is always stored in the RAM register and is never discarded.
- If at least one bit is electrically faulty (apart from M0M1), then the bit field E is set to 1 and stored as status bit in the receive register and the RAM register.
- The EM bit is set to 1 if any of M0 or M1 bits are electrically faulty. Note that the E and the EM bits can be set only when a valid start sequence has been detected.
- Set status bit field T in the receive register to 1 if the frame received is in the wrong slot (boundary violations). This bit is NOT set for asynchronous modes.
- Set status bit F if no frame is received in defined frame slot. This bit is NOT set for asynchronous modes.
- The incoming frames after the sync pulse are to be numbered from 1 to 6 (max). This is done by a 3-bit slot counter, which is incremented automatically after the end of each slot irrespective of whether a valid message is received or not. This 3-bit value is stored in the receive register and the RAM registers as the “slot_counter” field.

  T-bit Handling

The following paragraphs describe how the IP handles error cases that generate T bit errors. A number of examples are shown, including the error free case, which is used to explain the reception process.

60.4.2.3.3.1 Error Free Case

Figure 1500. Error Free Case



Message Name	Slot Counter	T-bit	F-bit	E/EM-bit	C-bit	SMC buffer	SMC data
MSG1	1	No	No	No	No	No	No
MSG2	2	No	No	No	No	2	M0/M1
MSG4	4	No	No	No	No	4	M0/M1

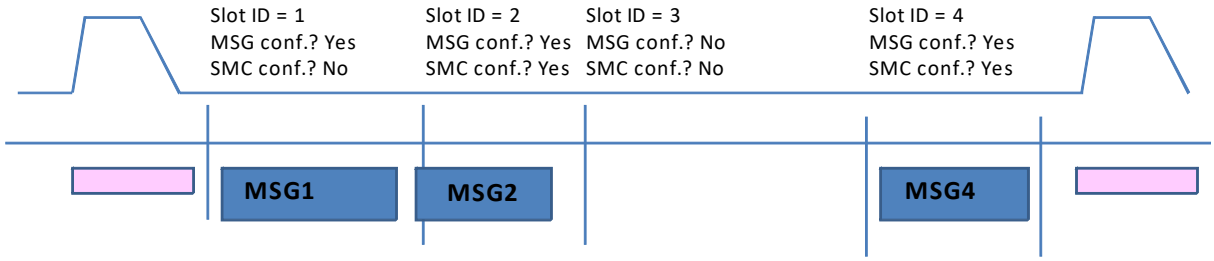
In this case, all messages are received in the correct slots.

- As soon as the Manchester decoder block is enabled (after the SYNC pulse has finished and any configured disable offset has expired, pink region) the IP checks for the first configured message and starts monitoring the output of the PSI5 transceiver.
- The IP identifies a correct start of message sequence (S0, S1)== '00', and captures the slot counter when the first rising edge of the start sequence is detected. This first rising edge detected is the mid-point transition of the Manchester encoded S0 bit of the message start sequence
- The expected number of bits configured in the message is received and decoded. Any error in the Manchester encoded bits of the message E bit is set - unless the error is in one of the first two bits and SMC message is configured, in which case the EM bit is set.
- After the final bit is received, a check is made that the message CRC or parity is correct. If the CRC or parity check is not correct, C bit is set for the message.
- Once the CRC checks have been made, the received message — along with the status bits, slot counter in which the start sequence was received, the timestamp of the message, the message data and CRC or parity is saved to the channel message FIFO.
- As soon as the received message has been stored, the idle state of the bus is verified and monitoring of the transceiver output restarts to check for the next configured message
- If the serial message channel (SMC) is configured, the first two data bits are extracted from a sequence of 7 PSI5 messages as long as no electrical error is detected in those bits (EM error). If the correct start sequence for the SMC message is detected on M1 (6 logical 1s and a logical 0) in those 7 messages, M0 and M1 from the following 11 messages are also extracted and saved in the SMC buffer allocated to the slot in which

the PSI5 message is received (in total, 18 PSI5 messages are needed to build one SMC message). When the SMC has been received, the CRC is checked and the status of the check is saved in the SMC buffer. If an EM error occurs after a start sequence has been detected and before the end of the message is reached, M0 and M1 are both saved as 0 and the rest of the message is received, at which point the CRC is checked as normal.

60.4.2.3.3.2 Message Too Early

Figure 1501. Message Too Early



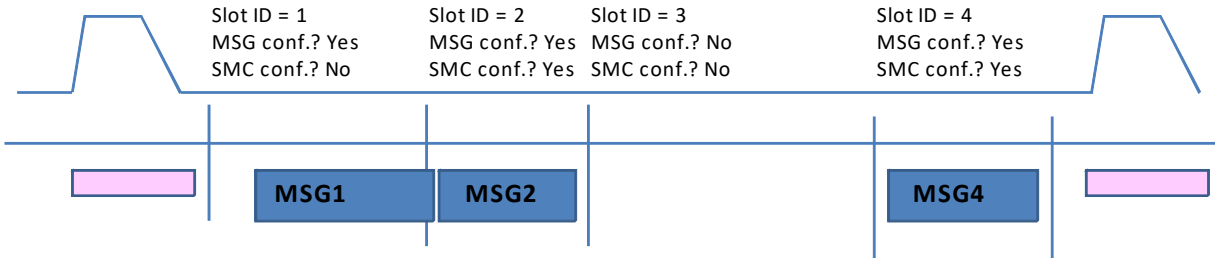
Message Name	Slot Counter	T-bit	F-bit	E/EM-bit	C-bit	SMC buffer	SMC data
MSG1	1	No	No	No	No	No	No
MSG2	1	Yes	No	No	No	2	M0/M1
MSG4	4	No	No	No	No	4	M0/M1

In this scenario, MSG2 reception starts early, before slot 2 boundary is crossed, and so the slot counter is captured as “1” and because the message starts in the wrong slot, the T bit is set. If no other errors are detected during reception, the assumption is that the correct MSG2 has been received but not entirely in its correct slot. The status of the message will include the T bit set and slot counter will be 1, so software can interpret that MSG2 was too early.

The SMC message is received correctly.

60.4.2.3.3.3 Message Too Late

Figure 1502. Message Too Late



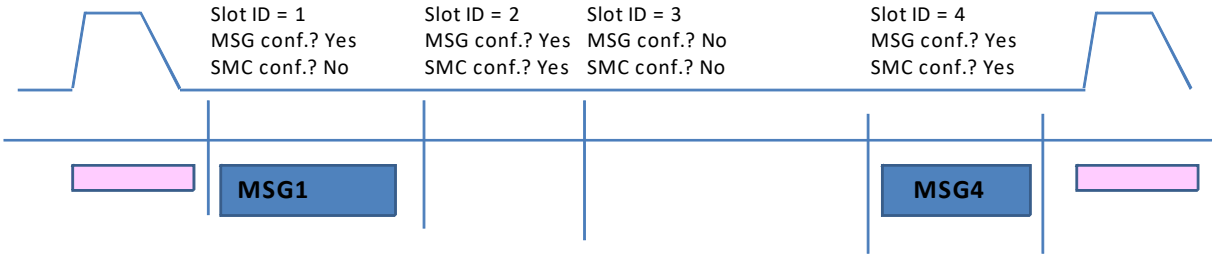
Message Name	Slot Counter	T-bit	F-bit	E/EM-bit	C-bit	SMC buffer	SMC data
MSG1	1	Yes	No	No	No	No	No
MSG2	2	No	No	No	No	2	M0/M1
MSG4	4	No	No	No	No	4	M0/M1

In this scenario, MSG1 reception starts later than expected and ends after slot 2 boundary is crossed. The slot counter is captured as “1” and because the message ends in the wrong slot, the T bit is set. If no other errors are detected during reception, the assumption is that the correct MSG1 has been received but not entirely in its correct slot. The status of the message will include the T bit set and slot counter will be “1”, so software can interpret that MSG1 was late.

The SMC message is received correctly.

60.4.2.3.3.4 Message Missing

Figure 1503. Message Missing



Message Name	Slot Counter	T-bit	F-bit	E/EM-bit	C-bit	SMC buffer	SMC data
MSG1	1	No	No	No	No	No	No
MSG2	2	No	Yes	No	No	2	0/0
MSG4	4	No	No	No	No	4	M0/M1

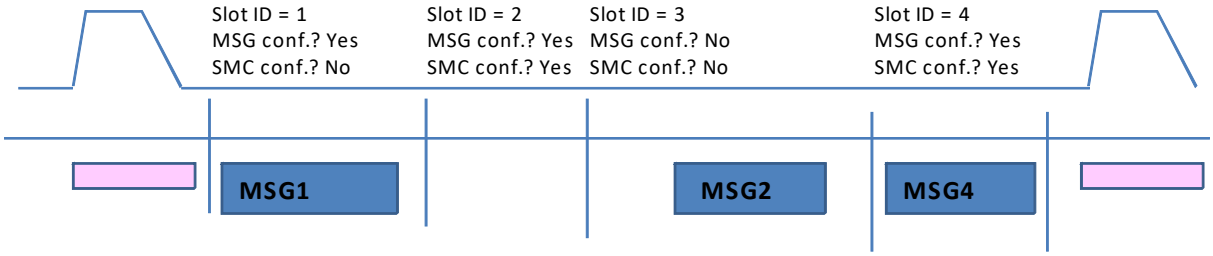
In this scenario, MSG2 is missing.

At the end of slot 2, the IP writes message buffer location 2 (MSG2 location) with slot counter 2 and the F bit set. The data and timestamp regions are filled with 0s.

The IP waits until the start of the next configured slot (slot 4) to update the SMC buffer 2 with “00”. If slot 4 was not configured, the SMC for the missing MSG2 would be updated at the end of the final slot.

60.4.2.3.3.5 Message Much Too Late (in unconfigured slot)

Figure 1504. Message Much Too Late (in unconfigured slot)



Message Name	Slot Counter	T-bit	F-bit	E/EM-bit	C-bit	SMC buffer	SMC data
MSG1	1	No	No	No	No	No	No
MSG2a	2	No	Yes	No	No	2	No update
MSG2b	3	Yes	No	No	No	2	M0/M1
MSG4	4	No	No	No	No	4	M0/M1

In this scenario, MSG2 reception does not start in its own slot, and the F bit is set for message buffer location 2 (MSG2a).

Because slot 3 is unconfigured, the IP continues to wait for MSG2 in slot 3 and does not update the SMC message buffer 2. The IP recognizes the MSG2 start condition in the normal way and updates PSI5 message buffer 3 (MSG2b), capturing slot 3 as the start slot counter and setting T bit, so that the software can recognize that MSG2 has come in slot 3.

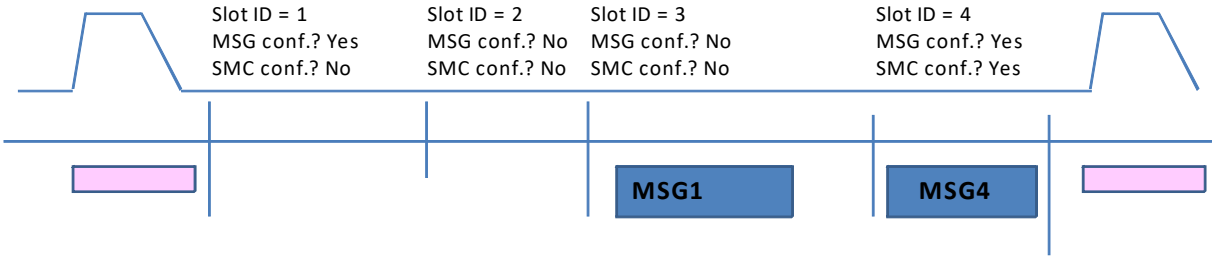
When MSG2 has been received, SMC message buffer 2 is updated at the end of slot 3. MSG4 is received normally.

Note: Any scenario where an expected message arrives in an unconfigured slot, leads to an issue when DMA access is used to read the PSI5 message buffer. This is because instead of the expected three entries in the PSI5 message buffer, there will be four, and assuming the DMA watermark is set for the expected three messages, the DMA request will be activated before MSG4 has been received.

In order to overcome this, if DMA access is required, no slots can be left unconfigured and the DMA watermark must be set to the number of slots in the SYNC period. In that way, the DMA will always be synchronized, but depending on how the 'empty' slot is configured, when a message is too late as described here, the result can be that the late message and its associated SMC content is lost.

60.4.2.3.3.6 Message Much Too Late (in unconfigured slot)

Figure 1505. Message Much Too Late (in unconfigured slot)



Message Name	Slot Counter	T-bit	F-bit	E/EM-bit	C-bit	SMC buffer	SMC data
MSG1a	1	No	Yes	No	No	No	No
MSG1b	3	Yes	No	No	No	No	No
MSG4	4	No	No	No	No	4	M0/M1

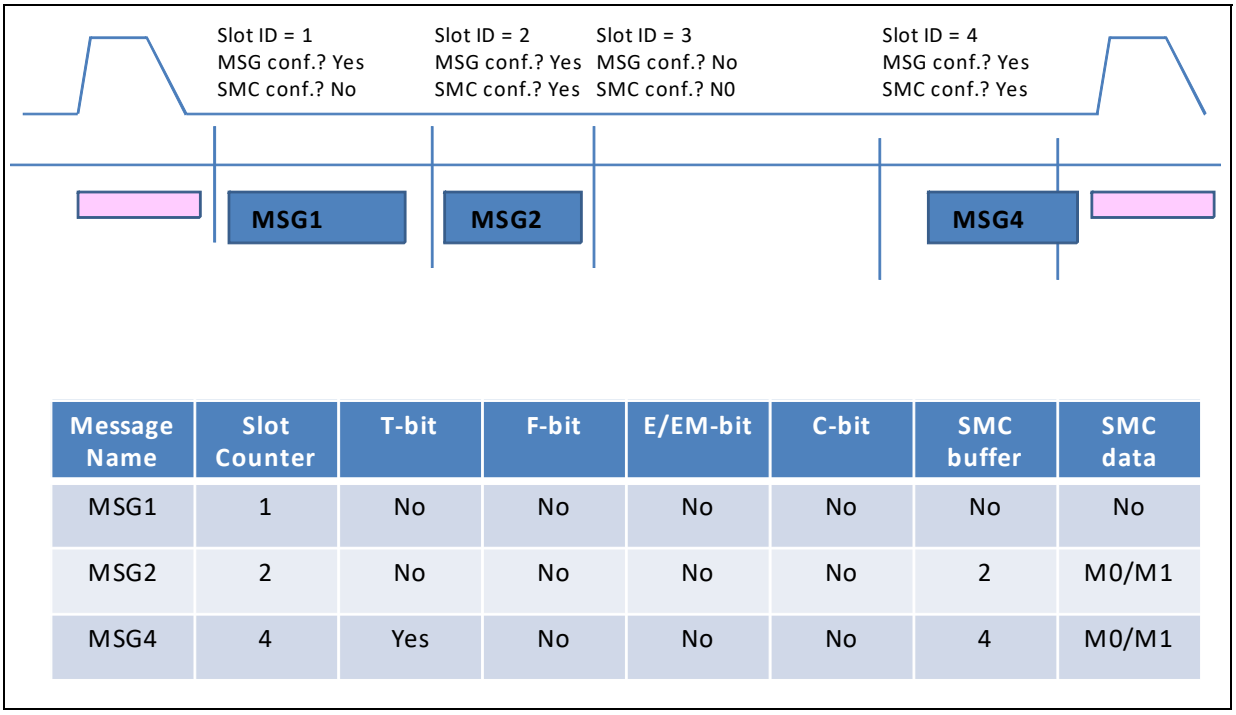
In this scenario, MSG1 is received much too late, in the second of two unconfigured slots.

When MSG1 does not arrive in slot 1, MSG1a is saved in the message buffer with F bit set. The IP waits throughout the two unconfigured slots and is received in slot 3 with slot counter 3. Because MSG1b is received completely in slot 3 it is saved in the message buffer with T bit set.

MSG4 is received normally.

60.4.2.3.3.7 Message Too Late (in last slot)

Figure 1506. Message Too Late (in last slot)



Message Name	Slot Counter	T-bit	F-bit	E/EM-bit	C-bit	SMC buffer	SMC data
MSG1	1	No	No	No	No	No	No
MSG2	2	No	No	No	No	2	M0/M1
MSG4	4	Yes	No	No	No	4	M0/M1

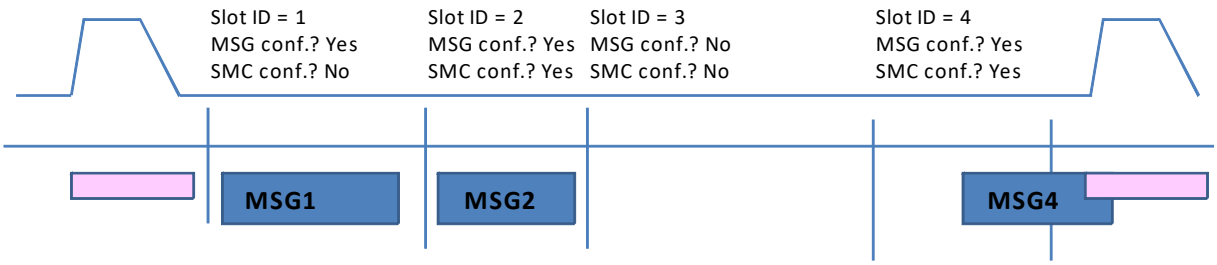
In this scenario, the last message is received late.

MSG1 and MSG2 are received error free.

MSG4 is late with the start sequence identified in the correct slot 4, with slot counter 4 captured. The end of the message occurs after the end of the final slot, but before the Manchester decoder is disabled by the SYNC pulse on the bus. The message is received correctly with the T bit set.

60.4.2.3.3.8 Message Too Late (in last slot)

Figure 1507. Message Too Late (in last slot)



Message Name	Slot Counter	T-bit	F-bit	E/EM-bit	C-bit	SMC buffer	SMC data
MSG1	1	No	No	No	No	No	No
MSG2	2	No	No	No	No	2	M0/M1
MSG4	4	Yes	No	Yes	Yes	4	M0/M1 or 0/0

In this scenario, the last message is received late in the last slot.

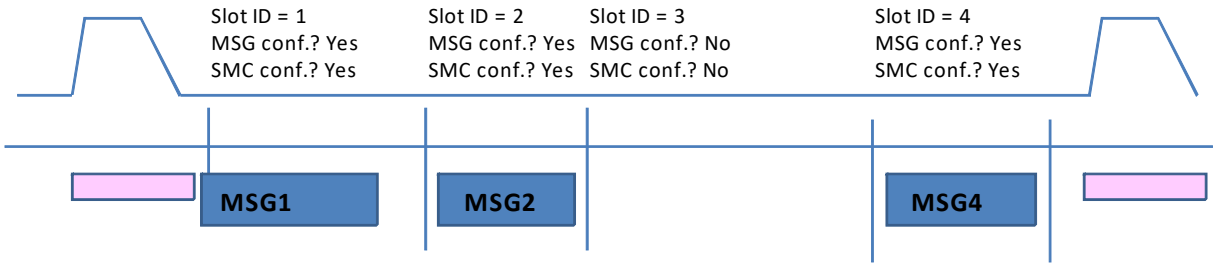
MSG1 and MSG2 are received error free.

MSG4 is late with the start sequence identified in the correct slot 4, with slot counter 4 captured. The end of the message occurs after the Manchester decoder is disabled by the SYNC pulse on the bus, so the last part of the message cannot be received correctly. Nevertheless, the message reception is completed and the message is stored in the message buffer with the T bit set and 0 written in the buffer for all bits received while the Manchester decoder is disabled. This leads to E and C bit errors.

In an extreme case, it could also lead to EM bit errors, in which case the SMC buffer will also be written with (0,0) otherwise, the correct bits are written to the SMC buffer.

60.4.2.3.3.9 Message Too Early (in first slot)

Figure 1508. Message Too Early (in first slot)



Message Name	Slot Counter	T-bit	F-bit	E/EM-bit	C-bit	SMC buffer	SMC data
MSG1	0	Yes	No	No	No	1	M0/M1
MSG2	2	No	No	No	No	2	M0/M1
MSG4	4	No	No	No	No	4	M0/M1

In this scenario, the first message is received early.

As soon as the Manchester decoder disable phase has ended, the IP verifies the idle condition on the signal line and then looks for the first message start sequence. MSG1 start sequence is received before the start of slot 1 with slot counter 0 captured. MSG1 is then received normally and saved with slot counter 0 and T bit set.

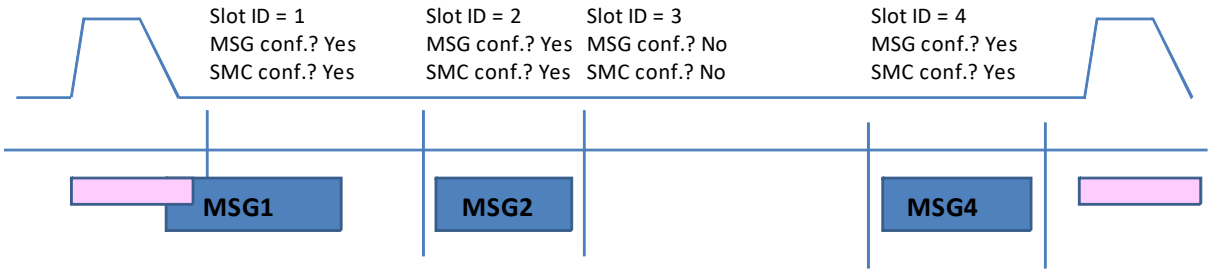
SMC bits M0/M1 are saved to SMC buffer 1.

MSG2 and MSG4 are received normally.

Note: This scenario can be avoided if the start of slot 1 is configured to coincide with the end of the Manchester decoder disable phase.

60.4.2.3.3.10 Message Too Early (in first slot and overlaps Manchester Decoder disable)

Figure 1509. Message Too Early (in first slot)



Message Name	Slot Counter	T-bit	F-bit	E/EM-bit	C-bit	SMC buffer	SMC data
MSG1	1	No	Yes	No	No	1	0/0
MSG2	2	No	No	No	No	2	M0/M1
MSG4	4	No	No	No	No	4	M0/M1

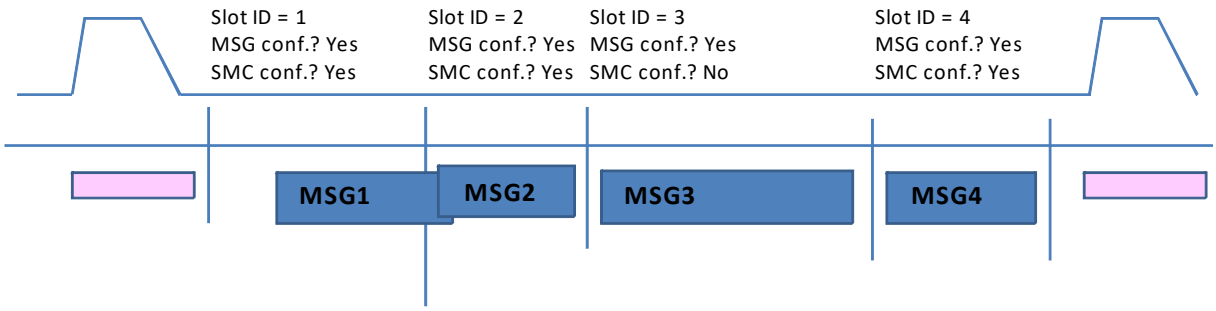
In this scenario, the first message is received early and overlaps with the Manchester decoder disable phase.

As soon as the Manchester decoder disable phase has ended, the IP starts looking for “idle condition” on the signal line by monitoring it for 42 counts of the sample clock. If the signal line is found to be initially high or if a valid transition occurs in the time frame of 42 samples of the sample clock, the Manchester decoder stops decoding until the end of slot 1. Decoding will restart at the beginning of slot 2.

MSG1 is assumed to be lost and the “F” bit is set for that message with the Slot Counter as “1”. If MSG1 SMC is configured then SMC is saved with “00” appended for the M0/M1 bits.

60.4.2.3.3.11 Message Too Late (overlaps next message)

Figure 1510. Message Too Late (overlaps next message)



Message Name	Slot Counter	T-bit	F-bit	E/EM-bit	C-bit	SMC buffer	SMC data
MSG1	1	Yes	No	Yes	Yes	1	M0/M1
MSG2	2	No	Yes	No	No	2	0/0
MSG3	3	No	No	No	No	No	No
MSG4	4	No	No	No	No	4	M0/M1

In this scenario, MSG1 is received too late and overlaps with MSG2.

MSG1 start is recognized in slot 1. The last bits of MSG1 are disrupted by the start of MSG2 leading to likely E and C bit errors.

At the end of MSG1, idle time is found to be violated by either the signal line being high as soon as NO_DETECT_STATE is entered or FIRST_TRANSITION_DETECT occurs within 42 counts of the idle detect sample clock. The Manchester decoder stops decoding until the end of slot 2. Decoding is restarted at the start of slot3. MSG2 is not received and MSG2 location in the memory will be saved with the F-bit set. The SMC buffer 2 will be updated with 00.

MSG3 and MSG4 are received correctly in their own slots.

60.4.2.3.4 Data Frame Storage

Once the data frame is received and all required status fields updated (for example, Time stamp, C, E, EM, T, F and Frame Counter), this 64-bit value in the data receive register is moved to the RAM Register (PMB).

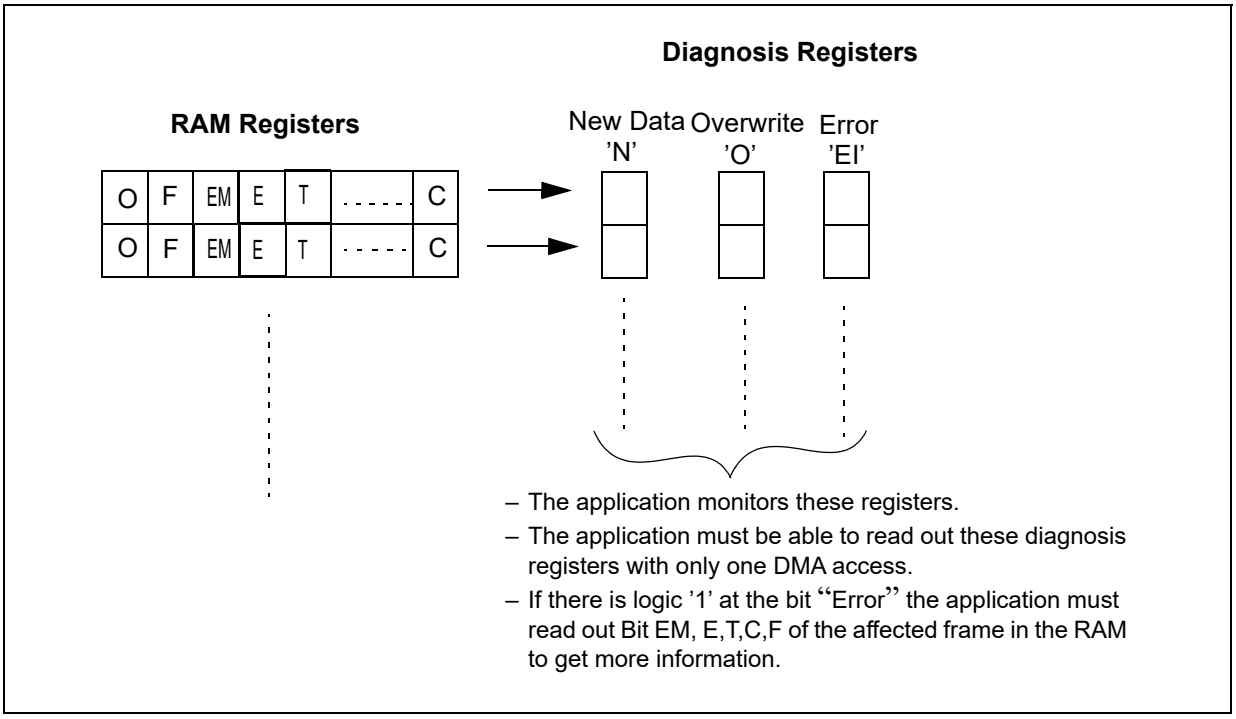
- This RAM buffer is a 32 (depth) × 64 (frame size) register array with DMA transfer capability (both single or burst).
- Also, the individual RAM locations can be accessed directly by the microcontroller core any time.
- Care needs to be taken that receive data is assigned to the sensor and frame slot unambiguously.

60.4.2.3.5 Message diagnostics

Apart from the regular status bits (mentioned above), a few more diagnostic checks are done and stored as status bits along with each data frame location in RAM. These are:

- Diagnosis registers are associated with each sensor data RAM buffer to indicate:
  - RAM buffer has been filled with data.
  - RAM buffer has been overwritten before data previously stored was read.
  - Data written to buffer includes an error bit (E, EM, C, T, or F, configurable).
- The diagnosis registers are able to be read with a single DMA access.
- After successfully completing a read access, these registers are reset by hardware. This is done by mapping these diagnosis bits byte-wise so that one byte access clears a single bit without affecting other bits.

Figure 1511. Diagnosis Information attached to every frame



Note: See Section 60.8.3: Implementation of EI Diagnostic Register for more detail.

60.4.2.3.6 SMC message extraction

The optional messaging channel M0, M1 (shown in *Figure 1496*) is used by sensors to send an SMC frame along with the normal PSI5 frame. It takes 18 PSI5 frames to send one SMC frame.

The extraction of SMC bits is done for each slot separately and stored separately into the SMC frame receive buffers. The size of the SMC frame receive buffers is 6 (no of slots) × 32 (width of SMC frame, as defined below).

The extraction logic works as follows:

- Search for SMC start sequence 111111 followed by 0 on M1 and keep storing last 6 bits {CRC} on M0.
- If the start sequence is detected it continues with further extraction and storage of M1 and M0.
  - Store the extracted frame on the corresponding slot buffer.
  - Move the SMC frame to corresponding SFR[i] register period. Once complete, the SMC frame is extracted and generates an interrupt, so that it can be read by the CPU.
  - If the start sequence is not detected, it continues with the search.

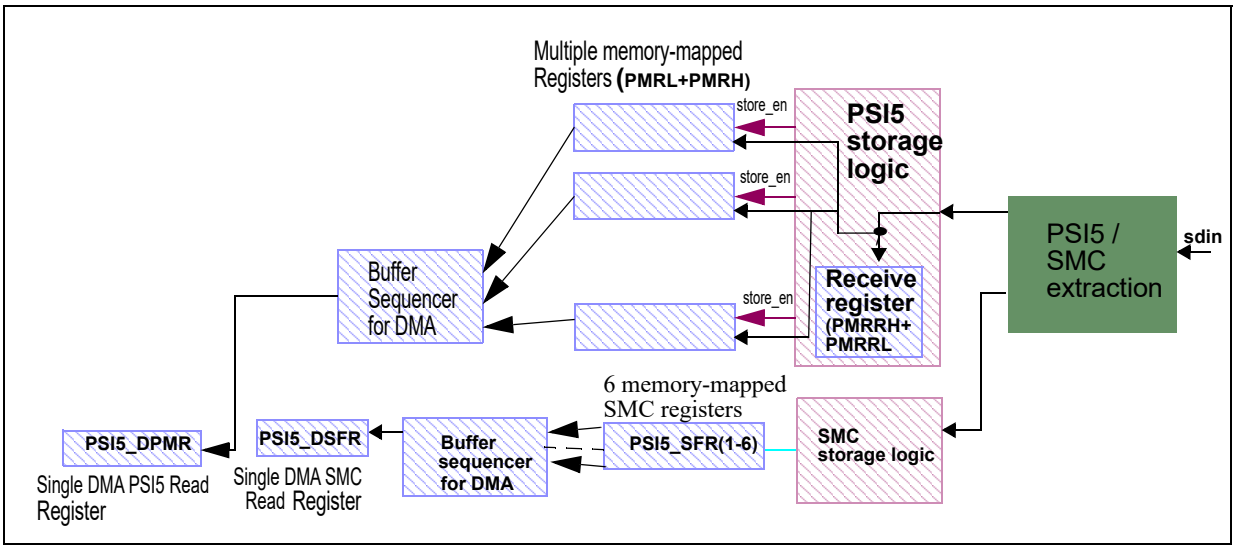
The SMC frame is extracted in hardware and stored separately as memory-mapped registers. The size of the SMC frame receive buffers is 6 (no of slots) × 32 (width of SMC frame).

60.4.2.4 Message Read Logic

PSI5 supports two kinds of read access to received sensor data.

- Read access through memory-mapped RAM registers. These registers can be read by the CPU any time or, specifically, when the concerned interrupts are raised. Applicable for both the SMC and the PSI5 messages.
- Read access through DMA request. Here the DMA sequencer treats the memory-mapped registers as a block and reads them sequentially. Note that even if the DMA is used, the CPU can read the RAM registers any time by addressing them. This is applicable for both the SMC and the PSI5 messages. When reading the PSI5 messages, it is possible to read only the PSI5 messages or PSI5 + diagnostic registers.

Figure 1512. PSI5 and SMC Message Read Logic Block



60.4.2.5 CRC Recalculation

Error detection is realized by a single bit even parity (recommended for 10 or fewer bits) or a 3-bit CRC (recommended for long data words). The transmission error detection is selectable (by frame configuration register) as:

- 1-bit even parity
- 3-bit CRC

The generator polynomial of the CRC is $g(x)=1+x+x^3$ with a binary CRC initialization value of 111. The transmitter extends the data bits by three 0s (as MSBs). This augmented data word is fed (LSB first) into the shift registers of the CRC check. Start bits are ignored in this check. When the last zero of the augmentation is calculated on the input adder and the result is shifted, the buffer registers contain the CRC checksum. These three check bits are transmitted in reverse order (MSB first: C2, C1, C0).

Figure 1513. CRC calculation scheme

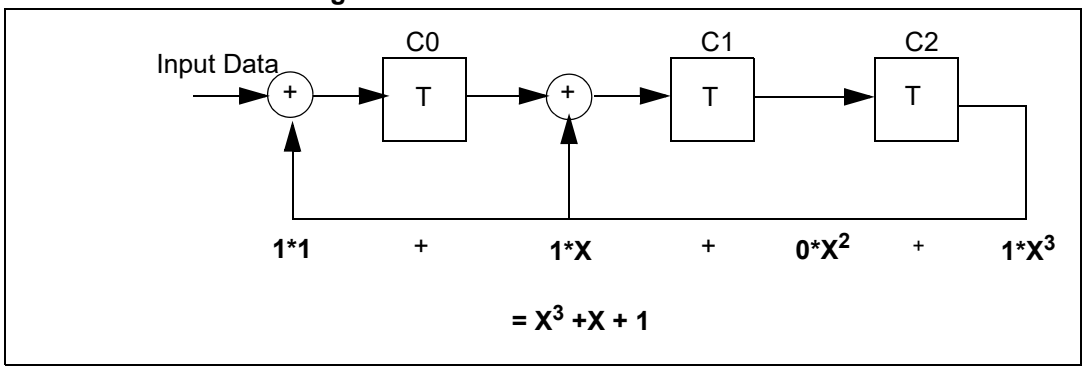


Table 1541. CRC Calculation (at transmitter side)

Sensor Data (LSB first)		0	1	1	1	0	1	1	1	0	0	0	0	0
C0	1	1	0	1	1	0	0	0	0	1	1	1	0	0
C1	1	0	0	0	1	1	1	1	1	1	0	0	1	0
C2	1	1	0	0	0	1	1	1	1	1	1	0	0	1

Table 1542. Transmitted and Received Sensor Data

Transmitted Sensor Data	S0	S1	D0	D1	D2	D3	D4	D5	D6	D7	D8	D9	C2	C1	C0
	0	0	0	1	1	1	0	1	1	1	0	0	1	0	0
Received Sensor Data	S0	S1	D0	D1	D2	D3	D4	D5	D6	D7	D8	D9	C2	C1	C0
	0	0	0	1	1	1	0	1	1	1	0	0	1	0	0

Table 1543. CRC Re-Calculation (At Receiver Side)

Sensor Data (LSB first)		0	1	1	1	0	1	1	1	0	0	1	0	0
C0	1	1	0	1	1	0	0	0	0	1	1	0	0	0
C1	1	0	0	0	1	1	1	1	1	1	0	0	0	0
C2	1	1	0	0	0	1	1	1	1	1	1	0	0	0

Syndrome = 000 → Transmission successful

The correct transmission is checked by the PSI5 receiver by comparing the syndrome (content of the shift register after reception of both data bits and CRC bits) with the zero vector. For error free reception, all shift registers have to be zero.

60.4.3 ECU-to-sensor communication

Whereas the sensor-to-ECU communication is realized by current modulation, voltage modulation on the supply lines is used to transmit commands to the sensors. A predefined start sequence allows the sensors to recognize that a command is being transmitted during all synchronous operation modes.

60.4.3.1 Physical Layer

A logical 1/0 is represented by data sync pulse of variable length, a logical 1 by the sync pulse of width W1(programmable from 0 to 127 µs) and logical ’0’ by sync pulse with W0{programmable from 0 to 127 µs}.

60.4.3.2 Data Link Layer

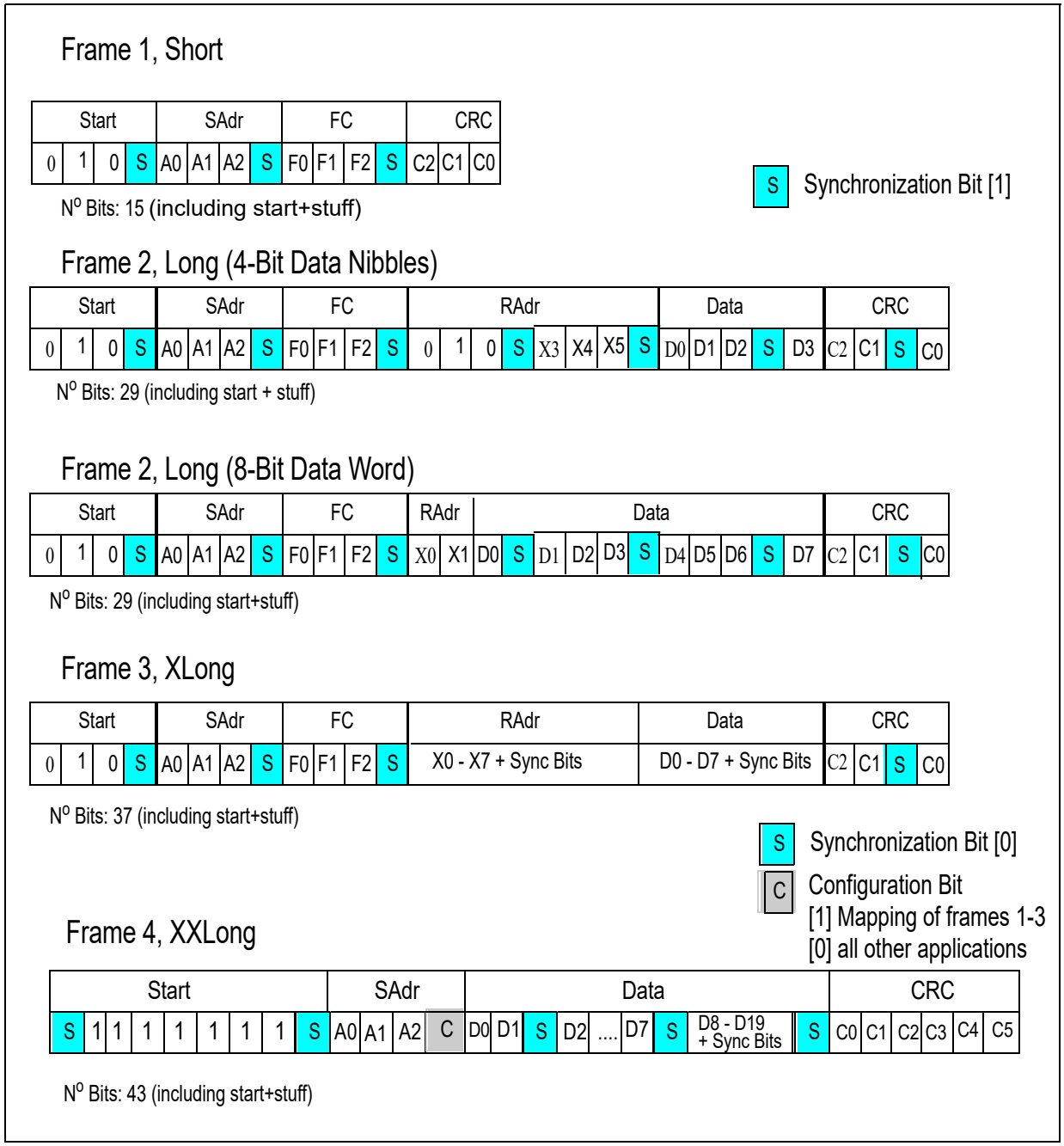
60.4.3.3 Bit coding

The bit period is the cycle time as specified for the operation mode, for example, 500 µs in the PSI5-P10P-500/3L mode. (For details about this topology, refer to *Section 60.1.3.3.1: PSI5 topology*.) The lower limit depends on the filter characteristics of the hardware circuitry used for the sync signal detection (subsequent sync signals might vary the internal $V_{ss,\ base}$ reference).

60.4.3.4 Data transmission

The IP supports the standard and the Non-Standard ECU to Sensor communication Formats. In the Non-Standard mode of communication it can be used to send any frame size programmable from 1-64 bits. The IP supports the PSI5 ECU to Sensor Communication formats as briefed below in *Figure 1514* and the text that follows it.

Figure 1514. Standard Frame Formats Supported



- Short: It consists of the following data stream (Order of transmission from IP is Left to Right. Thus C0 comes out Last) -- “010”, S, “A0, A1, A2”, S, “F0, F1, F2”, S, “C2, C1, C0” where “010” is the start sequence “S” is the stuff bit(1), A0-A2 is the sensor

address, F0-F2 are the functional codes and “C2-C0” is the CRC. Refer *Figure 1514* for the frame structure.

  - The CRC polynomial used is $x3 + x + 1$ with a binary initialization value of “111(C2:C0)”.
  - The transmitter extends the data bits by three zeros (as MSBs).
  - The CRC calculator is fed LSB first. Thus for CRC calculation the sequence given is [A0...A2,F0...F2,”000”] with “A0” being given first and “0” as the last. When the last zero of the augmentation is calculated on the input adder and the result is shifted, the CRC calculation has been achieved.
  - When mapping this sequence to the DPR register bits for calculating the CRC; then it becomes (DPR[0],DPR[1]....DPR[5],000) with DPR[0] being fed first in the CRC calculator.

- Long (4-bit Data Nibble): It consists of the following data stream (Order of transmission from IP is Left to Right. Thus C0 comes out Last) -- “010”, S, “A0, A1, A2”, S, “F0, F1, F2”, S, “X0, X1, X2”, S, “X3, X4, X5, S, “D0, D1, D2”, S,“D3, C2, C1”,S, “C0” where “010” is the start sequence “S” is the stuff bit(1), A0-A2 is the sensor address, F0-F2 are the functional codes, “X0-X5” is the Range Address, “D0-D3” is the data and “C2-C0” is the CRC. Refer to *Figure 1514* for the frame structure.
  - The CRC polynomial used is $x3 + x + 1$ with a binary initialization value of “111(C2:C0)”.
  - The transmitter extends the data bits by three zeros (as MSBs).
  - The CRC calculator is fed LSB first. Thus for CRC calculation the sequence given is [A0...A2,F0...F2,X0...X5,D0...D3,”000”] with “A0” being given first and “0” as the last. When the last zero of the augmentation is calculated on the input adder and the result is shifted, the CRC calculation has been achieved.
  - When mapping this sequence to the DPR register bits for calculating the CRC; then it becomes (DPR[0],DPR[1]....DPR[15],000) with DPR[0] being fed first in the CRC calculator.
- Long (8-bit Data Word): It consists of the following data stream (Order of transmission from IP is Left to Right. Thus C0 comes out Last) Data Stream - “010”, S, “A0, A1, A2”, S, “F0, F1, F2”, S, “X0, X1, D0”, S, “D1, D2, D3, S, “D4, D4, D6”, S,“D7, C2, C1”, S, “C0” where “010” is the start sequence “S” is the stuff bit(1), A0-A2 is the sensor address, F0-F2 are the functional codes,”X0-X1” is the Range Address, “D0-D7” is the data and “C2-C0” is the CRC. Refer *Figure 1514* for the frame structure.
  - The CRC polynomial used is $x3 + x + 1$ with a binary initialization value of “111(C2:C0)”.
  - The transmitter extends the data bits by three zeros (as MSBs).
  - The CRC calculator is fed LSB first. Thus for CRC calculation the sequence given is [A0...A2,F0...F2,X0...X1,D0...D7,”000”] with “A0” being given first and “0” as the last. When the last zero of the augmentation is calculated on the input adder and the result is shifted, the CRC calculation has been achieved.
  - When mapping this sequence to the DPR register bits for calculating the CRC; then it becomes (DPR[0],DPR[1]....DPR[15],000) with DPR[0] being fed first in the CRC calculator.
- X-Long: It consists of the following data stream (Order of transmission from IP is Left to Right. Thus C0 comes out Last) Data Stream - “010”, S, “A0, A1, A2”, S, “F0, F1, F2”, S, “X0, X1, X2”, S, “X3, X4, X5”, S, “X6, X7, D0”, S, “D1, D2, D3”, S, “D4, D5, D6”, S,“D7, C2, C1”, S, “C0” where “010” is the start sequence “S” is the stuff bit(1), A0-A2 is

the sensor address, F0-F2 are the functional codes,"X0-X5" is the Range Address, "D0-D3" is the data and "C2-C0" is the CRC. Refer to *Figure 1514* for the frame structure.

- The CRC polynomial used is $x3 + x + 1$ with a binary initialization value of "111(C2:C0)".
- The transmitter extends the data bits by three zeros (as MSBs).
- The CRC calculator is fed LSB first. Thus for CRC calculation the sequence given is [A0...A2,F0...F2,X0...X7,D0...D7,"000"] with "A0" being given first and "0" as the last. When the last zero of the augmentation is calculated on the input adder and the result is shifted, the CRC calculation has been achieved.
- When mapping this sequence to the DPR register bits for calculating the CRC; then it becomes (DPR[0],DPR[1]....DPR[21],000) with DPR[0] being fed first in the CRC calculator.

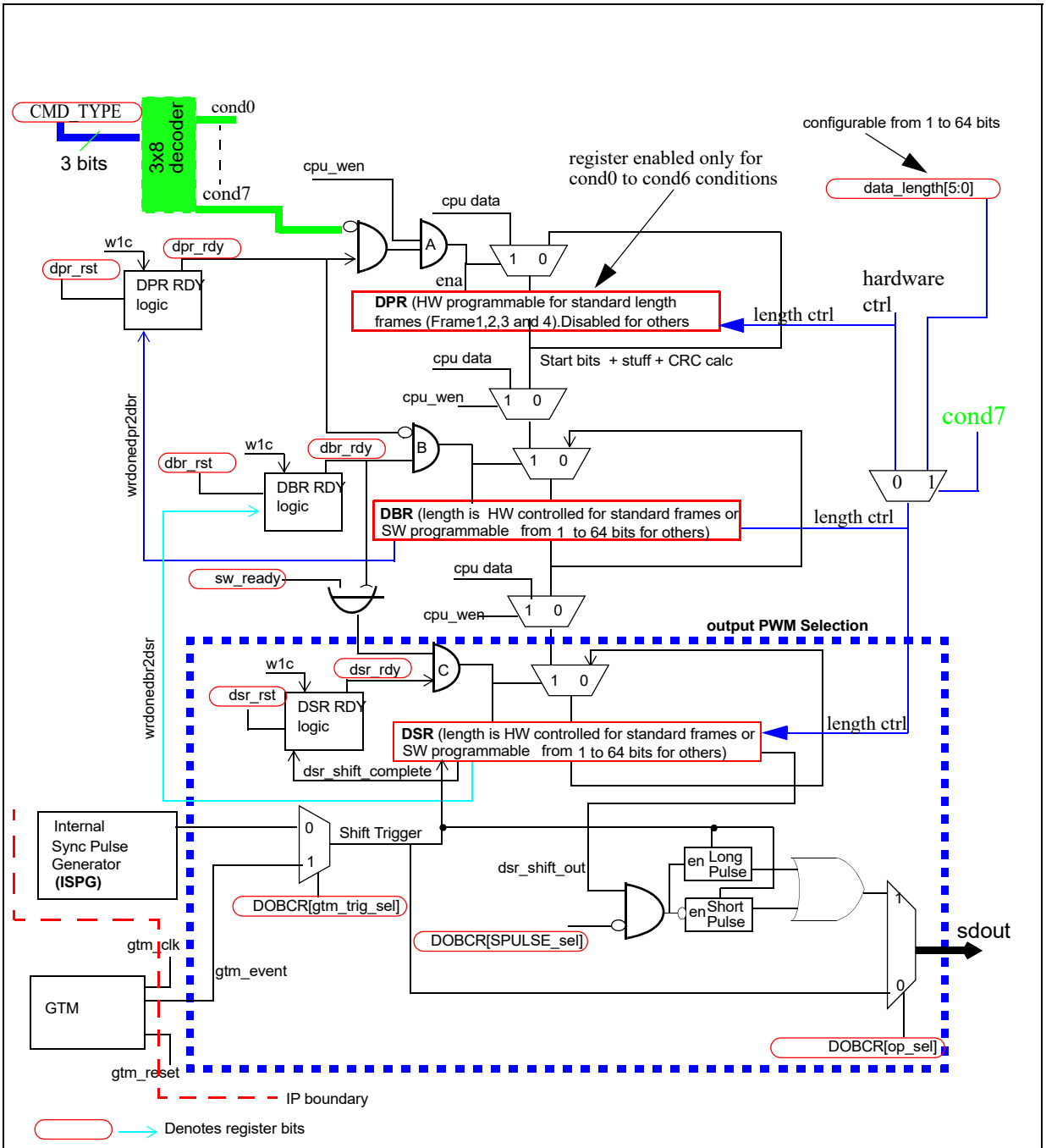
- XX-Long: It consists of the following data stream (Order of transmission from IP is Left to Right. Thus C5 comes Last) Data Stream - "011111110", "A0, A1, A2", "C", "D0, D1", S, "D2, D3, D4" D5, D6, D7, S, "D8, D9, D10, D11, D12, D13", S,"D14, D15, D16, D17, D18, D19", S, "C0, C1, C2, C3, C4, C5" where "011111110" is the start sequence "S" is the stuff bit(0), A0-A2 is the sensor address, "C" is the configuration bit, "D0-D19" is the data and "C0-C5" is the CRC. Refer to *Figure 1514* for the frame structure.
  - The CRC polynomial used is $x6 + x4 + x3 + 1$ with a binary CRC initialization value 010101(C5:C0).
  - The transmitter extends the data bits by six zeros (as MSBs).
  - The CRC calculator is fed LSB first. Thus for CRC calculation the sequence given is [A0...A2,D0...D19,"000000"] with "A0" being given first and "0" as the last. When the last zero of the augmentation is calculated on the input adder and the result is shifted, the CRC calculation has been achieved.
  - When mapping this sequence to the DPR register bits for calculating the CRC; then it becomes (DPR[0],DPR[1]....DPR[23],000000) with DPR[0] being fed first in the CRC calculator.

60.4.3.4.1 Transmission Data Framing

The output data transmission block is constructed of seven sub-blocks:

- Data preparation register
- Buffer register
- Data shift register
- Periodic Sync Pulse Trigger Generator
- "GTM driven" sync pulse trigger input (GTM external to PSI5)
- Pulse length modulation block
- OR gate

Figure 1515. Data Transmission Block



cond0 = Frame 1(short) - start condition: at least 31 "1s"
cond1 = Frame 1(short) - start condition: at least 5 "0s"
cond2 = Frame 2(long) - start condition: at least 31 "1s"
cond3 = Frame 2(long) - start condition: at least 5 "0s"
cond4 = Frame 3(x-long) - start condition: at least 31 "1s"
cond5 = Frame 3(x-long) - start condition: at least 5 "0s"
cond6 = Frame 4(xx-long)
cond7 = Non standard frames - programmable <data_length> 1 to 64 bits

When <sw_ready> == 0 the data from DBR to DSR is NOT transferred and remains pending even if DSR_RDY becomes 1. Only when <sw_ready> becomes 1, is the data from DBR to DSR transferred.

60.4.3.4.2 Data Preparation Register (64-bit register, DPRH + DPRL, DPRH is fixed at 0)

Up to 24 bits of this register can be written by the CPU. These register bits contain the ECU to sensor command that needs to be transmitted after being appended with the CRC, start bits and stuff bits. The hardware automatically calculates the CRC, adds the start and the stuff bits, before transferring the command to DBR.

- When DPR_RDY == 1, the data to be transmitted to sensors is written to this register via system bus with order data bit to be transmitted first at LSB and furthest on MSB. This register can only be used for writing standard length data frames. Data to this register can be written only when PSI5_DOBCR[CMD_TYPE]!= 7. When PSI5_DOBCR[CMD_TYPE] == 7 then DPR_RDY remains at “0” and the commands cannot be written to this register. Bits writable in different settings of the PSI5_DOBCR[CMD_TYPE] can be seen in *Section 60.3.2.36: Data Preparation Register Low (PSI5_CH0_DPRL)*. Note that PSI5_DPRH is always = 0.
- Having written the data to the DPR, the DPR_RDY bit has be cleared with a w1c. This indicates to the internal logic that the data in the DPR is valid and is to be processed.
- If DPR_RDY = 0 and a new command is written to the DPR then the command is rejected and GISR[IS_PROW] is set.
- If the CRC or parity bit calculation is finished, then the data is shifted to the Buffer register with the start bits and the sync bits appended as well. This transfer is done to the Buffer register only if the Buffer register is ready for new data, which is indicated by GISR[DBR_RDY] == 1.
- As mentioned above, once this transfer is done, the DPR_RDY becomes 1 and a new command can be written to it.

60.4.3.4.3 Data Buffer Register (64-bit register)

- This register has two sub registers DBRH (higher register) and DBRL (lower register). Both of them are collectively referred to as DBR. DBR is of length 64 bits DBR[63:0]
- The bits accessible for functionality are dependent on the value of PSI5_DOBCR[CMD_TYPE]. For the writable bits corresponding to different setting of the PSI5_DOBCR[CMD_TYPE] field refer bit description in *CMD_TYPE*
- When DPR_RDY == 0 and DBR_RDY goes to 1 then internal logic automatically transfers the data from the DPR (with CRC + stuff bits + sync bits) to the DBR. DBR_RDY goes low and DPR_RDY goes 1. In order to check the calculation of the DPR, register DBR can be read at this stage, that is, when DPR_RDY becomes “1”.
- If DBR_RDY == 1 AND DPR_RDY == 1, then the data to be transmitted to sensors can be written to this register directly via system bus skipping preparation register in case application includes the start and sync bits and calculates the CRC/Parity itself. If the CPU writes to the DBR when DBR_RDY == 1 but DPR_RDY == 0 and

PSI5_DOBCR[CMD_TYPE]!= 7, then the results can be unpredictable since the DPR has a pending data that needs to be transferred to DBR.

- If the CPU is writing to the DBR as mentioned above, then the DBR_RDY bit has to be cleared by a w1c. This indicates to the internal logic that the data in the DBR is valid and is to be used for the shift operation.
- If while DBR_RDY == 0, and a new command is directly written to the DBR then the command is rejected and GISR[IS_BROW] bit is set.
- If while DBR_RDY == 0 a new command HAS to be written in the DBR then DOBCR[DBR_RST] has to be written to 1. Writing a 1 to DBR_RST will reject the current command in the DBR and generate DBR_RDY = 1.
- If the Data Shift Register signals that it is ready for new data (PSI5_GISR[DSR_RDY] == 1), the data of the Buffer register is shifted into the data shift register provided PSI5_DOBCR[SW_READY] is 1.
- If PSI5_DOBCR[SW_READY] == 0 then the data from the DBR is NOT transferred to the DSR automatically even if the DSR_RDY goes as 1. In such a case the IP waits for the software to make the PSI5_DOBCR[SW_READY] bit as 1; only then does it transfer the contents to the DSR.

60.4.3.4.4 Data Shift Register (64-bit register)

- This register has two sub registers DSRH (higher register) and DSRL (lower register). Both of them are collectively referred to as DSR.DSR is of length 64 bits DSR[63:0]. This register has a 1-1 correspondence with the bits in the DBR register when this register is updated automatically by the hardware.
- The bits accessible for functionality are dependent on the value of PSI5_DOBCR[CMD_TYPE]. For the writable bits corresponding to different setting of the PSI5_DOBCR[CMD_TYPE] field refer to bit description in *CMD_TYPE*.
- When DBR_RDY == 0 and DSR_RDY goes to 1 then internal logic automatically transfers the data from the DBR to the DSR provided PSI5_DOBCR[SW_READY] == 1'b1, else this transfer remains pending until PSI5_DOBCR[SW_READY] is '0'. Once the transfer is complete, DSR_RDY goes low and DBR_RDY goes 1. DSR contents that would finally get transferred, can be read at this stage, that is, when DBR_RDY becomes "1".
- Making PSI5_DOBCR[SW_READY] now as '0' will not interrupt the ongoing transfer in the DSR.
- If DPR_RDY == 1 AND DBR_RDY == 1 AND DSR_RDY == 1, then the data to be transmitted to sensors can be written to this register directly via system bus skipping Buffer register. If the CPU writes to the DSR when DSR_RDY == 1 but DBR_RDY == 0 then the results can be unpredictable since the DBR has a pending data that needs to be transferred to DSR.
- When PSI5_DOBCR[CMD_TYPE] == 7 and If DBR_RDY == 1 AND DSR_RDY == 1, then the data to be transmitted to sensors can be written to this register directly via system bus skipping Buffer register. If the CPU writes to the DSR when DSR_RDY ==

1 but DBR_RDY == 0 then the results can be unpredictable since the DBR has a pending data that needs to be transferred to DSR.

- If the CPU is writing to the DSR as mentioned above then the DSR_RDY bit has to be cleared by a w1c. This indicates to the internal logic that the data in the DSR is valid and is to be used for the shift operation.
- If while DSR_RDY == 0, and a new command is directly written to the DSR then the command is rejected and GISR[IS_DSROW] is set.
- If while DSR_RDY == 0 a new command HAS to be written in the DSR then DOBCR[DSR_RST] has to be written to 1. Writing a 1 to DSR_RST will reject the current command in the DSR and generate DSR_RDY = 1.
- This register is clocked by the ISPG pulses or the GTM pulses, as described in *Section 60.4.3.4.5: Sync Pulse Trigger Generation*.
- Once the shift operation is complete in the DSR then DSR_RDY becomes 1.

60.4.3.4.5 Sync Pulse Trigger Generation

The sync pulse trigger can be used to generate the timing pulses or the data pulses.

The sync pulse can be generated based on triggers from two different sources:

- ISPG (Internal Sync Pulse Generator): Composed of CTPR, CIPR and Channel Target Counter (per channel). The internal sync pulse generator is clocked by either the GTM clock (max frequency is 3.9Mhz) or the internal 1 MHz clock. The Channel Target Counter operates with a 1 µs resolution (when using the 1Mhz clock) or GTM clock resolution (when using the GTM clock). The ISPG is implemented to allow generation of internal periodic trigger. Refer to *Figure 1522* for the ISPG structure and *Figure 1520* for the CTPR and CIPR explanation. The ISPG uses the Channel Target Counters (CTCs) for generating the sync pulse. The CTCs can be triggered globally (simultaneous for all channels) or locally (individual to each channel). The CTC_GED bit in the Global Control Register (GCR) ensures that all the CTCs start/stop simultaneously. The individual target values when the trigger pulse is generated are set in the Counter Target Pulse Register (CTPR) and the Channel Initial Pulse Register (CIPR). These registers are present for each channel. When staggering the channels, the CTC_GED bit should be used for providing a simultaneous trigger to the staggered channels. It is also possible for a subset of channels to work on the above arrangement while the remaining channels independently generate trigger pulses for which start/stop is controlled locally to the channels.
  - For each channel, it is possible to configure the trigger period by software using sync pulse generator control/configuration register and setting compare values in Counter Target Pulse Register(s). The Counter Initial Pulse Register (CIPR) can be used to provide the initial offset between various channels.
  - It is possible to deactivate each channel separately for the purpose of sync pulse generation.
  - The periodic pulses can be used to synchronize sensor-to-ECU communication (synchronization timing pulses) [State 1 of *Section 60.8.2: Valid states for*

integrated sync pulse generator] or can be used as data pulse during ECU-to-sensor communication (through the DOBCR) [State 2 of *Section 60.8.2*].

– When programming the CTPR, the Tsync period should be programmed >= 6 SP_TS_CLK cycles (For details of SP_TS_CLK, refer to Bit 23 of *Section 60.3.2.2: PSI5 Channel Control Register (PSI5_CH0_PCCR)*).

- GTM event controlled: Generation of sync trigger/pulse can be done based on events (for example, crank angle or software) through general timer module (GTM) output to the PSI5 interface.
  – Here the sync trigger pulses come from an external GTM module routed through a full XBAR, which is programmable at the system level.
  – The sync pulse programming (period, and so on) is done inside the GTM.
  – In *Figure 1515*, if DOBCR[gtm_trig_sel] == 1, DOBCR[SPULSE_sel] == 0 and DOBCR[OP_sel] == 1 then the GTM trigger is used for data pulse generation (data pulses for ECU-to-sensor communication). [State 4 of *Section 60.8.2*]
  – In *Figure 1515*, if DOBCR[gtm_trig_sel] == 1, DOBCR[SPULSE_sel] == 1 and DOBCR[OP_sel] == 1 the GTM trigger is used for timing pulse generation (synchronization pulse for sensor-to-ECU communication). [State 3 of *Section 60.8.2: Valid states for integrated sync pulse generator*]
  – In *Figure 1515*, if DOBCR[gtm_trig_sel] == 1, DOBCR[SPULSE_sel] == “x” and DOBCR[OP_sel] == 0 the GTM trigger is passed directly to data sync port without pulse length modulation. In this case, the sync pulse length is defined by the GTM pulse length. [State 5 of *Section 60.8.2*]
- Programming and control: For achieving the various states of ISPG control or the GTM control, the 3 equivalent bits (GTM_TRIG_SEL, SPULSE_SEL and OP_SEL) belonging to the DOBCR, can be programmed, as shown in *Figure 1515* and *Figure 1516*. The programming of various states mentioned above, with reference to the *Section 60.8.2: Valid states for integrated sync pulse generator* can be seen in *Table 1526* and also shown in *Figure 1524*. The sending of the ECU to sensor commands, can be controlled by PSI5_GISR[DPR_RDY,DBR_RDY,DSR_RDY] bits and the PSI5_DOBCR register bits.

60.4.3.4.5.1 Timing Pulses

Timing pulses can also be produced without pulse width modulation also. These timing pulses can be generated through two sources:

- Periodic sync pulse from ISPG. In this case the synchro pulse from the internal sync pulse generator can be made available without any modulation. Referring to *Figure 1516*, in synchronous mode, if DOBCR[GTM_TRIG_SEL] == 0 and DOBCR[OP_SEL] == 0 then the internal sync pulse generator output can be made available without any modulation. The high time of this pulse is fixed at 4 µs.
- Referring to *Figure 1516*, in synchronous mode, if DOBCR[GTM_TRIG_SEL] == 1 and DOBCR[OP_SEL] == 0, then the GTM trigger can be used for timing pulse generation without any Pulse Width Modulation.

60.4.3.4.5.2 Data pulse generation

Data pulses are produced by the Pulse Length Modulation Block based on triggers from one of the two selectable sources, the ISPG or the GTM, during ECU-to-sensor communication.

The two sync sources are selected by software programming of the Pulse Length Control/Configuration Register.

- A logical 1 in the Data Output Register LSB is coded as data sync pulse of width W1 {up to 127 µs}.
- A logical 0 in the Data Output Register LSB is coded as data sync pulse of width W0 {up to 127 µs}.
- The two widths are programmable in the Pulse Length Control/Configuration Register.
- The Data Shift Register is shifted right every time a bit is transmitted.

Figure 1515 shows the data transmission block.

Figure 1516. ECU-to-sensor communication registers

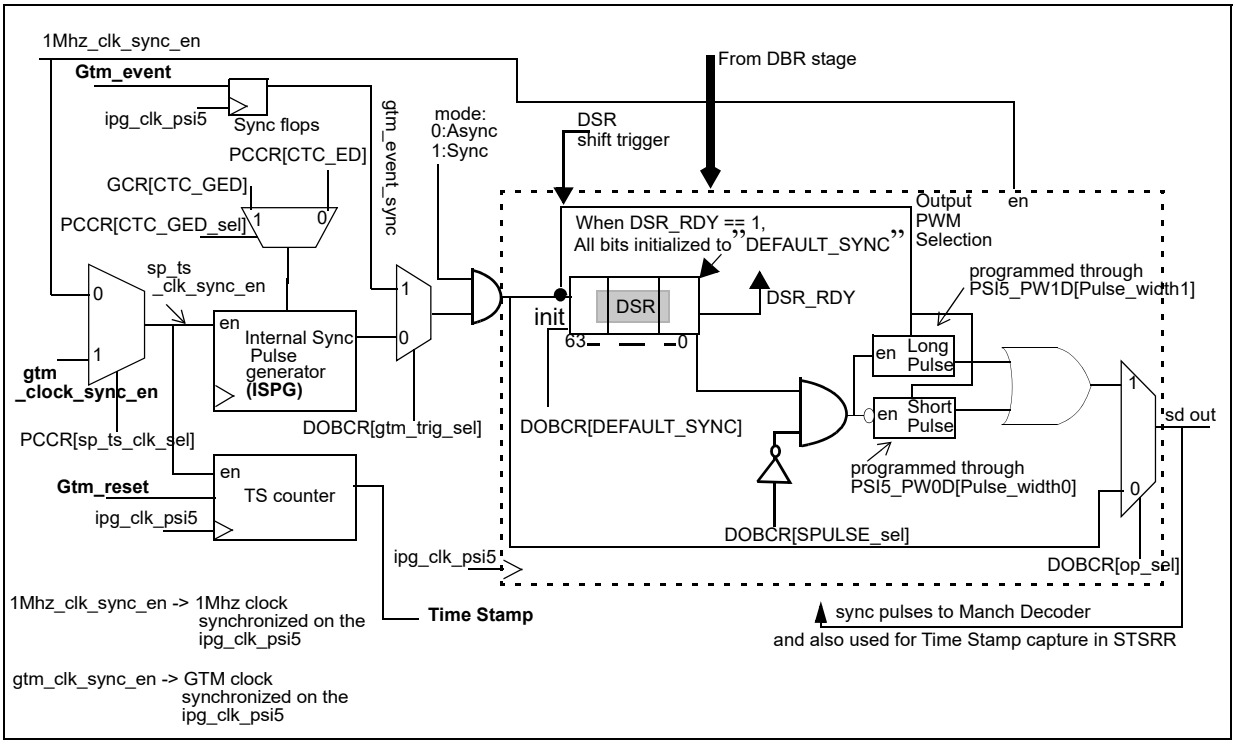


Figure 1516 details the implementation of *Figure 1515* and shows how the various bits mentioned in *Section 60.3.2.30: Data Output Block Configuration Register (PSI5_CH0_DOBCR)* control the behavior of the Output Command path. *Figure 1516* also shows the various clock sources to the Internal Sync Pulse Generator and the Time Stamp counter.

60.4.3.5 Response to ECU-to-sensor frames

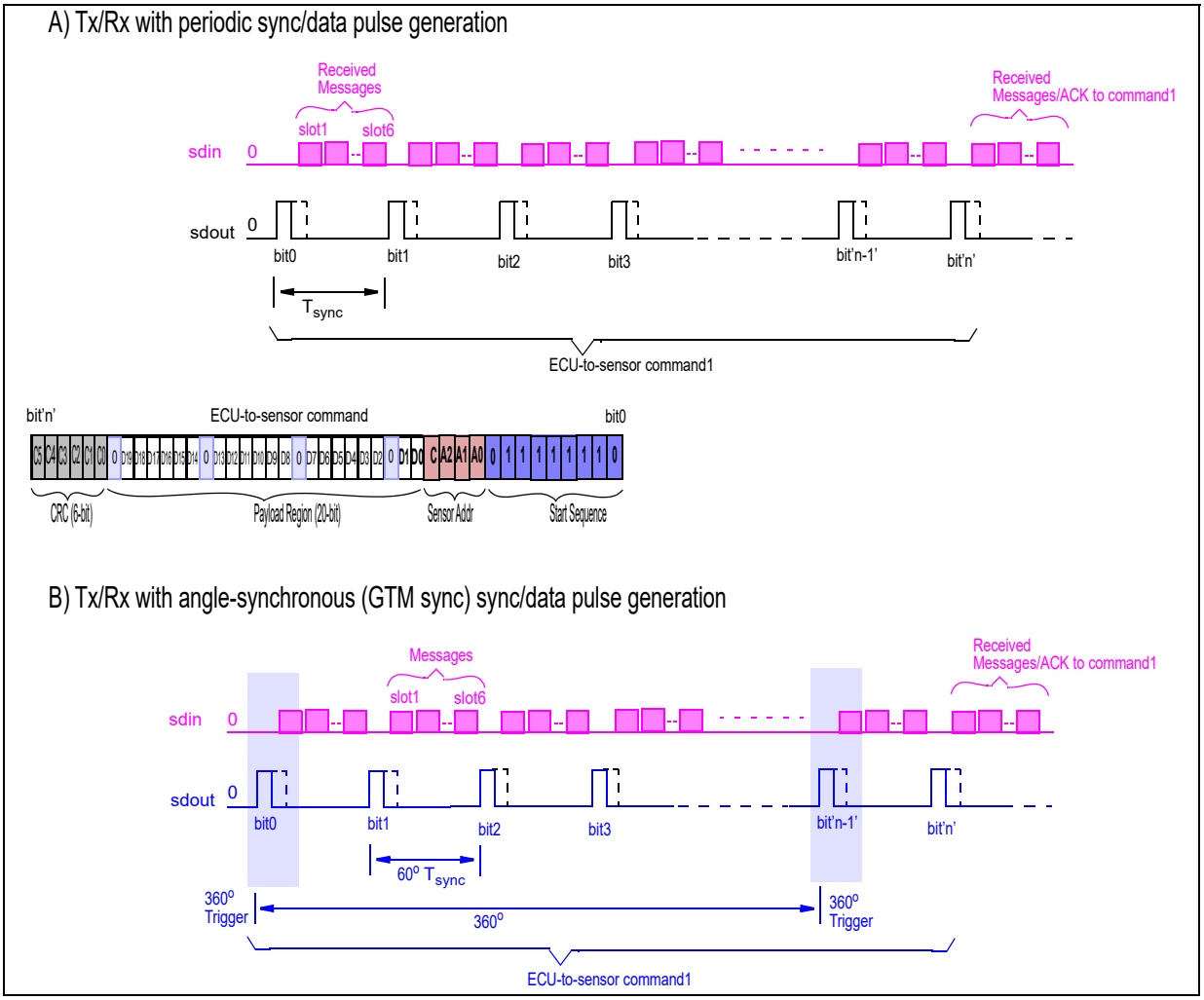
The response to an ECU-to-sensor frame can be optionally transmitted in two ways:

- Out-of range values of data frame (as fast response coded as normal PSI5 payload region)
- Serial messaging channel (verification of the response frames must be handled by the software)

60.4.4 Tx/Rx scenario

The diagram below shows the communication scenario in synchronous modes. For asynchronous mode, sync pulses are not generated and sensor transmits data asynchronously.

Figure 1517. Synchronous mode communication scenarios



60.4.5 Interrupt handling

Configurable interrupts based on events (for example, reception of valid message), errors, DMA transfers, and so on are implemented. The following types of event can generate an interrupt:

- Reception of a new PSI5 message
- Reception of a new SMC message
- Unread SMC message overwritten in the SMC buffer area (PSI5_SFR)
- Unread PSI5 message overwritten in the PSI5 buffer area (PSI5_PMRH, PSI5_PMRL)
- Dedicated interrupts for selectable errors, mentioned below:
  - Bit C is set
  - Bit E or EM, or both, are set
  - Bit T is set
  - No PSI5 frame was received in a defined frame slot (Bit F is set)
- ECU-to-sensor communication-related interrupts
  - The application tries to load new data into the data preparation register before the register has signalized that it is ready for new data (Note that the new command will be ignored in this case).
  - The application loads new data into the data buffer register before the register has signalized that it is ready for new data (Note that the new command will be ignored in this case).
  - The application loads new data into the data shift register before the register has signalized that it is ready for new data (Note that the new command will be ignored in this case)
- DMA related interrupts (transfer finished/FIFO_FULL/underflow)
- Apart from this a single one sum interrupt OR-ing all the above events.

Various interrupt signals that represent the above events are:

- PSI5 new message arrival interrupt (Status flags in PSI5_NDSR)This is asserted when any of the programmed storage PSI5 locations (max 32) receive a new PSI5 message

(error/error free). This interrupt also indicates that the PSI5 receive register (PMRRL/PMRRH) has been updated with a new PSI5 message.

- SMC message, ECU-to-sensor communication, and Time Stamp related interrupts (All Status flags in PSI5_GISR): This is asserted during following events:
  - When a new SMC message is received in its corresponding slot OR
  - When there is a CRC error in any SMC message OR
  - When an unread message in some corresponding SMC location (in PSI5_SFR) gets overwritten by a new SMC message OR
  - When DPR is ready for receiving a new data OR
  - When DBR is ready for receiving a new data OR
  - When DSR is ready for receiving a new data OR
  - When there is an attempt to write to DPR while it is NOT ready OR
  - When there is an attempt to write to DBR while it is NOT ready OR
  - When there is an attempt to write to DSR while it is NOT ready OR
  - When data time stamp register is updated with new time stamp value OR
  - When the sync time stamp read register is updated with a new time stamp.
- PSI5 Message Overwrite Interrupt (Status Flags in PSI5_OWSR): This is asserted when any of the programmed storage locations in the memory (max 32) have an unread PSI5 message that is overwritten by a new upcoming PSI5 message. Corresponds to OWSR.
- PSI5 Error interrupt (Status Flags in PSI5_EISR): This is asserted when any of the programmed storage locations in the memory (max32) have an error in the PSI5 message. It is used only for PSI5 messages.
- DMA Interrupt (Status Flags in PSI5_DSR):
  - When the DMA transfer corresponding to conf2, conf3 or conf4 finishes OR
  - When the DMA transfer corresponding to SMC message gets over OR
  - When there is an FIFO_FULL/Underflow in the PSI5 RAM register area OR
  - When there is an underflow in the SMC message area.
- Global Interrupt: Single interrupt OR-ing all above interrupt events.

60.4.5.1 Error conditions

- C: CRC/parity error on received data, evaluated after recalculating CRC on received data and comparing with received CRC or recalculating parity on received data and comparing with received parity.
- E: Electrical error during any Tbit period (excluding Messaging field M0M1) during frame reception (not applicable during Idle state); that is, if at least one bit is absent during its Tbit period.
- EM: Electrical error on M0M1 field.
- T: Timing error during frame reception violates the slot boundary. Here, all the corner cases and conditions are taken care of (for example, the isolation of correct slot to correct message if the message is too early or too late with respect to its expected position, the crossing of the slot boundary when one frame overlaps the other, and so on).
- F: No frame received during a particular slot time in case of sync mode.

60.4.6 DMA support

- The PSI5 module generates separate DMA transfer requests for PSI5 message/diagnostic status and SMC messages.
- A single DMA request is used for PSI5 messages and diagnostic data. The request is triggered based on the selected DMA configuration (either PSI5 message or diagnostic data or both). After choosing the configuration, data is available on respective DMA pop-up registers.
- DMA requests for PSI5 messages and diagnostic registers are based on the watermark level of message buffers (to be treated as FIFO for the DMA). PSI5 message buffer FIFOs get data from the PSI5 receiver in the order they are received.
- DMA requests for SMC messages are based on successful storage of each SMC frame to the message buffers. The SMC message buffers get data from SMC extraction logic in fixed round-robin arbitration sequence.
- Refer to *Section 60.4.6.1: Single DMA request for PSI5/diagnostic message buffers* for PSI5 DMA handling and to *Section 60.4.6.2: Single DMA request for SMC received messages* for SMC DMA handling.

60.4.6.1 Single DMA request for PSI5/diagnostic message buffers

A common DMA request (per PSI5 block) is generated for PSI5 message/diagnostic buffers. The depth (up to 32 in present context) of the message buffer is programmable via the MEM_DEPTH filled in the PCCR. The width of each location of the PSI5 message buffer is 64 bits. When reading PSI5 message through the DMA, the 32-bit DPMR is sequenced with the data from the PSI5 message buffer. When reading the diagnostic registers through the DMA, the 32-bit DDSR is sequenced as NDSR followed by EISR. The DMA operation will happen as follows:

- Based on the DMA configuration chosen, one of the three DMA transfer sequence is initiated:
  - PSI5 buffer followed by diagnostic data sequenced to DPMR
  - PSI5 data sequenced to DPMR
  - Diagnostic buffer data sequenced to DDSR
- The DMA request is asserted when the number of messages stored in the buffer crosses the configured watermark number of messages.
- DMA request is deasserted once the “watermark” number of messages have been read.
- If the number of unread messages is still greater than the watermark programmed, then the DMA request is reasserted.
- The received message data is read from a common memory-mapped 32-bit DMA PSI5 message register (DPMR). Thus, a single PSI5 message (64 bits wide) will require two IPS read operations. The diagnostic data is read either through the DPMR or through the DDSR (single 32-bit register) by the DMA controller via the IPS interface.
- It is assumed that the DMA Controller’s Transfer Control Descriptor (TCD) would be set to transfer bytes equal to the messages indicated by watermark level, as described in *DMA_PM _DS_WM* in *Section 60.3.2.3: DMA Control Register (PSI5_CH0_DCR)*. It is assumed that when responding to a DMA request, the DMA controller will read faster

than the rate at which messages are received and stored in the buffer/FIFO, which will cause the DMA request to deassert.

- The next received message will be updated on the common memory map register once the previous message is read by the DMA (by the DMA sequencer).
- The store signal from the synchronization logic will push the message into the memory-mapped RAM registers and in parallel also into the PSI5 receive register. *Figure 1512* shows the format in which the messages will be stored in any message buffer (RAM registers) and finally when being read from DMA access register.
- For more details about DMA programming refer to *Section 60.3.2.2: PSI5 Channel Control Register (PSI5_CH0_PCCR)*, *Section 60.3.2.3: DMA Control Register (PSI5_CH0_DCR)*, *Section 60.3.2.4: DMA Status Register (PSI5_CH0_DSR)*, *Section 60.3.2.10: DMA PSI5 Message Register (PSI5_CH0_DPMR)* and *Section 60.3.2.12: DMA Diagnostic Status Register (PSI5_CH0_DDSR)*.

60.4.6.2 Single DMA request for SMC received messages

A separate DMA request (each channel) is generated for the SMC message buffer. Each SMC received message would be stored in the internal registers and read out by DMA controller via a common memory-mapped DMA SMC Frame Register (DSFR) (single 32-bit register for each message buffer). The data flow will happen as follows:

- A request is asserted when the system buffer successfully receives one complete SMC message, and remains asserted until all complete messages have been read. Data is read from the DMA PSI5 Message Register in the memory map.
- It is the job of an internal sequencing logic to provide the next complete received message. The internal sequencer works in a round-robin fashion checking for the slot-specific registers for each SMC message. Thus the sequence will be slot1— slot2—slot3—slot4—slot5—slot6—slot1 and so on. In whichever slot it encounters the first complete SMC message, the DMA request is asserted and the sequencer will wait for this data to be read before moving on to the next slot in its loop. If a particular slot message remains unread and a new SMC message for the same slot comes in, then the previous message would be overwritten and the OW bit would be set.
- The DMA controller should be programmed to read only one message in one DMA transfer. Otherwise, if it is programmed to read more than one message in one DMA transfer, underflow in buffers might occur. User software should ensure there is no overflow.
- For more details about DMA programming refer to *Section 60.3.2.2: PSI5 Channel Control Register (PSI5_CH0_PCCR)*, *Section 60.3.2.3: DMA Control Register (PSI5_CH0_DCR)*, *Section 60.3.2.4: DMA Status Register (PSI5_CH0_DSR)* and to *Section 60.3.2.12: DMA Diagnostic Status Register (PSI5_CH0_DDSR)*.

60.4.7 Message read through interrupt generation

There is a separate interrupt request for received messages for each PSI5 channel. Each PSI5 channel can be configured individually to generate an interrupt on reception of messages. An interrupt is asserted when the corresponding PSI5 channel has received a complete message (PSI5 or SMC). Refer to *Section 60.4.5* for more details about interrupts. The interrupt routine should be coded in such a way to take the least amount of overhead (in CPU cycles) while entering the routine and start reading the available messages. The routine should be capable of accessing the corresponding PSI5 channel that is enabled to give out messages via interrupts through the IPS Interface. In response to these interrupts,

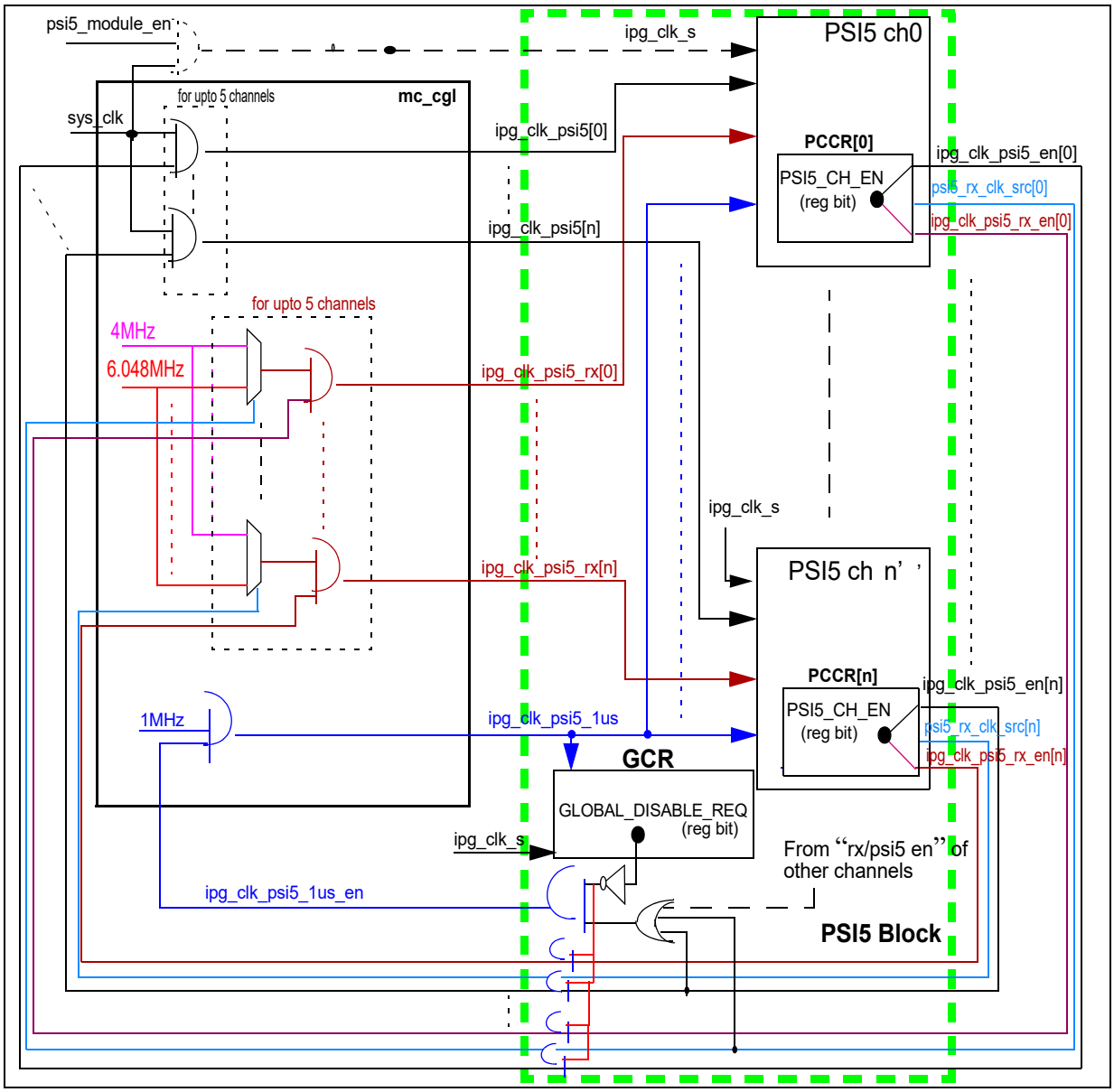
the CPU is expected to read interrupt status register for individual PSI5 Channel (depending on which interrupt was asserted) and store it in system memory.

60.5 Clock and Reset

The proposed clock scheme of the PSI5 block is shown in *Figure 1518*. Global clock and clock divider are taken care of inside the CGM. Following is a brief description of these clocks with respect to the CGM:

- Global clock (refer to *Figure 1518*)
  - 1 Global clock generated from a single source but separately controllable for each PSI5 channel (“ipg_clk_psi5” for each channel that is generated from sys_clk).
  - It has a jitter of less than 5 ns at 1 MHz.
  - 1 global clock, which is enabled during any register read/write from IPS interface. (“ipg_clk_s”, which is common to all the PSI5 channels).
- Clock divider
  - Only one clock divider for all PSI5 interfaces (divider to generate 4 MHz/6.048 MHz from the system clock).
  - The clock divider generates all clocks that are needed for up to five interfaces.
  - The clock divider is configurable by software.
  - The output of the clock divider has a jitter of less than 5 ns at 1 MHz.
- Separate GTM clock (not shown in *Figure 1518*) for each channel, which is available from at the system level. For reliable operation, the GTM clock should be limited in frequency to up to 3.9 MHz.

Figure 1518. PSI5 clock scheme



Note: Refer to the Device Configuration chapter for value of PSI5_NUM_CH.

60.5.1 Latency and Jitter related requirements

- Latency related
  - From the detection of the last bit of a frame (end of last CRC bit of message) at the input pin of the controller, up to provision of the data for the software (including CRC calculation): ≤ 3µs.
  - From the sync pulse trigger output of the Sync Pulse Generator, up to the output of the sync pulse trigger at the data output pin of the controller: ≤ 2µs.
- Jitter related
  - From the detection of the last bit of a frame (end of last CRC bit of message) at the input pin of the controller, up to provision of the data for the software (including CRC calculation): <= $1T_{ipg_clk_psi5}$
  - From the sync pulse trigger output of the Sync Pulse Generator, up to the output of the sync pulse trigger at the data output pin of the controller: <= 1 µs +$T_{ipg_clk_psi5}$

60.5.2 Reset

The device has a single global asynchronous reset input to the module that resets the whole module bringing it to an inactive state upon its assertion. The user software must program the module registers to bring up the receiver operations.

60.6 Initialization information

60.6.1 Initialization

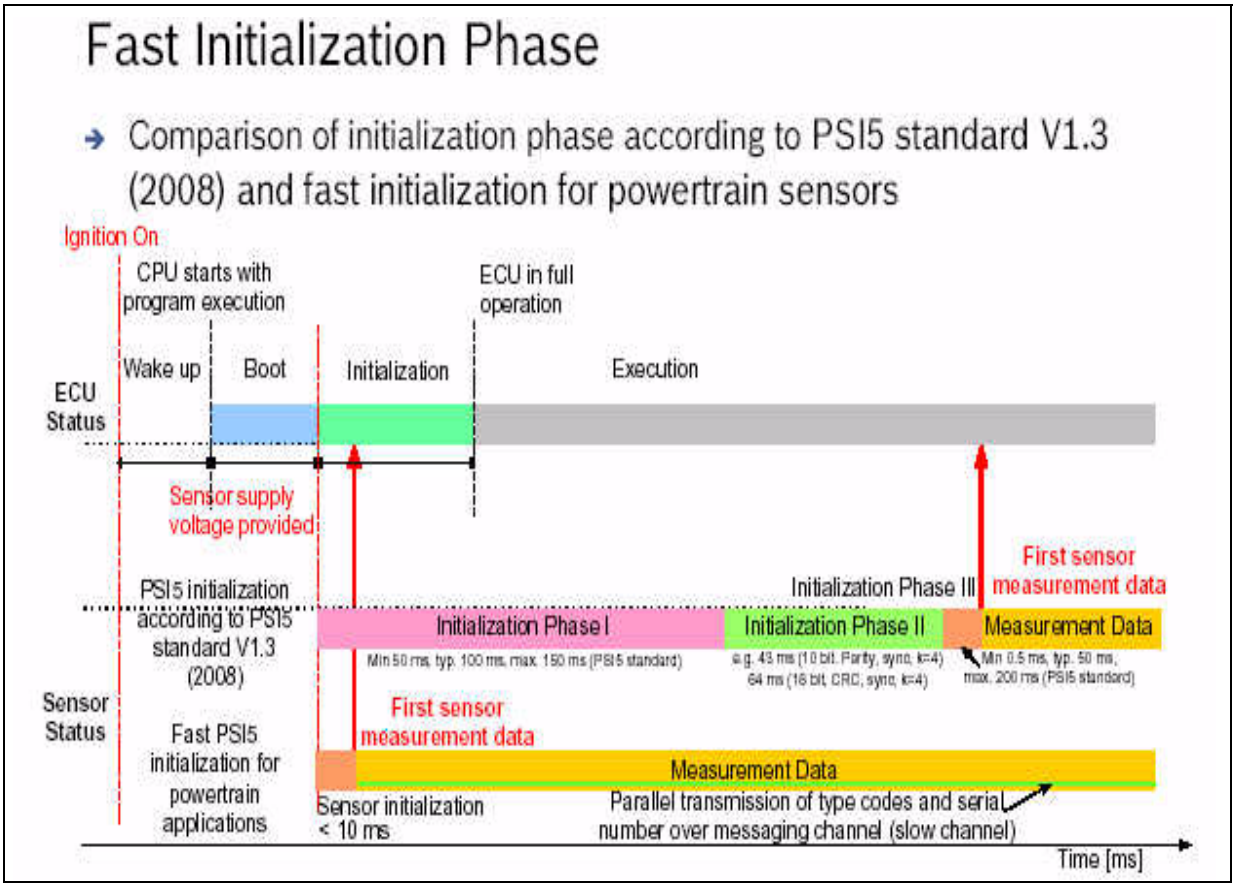
60.6.1.1 Sensor startup and initialization

After each power-on or under-voltage reset, the sensor performs an internal initialization, which is divided into three phases:

- Initialization Phase1: No data transmission.
- Initialization Phase 2: Transmission of type code and serial number of sensor.
- Initialization Phase 3: Transmission of “Sensor Ready” or “Sensor Defect” or other sensor specific data.

As this PSI5 implementation is targeted towards powertrain devices/sensors the standard initialization phase is modified and tuned for those applications. The comparison of the original scheme and enhanced schemes are shown in *Figure 1519*. *Figure 1519* also describes that the initialization content is the same as defined in V1.3, and the only modification is that it is transmitted using the SMC channel when a fast startup is required. “Fast Initialization Phase” below means that the sending of first sensor data is fast, but the sending of sensor initialization data will actually be much slower.

Figure 1519. Fast initialization phase



60.7 Design Assumptions and Limitations

60.7.1 Assumptions

- Slot Configuration Register would always be programmed with valid number of bits for frame configuration.
- There will be no Timing error bits (T) in async mode. Also, in this mode the IP waits indefinitely for the detection of a valid start sequence, hence the F bit (NFR) is not set in Async mode. Rest all of the errors are available (E, EM, and C).

60.7.2 Limitations

As this is a PSI5 receiver, designed primarily to only receive the PSI5/SMC data and store as it is into the RAM buffer. So, it would not be responsible neither to recognize the content of the received messages nor do any decision making based on the content. Similarly, for Command it would not recognize the content of command and would transmit as it is writing into corresponding DSR and reception of PSI5 messages would not in any way depend on the content of command as far as PSI receiver is concerned. Its the responsibility of SW to sequence the issue of commands across the PSI5 channels and thereby recognize the content of received message and infer it.

60.8 Miscellaneous descriptions

60.8.1 Internal SYNC Pulse generation and coordination across PSI5 channels

Each PSI5 channel has the Channel Target Pulse Registers (CTPR) and Channel Initialize Pulse Register (CIPR). The CTPR defines the period of the internal sync pulse generation and the CIPR defines the initial offset time with respect to the sync pulse trigger. The sync pulse trigger can be generated from either the PSI5_GCR[CTC_GED] global bit or the PSI5_PCCR[CTC_ED] channel specific bit.

When staggering of the channels is required then the PSI5_GCR[CTC_GED] global bit is used as a common trigger for all the PSI5 channels in which PSI5_PCCR[CTC_GED_SEL] bit is set to 1. Setting this bit to 1 triggers the internal sync pulse generators of all the PSI5 channels and the trigger of each channel is then automatically staggered with respect to the other channel, depending on the value of the CIPR.

When the specific PSI5 channel is not to be kept as a part of staggered channels, then the channel-specific PSI5_PCCR[CTC_GED_SEL] bit has to be kept as 0. The channel in this case, has to be triggered locally by writing a 1 to the PSI5_PCCR[CTC_ED] of that specific channel. This triggers the internal Sync Pulse Generator of that specific channel without affecting other channels.

Following is the procedure for starting the internal Sync Pulse Generators:

- If in Normal mode, then stop and reset all Channel Trigger Counters (CTCs) by setting the global PSI5_GCR[CTC_GED] bit to 0 or the channel specific PSI5_PCCR[CTC_ED] to 0.
- Enter Config mode.
- In Config mode, load all CTCs with the initialization value (offset) from CIPR.
- In Config mode, load all Counter Target Pulse Registers (CTPR) with the needed sync pulse period.
- Enter Normal mode.
- Send global start trigger (using PSI5_GCR[CTC_GED] bit) to all CTCs for synchronized start for the staggered channels OR send a local start trigger (by using the PSI5_PCCR[CTC_ED]) for channels not staggered.
- If the Compare Unit detects that the value in the CTC is equal to the value in CTPR, the CTC is automatically reset by hardware and restarts with 0.
- In Normal mode, if required, the CTC can be stopped and reset by writing a “0” to the PSI5_GCR[CTC_GED] or the PSI5_PCCR[CTC_ED] as the case may be.
- A sync pulse trigger is created. (See below.)

For the details about the operation of the below interface, refer to *Section 60.3.2.34: Counter Target Pulse Register (PSI5_CH0_CTPR)*, *Section 60.3.2.35: Counter Initialize Pulse Register (PSI5_CH0_CIPR)* and to *Section 60.4.3.4.5: Sync Pulse Trigger Generation*.

Figure 1520. CIPR and CTPR

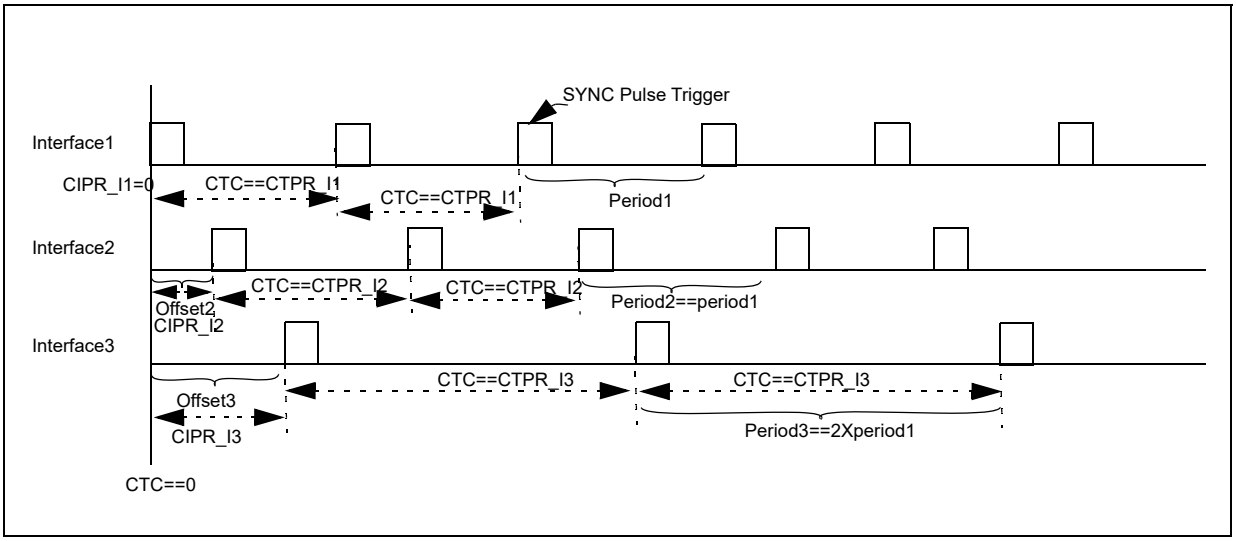


Figure 1520 shows the CIPR and the CTPR definition w.r.t the staggering of the channels. *Figure 1521* is a more detailed representation of these parameters along with the associated jitter, across different channels when these channels are staggered. (The figure also shows some other parameters, such as the modulated pulse width parameters and the Manchester decoder disable parameters.)

Figure 1521. Sync Pulse Generation Parameters and Jitters

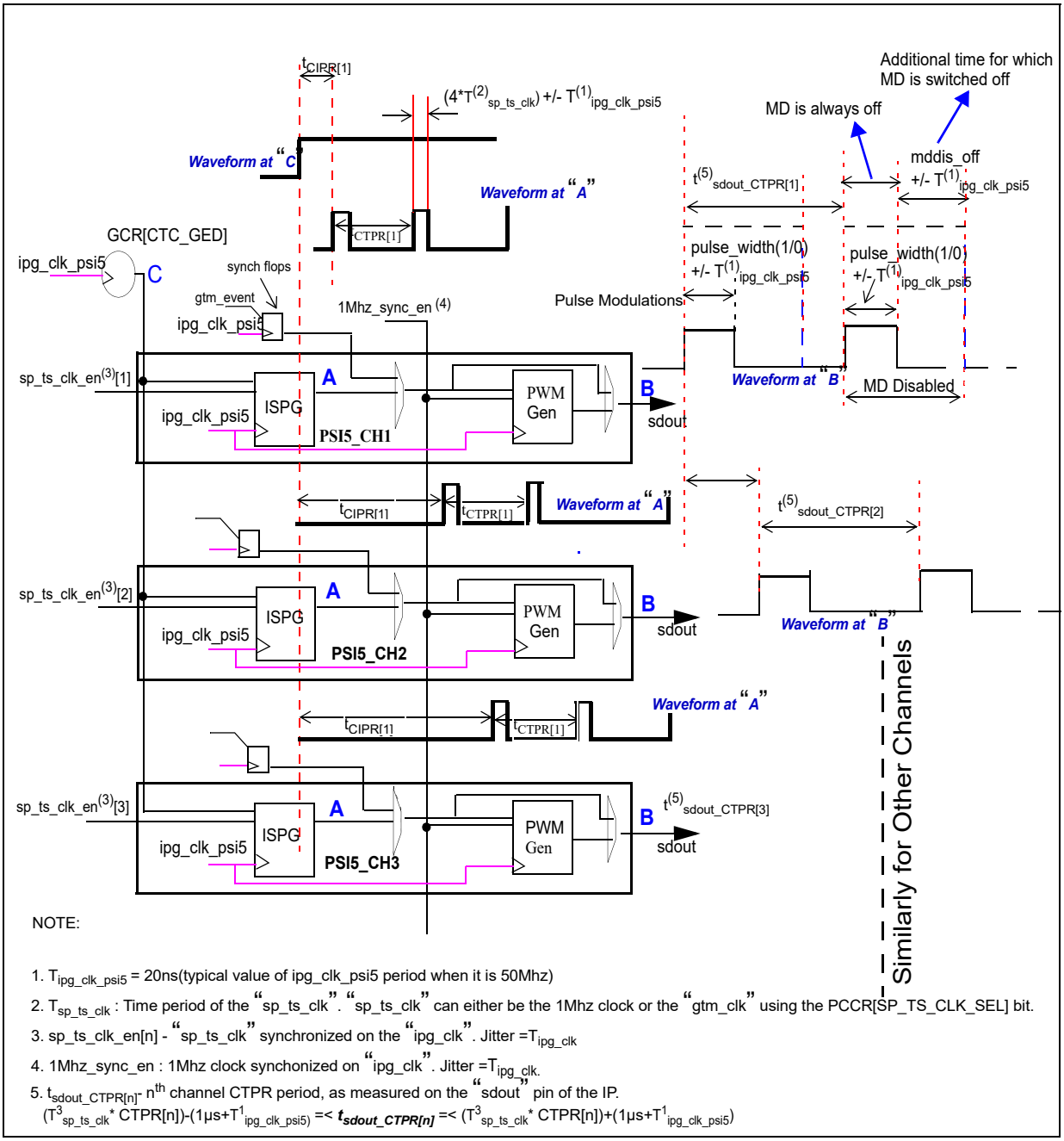
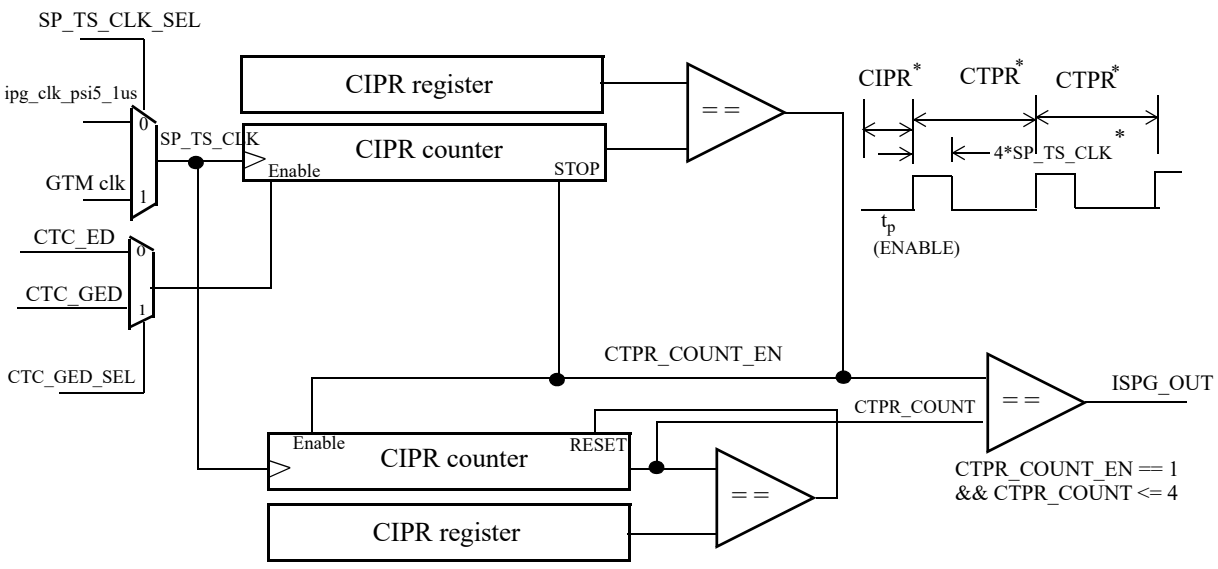


Figure 1521 shows the various jitters.

Figure 1522. ISPG (Internal Sync Pulse Generator) Structure



* - For exact values and more details refer to *Figure 1521*

Note 1: All channels with CTC_GD selected will be enabled at the same time, enabling staggering of the ispg_out signals.
Note 2: This structure is replicated in each channel, with CTC_GED being a common signal to all channels.

60.8.2 Valid states for integrated sync pulse generator

Figure 1523. ECU to Sensor Communication States (part 1)

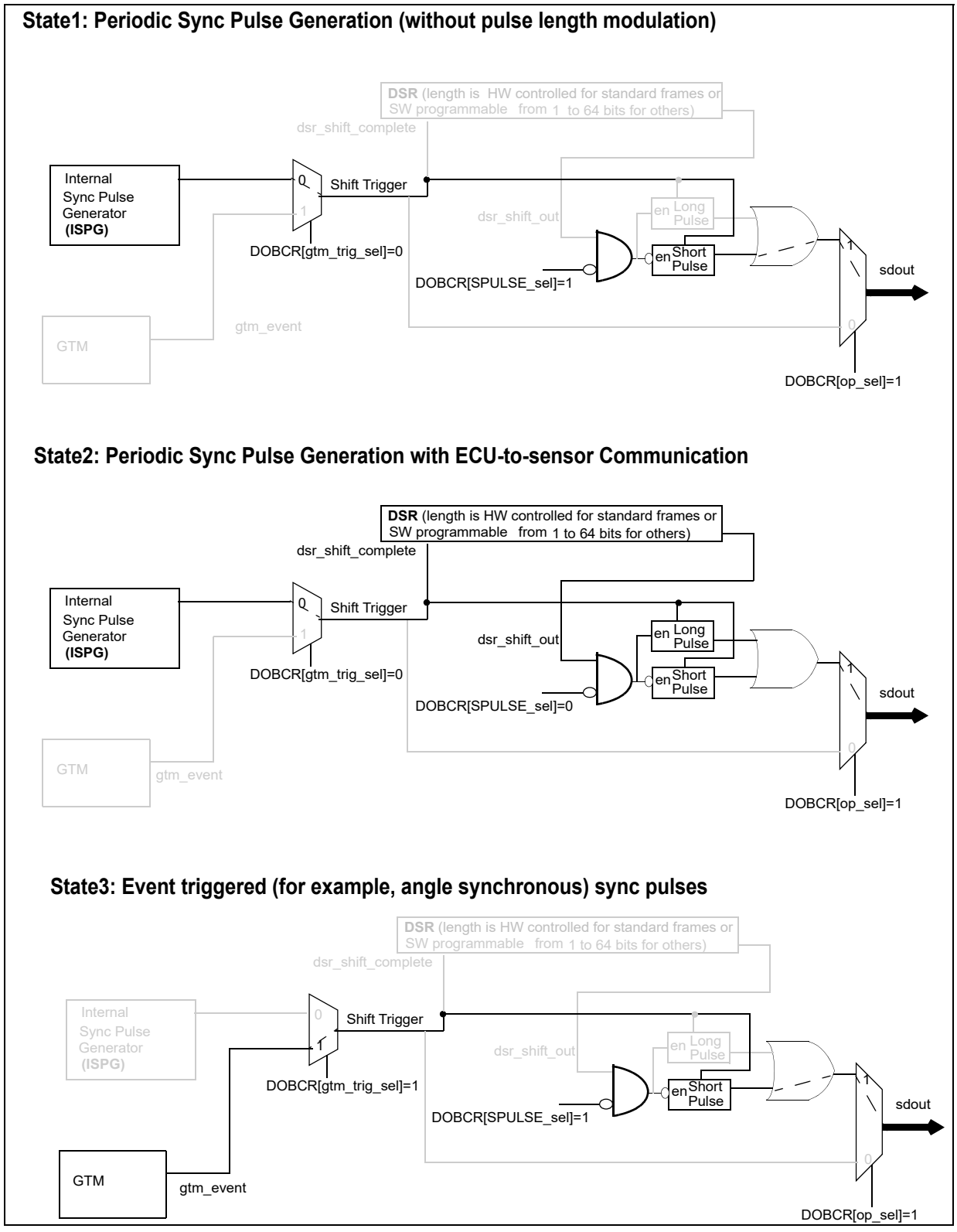
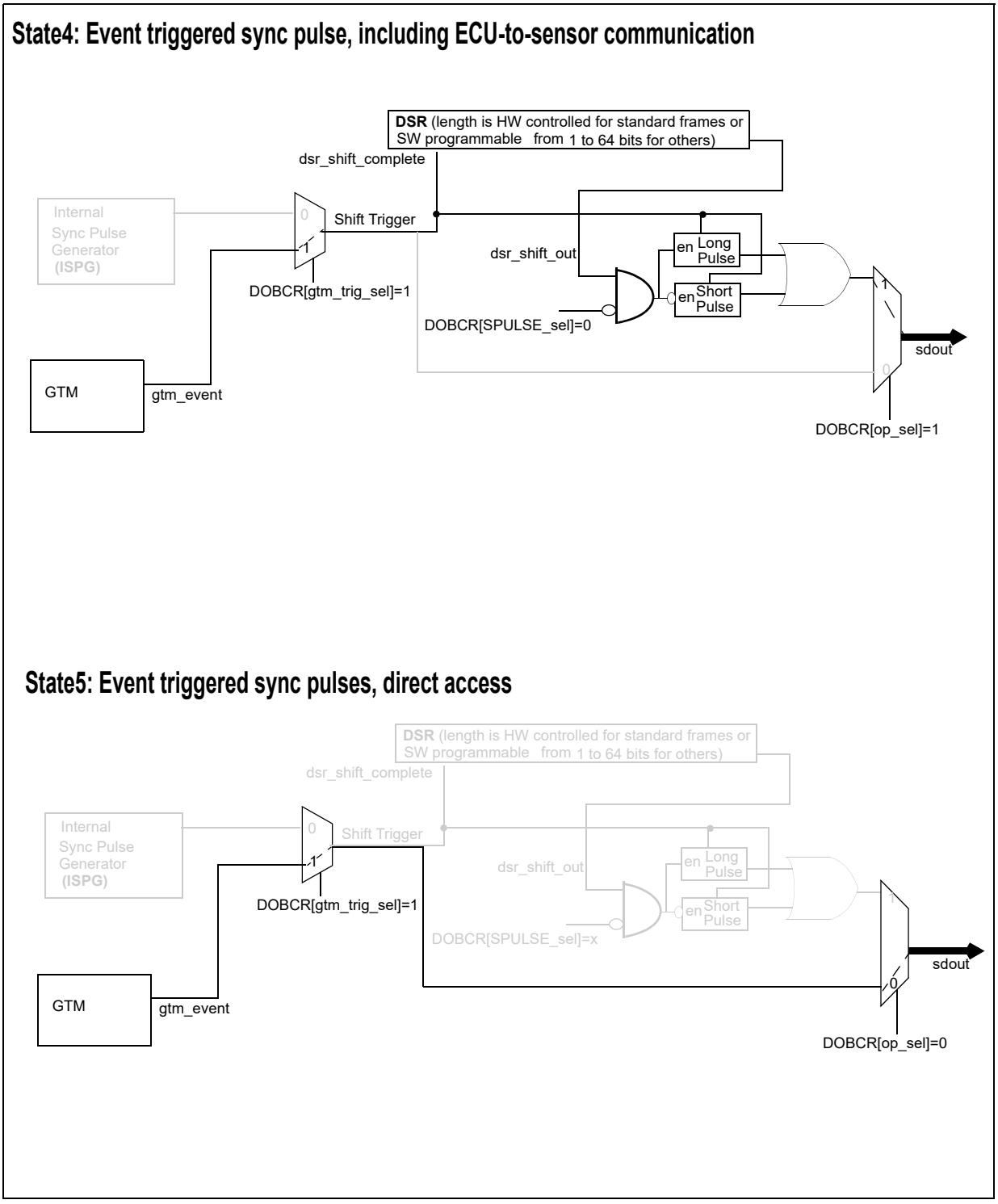
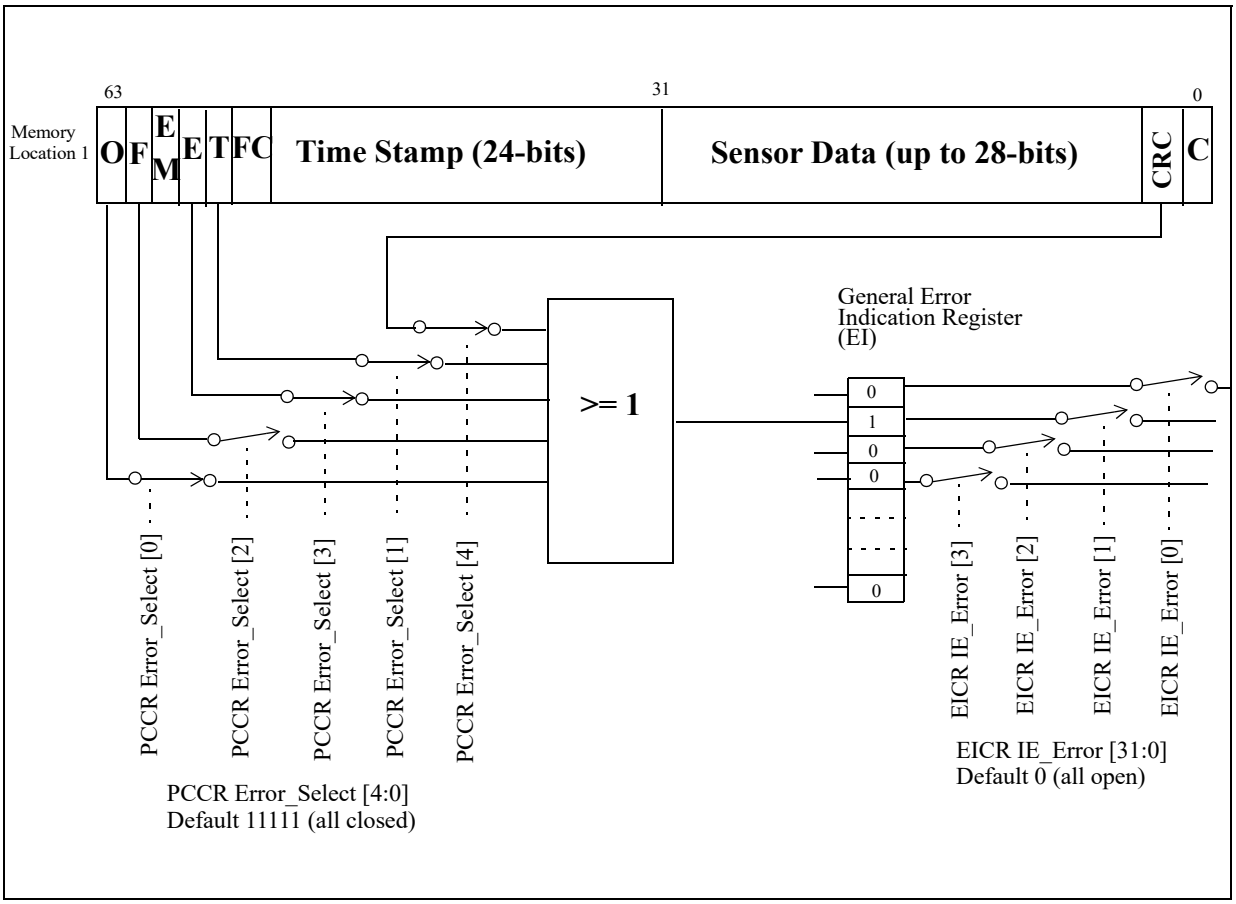


Figure 1524. ECU to Sensor Communication States (part 2)



60.8.3 Implementation of EI Diagnostic Register

Figure 1525. Implementation of EI Diagnostic Register



61 Peripheral Sensor Interface-Support module (PSI5-S)

61.1 Introduction

The PSI5 Support IP (PSI5-S) module is used to interface directly to a UART compatible PSI5 transceiver over a high speed UART link. The transceiver is required to implement a compliant PSI5 module for all vehicle applications as is currently defined in the PSI5 protocol specification V2.0.

PSI5 messages from sensors received by the transceiver are decomposed into UART bytes and sent to the PSI5-S which in turn reconstructs the PSI5 messages from the UART packet, and makes them available for the application in system RAM.

ECU-to-sensor commands are sent from the PSI5-S module to the transceiver, which in turn generates the appropriate logical 1's and 0's in the SYNC pulses sent on the bus. The PSI5-S supports all command formats of the PSI5 V2.0 including the "tooth gap" method and "variable length" pulses.

At the system level, the PSI5-S supports one transceiver with up to seven PSI5 channels. Apart from these seven channels, there is an eighth channel available for special purposes, including direct communication with the transceiver and storage of corrupt messages.

Note: For details on PSI5-S system, refer to Figure 1570.

61.1.1 Overview

PSI5 is an interface for automotive sensor applications. PSI5 is an open standard based on existing sensor interfaces for peripheral airbag sensors, already proven in millions of safety airbag systems. The technical characteristics, the low implementation overhead, as well as the attractive cost make PSI5 also suitable for many other automotive sensor applications. The development goal of PSI5 is a flexible, reliable communication standard for automotive sensor applications that can be used and implemented license free.

The main features of PSI5 are high-speed and high-reliability data transfer at the lowest possible implementation overhead cost, offering a universal and flexible solution for multiple sensor applications.

The main features of the PSI5 standard are:

- Two-wire current interface
- Manchester-coded digital data transmission from sensor to ECU
- High data transmission speed of 125 Kbit/s or optional 189 Kbit/s
- High EMC robustness and low emission
- Wide range of sensor supply current
- Configurable data word length (8 to 28 bits)
- Asynchronous or synchronous operation
- Bidirectional communication

The PSI5-S module can be configured to support the following PSI5 features:

- Configurable data word length (8 to 28 bits)
- Asynchronous or synchronous operation
- Bidirectional communication

The PSI5-S module requires the physical and data link layer of the PSI5 to be implemented outside its interface. It also requires the PSI5 information to be embedded in the form of UART packet, from where it reconstructs the PSI5 message.

The module receives PSI5 message, from up to seven PSI5 channels each with up to six sequential frames. This is achieved via the external UART compliant transceiver. The PSI5 message in each of these frames are decomposed into three-to-six UART bytes (depending on PSI5 data payload) and streamed on a UART link at up to 6.25 Mbit/s to the PSI5-S module. In addition, via the PSI5-S interface, the application sends commands to the transceiver. The transceiver interprets them as either SYNC pulse commands (which will result in short, long or no SYNC pulses on the PSI5 bus), configuration commands (which will result in updates to the transceiver configuration) or diagnostic commands (which will result in the transceiver responding directly over the UART connection). In addition to the seven channels above there is an additional channel, known as Channel0, which is used for special purposes like storing the direct responses from the transceiver or storing the unrecoverable messages.

The PSI5-S consists of three main functional blocks.

1. UART front end: used as PSI5-S front end.
   When PSI5 functionality is not required then it is possible to use this as a standard UART (without LIN) also.
2. Message Recovery Unit: it consists of two state machines.
   – The first state machine interprets the input UART data stream and rebuilds the PSI5 message contained in the up to six UART bytes and also checks for the status of the correct reception of PSI5 message.
   – The second state machine sends SYNC pulse commands via the UART TxBuffer, to the external UART compliant transceiver, which in turn outputs SYNC pulses (short, long, or no SYNC pulse) on the PSI5 bus used to synchronize sensor responses in a bus configuration, and where required, also sends commands to sensors as defined in the PSI5 protocols V2.0.
3. PSI5 Message buffer & interface: It is composed of two MRU buffers, MRU buffer1 and MRU buffer2.
   Both buffers are collectively referred to as MRU output buffer. The MRU buffer1 is internal to the IP and not accessible from outside the IP. It is three word deep. MRU buffer2 is accessible to the application and is 4 word deep. MRU buffer2 can be read by the DMA interface, to transfer the stored PSI5 messages, to Channel specific mailboxes in system RAM directly.

Note: General description:

• The location in system RAM where the messages are stored is referred to as the Mailbox. The mailbox is divided on the basis of channel and each channel is divided on the basis of slots on the PSI5 bus. Thus Mailbox1, location1 would mean Mailbox location for Channel1 and Slot No "1". This is also denoted by MBOX1, LOC1 or $MBOX_1$, LOC_1. This terminology would be followed in the subsequent sections of the doc.

• Each "location" of each mailbox channel is made up of 3x32 bits. The maximum size of each mailbox can be 6 "locations". There are 8 such mailboxes. For channels operating in asynchronous mode, the whole of the Mailbox would act like a ring buffer and all the 6 locations would be filled in that manner.

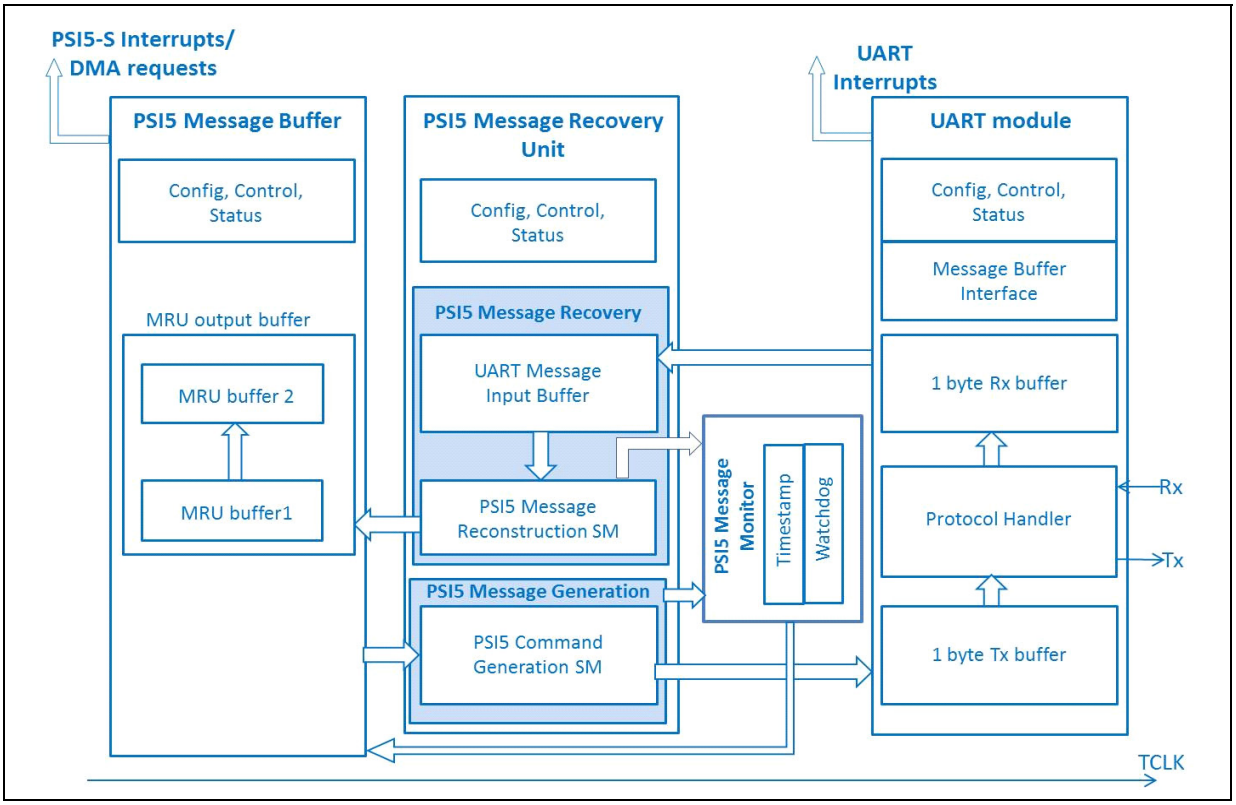
• Mailbox0 only has two locations MBOX0, LOC0 and MBOX0, LOC1. MBOX0, LOC0 is reserved for messages intended for Channel0 (used for transceiver specific messages) while

MBOX0, LOC1 contain "unrecoverable" messages for any channel. Other 4 locations of the MBOX0 are reserved.

- *Also a "UART byte" would mean the "byte" that is received by the UART receiver. "UART packet" would mean the group of UART bytes from which the PSI5 message would be extracted. "PSI5 message" means the message contained in the MRU buffers which contains the actual PSI5 message along with the timestamp and the error status bits.*

Figure 1526 shows the block diagram of the PSI5-S module.

Figure 1526. PSI5-S digital interface block diagram



61.1.2 Features

- IPS interface for register read/write.
- Support for up to 7 PSI5 channels (Channel1 to Channel7) with each channel supporting up to 6 different Frame Streams (Slots). Each of the Frame Streams can be individually configured for Parity/CRC, payload Length, Synchronous/Asynchronous Operation, Timestamp Capture event, Timestamp A or Timestamp B selection.
- Additional Channel0 for special functions.
- Supports bidirectional full duplex operations, via the UART Rx and Tx links.
- Provision of Extended CRC (XCRC) to check the coherence of the UART packet that comprise the PSI5 message.
- Message Reconstruction Unit (MRU) buffer2 (consisting of 4, 32-bit words). Three of these words contain the PSI5 messages (PSI5 message payload, Message Status

flags, Time Stamp) and the fourth one contains the mailbox address to which the three PSI5 words need to be transferred.

- A 24-bit time stamp register, clocked with either the module clock or the GTM clock (gtm_trig). The time stamp appended to the PSI5 message in the MRU, can be configured to be, either the one that is captured when a new PSI5 message is detected, or the one that is captured when the Output ECU to Sensor command is transferred from the UART TX buffer to the UART Tx Shift register.
- IPS interface DMA support, for transferring the data in the MRU output buffer to system memory.
- Downstream Data Shift Register (DDSR) on per channel basis supporting the following:
  - Application can write the ECU to Sensor commands to the DDSR.
  - Supports automatic addition of the start bits/stuff bits /CRC calculation to the contents of DDSR.
  - Provision of resetting and rejecting the contents of DDSR at any stage of command transmission.
- Support for conversion of the logic levels contained in the DDSR, to UART commands to be sent to the transceiver.
- Provision of writing the Software calculated commands directly to the UART TX Buffer. This can be used for direct writing of the transceiver status or diagnostic commands, or both.
- Configurable UART message idle time counter, that indicates to the PSI5 MRU when an idle time at least as long as the configured length (number of bit times), has occurred on the Rx line.
- Provision of configuring the PSI5-S front-end as a standalone UART module, when the PSI5-S module is disabled. In this mode the module, does not have any relation with the PSI5 protocol.
- Support of up to 6.25 Mbit/s UART Rx Baud Rate.
- An output clock for the PSI5 transceiver UART interface.

61.1.3 PSI5-S global modes of operation

The PSI5-S module can be configured in 4 different global modes of operation. These are UART_STDALONE, PS_DISABLE, PS_CONFIG and PS_NORMAL. These global modes can be entered by programming the PS_GLCR[GLOBAL_MODE] bits appropriately. *Figure 1527* shows the state transition diagram for these bits.

The PS_NORMAL, PS_CONFIG and PS_DISABLE are collectively referred to as the “PSI5-S” modes (refer to *Figure 1527*). Once the PS_GLCR[GLOBAL_MODE] bits are appropriately configured for the “PSI5-S” modes, the information that the intended mode of operation is fully achieved is reflected by the setting of the PS_GLSR[GL_MODETR_DONE] status bit. An interrupt can also be generated for this event, in case the PS_GLCR[GL_MODETR_DONE_EN] bit is set to “1”. Thus, only when PS_GLSR[GL_MODETR_DONE] goes to “1” it should be considered that the IP has entered the desired “PSI5-S” mode.

This bit is reset to “0” when the mode is changed from any of the PSI5-S modes to the UART_STDALONE mode and the corresponding interrupt is automatically deasserted.

Following are the descriptions of the different global modes.

61.1.3.1 Standalone UART mode (UART_STDALONE mode)

This is the default state after the module reset. In this mode, only the PS_GLCR[GLOBAL_MODE] register bits and the registers of the UART module are accessible for writing. This is the mode which is to be used when the PSI5-S front end has to be used as a standalone UART. This state indicates that the UART will now function as a standard UART and the PSI5-S core will be disabled.

61.1.3.2 Disable mode (PS_DISABLE)

PS_Disable mode in the PSI5-S reduces power consumption. In this mode the RX, TX and common time base counters are disabled for any operation. This mode is entered by the specific combination of PS_GLCR[GLOBAL_MODE] bits. Refer to *Table 1544*, for the details of these bits. Note that when entering PS_Disable mode from any other mode, the register contents are retained and are not changed. The timestamp counter is however reset when entering Disable mode.

In PS_Disable mode, only PS_GLCR[GLOBAL_MODE] register bits are accessible for read/write. Accessing any other register bits will generate transfer error.

There is no software reset bit in the PSI5 module. The same has to be managed outside the IP at the system level. Channels will enter “Configuration/Normal mode” based on the value of the PS_GLCR[GLOBAL_MODE] bits.

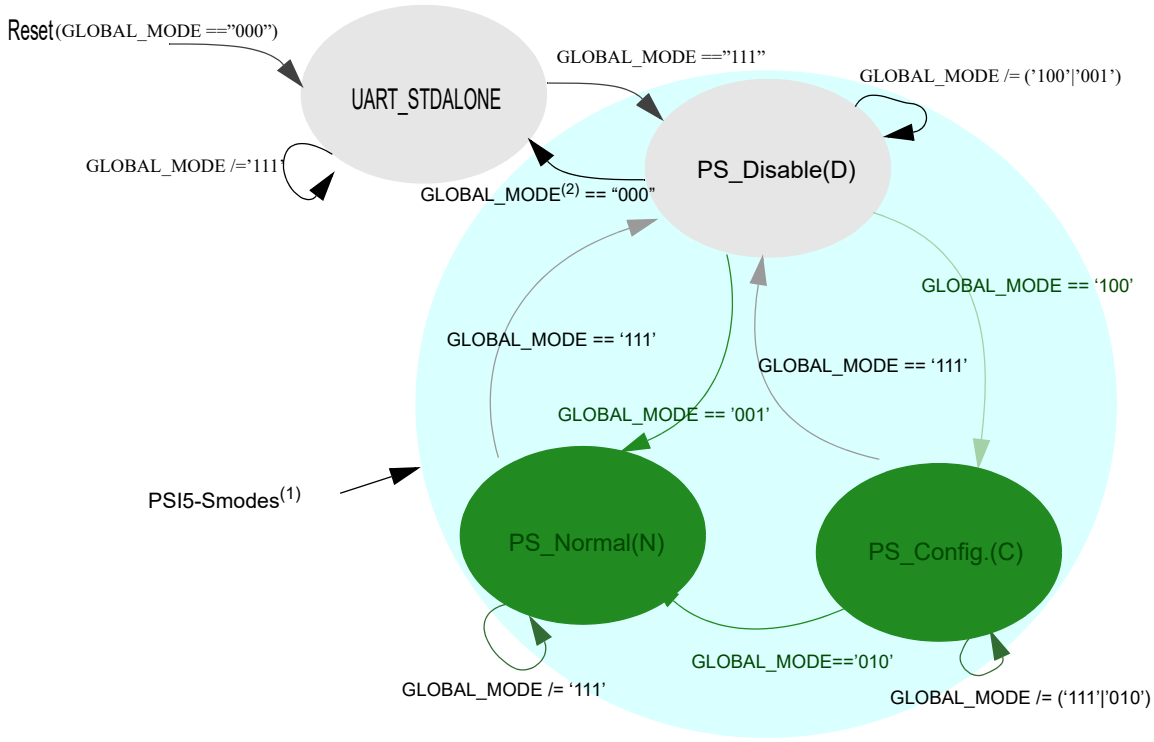
61.1.3.3 Configuration mode (PS_CONFIG mode)

In the configuration mode all PSI5-S global, channel specific configuration/control registers and UART registers can be written. The receiver and transmitter sub-blocks remain disabled in this mode. This mode is entered by the specific combination of PS_GLCR[GLOBAL_MODE] bits. Refer to *Table 1544*, for the details of these bits.

61.1.3.4 Normal mode (PS_NORMAL mode)

In PS_NORMAL mode, the receiver and the transmitter blocks become active, error handling is enabled, UART starts its operation and all the PSI5 protocol functions are also enabled. In this mode, all the PSI5-S global and channel-specific configuration registers are locked for writing, although these registers can still be read. This mode is entered by the specific combination of PS_GLCR[GLOBAL_MODE] bits. Refer to *Table 1544* for the details of these bits. Note that there are still some register bits that can be written in this mode also (refer to the detailed register description).

Figure 1527. Operating modes



1. PSI5-S module can be configured in 4 different Global modes of operation. These are UART_STDALONE, PS_NORMAL, PS_CONFIG and PS_DISABLE. The PS_NORMAL,PS_CONFIG and PS_DISABLE are collectively referred to as “PSI5-S” modes.
2. PS_GLSR[GL_MODETR_DONE] status bit is reset to “0” when the transition occurs from PS_DISABLE to UART_STDALONE mode.

Table 1544. State transition Table

Present State	PS_GLCR (State Variables)			Next State
	PS_GLCR[GLOBAL_MODE(2)]	PS_GLCR[GLOBAL_MODE(1)]	PS_GLCR[GLOBAL_MODE(0)]	
UART_STDALONE	1	1	1	PS_Disable
PS_Disable	0	0	0	UART_STDALONE
PS_Disable	0	0	1	PS_Normal
PS_Disable	1	0	0	PS_Config
PS_Config	1	1	1	PS_Disable
PS_Config	0	1	0	PS_Normal
PS_Normal	1	1	1	PS_Disable

61.1.4 PSI5-S system modes of operation

61.1.4.1 Stop mode

For PSI5-S modes:

In this mode, upon the STOP request from the system level the IP generates a system level acknowledgment with the information that the clocks to the IP can be switched off. The process of the same is as described below.

At the PSI5-S Rx side, as soon as stop mode request is received by the IP then the current PSI5 message under construction (in the internal PSI5 message extraction logic) is immediately saved to the MRU output buffer1. Thus the assertion of the stop mode acts like a “forced” flushing for the current PSI5 message completion. Depending on the state of the PSI5 message under construction, the resultant PSI5 message might or might not have errors. Any byte that is being received by the UART is henceforth rejected. The internal PSI5 message extraction logic goes back to an “idle” state and remains so till the STOP mode request remains asserted.

At the PSI5-S Tx side, on the reception of the stop mode request, the internal sync pulse generation counters are stopped immediately (DDSR trigger counters) and no further command bits are shifted onto the UART Tx side. The UART Tx completes the current byte that is being transmitted and enters the idle state. No further transmissions take place. On deassertion of the stop mode, once the clock restarts, the transmission of the commands resumes from where they had halted.

Once both the above processes are completed the IP generates acknowledgment for switching of the clocks to the PSI5-S IP. Also, refer to *Section 61.4.2.2: PSI5 “recoverable and unrecoverable” messages*.

For UART STDALONE mode, refer to *Chapter 63: LINFlexD*.

61.1.4.2 Debug mode

The IP enters this mode on system request. Any register can be read or written to by the debugger. The IP can enter the debug mode in addition to it being in any of the other modes (UART Standalone/Disable/Normal/Config). In this mode the hardware clearing of the register PS_WDGTSSR remains disabled and any read of this register has no effect on the contents of this register. Apart from this there is no other difference as to whether the IP is in debug mode or not in the debug mode of operation.

61.2 External signal description

Table 1545. External signal description

S.No	Name	Function	I/O	Pull-up
Signals to/from pads				
1.	ipp_ind_linrx	LINUART receive pin. Used for Receiving UART packets from transceiver which contain information to enable the rebuilding of the PSI5 message within the packet of between three and six UART bytes	I	Active

Table 1545. External signal description (continued)

S.No	Name	Function	I/O	Pull-up
2.	ipp_do_lintx	LINUART transmit pin. Used for transmitting command bytes from TxFIFO to transceiver for generation of SYNC pulses (long, short or no) on the PSI5 bus or direct diagnostic/status commands for the transceiver itself	O	Passive
3.	ipp_port_en_tx	LINUART transmission port enable: Connected to the enable pin of the pad which is to be used for “LINTX” functionality. When “1” it indicates that the IP is doing transmit of data.	O	Passive
4.	ipp_psi5_uart_tclk	Transceiver clock output, which drives the UART interface on the transceiver. The clock output frequency is configurable between 4 and 8 x UART baud rate on the interface between PSI5-S and transceiver.Frequency can be controlled at the system level through the Clock Generator Module (MC_CGM). The duty cycle of the transceiver clock is between 40% and 60%	O	Passive

61.3 Memory map and register description

61.3.1 Memory map

Table 1546 shows the memory map.

Table 1546. PSI5-S memory map

Offset (Hex)	Register	Access(1)			Reset value(2)	Location
		D	C	N		
0x0000	PSI5-S LIN Control register 1 (PS_LINCR1)	R	R/W	R	0x0000_0092	*Section 61.3.3.1*
0x0004	PSI5-S LIN Interrupt Enable register (PS_LINIER)	R	R/W	R	0x0000_0000	*Section 61.3.3.2*
0x0008	PS_LINSR (PS_NOUSE_REG(3))	Read value in PSI5-S modes is “dont care”. Also see (4) below.			0x0000_0040	*Section 61.3.3.3*

Table 1546. PSI5-S memory map (continued)

Offset (Hex)	Register	Access[1]			Reset value[2]	Location
		D	C	N		
0x000C–0x000F	Reserved	—	—	—	0x0000_0000	—
0x0010	PSI5-S UART Mode Control register (PS_UARTCR)	R	R/W	R	0x0000_0000	*Section 61.3.3.4*
0x0014	PSI5-S UART Mode Status register (PS_UARTSR)	R	R/W1C	R/W1C	0x0000_0000	*Section 61.3.3.5*
0x0018–0x001B	Reserved	—	—	—	0x0000_0200	—
0x001C–0x001F	Reserved	—	—	—	0x0000_FFFF	—
0x0020–0x0023	Reserved	—	—	—	0x0000_0E2C	—
0x0024	PSI5-S LIN Fractional Baud Rate register (PS_LINFBRR)	R	R/W	R	0x0000_0000	*Section 61.3.3.6*
0x0028	PSI5-S LIN Integer Baud Rate register (PS_LINIBRR)	R	R/W	R	0x0000_0000	*Section 61.3.3.7*
0x002C–0x002F	Reserved	—	—	—	0x0000_0000	—
0x0030	PSI5-S LIN Control register 2 (PS_LINCR2)	R	R/W	R	0x0000_4000	*Section 61.3.3.8*
0x0034–0x0037	Reserved	—	—	—	0x0000_0000	—
0x0038	PSI5-S Buffer Data Register Least Significant (PS_BDRL)	R	R/W	R	0x0000_0000	*Section 61.3.3.9*
0x003C	PSI5-S Buffer Data Register Most Significant (PS_BDRM)	R	R/W	R	0x0000_0000	*Section 61.3.3.10*
0x0040–0x0043	Reserved	—	—	—	0x0000_0000	—
0x0044–0x0047	Reserved	—	—	—	0x0000_0000	—
0x0048–0x004B	Reserved	—	—	—	0x0000_0000	—
0x004C	PSI5-S Global Control register (PS_GCR)	R	R/W	R	0x0000_0000	*Section 61.3.3.11*

Table 1546. PSI5-S memory map (continued)

Offset (Hex)	Register	Access[1]			Reset value[2]	Location
		D	C	N		
0x0050	PSI5-S UART Preset Timeout register (PS_UARTPTO)	R	R/W	R	0x00000FFF	*Section 61.3.3.12*
0x0054	UPSI5-S ART Current Timeout register (PS_UARTCTO)	R	R	R	0x0000_0000	*Section 61.3.3.13*
0x0058	DMATXE (PS_NOUSE_REG[3])	For PSI5-S modes always program with 0x0000_00000 in Config Mode. Also see [4] below.			0x0000_0000	*Section 61.3.3.14*
0x005C	DMARXE (PS_NOUSE_REG[3])	For PSI5-S modes always program with 0x0000_00000 in Config Mode. Also see [4] below.			0x0000_0000	*Section 61.3.3.15*
0x0060	PS_PTD	R	R/W	R	0x0000_0000	*Section 61.3.3.16*
0x0064–0x009F	Reserved for UART registers					
0x00A0–0x00B3	Reserved for UART registers					
PSI5-S GLOBAL REGISTERS						
0x00B4	Global Control Register (PS_GLCR)	R/W[5]	R/W	R/W[5]	0x0000_0000	*Section 61.3.4.1*
0x00B8	Global Status Register (PS_GLSR)	R/W1C[5]	R/W1C[5]	R/W1C[5]	0x0010_0000	*Section 61.3.4.2*
0x00BC	Channel_Base_Address (PS_CH_BASE_ADDR)	R	R/W	R	0x0000_0000	*Section 61.3.4.3*
0x00C0	MRU OUTPUT BUFFER2 REGISTER0 (PS_MRU_BUF2_REG0)	R	R	R	0x0000_0000	*Section 61.3.4.4*
0x00C4	MRU OUTPUT BUFFER2 REGISTER1 (PS_MRU_BUF2_REG1)	R	R	R	0x0000_0000	*Section 61.3.4.5*
0x00C8	MRU OUTPUT BUFFER2 REGISTER2 (PS_MRU_BUF2_REG2)	R	R	R	0x0000_0000	*Section 61.3.4.6*
0x00CC	MRU OUTPUT BUFFER2 REGISTER3 (PS_MRU_BUF2_REG3)	R	R	R	0x0000_0000	*Section 61.3.4.7*
0x00D0–0x00DF	Reserved					—

Table 1546. PSI5-S memory map (continued)

Offset (Hex)	Register	Access[1]			Reset value[2]	Location
		D	C	N		
0x00E0	PSI5-S Mbox Status Irq (PS_MBOX_SR_IRQ)	R	R	R	0x0000_0000	*Section 61.3.4.8*
0x00E4	PSI5-S Error Status Irq (PS_ERR_SR_IRQ)	R	R/W1C	R/W1C	0x0000_0000	*Section 61.3.4.9*
0x00E8	PSI5-S Mbox select Irq0 (PS_MBOX_SEL_IRQ0)	R	R/W	R/W	0x0000_0000	*Section 61.3.4.10*
0x00EC	PSI5-S Error Select Irq0 (PS_ERR_SEL_IRQ0)	R	R/W	R/W	0x0000_0000	*Section 61.3.4.11*
0x00F0	PSI5-S Mbox Select Irq1 (PS_MBOX_SEL_IRQ1)	R	R/W	R/W	0x0000_0000	*Section 61.3.4.11*
0x00F4	PSI5-S Error Select Irq1 (PS_ERR_SEL_IRQ1)	R	R/W	R/W	0x0000_0000	*Section 61.3.4.11*
0x00F8	PSI5-S Mbox Select Irq2 (PS_MBOX_SEL_IRQ2)	R	R/W	R/W	0x0000_0000	*Section 61.3.4.11*
0x00FC	PSI5-S Error Select Irq2 (PS_ERR_SEL_IRQ2)	R	R/W	R/W	0x0000_0000	*Section 61.3.4.11*
0x0100	PSI5-S Mbox Select Irq3 (PS_MBOX_SEL_IRQ3)	R	R/W	R/W	0x0000_0000	*Section 61.3.4.11*
0x0104	PSI5-S Error Select Irq3 (PS_ERR_SEL_IRQ3)	R	R/W	R/W	0x0000_0000	*Section 61.3.4.11*
0x0108	PSI5-S Mbox Select Irq4 (PS_MBOX_SEL_IRQ4)	R	R/W	R/W	0x0000_0000	*Section 61.3.4.11*
0x010C	PSI5-S Error Enable Irq4 (PS_ERR_SEL_IRQ4)	R	R/W	R/W	0x0000_0000	*Section 61.3.4.11*
0x0110	PSI5-S Mbox Select Irq5 (PS_MBOX_SEL_IRQ5)	R	R/W	R/W	0x0000_0000	*Section 61.3.4.11*
0x0114	PSI5-S Error Select Irq5 (PS_ERR_SEL_IRQ5)	R	R/W	R/W	0x0000_0000	*Section 61.3.4.11*
0x0118	PSI5-S Mbox Select Irq6 (PS_MBOX_SEL_IRQ6)	R	R/W	R/W	0x0000_0000	*Section 61.3.4.11*
0x011C	PSI5-S Error Select Irq6 (PS_ERR_SEL_IRQ6)	R	R/W	R/W	0x0000_0000	*Section 61.3.4.11*
0x0120	PSI5-S Mbox Select Irq7 (PS_MBOX_SEL_IRQ7)	R	R/W	R/W	0x0000_0000	*Section 61.3.4.11*

Table 1546. PSI5-S memory map (continued)

Offset (Hex)	Register	Access[1]			Reset value[2]	Location
		D	C	N		
0x0124	PSI5-S Error Select Irq7 (PS_ERR_SEL_IRQ7)	R	R/W	R/W	0x0000_0000	*Section 61.3.4.11*
0x0128	PSI5-S Watchdog Frame Error Status and Watchdog Frame Errror Timestamp capture register (PS_WDGTSSR)	R	R/ HC[6]	R	0x0000_0000	*Section 61.3.4.12*
0x012C	PSI5-S ECU to Sensor Direct Command write register (PS_DIRCMD)	R	R/W	R/W	0x0000_0000	*Section 61.3.4.13*
0x0130–0x016B	Reserved					—
Channel0						
0x016C	PSI5-S channel0 message configuration register A (PS_MSGA_CH0)	R	R/W	R	0x0000_0051	*Section 61.3.4.14*
0x0170	PSI5-S channel0 message configuration register B (PS_MSGB_CH0)	R	R/W	R	0x0000_0008	*Section 61.3.4.15*
0x0174–0x0177	Reserved					—
0x0178	PSI5-S Mailbox status register channel0 (PS_MBOX_SR_CH0)	R	R/W1C	R/W1C	0x0000_0000	*Section 61.3.4.16*
0x017C–0x018F	Reserved					—
Channel1						
0x0190	PSI5-S channel1 message configuration register A (PS_MSGA_CH1)	R	R/W	R	0x0000_0000	*Section 61.3.4.17*
0x0194	PSI5-S channel1 message configuration register B (PS_MSGB_CH1)	R	R/W	R	0x1084_2108	*Section 61.3.4.18*
0x0198–0x019B	Reserved					—
0x019C	PSI5-S Mailbox status register channel1 (PS_MBOX_SR_CH1)	R	R/W1C	R/W1C	0x0000_0000	*Section 61.3.4.19*
0x01A0	PSI5-S channel1 watchdog configuration register (PS_WD_CFGR_CH1)	R	R/W[5]	R/W[5]	0x0000_0000	*Section 61.3.4.20*
0x01A4	PSI5-S DDSR Trigger offset register channel1 (PS_DDTRIG_OFFR_CH1)	R	R/W	R	0x0000_0000	*Section 61.3.4.21*

Table 1546. PSI5-S memory map (continued)

Offset (Hex)	Register	Access[1]			Reset value[2]	Location
		D	C	N		
0x01A8	PSI5-S DDSR Trigger period register channel1 (PS_DDTRIG_PERR_CH1)	R	R/W	R	0x0000_0000	*Section 61.3.4.22*
0x01AC	PSI5-S channel1 ECU to Sensor Control register (PS_E2SCR_CH1)	R	R/W	R/W[5]	0x0000_0800	*Section 61.3.4.23*
0x01B0	PSI5-S channel1 ECU to Sensor Status register (PS_E2SSR_CH1)	R	R/W1C	R/W1C	0x0000_0020	*Section 61.3.4.24*
0x01B4	PSI5-S channel1 ECU to Sensor Downstream Data Shift register (PS_DDSR_H_CH1)	R	R	R	0x0000_0000	*Section 61.3.4.25*
0x01B8	PSI5-S channel1 ECU to Sensor Downstream Data Shift register (PS_DDSR_L_CH1)	R	R/W[5]	R/W[5]	0x0000_0000	*Section 61.3.4.26*
0x01BC – 0x01CB	Reserved					—
Channel2						
0x01CC	PSI5-S channel1 message configuration register A (PS_MSGA_CH2)	R	R/W	R	0x0000_0000	*Section 61.3.4.17*
0x01D0	PSI5-S channel1 message configuration register B (PS_MSGB_CH2)	R	R/W	R	0x1084_2108	*Section 61.3.4.18*
0x01D4–0x01D7	**Reserved**					—
0x01D8	PSI5-S Mailbox status register channel2 (PS_MBOX_SR_CH2)	R	R/W1C	R/W1C	0x0000_0000	*Section 61.3.4.19*
0x01DC	PSI5-S channel1 watchdog configuration register (PS_WD_CFGR_CH2)	R	R/W[5]	R/W[5]	0x0000_0000	*Section 61.3.4.20*
0x01E0	PSI5-S DDSR Trigger offset register channel1 (PS_DDTRIG_OFFR_CH2)	R	R/W	R	0x0000_0000	*Section 61.3.4.21*
0x01E4	PSI5-S DDSR Trigger period register channel1 (PS_DDTRIG_PERR_CH2)	R	R/W	R	0x0000_0000	*Section 61.3.4.22*
0x01E8	PSI5-S channel1 ECU to Sensor Control register (PS_E2SCR_CH2)	R	R/W	R/W[5]	0x0000_0800	*Section 61.3.4.23*
0x01EC	PSI5-S channel1 ECU to Sensor Status register (PS_E2SSR_CH2)	R	R/W1C	R/W1C	0x0000_0020	*Section 61.3.4.24*

Table 1546. PSI5-S memory map (continued)

Offset (Hex)	Register	Access[1]			Reset value[2]	Location
		D	C	N		
0x01F0	PSI5-S channel1 ECU to Sensor Downstream Data Shift register (PS_DDSR_H_CH2)	R	R	R	0x0000_0000	*Section 61.3.4.25*
0x01F4	PSI5-S channel1 ECU to Sensor Downstream Data Shift register (PS_DDSR_L_CH2)	R	R/W[5]	R/W[5]	0x0000_0000	*Section 61.3.4.26*
0x01F8–0x0207	Reserved					—
Channel3						
0x0208	PSI5-S channel1 message configuration register A (PS_MSGA_CH3)	R	R/W	R	0x0000_0000	*Section 61.3.4.17*
0x20C	PSI5-S channel1 message configuration register B (PS_MSGB_CH3)	R	R/W	R	0x1084_2108	*Section 61.3.4.18*
0x0210–0x0213	Reserved					—
0x0214	PSI5-S Mailbox status register channel3 (PS_MBOX_SR_CH3)	R	R/W1C	R/W1C	0x0000_0000	*Section 61.3.4.19*
0x0218	PSI5-S channel1 watchdog configuration register (PS_WD_CFGR_CH3)	R	R/W[5]	R/W[5]	0x0000_0000	*Section 61.3.4.20*
0x21C	PSI5-S DDSR Trigger offset register channel1 (PS_DDTRIG_OFFR_CH3)	R	R/W	R	0x0000_0000	*Section 61.3.4.21*
0x0220	PSI5-S DDSR Trigger period register channel1 (PS_DDTRIG_PERR_CH3)	R	R/W	R	0x0000_0000	*Section 61.3.4.22*
0x0224	PSI5-S channel1 ECU to Sensor Control register (PS_E2SCR_CH3)	R	R/W	R/W[5]	0x0000_0800	*Section 61.3.4.23*
0x0228	PSI5-S channel1 ECU to Sensor Status register (PS_E2SSR_CH3)	R	R/W1C	R/W1C	0x0000_0020	*Section 61.3.4.24*
0x022C	PSI5-S channel1 ECU to Sensor Downstream Data Shift register (PS_DDSR_H_CH3)	R	R	R	0x0000_0000	*Section 61.3.4.25*
0x0230	PSI5-S channel1 ECU to Sensor Downstream Data Shift register (PS_DDSR_L_CH3)	R	R/W[5]	R/W[5]	0x0000_0000	*Section 61.3.4.26*
0x0234–0x0243	Reserved					—

Table 1546. PSI5-S memory map (continued)

Offset (Hex)	Register	Access[1]			Reset value[2]	Location
		D	C	N		
Channel4						
0x0244	PSI5-S channel1 message configuration register A (PS_MSGA_CH4)	R	R/W	R	0x0000_0000	*Section 61.3.4.17*
0x0248	PSI5-S channel1 message configuration register B (PS_MSGB_CH4)	R	R/W	R	0x1084_2108	*Section 61.3.4.18*
0x024C–0x024F	Reserved					—
0x0250	PSI5-S Mailbox status register channel4 (PS_MBOX_SR_CH4)	R	R/W1C	R/W1C	0x0000_0000	*Section 61.3.4.19*
0x0254	PSI5-S channel1 watchdog configuration register (PS_WD_CFGR_CH4)	R	R/W[5]	R/W[5]	0x0000_0000	*Section 61.3.4.20*
0x0258	PSI5-S DDSR Trigger offset register channel1 (PS_DDTRIG_OFFR_CH4)	R	R/W	R	0x0000_0000	*Section 61.3.4.21*
0x025C	PSI5-S DDSR Trigger period register channel1 (PS_DDTRIG_PERR_CH4)	R	R/W	R	0x0000_0000	*Section 61.3.4.22*
0x0260	PSI5-S channel1 ECU to Sensor Control register (PS_E2SCR_CH4)	R	R/W	R/W[5]	0x0000_0800	*Section 61.3.4.23*
0x0264	PSI5-S channel1 ECU to Sensor Status register (PS_E2SSR_CH4)	R	R/W1C	R/W1C	0x0000_0020	*Section 61.3.4.24*
0x268	PSI5-S channel1 ECU to Sensor Downstream Data Shift register (PS_DDSR_H_CH4)	R	R	R	0x0000_0000	*Section 61.3.4.25*
0x26C	PSI5-S channel1 ECU to Sensor Downstream Data Shift register (PS_DDSR_L_CH4)	R	R/W[5]	R/W[5]	0x0000_0000	*Section 61.3.4.26*
0x0270–0x027F	Reserved					—
Channel5						
0x0280	PSI5-S channel1 message configuration register A (PS_MSGA_CH5)	R	R/W	R	0x0000_0000	*Section 61.3.4.17*
0x0284	PSI5-S channel1 message configuration register B (PS_MSGB_CH5)	R	R/W	R	0x1084_2108	*Section 61.3.4.18*
0X0288–0x028B	Reserved					—

Table 1546. PSI5-S memory map (continued)

Offset (Hex)	Register	Access[1]			Reset value[2]	Location
		D	C	N		
0x028C	PSI5-S Mailbox status register channel5 (PS_MBOX_SR_CH5)	R	R/W1C	R/W1C	0x0000_0000	*Section 61.3.4.19*
0X0290	PSI5-S channel1 watchdog configuration register (PS_WD_CFGR_CH5)	R	R/W[5]	R/W[5]	0x0000_0000	*Section 61.3.4.20*
0X0294	PSI5-S DDSR Trigger offset register channel1 (PS_DDTRIG_OFFR_CH5)	R	R/W	R	0x0000_0000	*Section 61.3.4.21*
0x0298	PSI5-S DDSR Trigger period register channel1 (PS_DDTRIG_PERR_CH5)	R	R/W	R	0x0000_0000	*Section 61.3.4.22*
0x029C	PSI5-S channel1 ECU to Sensor Control register (PS_E2SCR_CH5)	R	R/W	R/W[5]	0x0000_0800	*Section 61.3.4.23*
0x2A0	PSI5-S channel1 ECU to Sensor Status register (PS_E2SSR_CH5)	R	R/W1C	R/W1C	0x0000_0020	*Section 61.3.4.24*
0x2A4	PSI5-S channel1 ECU to Sensor Downstream Data Shift register (PS_DDSR_H_CH5)	R	R	R	0x0000_0000	*Section 61.3.4.25*
0x2A8	PSI5-S channel1 ECU to Sensor Downstream Data Shift register (PS_DDSR_L_CH5)	R	R/W[5]	R/W[5]	0x0000_0000	*Section 61.3.4.26*
0x02AC –0x02BB	Reserved					—
Channel6						
0x02BC	PSI5-S channel1 message configuration register A (PS_MSGA_CH6)	R	R/W	R	0x0000_0000	*Section 61.3.4.17*
0x02C0	PSI5-S channel1 message configuration register B (PS_MSGB_CH6)	R	R/W	R	0x1084_2108	*Section 61.3.4.18*
0x02C4– 0x02C7	Reserved					—
0x02C8	PSI5-S Mailbox status register channel6 (PS_MBOX_SR_CH6)	R	R/W1C	R/W1C	0x0000_0000	*Section 61.3.4.19*
0x02CC	PSI5-S channel1 watchdog configuration register (PS_WD_CFGR_CH6)	R	R/W[5]	R/W[5]	0x0000_0000	*Section 61.3.4.20*
0x02D0	PSI5-S DDSR Trigger offset register channel1 (PS_DDTRIG_OFFR_CH6)	R	R/W	R	0x0000_0000	*Section 61.3.4.21*
0x02D4	PSI5-S DDSR Trigger period register channel1 (PS_DDTRIG_PERR_CH6)	R	R/W	R	0x0000_0000	*Section 61.3.4.22*

Table 1546. PSI5-S memory map (continued)

Offset (Hex)	Register	Access[1]			Reset value[2]	Location
		D	C	N		
0x2D8	PSI5-S channel1 ECU to Sensor Control register (PS_E2SCR_CH6)	R	R/W	R/W[5]	0x0000_0800	*Section 61.3.4.23*
0x2DC	PSI5-S channel1 ECU to Sensor Status register (PS_E2SSR_CH6)	R	R/W1C	R/W1C	0x0000_0020	*Section 61.3.4.24*
0x2E0	PSI5-S channel1 ECU to Sensor Downstream Data Shift register (PS_DDSR_H_CH6)	R	R	R	0x0000_0000	*Section 61.3.4.25*
0x2E4	PSI5-S channel1 ECU to Sensor Downstream Data Shift register (PS_DDSR_L_CH6)	R	R/W[5]	R/W[5]	0x0000_0000	*Section 61.3.4.26*
0x02E8–0x02F7	Reserved					—
Channel7						
0x02F8	PSI5-S channel1 message configuration register A (PS_MSGA_CH7)	R	R/W	R	0x0000_0000	*Section 61.3.4.17*
0x02FC	PSI5-S channel1 message configuration register B (PS_MSGB_CH7)	R	R/W	R	0x1084_2108	*Section 61.3.4.18*
0x0300–0x0303	Reserved					—
0x0304	PSI5-S Mailbox status register channel1 (PS_MBOX_SR_CH7)	R	R/W1C	R/W1C	0x0000_0000	*Section 61.3.4.19*
0x0308	PSI5-S channel1 watchdog configuration register (PS_WD_CFGR_CH7)	R	R/W[5]	R/W[5]	0x0000_0000	*Section 61.3.4.20*
0x030C	PSI5-S DDSR Trigger offset register channel1 (PS_DDTRIG_OFFR_CH7)	R	R/W	R	0x0000_0000	*Section 61.3.4.21*
0x0310	PSI5-S DDSR Trigger period register channel1 (PS_DDTRIG_PERR_CH7)	R	R/W	R	0x0000_0000	*Section 61.3.4.22*
0x0314	PSI5-S channel1 ECU to Sensor Control register (PS_E2SCR_CH7)	R	R/W	R/W[5]	0x0000_0800	*Section 61.3.4.23*
0x0318	PSI5-S channel1 ECU to Sensor Status register (PS_E2SSR_CH7)	R	R/W1C	R/W1C	0x0000_0020	*Section 61.3.4.24*
0x031C	PSI5-S channel1 ECU to Sensor Downstream Data Shift register (PS_DDSR_H_CH7)	R	R	R	0x0000_0000	*Section 61.3.4.25*

Table 1546. PSI5-S memory map (continued)

Offset (Hex)	Register	Access[1]			Reset value[2]	Location
		D	C	N		
0x0320	PSI5-S channel1 ECU to Sensor Downstream Data Shift register (PS_DDSR_L_CH7)	R	R/W[5]	R/W[5]	0x0000_0000	*Section 61.3.4.26*
0x0324–0x0330	Reserved					—

1. In this column, R/W = Read/Write, R = Read-only, W = Write-only, W1C = write-1-to-clear and W1S = “write-to-1-shot wherein writing a 1 causes an effect on the register bit while writing a 0 has no effect; a read always returning a 0”, D = Disable mode, C = Config mode, N = Normal mode. All 32-bit registers are word/half word/byte addressable for write. Read is always word addressable.
2. Locations from 0x0000 to 0x005C are for UART registers.
3. PS_NOUSE_REG: Stands for register NOT REQUIRED in PSI5-S modes. It will be required in UART STANDALONE mode.
4. Register has significance in UART STANDALONE mode. This document would only mention the bit field names in the corresponding register description. For detailed bit description and the functioning of this register in UART modes (sleep, init and normal) belonging to UART STANDALONE mode, refer to *Chapter 63: LINFlexD*.
5. Not all register bits are of this type (refer to the register bit description for more details).
6. Register Bits are automatically hardware cleared on read of any bit of this register.

61.3.2 Transfer error generation

The IP generates transfer error cases in a number of scenarios as detailed below:

1. Trying to write to any register in the PS_DISABLE mode (apart from the PS_GLCR and PS_GLSR) will result in the generation of the transfer error.
2. Trying to write to any “Read Only” register (those which have “R,R,R” in all the D,C,N columns in *Table 1546*) will generate a transfer error in any of the PS_DISABLE, PS_NORMAL and PS_CONFIG modes. Apart from the above “Read Only” registers, trying to write to any register bits outside there intended mode of operation (Normal/Config) would be ignored without the generation of any transfer error. For bit level operation and mode access of each register, refer to *Section 61.3.3* and *Section 61.3.4*.
3. Trying to access (R/W) any register space which is declared as reserved in *Table 1546* or which is unimplemented will generate a transfer error.
4. Reading of any register in any mode and if that register is NOT in unimplemented/reserved space, then a transfer error will not be generated.
5. Trying to write to any PSI5-S register (starting from locations 0x00B4, but apart from PS_GLCR and PS_GLSR) in the UART_STDALONE mode will generate a transfer error.
6. In UART Standalone mode the UART behaves as a standalone IP and for detailed transfer error cases in this mode, refer to *Chapter 63: LINFlexD*.

61.3.3 UART register configuration

This section contains information of the UART registers that are specifically required for correct functioning of the IP in the “PSI5-S” modes.

In order to operate the PSI5-S module, the UART configuration registers need to be programmed with appropriate values. The values that are required for different register fields are defined in the following register descriptions. Some bits have to be programmed with definite values of “1” or “0” only. This configuration has to be strictly followed for a predictable operation of the PSI5-S core. All these bits should be programmed in the PS_CONFIG mode before enabling the UART RX or the TX paths.

In the following UART register description “PS_NOUSE_BIT” against a bit name and bit description, means that the specific register bit is present in the UART module but it is NOT REQUIRED in the PSI5-S modes of operation. For correct operation in the PSI5-S modes, the corresponding register bit might be required to be programmed as “1” or a “0”; the same would be mentioned in the bit description.

Note that a bit with “PS_NOUSE_BIT” against its description would still be required in the UART STDALONE mode. For detailed information of the UART register bit functioning in the UART STDLONE mode, refer to the separate document describing the standalone LINUART module.There are some other register bits which do not have a “PS_NOUSE_BIT” against them. These can be programmed as required. A bit mentioned as “Reserved” would mean that neither it is available in the UART STANDLONE mode nor in the PSI5-S modes.

61.3.3.1 PSI5-S LIN control register 1 (PS_LINCR1)

The PS_LINCR1 register contains control bits used to configure features of the PSI5-S UART Front end.

Address: 0x0000 Access: User read/write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	AUTOWU	0	0	0	0	1	SLFM	LBKM	1	0	RBLM	SLEEP	INIT
W																
Reset	0	0	0	0	0	0	0	0	1	0	0	1	0	0	1	0

Figure 1528. PSI5-S LIN control register 1 (PS_LINCR1)

Table 1547. PS_LINCR1 field descriptions

Field	Description
19 AUTOWU	PS_NOUSE_BIT Always program as “0” in PSI5-S modes.
25 SLFM	Self test mode 0 Self test mode disabled 1 Self test mode enabled To be used along with the LBKM bit. When this bit is set then in addition to the Loop Back mode; the “ipp_do_lintx” is driven to “1”.
26 LBKM	Loop Back mode 0 Loop Back Mode disabled 1 Loop Back mode enabled Programmed in PS_CONFIG mode. When set then the “ipp_do_lintx” output is internally looped back and connected to “ipp_ind_linrx” pin.
29 RBLM	PS_NOUSE_BIT Always program as 0 in PSI5-S modes.
30 SLEEP	PS_NOUSE_BIT **Note:** This bit is Read Only Bit in PSI5-S modes and R/W in UART Standalone mode. This bit is read as “1” in PS_DISABLE mode but is read as “0” in PS_NORMAL and PS_CONFIG mode.
31 INIT	PS_NOUSE_BIT **Note:** This bit is Read Only Bit in PSI5-S modes and R/W in UART Standalone mode. This bit is read as “0” in PS_DISABLE and PS_NORMAL mode, and “1” in PS_CONFIG mode.

61.3.3.2 PSI5-S LIN interrupt enable register (PS_LINIER)

Address: 0x0004 Access: User read/write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	SZIE	OCIE	0	0	0	0	0	FEIE	BOIE	0	WUIE	0	TOIE	DRIE	DTIE	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1529. PSI5-S LIN interrupt enable register (PS_LINIER)

Table 1548. PS_LINIER field descriptions

Field	Description
16 SZIE	Stuck at Zero Interrupt Enable 0 No interrupt 1 Interrupt generation enabled An interrupt is generated if this bit is set and the Stuck at Zero Flag (SZF) in PS_UARTSR is set. This interrupt can be used for monitoring purpose. It is not necessary to enable this interrupt for the correct operation of the PSI5-S core. **Note:** This bit is also writable in PS_NORMAL mode.
17 OCIE	PS_NOUSE_BIT Always program as 0 in PSI5-S modes. **Note:** This bit is also writable in PS_NORMAL mode.
23 FEIE	Frame Error Interrupt Enable 0 No interrupt 1 Interrupt generation enabled An interrupt is generated if this bit is set and the Stuck at Zero Flag (FEF) in PS_UARTSR is set. This interrupt can be used for monitoring purpose. It is not necessary to enable this interrupt for the correct operation of the PSI5-S core. **Note:** This bit is also writable in PS_NORMAL mode.
24 BOIE	Buffer Overrun Error Interrupt Enable 0 No interrupt 1 Interrupt generation enabled An interrupt is generated if this bit is set and the Buffer Overrun Flag (BOF) in PS_UARTSR is set. This interrupt can be used for monitoring purpose. It is not necessary to enable this interrupt for the correct operation of the PSI5-S core. **Note:** This bit is also writable in PS_NORMAL mode.
26 WUIE	PS_NOUSE_BIT Always program as 0 in PSI5-S modes. **Note:** This bit is also writable in PS_NORMAL mode.
28 TOIE	Timeout Interrupt Enable 0 No interrupt 1 Interrupt generation enabled. An interrupt is generated if this bit is set and the Timeout Flag (TO) in PS_UARTSR is set. This interrupt can be used for monitoring purpose. It is not necessary to enable this interrupt for the correct operation of the PSI5-S core. **Note:** This bit is also writable in PS_NORMAL mode.

Table 1548. PS_LINIER field descriptions (continued)

Field	Description
29 DRIE	Data Reception complete Interrupt enable 0 No interrupt 1 Interrupt generation enabled An interrupt is generated if this bit is set and the Data Received Flag (DRF) in PS_UARTSR is set. This interrupt can be used for monitoring purpose. It is not necessary to enable this interrupt for the correct operation of the PSI5-S core. **Note:** This bit is also writable in PS_NORMAL mode.
30 DTIE	Data Transmitted Interrupt enable 0 No interrupt 1 Interrupt generation enabled An interrupt is generated when this bit is set and Data Transmitted flag (DTF) in PS_UARTSR is set. This interrupt can be used by the Application for debugging when it wants to read the byte that is under transmission. This byte can be read from the PS_BDRL[DATA_TX[0]] location. It is not necessary to enable this interrupt for the correct operation of the PSI5-S core. **Note:** This bit is also writable in PS_NORMAL mode.

61.3.3.3 PSI5-S LIN status register (PS_LINSR)

This register is used only in the UART STANDALONE mode and has no significance in the PSI5-S modes

Address: 0x0008 Access: User read/write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	LINS3	LINS2	LINS1	LINS0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0

Figure 1530. LIN status register (LINSR)

Table 1549. LINSR field descriptions

Field	Description
16:19 LINS[3:0]	PS_NOUSE_BITS Read Value is “don’t care” in PSI5-S modes

61.3.3.4 PSI5-S UART mode control register (PS_UARTCR)

Address: 0x0010 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	MIS	CSP			OSR				ROSE	NEF			PCE_Tx/DTU	SBUR[1:0]		WLS
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	TDFL/TFC			RDFL/RFC			RFBM	TFBM	WL1	PC1	RxEn	TxEn	PC0	PCE_Rx	WL0	UART
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1531. PSI5-S UART mode control register (PS_UARTCR)

Table 1550. PS_UARTCR field descriptions

Field	Description
0 MIS	PS_NOUSE_BIT Always program as “1” in PSI5-S modes.
1:3 CSP	Configurable Sample Point These bits will decide the sample point during reduced over sampling (ROSE=1). CSP can take the following range of values for a certain over sampling rate: **OSR** / **CSP** 4 / 2, 3 5 / 2, 3, 4 6 / 3, 4, 5 8 / NA For any values of CSP other than the ones mentioned in the above table, data sampled is erroneous.
4:7 OSR	Over Sampling Rate These bits are programmable by the user to configure the number of samples taken for a bit when reduced over sampling is enabled (ROSE=1). For proper functionality, OSR can only take values 4, 5, 6 and 8. For any other values start bit of data is not detected by receiver and thus the reception will never start.
8 ROSE	Reduced Over Sampling Enable 0 Each bit is over sampled sixteen times. 1 OSR bits decide the over sampling rate. The default value of this bit is 0.
9:11 NEF	PS_NOUSE_BIT Always program as “000” in PSI5-S modes.

Table 1550. PS_UARTCR field descriptions (continued)

Field	Description
12 PCE_Tx/ DTU	In PSI5-S mode Parity Control Enable for Tx path (In UART STANDLONE mode used to disable UART timeout, refer to *Chapter 63: LINFlexD* for more details): 1 Parity generation for Tx path enabled. 0 Parity generation for Tx path disabled. Can be programmed in PS_CONFIG mode only and when PS_UARTCR[31] == 1. When programmed as “1” then a 1 bit additional parity bit is added to the transmitted 8 bits of UART data, so a total of 9 bits including parity are present between a start and a stop bit, in the transmitted data. When programmed as “0” then total of 8bits data is present between a start and a stop bit in the transmitted data.
13:14 SBUR	PS_NOUSE_BIT Always program as “00” in PSI5-S modes.
15 WLS	PS_NOUSE_BIT Always program as “0” in PSI5-S modes.
16:18 TDFL/TFC	PS_NOUSE_BIT Always program as “000” in PSI5-S modes.
19:21 RDFL/RFC	PS_NOUSE_BIT Always program as “000” in PSI5-S modes.
22 RFBM	PS_NOUSE_BIT Always program as “0” in PSI5-S modes. Can be programmed in PS_CONFIG mode only and when PS_UARTCR[UART] == 1
23 TFBM	PS_NOUSE_BIT Always program as “0” in PSI5-S modes. Can be programmed in PS_CONFIG mode only and when PS_UARTCR[UART] == 1
24 WL1	PS_NOUSE_BIT Always program as “0” in PSI5-S modes. Can be programmed in PS_CONFIG mode only and when PS_UARTCR[UART] == 1
25 PC1	Parity Control {PC1, PC0} 00 Parity is Even for both the Rx/Tx paths. 01 Parity is Odd for both the Rx/Tx paths. 10 A logical 0 is taken as parity bit for both the Rx and the Tx paths. 11 A logical 1 is taken as parity bit for both the Rx and the Tx paths. Can be programmed only in the PS_CONFIG mode only and when PS_UARTCR[UART] == 1.
26 RxEn	Receiver Enable 0 Receiver disabled 1 Receiver enabled This bit can be programmed only when PS_UARTCR[31] == 1.
27 TxEn	Transmitter Enable 0 Transmitter disabled 1 Transmitter enabled, transmission starts only when this bit is set and Data byte 0 (DATA0) is programmed This bit can be programmed only when PS_UARTCR[UART] == 1

Table 1550. PS_UARTCR field descriptions (continued)

Field	Description
28 PC0	Parity Control {PC1, PC0} 00 Parity is Even for both the Rx/Tx paths 01 Parity is Odd for both the Rx/Tx paths. 10 A logical 0 is taken as parity bit for both the Rx and the Tx paths 11 A logical 1 is taken as parity bit for both the Rx and the Tx paths. Can be programmed in PS_CONFIG mode only and when PS_UARTCR[UART] == 1.
29 PCE_Rx	Receive Parity check enable. 1 Enable the parity checking on the Rx path 0 Disable the parity checking on the Rx path Can be programmed in PS_CONFIG mode only and when PS_UARTCR[UART] == 1. When programmed as “1” then a 1 bit additional parity bit is expected along with the received 8 bits of UART data, so a total of 9 bits including parity are expected between a start and a stop bit, in the received data. When programmed as “0” then total of 8bits data is expected between a start and a stop bit in the received data.
30 WL0	PS_NOUSE_BIT Always program as “1” in PSI5-S modes. Can be programmed in PS_CONFIG mode only and when PS_UARTCR[UART] == 1.
31 UART	PS_NOUSE_BIT Always program as “1” in PSI5-S modes. Can be programmed in PS_CONFIG mode only.

61.3.3.5 PSI5-S UART mode status register (PS_UARTSR)

Address: 0x0014 Access: User read/write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	SZF	OCF	PE3	PE2	PE1	PE0	RMB	FEF	BOF	RDI	WUF	RFNE	TO	DRF	DTF	NF
W	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c		w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1532. PSI5-S UART mode status register (PS_UARTSR)

Table 1551. PS_UARTSR field descriptions

Field	Description
16 SZF	Stuck at Zero flag This bit is set by hardware when 100 dominant bits are detected. It should be cleared by software. An interrupt will be generated on the UART interrupt lines if the SZIE bit in PS_LINIER is set.
17 OCF	PS_NOUSE_BIT Read Only Bit.
18:20 PE[3:1]	PS_NOUSE_BIT Read Only Bit.
21 PE0	Parity Error flag for Data0 byte These bits indicate if there is a Parity Error in the received byte. No interrupt is generated if this error occurs. 0 No parity error 1 Parity error in the corresponding received byte This flag is automatically cleared by hardware. The software can poll this flag for debugging purposes.
22 RMB	PS_NOUSE_BIT Read Only Bit.
23 FEF	Framing Error flag This bit is set by hardware when there is a framing error (invalid stop bit). FEF is automatically cleared by hardware. An interrupt is generated if this bit is set and the FEIE bit of PS_LINIER is set. This interrupt can be used for monitoring purpose. It is not necessary to enable this interrupt for the correct operation of the PSI5-S core
24 BOF	Buffer overrun flag This bit is set by hardware when there is a new byte received and the previous byte has not yet been read from the UART Rx buffer due to some fault in PSI5-S core reading logic. It is automatically cleared by hardware. An interrupt is generated if this bit is set and the BOIE bit of PS_LINIER is set. This interrupt can be used for monitoring purpose. It is not necessary to enable this interrupt for the correct operation of the PSI5-S core.
25 RDI	Receiver Data Input signal This bit reflects the current status of the RX pin.
26 WUF	PS_NOUSE_BIT Read-only Bit.
27 RFNE	PS_NOUSE_BIT Read-only Bit.

Table 1551. PS_UARTSR field descriptions (continued)

Field	Description
28 TO	Timeout This bit is set by hardware when a UART timeout occurs — in other words, the value of UARTCTO becomes equal to the preset value of the timeout (UARTPTO register setting). TO is automatically cleared by hardware. An interrupt is generated if this bit is set and the TOIE bit of PS_LINIER is set. This interrupt can be used for monitoring purpose. It is not necessary to enable this interrupt for the correct operation of the PSI5-S core.
29 DRF	Data Reception Completed Flag DRF is set by hardware in UART buffer mode and indicates that one byte has been received and is ready for being read. DRF is automatically cleared by hardware. An interrupt is generated if this bit is set and the DRIE bit of PS_LINIER is set. This interrupt can be used for monitoring purpose. It is not necessary to enable this interrupt for the correct operation of the PSI5-S core.
30 DTF	Data Transmission Completed Flag DTF is set by hardware in UART buffer mode and indicates that the current byte which was written to the PS_BDRL[0] location has been successfully transmitted by the UART. DTF is automatically cleared by hardware when a new byte transmission is started. An interrupt will be generated if the DTIE bit in PS_LINIER is set.
31 NF	Noise flag This bit is set by hardware when noise is detected in the received character. It should be cleared by software. During reduced oversampling (ROSE bit = 1), it is enabled only when OSR = 8.

61.3.3.6 PSI5-S LIN fractional baud rate register (PS_LINFBRR)

This register consists of bits that decide the fractional part of the LIN Baud Rate. It can be programmed only in Initialization mode.

Address: 0x0024 Access: User read/write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	FBR3(1)	FBR2(1)	FBR1(1)	FBR0(1)
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. Register bit can be read in any mode, written only in PS_CONFIG mode.

Figure 1533. PSI5-S LIN fractional baud rate register (PS_LINFBRR)

Table 1552. PS_LINFBRR field descriptions

Field	Description
28:31 FBR[3:0]	Fractional Baud rates. Refer to the *Note: on page 2564*. **LINFBRR[4:0] / Fraction (LDIV)** 0x0 / 0 0x1 / 1/16 ... / ... 0xE / 14/16 0xF / 15/16

61.3.3.7 LIN integer baud rate register (PS_LINIBRR)

This register consists of control bits that decide the baud rate along with the PS_LINFBRR. It can be programmed only in Initialization mode.

Address: 0x0028 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	IBR19(1)	IBR18(1)	IBR17(1)	IBR16(1)
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	IBR15(1)	IBR14(1)	IBR13(1)	IBR12(1)	IBR11(1)	IBR10(1)	IBR9(1)	IBR8(1)	IBR7(1)	IBR6(1)	IBR5(1)	IBR4(1)	IBR3(1)	IBR2(1)	IBR1(1)	IBR0(1)
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. Register bit can be read in any mode, written only in PS_CONFIG mode.

Figure 1534. PSI5-S LIN integer baud rate register (PS_LINIBRR)

Table 1553. PS_LINIBRR field descriptions

Field	Description
12:31 IBR[19:0]	Integer Baud rates These bits along with the fractional baud rate bits decide the LIN baud rate as below. **LINIBRR[19:0] / Mantissa (LDIV)** 0x00000 / LIN clock disabled 0x00001 / 1 ... / ... 0xFFFFE / 1048574 0xFFFFF / 1048575

Note: The UART baud rate is programmed in two registers: the LIN Integer Baud Rate register (PS_LINIBRR) and the LIN Fraction Baud Rate register (PS_LINFBRR). The baud rate registers can be programmed only during the PS_CONFIG mode.

The baud rate is calculated with the following formula for both receiver and transmitter:

Tx_{BR} = Rx_{BR} = ipg_baud_clk/(16 × LDIV) (when UARTCR[ROSE] == 0)

Tx_{BR} = Rx_{BR} = ipg_baud_clk/UARTCR[OSR] x LINIBRR (when UARTCR[ROSE] == 1)

LDIV is an unsigned fixed point number which comprises of the mantissa and the fraction. The mantissa is coded into 20 bits of LINIBRR, and the fraction is coded on 4 bits of LINFBRR.

When reduced oversampling is enabled (UARTCR[ROSE] ==1) then PS_LINFBRR is not used. Hence LDIV contains only the integer part programmed in LINIBRR. Following are 2 examples for two different baud rate calculations.

Example 1:

With UARTCR[ROSE] = 0.

LDIV = 1041.67, ipg_baud_clk = 80 MHz

LINIBRR = 1041d = 411 h

LINFBRR = 11d = B h

Tx_{BR} = Rx_{BR} = 80 Mhz/ (16 x 1041,67) = 4.8 Kbit/s

Example 2:

When UARTCR[ROSE] = 1:

LDIV = 10 d, LIN_CLK = 80 MHz

LINIBRR = 10 d, UARCR[OSR] = 4

Tx_{BR} = Rx_{BR} = ipg_baud_clk/ (UART[OSR] × LDIV) = 80 MHz / (4 × 10) = 2 Mbit/s

61.3.3.8 PSI5-S LIN control register 2 (PS_LINCR2)

This register includes control status bits related to buffer operations.

Address: 0x0030 Access: User read/write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	1	0	0	0	0	ABRQ(1)	0	0	0	0	0	0	0	0	0
W																
Reset	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. Register bit can be read/set by software.

Figure 1535. PSI5-S LIN control register 2 (PS_LINCR2)

Table 1554. PS_LINCR2 field descriptions

Field	Description
22 ABRQ	Abort Request Set by software to abort the current transmission. Cleared by hardware when the transmission has been aborted. Transmission is aborted at the end of the current bit. **Note:** This bit is also writable in PS_NORMAL mode.

61.3.3.9 PSI5-S buffer data register least significant (PS_BDRL)

Address: 0x0038 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	DATA_TX3								DATA_TX2							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	DATA_TX1								DATA_TX0							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1536. PSI5-S buffer data register least significant (PS_BDRL)

Table 1555. PS_BDRL field descriptions

Field	Description
0:7 DATA_TX3	PS_NOUSE_BIT Writing has no effect in fully configured(1) PSI5-S mode.
8:15 DATA_TX2	PS_NOUSE_BIT Writing has no effect in fully configured (1)PSI5-S mode.
16:23 DATA_TX1	PS_NOUSE_BIT Writing has no effect in fully configured(1) PSI5-S mode.
24:31 DATA_TX0	Data Byte for Tx. In fully configured PSI5-S modes the application has to take care to NEVER write to this register else the transmission of the UART can become unpredictable. This register can however be read by the application for debugging purpose, since it gives the byte that has been written by the PSI5-S core to the UART, for transmission.

1. All UART control registers are programmed as mentioned in this document.

61.3.3.10 PSI5-S buffer data register most significant (PS_BDRM)

Address: 0x003C Access: User read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	DATA_RX3								DATA_RX2							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	DATA_RX1								DATA_RX0							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1537. PSI5-S buffer data register most significant (PS_BDRM)

Table 1556. PS_BDRM field descriptions

Field	Description
0:7 DATA_RX3	PS_NOUSE_BIT Always read as “0” in fully configured(1) PSI5-S mode.
8:15 DATA_RX2	PS_NOUSE_BIT Always read as “0” in fully configured(1) PSI5-S mode.
16:23 DATA_RX1	PS_NOUSE_BIT Always read as “0” in fully configured(1) PSI5-S mode.
24:31 DATA_RX0	Receive data byte for Rx Receive data byte of the data field.This register can be read by the application for debugging purpose, since it gives the byte that has been received by the UART core and will be send to the PSI5-S core.

1. All UART control registers are programmed as mentioned in this document.

61.3.3.11 PSI5-S global control register (PS_GCR)

Address: 0x004C Access: User read/write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	TDFBM(1)	RDFBM(1)	TDLIS(1)	RDLIS(1)	STOP	SR(2)
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. Register bit can be read in any mode, written only in PS_CONFIG mode.
2. Register bit can be written only by software in PS_CONFIG mode. Bit is always read 0 by software.

Figure 1538. PSI5-S global control register (PS_GCR)

Table 1557. PS_GCR field descriptions

Field	Description
26 TDFBM	Transmit data first bit MSB This bit controls the first bit of transmit data (payload only) as MSB/LSB: 0 The first bit of transmitted data is LSB — in other words, the first bit transmitted is the LSB bit (of DATA_TX in PS_BDRL register) 1 The first bit of transmitted data is MSB — in other words, first bit transmitted is the MSB bit (of DATA_TX in PS_BDRL register)
27 RDFBM	Received data first bit MSB This bit controls the first bit of received data (payload only) as MSB/LSB: 0 The first bit of received data is LSB — in other words, the first bit received is mapped on LSB bit (of DATA_RX in PS_BDRM register) 1 The first bit of received data is MSB — in other words, first bit received is mapped on MSB bit (of DATA_RX in PS_BDRM register)
28 TDLIS	Transmit data level inversion selection This bit controls the data inversion of transmitted data (payload only). 0 Transmitted data is not inverted 1 Transmitted data is inverted
29 RDLIS	Received data level inversion selection This bit controls the data inversion of received data (payload only). 0 Received data is not inverted 1 Received data is inverted
30 STOP	PS_NOUSE_BIT Always program as “0” in PSI5-S modes.
31 SR	Soft reset SR executes a soft reset of the UART controller (FSMs, FIFO pointers, counters, timers, status and error registers) without modifying the configuration registers when a 1 write operation is performed. This bit should be cleared by software to perform further operations (this bit is not cleared by hardware). This bit is always read as 0.

61.3.3.12 PSI5-S UART preset timeout register (PS_UARTPTO)

This register contains the preset value of the timeout register in UART mode and is programmed to monitor the idle state of the reception line. This register can be written by software any time.

Address: 0x0050 Access: User read/write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	PTO11	PTO10	PTO9	PTO8	PTO7	PTO6	PTO5	PTO4	PTO3	PTO2	PTO1	PTO0
W																
Reset	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1

Figure 1539. PSI5-S UART preset timeout register (PS_UARTPTO)

Table 1558. PS_UARTPTO field descriptions

Field	Description
20:31 PTO[11:0]	Preset Timeout PTO defines the preset value of timeout counter. A zero-value is forbidden, otherwise the UARTSR[TO] status bit is immediately set. The register must be programmed to a value of “Idle Timeout required in terms of number of bits received” at that particular baud rate. The idle timeout is measured after the stop bit and measures the minimum amount of time (in terms of bits) for which the line stays high. Refer also to register UARTCTO.

61.3.3.13 PSI5-S UART current timeout register (PS_UARTCTO)

This register contains the current timeout value in UART mode, and is used in conjunction with the UARTPTO register (refer to *Section 61.3.3.12: PSI5-S UART preset timeout register (PS_UARTPTO)* to monitor the idle state of the reception line.

The timeout counter:

- Starts at zero and counts upward.
- Is clocked with baud clock frequency of Tx_{BR}/Rx_{BR}. Also refer to *Note: on page 2564*.
- Is automatically enabled when UARTCR[RXEN] = 1.

Note: Due to internal synchronization there is a delay between 4 to 6 clock cycles of “ipg_clk_ps”, for the internal counter’s value (which is clocked with baud frequency of Tx_{BR}/Rx_{BR}) to reflect on to UARTCTO.

UARTCTO is reset when:

- UARTCTO becomes equal to UARTPTO.
- Whenever UARTPTO is written.
- When the PSI5-S goes in PS_DISABLE mode.

Address: 0x0054 Access: User read

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	CTO11	CTO10	CTO9	CTO8	CTO7	CTO6	CTO5	CTO4	CTO3	CTO2	CTO1	CTO0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1540. PSI5-S UART current timeout register (PS_UARTCTO)

Table 1559. PS_UARTCTO field descriptions

Field	Description
20:31 CTO[11:0]	Current Timeout CTO defines the current value of the timeout counter. CTO is a read-only field. CTO is reset every time UARTPTO is re-initialized, or UARTCTO = UARTPTO, or by hard/soft reset. When the CTO value matches the preset value (UARTPTO), the status bit UARTSR[TO] is set.

61.3.3.14 DMA Tx enable register (DMATXE)

This register enables the DMA TX interface. This register is used only in the UART STANDALONE mode and has no significance in the PSI5-S modes.

Address: 0x0058 Access: User read/write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	DTE0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1541. DMA Tx enable register (DMATXE)

Table 1560. DMATXE field descriptions

Field	Description
31 DTE0	PS_NOUSE_BIT Always program with “0” in PSI5-S modes

61.3.3.15 DMA Rx enable register (DMARXE)

This register enables the DMA RX interface. This register is used only in the UART STANDALONE mode and has no significance in the PSI5-S modes.

Address: 0x005C Access: User read/write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	DRE0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1542. DMA Rx enable register (DMARXE)

Table 1561. DMARXE field descriptions

Field	Description
31 DRE0	PS_NOUSE_BIT Always program with “0” in PSI5-S modes

61.3.3.16 PSI5-S UART Tx idle delay time (PS_PTD)

This register is used to add a programmable idle time between the data that is transmitted on the UART Tx line. The duration of this delay is programmable between 1 to 16 bit times.The programmable delay is such that when the normal ECU to Sensor communication (no direct command) is happening, then it comes into play between each byte of transmitted data. When the ECU to Sensor direct command is used then this delay comes into play between each direct command. The direct commands can be programmed as 1, 2 and 4 bytes depending on the PS_GLCR[DIRCMD_LEN] register. Refer to *Section 61.3.4.13: PSI5-S ECU to sensor direct command write register (PS_DIRCMD)* and *Section 61.3.4.1: PSI5-S global control register (PS_GLCR)*, for more details.

Address: 0x0060 Access: User read/write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	IFD					EN
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1543. PSI5 UART Tx delay (PS_PTD)

Table 1562. PS_PTD field descriptions

Field	Description
26:30 IFD	Interframe Delay The idle time added between the UART data on the TX line is IFD+1 bit times. The UART data can be sized to be of either one byte or 1, 2 and 4 bytes.
31 EN	IFD Enable 0 The IFD is active in the UART Tx path 1 The IFD is inactive

61.3.4 PSI5-S register description

Unless specifically stated, the register bits are readable in all modes, but writable only in Config mode.

PS_GLCR[GLOBAL_MODE] are the only register bits that can be written in the PS_DISABLE mode.

All word locations are byte/half word/word accessible for write. Read is always a word access. Trying to write to “read only” bits of implemented register space, will be ignored without the generation of any error.

61.3.4.1 PSI5-S global control register (PS_GLCR)

The Global Control Register (PS_GLCR) provides control functions for all the PSI5-S channels simultaneously.

Address: 0x00B4 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	DIRCMD_LEN		0	IE_DIRCMD_RDY	0	DEBUG_EN	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	GL_DDSR_TRIG	GL_MODETR_DONE_EN	MRU_ERR_EN	TSCS_B	TSCS_A	0	0	0	CLRTSCNT_G_L	TSCNT_EN_B	TSCNT_EN_A	TSCNTEN_G	TSCNTEN_G_L	GLOBAL_MODE[2:0]		
W						CLR_CNTR_B	CLR_CNTR_A	CLRTSCNT_G								
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1544. PSI5-S global control register (PS_GLCR)

Table 1563. PS_GLCR field descriptions

Field	Description
8:9 DIRCMD_LEN	Direct Command Length Register This defines the length of the Direct Command that is transferred to the UART Tx line. The Direct command is written to the PS_DIRCMD register in the format {DIRCMD_BYTE3, DIRCMD_BYTE2, DIRCMD_BYTE1, DIRCMD_BYTE0}.Following is the bit description: 00 1 byte command: 1 byte {DIRCMD_BYTE0} of the PS_DIRCMD is transferred to the UART Tx. Rest bytes of the PS_DIRCMD are ignored. 01 2 byte command: 2 bytes {DIRCMD_BYTE1, DIRCMD_BYTE0} of the PS_DIRCMD is transferred to the UART Tx without any idle time between DIRCMD_BYTE1 and DIRCMD_BYTE0. Rest bytes of the PS_DIRCMD are ignored. 10 4 byte command: 4 bytes 11 4 byte command: 4 bytes {DIRCMD_BYTE3, DIRCMD_BYTE2, DIRCMD_BYTE1, DIRCMD_BYTE0} of the PS_DIRCMD is transferred to the UART Tx line without any idle time between these bytes. The idle time between the commands is still governed by the PS_PTD register in the LINUART module. This register is also writable in the NORMAL mode.
11 IE_DIRCMD_RDY	Interrupt enable bit for DIRCMD_RDY 0 PS_GLCR[DIRCMD_RDY], when goes as “1”, does not generate any interrupt on the “ipi_ps_glbl” interrupt line. 1 PS_GLCR[DIRCMD_RDY], when goes as “1”, generates an interrupt on the “ipi_ps_glbl” interrupt line. **Note:** This bit is also writable in PS_NORMAL mode.
13 DEBUG_EN	Debug enable bit 0 The IP does not enter the debug mode when a request is made to the IP for entering into the debug mode. It continues its normal operation. 1 The IP enters the debug mode immediately, as soon as a request is made to the IP for entering into the debug mode. For debug mode operation, refer to *Section 61.1.4.2: Debug mode*.
14:15 GTM_TRIG_SEL	GTM_TRIG_SEL bits. These bits are used to select the trigger for the Timestamp counters as shown in *Figure 1587: Clock distribution in PSI5-S*. The GTM trigger signals control the frequency of operation of the timestamp counters. In other words they control the clocking (clkA and clkB) of the timestamp counters A and B. 00 gtm_trig[0] is selected for clocking the timestamp counter. 01 gtm_trig[1] is selected for clocking the timestamp counter. 10 gtm_trig[2] is selected for clocking the timestamp counter. 11 gtm_trig[3] is selected for clocking the timestamp counter.
16 GL_DDSR_TRIG	This is the global shift trigger of all the DDSRs of each channel. When the corresponding PS_E2SCR_CH[n][GL_TRIG_SEL] == 1, only then is this bit effective, else this bit has no effect. 1 The shift triggers of all the DDSR’s start synchronously. 0 Shift triggers of all the DDSR registers are halted (default state). If bit is cleared during runtime, PS_DDTRIG_PERR and PS_DDTRIG_OFFR counters are reset. **Note:** If bit is subsequently set after being cleared, then the counters will restart and the offsets in PS_DDTRIG_PERR and PS_DDTRIG_OFFR registers will be applied again. This bit is also writable in PS_NORMAL mode

Table 1563. PS_GLCR field descriptions (continued)

Field	Description
17 GL_MODETR_ DONE_EN	Interrupt enable for PS_GLSR[GL_MODETR_DONE] bit 1 Generates the interrupt when PS_GLSR[GL_MODETR_DONE] bit is set. 0 No interrupt is generated on setting of the PS_GLSR[GL_MODETR_DONE] bit **Note:** This bit is also writable in all modes (PSI5-S and UART_NORMAL)
18 MRU_ERR_EN	Enables the interrupt for MRU overwrite when the unread contents of MRU Buffer1 are overwritten by a new message. 0 MRU overwrite error is not enabled. 1 MRU overwrite error is enabled. **Note:** This bit is also writable in ps_normal mode
19 TSCS_B	This bit controls the selection of clock input to timestamp counter B, either by the external periodic clock or by the GTM module. 1 gtm_trig is selected as the clock. 0 External clock (ipg_clk_ps_ts) is selected.
19 TSCS_B	This bit controls the selection of clock input to timestamp counter B, either by the external periodic clock or by the GTM module. 1 gtm_trig is selected as the clock. 0 External clock (ipg_clk_ps_ts) is selected.
21 CLR_CNTR_B	This bit will reset timestamp counter B when PS_GLCR[CLRTSCNT_G_L] is NOT set. 1 Timestamp counter B is reset. This bit is a one shot and is always read as 0. **Note:** This bit is also writable in ps_normal mode
22 CLR_CNTR_A	This bit will reset timestamp counter A when PS_GLCR[CLRTSCNT_G_L] is NOT set. 1 Timestamp counter A is reset. This bit is a one shot and is always read as 0. **Note:** This bit is also writable in ps_normal mode
23 CLRTSCNT_G	This bit controls the global clearing of the timestamp counters A and B when CLRTSCNT_G_L == 1. When CLRTSCNT_G_L == 0 then this bit has no effect. 1 Both timestamp counters are cleared at the same time. This bit is a one shot and is always read as 0. **Note:** This bit is also writable in ps_normal mode
24 CLRTSCNT_G_L	This bit control whether the timestamp counters counter A and B are cleared simultaneously or independently: 0 Timestamp counters A and B are cleared independently. When PS_GLCR[CLR_CNTR_A] == 1 timestamp counter A is cleared. When PS_GLCR[CLR_CNTR_B]== 1 timestamp counter B is cleared. 1 Both timestamp counters are cleared simultaneously as soon as CLRTSCNT_G == 1.
25 TSCNT_EN_B	This bit controls enabling of timestamp counter A when PS_GLCR[TSCNTEN_G_L] is not set. 0 Timestamp counter A not enabled 1 Timestamp counter B is enabled **Note:** This bit is also writable in ps_normal mode.

Table 1563. PS_GLCR field descriptions (continued)

Field	Description
26 TSCNT_EN_A	This bit controls enabling of timestamp counter A when PS_GLCR[TSCNTEN_G_L] is not set. 0 Timestamp counter A not enabled 1 Timestamp counter A is enabled **Note:** This bit is also writable in ps_normal mode.
27 TSCNTEN_G	This bit controls the global trigger of the timestamp counters A and B when TSCNTEN_G_L == 1. When TSCNTEN_G_L == 0 then this bit has no effect. 0 Both timestamp counters are disabled simultaneously. 1 Both timestamp counters start simultaneously. **Note:** This bit is also writable in ps_normal mode.
28 TSCNTEN_G_L	This bit control whether the timestamp counters counter A and B are started simultaneously or independently. 0 Timestamp counters A and B are started independently. When PS_GLCR[TSCNT_A_EN] == 1 timestamp counter A starts When PS_GLCR[TSCNT_B_EN] == 1 timestamp counter B starts. 1 Both timestamp counters are triggered simultaneously as soon as TSCNTEN_G == 1.
29:31 GLOBAL_MODE	These are the bits which decide Global Mode of the IP. There are 4 modes, UART_STDALONE, PS_DISABLE, PS_CONFIG and PS_NORMAL. For details of modes related to GLOBAL_MODE bit setting, refer to *Table 1544: State transition Table*. **Note:** These bits are writable in all modes (PSI5-S and UART_STANDALONE).

61.3.4.2 PSI5-S global status register (PS_GLSR)

Address: 0x00B8 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	DIRCMD_RDY	0	0	0	0
W												w1c				
Reset	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	GL_MODETR_DONE	MRU_ERR	0	0	0	0	0	0	0	FID		CHID			
W		w1c	w1c													
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1545. PSI5-S global status register (PS_GLSR)

Table 1564. PS_GLSR field descriptions

Field	Description
11 DIRCMD_RDY	Bit used to indicate the “Ready for Direct Command write” status: 0 Direct Command that is written to the PS_DIRCMD register is currently under processing. Any further command written to the PS_DIRCMD register would be rejected without any information. 1 The PS_DIRCMD register is empty and the application can write the direct ECU to sensor transceiver command to the PS_DIRCMD register. Once the command is written to the PS_DIRCMD, then a “w1c” to this register should be done so that the command written to the PS_DIRCMD register is queued for UART transmission with the highest priority. When the command has been transferred from thie register to the UART, then this bit goes as “1”. If the PS_GLCR[IE_DIRCMD_RDY] bit is set to “1” then this event generates an interrupt on the ipi_ps_glbl line. **Note:** This bit is also writeable in the PS_NORMAL mode.
17 GL_MODETR_DONE	This bit indicates, as to when the global mode transitions, as programmed by PS_GLCR[GLOBAL_MODE] have actually been accomplished: 1 The global mode as programmed in the PS_GLCR[GLOBAL_MODE] have been accomplished 0 The intended global mode, as programmed in the PS_GLCR[GLOBAL_MODE] register bits, is under processing. **Note:** This bit is cleared by w1c. The bit is also writable in PS_NORMAL and PS_DISABLE mode. This bit is reset to “0” when the mode is changed from PS_DISABLE to the UART_STDALONE mode.
18 MRU_ERR	This bit shows whether unread MRU buffer1 contents have been overwritten by a new message: 1 overwrite occurred 0 no overwrite occurred **Note:** This bit is cleared by w1c. The bit is also writable in PS_NORMAL mode.
26:28 FID	Points to the frame whose data in MRU buffer 1 was overwritten
29:31 CHID	Points to the channel whose data in MRU buffer 1 was overwritten

61.3.4.3 PSI5-S channel base address register (PS_CH_BASE_ADDR)

Address: 0x00BC Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	MAILBOX_BASE_ADDR															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	MAILBOX_BASE_ADDR															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1546. PSI5-S channel base address register (PS_CH_BASE_ADDR)

Table 1565. PS_CH_BASE_ADDR field descriptions

Field	Description
0:31 MAILBOX_BASE_ADDR	This register will provide the base address of the mailbox in the system RAM.This value needs to be programmed by the software.

61.3.4.4 PSI5-S MRU output buffer 2 register 0 (PS_MRU_BUF2_REG0)

Address: 0x00C0 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	CHANNEL_SPECIFIC_MAILBOX_ADDR															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	CHANNEL_SPECIFIC_MAILBOX_ADDR															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1547. PSI5-S MRU output buffer 2 register 0 (PS_MRU_BUF2_REG0)

Table 1566. PS_MRU_BUF2_REG0 field descriptions

Field	Description
0:31 CHANNEL_SPECIFIC_MAILBOX_ADDR	This register will provide the channel specific mailbox address corresponding to channel and slot for which data has been received.This value is computed by the IP based on the PS_CH_BASE_ADDR[MAILBOX_BASE_ADDR] and FID, CID. Following is the formula used to calculate this address: – For messages with CID==0 in the header byte, destined for Channel0 location0, value is: PS_CH_BASE_ADDR[MAILBOX_BASE_ADDR] – For “unrecoverable” messages destined for Channel0,location1, value is: PS_CH_BASE_ADDR[MAILBOX_BASE_ADDR]+0xC – For other messages with CID = 1 to 7 the value can be calculated as: (PS_CH_BASE_ADDR[MAILBOX_BASE_ADDR] + (6*4*3(MSG_CH_ID)+(MSG_FRAME_ID)*4*3)) When using the DMA for reading the MRU buffer, this register forms the Access1 of the DMA transfer.

61.3.4.5 PSI5-S MRU output buffer 2 register 1 (PS_MRU_BUF2_REG1)

Address: 0x00C4 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	DCI				0	R_UVL_ERR	N_ERR	CHID			FID			R_OVL_ERR	F_WD_ERR	SCI_O_ERR
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	SCI_F_ERR	SCI_P_ERR	HD_ERR	ERR		CRC_ERR/ P_ERR	CRC			XCRC_ERR	XCRC					
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1548. PSI5-S MRU output buffer 2 register 1 (PS_MRU_BUF2_REG1)

Table 1567. PS_MRU_BUF2_REG1 field descriptions

Field	Description
0:3 DCI	To ensure that the Application reads the PSI5 message and its timestamp consistently, Data Consistency Indicator (DCI) is added to each word of the Message Reconstruction Unit Output buffer and subsequently is transferred to the PSI5 Mailbox in system RAM
5 R_UVL_ERR	Message underflow It is set when less than three UART bytes are received in the UART packet. In case of XCRC error the message is treated as unrecoverable and written in mail box location1 of channel 0 1 Error occurred 0 No error occurred Refer to *Section 61.4.2.1: PSI5 message extraction*, for more details.
6 N_ERR	Indicates that the number of UART bytes in the received UART packet does not match the number specified in Channel Configuration Register A indicated by CID and FID that is the number of UART bytes does not match the value programmed in PS_MSGA_CH_{CID}[F_{FID}_bytes]. Also, refer to note in *Section 61.3.4.18: PSI5-S channel message configuration register B (PS_MSGB_CHn)*. 1 Error occurred 0 No error occurred Refer to *Section 61.4.2.1: PSI5 message extraction*, for more details.
7:9 CHID	Indicates Channel id of PSI5 message present in the UART header byte.
10:12 FID	Indicates Frame ID of PSI5 message present in the UART header byte.
13 R_OVL_ERR	Message overflow.It is set when more than six UART bytes are received in the UART packet.The message may be recoverable (XCRC_ERR == 0) or unrecoverable (XCRC_ERR==1). Refer to *Section 61.4.2.1: PSI5 message extraction*, for more details. 1 Error occurred 0 No error occurred

Table 1567. PS_MRU_BUF2_REG1 field descriptions (continued)

Field	Description
14 F_WD_ERR	Frame watchdog error in the received PSI5 message In case there is no XCRC error in the received message, then this bit will be the same as the PS_WDGTSSR[F_WD_ERR[CID]] bit, where CID is the channel ID of the received PSI5 message. In case the received PSI5 message has an XCRC error then this bit will be the logical OR of all the PS_WDGTSSR[F_WD_ERR[CHn]] bits that is in this case it will be "PS_WDGTSSR[F_WD_ERR[1]] OR PS_WDGTSSR[F_WD_ERR[2]] OR....PS_WDGTSSR[F_WD_ERR[7]]". Refer to *Section 61.4.5: PSI5-S watchdog operation*, *Section 61.3.4.18: PSI5-S channel message configuration register B (PS_MSGB_CHn)* and *Section 61.3.4.12: PSI5-S watchdog error status and watchdog timestamp status register (PS_WDGTSSR)*, for more details. 1 Error occurred 0 No error occurred **Note:** Whenever the PS_MRU_BUF2_REG1[F_WD_ERR] bit is set, PS_WDGTSSR (which is a hardware clearing register) should always be read. Failing to do so will result in the PS_MRU_BUF2_REG1[F_WD_ERR] bit continuing to be set for respective "CID", whenever the PS_WDGTSSR[F_WD_ERR_STATUS[CID]] bit is set. This would happen in subsequent messages saved to the mailbox even though no further watchdog error has occurred for the respective CID.This happens since for each received PSI5 message in the MRU buffers the status of the corresponding PS_MRU_BUF2_REG1[F_WD_ERR] is picked from the PS_WDGTSSR[F_WD_ERR_STATUS[CID]] bit.
15 SCI_O_ERR	UART overrun error has occurred in at least one UART byte of the received UART packet, due to which some UART byte has been lost. 1 Error occurred 0 No error occurred
16 SCI_F_ERR	UART framing error has occurred in the current received UART packet. 1 Error occurred 0 No error occurred
17 SCI_P_ERR	UART message parity error has occurred in at least one UART byte of the received UART packet. 1 Error occurred 0 No error occurred
18 HD_ERR	This bit is set when any of the ERR[1:0] is "1" and indicates a transceiver error. 1 Error occurred 0 No error occurred
19:20 ERR	Transceiver error flags received in the Message header byte of the UART packet indicated by CID and FID
21 CRC_ERR/ P_ERR	Indicates whether "PSI5 message" crc_err or the parity error in the PSI5 message indicated by CID and FID has occurred. The Parity is taken as even for PSI5 calculation: 1 Indicates crc_error/Parity Error has occurred. 0 No crc_error/parity error has occurred. **Note:** This bit always remains "0" for messages having CID and FID both as "0".
22:24 CRC	Contains the received CRC or Parity Bits of the PSI5 message, of indicated by CID and FID

Table 1567. PS_MRU_BUF2_REG1 field descriptions (continued)

Field	Description
25 XCRC_ERR	Indicates xcrc_err in the received UART packet: 1 Indicates xcrc_error occurred 0 No xcrc_error occurred
26:31 XCRC	This is the value of XCRC received in the last six bits of the UART byte, which occurs just before the information of a “new UART packet received” has been generated. Refer to *Section 61.4.1: PSI5-S UART message structure* and *Section 61.4.2.5: PSI5-S XCRC handling*, for more details.

61.3.4.6 PSI5-S MRU output buffer 2 register 2 (PS_MRU_BUF2_REG2)

Address: 0x00C8 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	DCI				PS_DATA											
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	PS_DATA															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1549. PSI5-S MRU output buffer 2 register 2 (PS_MRU_BUF2_REG2)

Table 1568. PS_MRU_BUF2_REG2 field descriptions

Field	Description
0:3 DCI	To ensure that the Application reads the PSI5 message and its timestamp consistently, Data Consistency Indicator (DCI) is added to each word of the Message Reconstruction Unit Output buffer and subsequently is transferred to the PSI5 Mailbox in system RAM.
4:31 PS_DATA	PSI5 data bits extracted from the received UART bytes.

61.3.4.7 PSI5-S MRU output buffer 2 register 3 (PS_MRU_BUF2_REG3)

Address: 0x00CC Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	DCI				0	0	0	0	TIMESTAMP							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	TIMESTAMP															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1550. PSI5-S MRU output buffer 2 register 3 (PS_MRU_BUF2_REG3)

Table 1569. PS_MRU_BUF2_REG3 field descriptions

Field	Description
0:3 DCI	To ensure that the Application reads the PSI5 message and its timestamp consistently, Data Consistency Indicator (DCI) is added to each word of the Message Reconstruction Unit Output buffer and subsequently is transferred to the PSI5 Mailbox in system RAM.
8:31 TIMESTAMP	Time stamp value appended based on the various timestamp bit settings in the PS_GLCR register, PS_PCCR_CH[n] register and the errors occurred. Refer to *Section 61.4.4: Timestamp*, for more details.

61.3.4.8 PSI5-S mbox status IRQ (PS_MBOX_SR_IRQ)

Address: 0x00E0 Access: User read/write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	MBOX_CH7	MBOX_CH6	MBOX_CH5	MBOX_CH4	MBOX_CH3	MBOX_CH2	MBOX_CH1	MBOX_CH0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1551. PSI5-S mbox status IRQ (PS_MBOX_SR_IRQ)

Table 1570. PS_MBOX_SR_IRQ field descriptions

Field	Description
24 MBOX_CH7	When set it Indicates at least one message in mailbox 7 is pending for read by the application. This bit is a “read status” of the OR of PS_MBOX_SR_CH7[*F(0-5)_READ*] bits that is MBOX_CH7 = F0_READ OR F1_READ OR...F5_READ. Hence this bit is cleared when all of the corresponding Fn_READ bits of Channel7 are cleared.
25 MBOX_CH6	When set it Indicates at least one message in mailbox 6 is pending for read by the application. This bit is a “read status” of the OR of PS_MBOX_SR_CH6[*F(0-5)_READ*] bits that is MBOX_CH6 = F0_READ OR F1_READ OR...F5_READ. Hence this bit is cleared when all of the corresponding Fn_READ bits of Channel6 are cleared.
26 MBOX_CH5	When set it Indicates at least one message in mailbox 5 is pending for read by the application. This bit is a “read status” of the OR of PS_MBOX_SR_CH5[*F(0-5)_READ*] bits that is MBOX_CH5 = F0_READ OR F1_READ OR...F5_READ. Hence this bit is cleared when all of the corresponding Fn_READ bits of Channel5 are cleared.
27 MBOX_CH4	When set it Indicates at least one message in mailbox 4 is pending for read by the application. This bit is a “read status” of the OR of PS_MBOX_SR_CH4[*F(0-5)_READ*] bits that is MBOX_CH4 = F0_READ OR F1_READ OR...F5_READ. Hence this bit is cleared when all of the corresponding Fn_READ bits of Channel4 are cleared.
28 MBOX_CH3	When set it Indicates at least one message in mailbox 3 is pending for read by the application. This bit is a “read status” of the OR of PS_MBOX_SR_CH3[*F(0-5)_READ*] bits that is MBOX_CH3 = F0_READ OR F1_READ OR...F5_READ. Hence this bit is cleared when all of the corresponding Fn_READ bits of Channel3 are cleared.
29 MBOX_CH2	When set it Indicates at least one message in mailbox 2 is pending for read by the application. This bit is a “read status” of the OR of PS_MBOX_SR_CH2[*F(0-5)_READ*] bits that is MBOX_CH2 = F0_READ OR F1_READ OR...F5_READ. Hence this bit is cleared when all of the corresponding Fn_READ bits of Channel2 are cleared.

Table 1570. PS_MBOX_SR_IRQ field descriptions (continued)

Field	Description
30 MBOX_CH1	When set it Indicates at least one message in mailbox 1is pending for read by the application. This bit is a “read status” of the OR of PS_MBOX_SR_CH1[*F(0-5)_READ*] bits that is MBOX_CH1 = F0_READ OR F1_READ OR...F5_READ. Hence this bit is cleared when all of the corresponding Fn_READ bits of Channel1 are cleared.
31 MBOX_CH0	When set it Indicates at least one message in mailbox 0is pending for read by the application. This bit is a “read status” of the OR of PS_MBOX_SR_CH0[*F(0-1)_READ*] bits that is MBOX_CH0 = F0_READ OR F1_READ. Hence this bit is cleared when all of the corresponding Fn_READ bits of Channel0 are cleared.

Note: Also refer to Figure 1584: IRQ[n] interrupts, for bit fields details.

61.3.4.9 PSI5-S error status IRQ (PS_ERR_SR_IRQ)

Address: 0x00E4 Access: User read/write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	R_UVL_ERR	N_ERR	R_OVL_ERR	F_WD_ERR	SCI_O_ERR	SCI_F_ERR	SCI_P_ERR	HD_ERR	CRC_ERR/ P_ERR	XCRC_ERR
W							w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1552. PSI5-S error status IRQ (PS_ERR_SR_IRQ)

Table 1571. PS_ERR_SR_IRQ field descriptions

Field	Description
22 R_UVL_ERR	When set indicates that message underflow error (number of UART bytes received are less than 3) is present in the message currently present in MRU buf2. This bit is the reflection of PS_MRU_BUF2_REG1[R_UVL_ERR] bit. This bit is cleared by w1c. Clearing this bit would also clear the PS_MRU_BUF2_REG1[R_UVL_ERR] bit. Correspondingly when a new message arrives in MRU_BUF2 then this bit is updated accordingly. **Note:** The bit is also writable in PS_NORMAL mode.
23 N_ERR	When set indicates that N_ERR (Number of UART bytes are not equal to PS_MSGA_CHn[Fn_byte]) is present in the message currently present in MRU buf2. This bit is the reflection of PS_MRU_BUF2_REG1[N_ERR] bit. This bit is cleared by w1c. Clearing this bit would also clear the PS_MRU_BUF2_REG1[N_ERR] bit. Correspondingly when a new message arrives in MRU_BUF2 then this bit is updated accordingly. **Note:** The bit is also writable in PS_NORMAL mode.
24 R_OVL_ERR	When set indicates that message Overflow error (number of UART bytes received have exceeded 6) is present in the message currently present in MRU buf2. This bit is the reflection of PS_MRU_BUF2_REG1[R_OVL_ERR] bit. This bit is cleared by w1c. Clearing this bit would also clear the PS_MRU_BUF2_REG1[R_OVL_ERR] bit. Correspondingly when a new message arrives in MRU_BUF2 then this bit is updated accordingly. **Note:** The bit is also writable in PS_NORMAL mode.
25 F_WD_ERR	When set indicates that Frame Watchdog error is present in the message currently present in MRU buf2. This bit is the reflection of PS_MRU_BUF2_REG1[F_WD_ERR] bit. This bit is cleared by w1c. Clearing this bit would also clear the PS_MRU_BUF2_REG1[F_WD_ERR] bit. Correspondingly when a new message arrives in MRU_BUF2 then this bit is updated accordingly. **Note:** The bit is also writable in PS_NORMAL mode.
26 SCI_O_ERR	When set indicates that UART Overrun error (from the UART side) is present in the message currently present in MRU buf2. When “1”, then this bit indicates that some UART byte belonging to the concerned UART packet, has been overwritten by another UART byte and is lost. This bit is the reflection of PS_MRU_BUF2_REG1[SCI_O_ERR] bit. This bit is cleared by w1c. Clearing this bit would also clear the PS_MRU_BUF2_REG1[SCI_O_ERR] bit. Correspondingly when a new message arrives in MRU_BUF2 then this bit is updated accordingly. **Note:** The bit is also writable in PS_NORMAL mode.

Table 1571. PS_ERR_SR_IRQ field descriptions (continued)

Field	Description
27 SCI_F_ERR	When set indicates that UART Framing error is present in the message currently present in MRU buf2. If this bit is set then it indicates to the application that the synchronization of the UART module is lost and any subsequent UART bytes received will be error prone. In this case the application is required to reset the UART module by using the PS_GCR[SR] bit. Subsequently it should restart the transmission. This bit is the reflection of PS_MRU_BUF2_REG1[SCI_F_ERR] bit. This bit is cleared by w1c. Clearing this bit would also clear the PS_MRU_BUF2_REG1[SCI_F_ERR] bit. Correspondingly when a new message arrives in MRU_BUF2 then this bit is updated accordingly. **Note:** The bit is also writable in PS_NORMAL mode.
28 SCI_P_ERR	When set indicates that UART parity error is present in the message currently present in MRU buf2. This bit will be set when there is a parity error in any UART byte using which the concerned UART packet was formed. This bit is the reflection of PS_MRU_BUF2_REG1[SCI_P_ERR] bit. This bit is cleared by w1c. Clearing this bit would also clear the PS_MRU_BUF2_REG1[SCI_P_ERR] bit. Correspondingly when a new message arrives in MRU_BUF2 then this bit is updated accordingly. **Note:** The bit is also writable in PS_NORMAL mode.
29 HD_ERR	When set indicates that Header error (E0/E1) is present in the message currently present in MRU buf2. This bit is the reflection of PS_MRU_BUF2_REG1[HD_ERR] bit. This bit is cleared by w1c. Clearing this bit would also clear the PS_MRU_BUF2_REG1[HD_ERR] bit. Correspondingly when a new message arrives in MRU_BUF2 then this bit is updated accordingly. **Note:** The bit is also writable in PS_NORMAL mode.
30 CRC_ERR/P_ERR	When set indicates that PSI5 CRC/Parity error is present in the message currently present in MRU buf2. This bit is the reflection of PS_MRU_BUF2_REG1[CRC_ERR/P_ERR] bit. This bit is cleared by w1c. Clearing this bit would also clear the PS_MRU_BUF2_REG1[CRC_ERR/P_ERR] bit. Correspondingly when a new message arrives in MRU_BUF2 then this bit is updated accordingly. The bit is also writable in PS_NORMAL mode. **Note:** This bit always remains “0” for messages having CID and FID both as “0”.
31 XCRC_ERR	When set indicates that XCRC error is present in the message currently present in MRU buf2. This bit is the reflection of PS_MRU_BUF2_REG1[XCRC_ERR] bit. This bit is cleared by w1c. Clearing this bit would also clear the PS_MRU_BUF2_REG1[XCRC_ERR] bit. Correspondingly when a new message arrives in MRU_BUF2 then this bit is updated accordingly. **Note:** The bit is also writable in PS_NORMAL mode.

Note: Also refer to Figure 1584: IRQ[n] interrupts, for bit fields details.

If an interrupt is triggered by this bit it cannot be guaranteed that the message containing this error flag is still in MRU_BUF2, in case the DMA is being used to transfer messages to the mailbox. The subsequent message in MRU_BUF2 may not have this flag set, in which case it will no longer be visible to an ISR triggered by the previous message. The error flags are stored with the message in the mailbox and can be checked when reading the message

61.3.4.10 PSI5-S mailbox select IRQ[irq_n] (PS_MBOX_SEL_IRQ[irq_n])

Address: 0x00E8 + [irq_n]*0x8 (irq_n=0 to 7) Access: User read/write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	MBOX_CH7_SEL	MBOX_CH6_SEL	MBOX_CH5_SEL	MBOX_CH4_SEL	MBOX_CH3_SEL	MBOX_CH2_SEL	MBOX_CH1_SEL	MBOX_CH0_SEL
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1553. PSI5-S mailbox select IRQ[irq_n] (PS_MBOX_SEL_IRQ[irq_n])

Table 1572. PS_MBOX_SEL_irq[irq_n] field descriptions

Field	Description
24 MBOX_CH7_SEL	Interrupt Selection source for irq[irq_n] interrupt 0 PS_MBOX_SR_IRQ[MBOX_CH7] does not contribute to generation of irq[irq_n] interrupt. 1 PS_MBOX_SR_IRQ[MBOX_CH7] contributes to generation of irq[irq_n] interrupt. It is also writable in ps_normal mode.
25 MBOX_CH6_SEL	Interrupt Selection source for irq[irq_n] interrupt 0 PS_MBOX_SR_IRQ[MBOX_CH6] does not contribute to generation of irq[irq_n] interrupt. 1 PS_MBOX_SR_IRQ[MBOX_CH6] contributes to generation of irq[irq_n] interrupt. It is also writable in ps_normal mode.
26 MBOX_CH5_SEL	Interrupt Selection source for irq[irq_n] interrupt 0 PS_MBOX_SR_IRQ[MBOX_CH5] does not contribute to generation of irq[irq_n] interrupt. 1 PS_MBOX_SR_IRQ[MBOX_CH5] contributes to generation of irq[irq_n] interrupt. It is also writable in ps_normal mode.

Table 1572. PS_MBOX_SEL_irq[irq_n] field descriptions (continued)

Field	Description
27 MBOX_CH4_SEL	Interrupt Selection source for irq[irq_n] interrupt 0 PS_MBOX_SR_IRQ[MBOX_CH4] does not contribute to generation of irq[irq_n] interrupt. 1 PS_MBOX_SR_IRQ[MBOX_CH4] contributes to generation of irq[irq_n] interrupt. It is also writable in ps_normal mode.
28 MBOX_CH3_SEL	Interrupt Selection source for irq[irq_n] interrupt 0 PS_MBOX_SR_IRQ[MBOX_CH3] does not contribute to generation of irq[irq_n] interrupt. 1 PS_MBOX_SR_IRQ[MBOX_CH3] contributes to generation of irq[irq_n] interrupt. It is also writable in ps_normal mode.
29 MBOX_CH2_SEL	Interrupt Selection source for irq[irq_n] interrupt 0 PS_MBOX_SR_IRQ[MBOX_CH2] does not contribute to generation of irq[irq_n] interrupt. 1 PS_MBOX_SR_IRQ[MBOX_CH2] contributes to generation of irq[irq_n] interrupt. It is also writable in ps_normal mode.
30 MBOX_CH1_SEL	Interrupt Selection source for irq[irq_n] interrupt 0 PS_MBOX_SR_IRQ[MBOX_CH1] does not contribute to generation of irq[irq_n] interrupt. 1 PS_MBOX_SR_IRQ[MBOX_CH1] contributes to generation of irq[irq_n] interrupt. It is also writable in ps_normal mode.
31 MBOX_CH0_SEL	Interrupt Selection source for irq[irq_n] interrupt 0 PS_MBOX_SR_IRQ[MBOX_CH0] does not contribute to generation of irq[irq_n] interrupt. 1 PS_MBOX_SR_IRQ[MBOX_CH0] contributes to generation of irq[irq_n] interrupt. It is also writable in ps_normal mode.

Note: Also refer to Figure 1584: IRQ[n] interrupts, for bit fields details.

61.3.4.11 PSI5-S error select IRQ[iq_n] (PS_ERR_SEL_IRQ[irq_n])

Address: 0x00EC + [irq_n]*0x8 (irq_n=0 to 7) Access: User read/write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	R_UVL_ERR_SEL	N_ERR_SEL	R_OVL_ERR_SEL	F_WD_ERR_SEL	SCI_O_ERR_SEL	SCI_F_ERR_SEL	SCI_P_ERR_SEL	HD_ERR_SEL	CRC_ERR_SEL	XCRC_ERR_SEL
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1554. PSI5-S error enable irq[irq_n] (PS_ERR_SEL_IRQ[irq_n])

Table 1573. PS_ERR_SEL_IRQ[irq_n] field descriptions

Field	Description
22 R_UVL_ERR_SEL	Interrupt Selection source for irq[irq_n] interrupt 0 PS_ERR_SR_IRQ[R_UVL_ERR_SEL] does not contribute to generation of irq[irq_n] interrupt. 1 PS_ERR_SR_IRQ[R_UVL_ERR_SEL] contributes to generation of irq[irq_n] interrupt. It is also writable in ps_normal mode.
23 N_ERR_SEL	Interrupt Selection source for irq[irq_n] interrupt 0 PS_ERR_SR_IRQ[N_ERR_SEL] does not contribute to generation of irq[irq_n] interrupt. 1 PS_ERR_SR_IRQ[N_ERR_SEL] contributes to generation of irq[irq_n] interrupt. It is also writable in ps_normal mode.
24 R_OVL_ERR_SEL	Interrupt Selection source for irq[irq_n] interrupt 0 PS_ERR_SR_IRQ[R_OVL_ERR_SEL] does not contribute to generation of irq[irq_n] interrupt. 1 PS_ERR_SR_IRQ[R_OVL_ERR_SEL] contributes to generation of irq[irq_n] interrupt. It is also writable in ps_normal mode.
25 F_WD_ERR_SEL	Interrupt Selection source for irq[irq_n] interrupt 0 PS_ERR_SR_IRQ[F_WD_ERR_SEL] does not contribute to generation of irq[irq_n] interrupt. 1 PS_ERR_SR_IRQ[F_WD_ERR_SEL] contributes to generation of irq[irq_n] interrupt. It is also writable in ps_normal mode.

Table 1573. PS_ERR_SEL_IRQ[irq_n] field descriptions (continued)

Field	Description
26 SCI_O_ERR_SEL	Interrupt Selection source for irq[irq_n] interrupt 0 PS_ERR_SR_IRQ[SCI_O_ERR_SEL] does not contribute to generation of irq[irq_n] interrupt. 1 PS_ERR_SR_IRQ[SCI_O_ERR_SEL] contributes to generation of irq[irq_n] interrupt. It is also writable in ps_normal mode.
27 SCI_F_ERR_SEL	Interrupt Selection source for irq[irq_n] interrupt 0 PS_ERR_SR_IRQ[SCI_F_ERR_SEL] does not contribute to generation of irq[irq_n] interrupt. 1 PS_ERR_SR_IRQ[SCI_F_ERR_SEL] contributes to generation of irq[irq_n] interrupt. It is also writable in ps_normal mode.
28 SCI_P_ERR_SEL	Interrupt Selection source for irq[irq_n] interrupt 0 PS_ERR_SR_IRQ[SCI_P_ERR_SEL] does not contribute to generation of irq[irq_n] interrupt. 1 PS_ERR_SR_IRQ[SCI_P_ERR_SEL] contributes to generation of irq[irq_n] interrupt. It is also writable in ps_normal mode.
29 HD_ERR_SEL	Interrupt Selection source for irq[irq_n] interrupt 0 PS_ERR_SR_IRQ[HD_ERR_SEL] does not contribute to generation of irq[irq_n] interrupt. 1 PS_ERR_SR_IRQ[HD_ERR_SEL] contributes to generation of irq[irq_n] interrupt. It is also writable in ps_normal mode.
30 CRC_ERR_SEL	Interrupt Selection source for irq[irq_n] interrupt 0 PS_ERR_SR_IRQ[CRC_ERR_SEL] does not contribute to generation of irq[irq_n] interrupt. 1 PS_ERR_SR_IRQ[CRC_ERR_SEL] contributes to generation of irq[irq_n] interrupt. It is also writable in ps_normal mode.
31 XCRC_ERR_SEL	Interrupt Selection source for irq[irq_n] interrupt 0 PS_ERR_SR_IRQ[XCRC_ERR_SEL] does not contribute to generation of irq[irq_n] interrupt. 1 PS_ERR_SR_IRQ[XCRC_ERR_SEL] contributes to generation of irq[irq_n] interrupt. It is also writable in ps_normal mode.

Note: Also refer to Figure 1584: IRQ[n] interrupts, for bit fields details.

61.3.4.12 PSI5-S watchdog error status and watchdog timestamp status register (PS_WDGTSSR)

Address: 0x0128 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	F_WD_ERR_STATUS							0	WDGTS_STATUS							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	WDGTS_STATUS															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1555. PSI5-S watchdog error status and watchdog timestamp status register (PS_WDGTSSR)

Table 1574. PS_WDGTSSR bit field details

Field	Description
0:6 F_WD_ERR_STATUS	F_WD_ERR_STATUS[7:1] These bits give the watchdog error status for the watchdog of each corresponding channel. Thus F_WD_ERR_STATUS[1] will give the watchdog error status for watchdog of channel1. Similarly F_WD_ERR_STATUS[2] will give the watchdog error status for watchdog of channel2 and so on till F_WD_ERR_STATUS[7], which gives the watchdog error status for watchdog of channel7. As soon as the watchdog of the corresponding channel has a watchdog overrun error, the error status is latched in the corresponding bits of this register. **Note:** PS_WDGTSSR is a hardware cleared register. These bits are automatically cleared by the hardware when PS_WDGTSSR is read by the application. This register should ALWAYS be read whenever the PS_MRU_BUF2_REG1[F_WD_ERR] bit is set. Failing to do so will result in the PS_MRU_BUF2_REG1[F_WD_ERR] bit continuing to be set for respective “CID” whenever the PS_WDGTSSR[F_WD_ERR_STATUS[CID]] bit is set. This would happen in subsequent messages saved to the mailbox even though no further watchdog error has occurred for the respective CID.This happens since for each received PSI5 message in the MRU buffers the status of the corresponding PS_MRU_BUF2_REG1[F_WD_ERR] is picked from the PS_WDGTSSR[F_WD_ERR_STATUS[CID]] bit.
8:31 WDGTS_STATUS	Timestamp value of that specific channel, the timestamp of which is captured when the watchdog error of that specific channel has occurred. This register always contains the timestamp value corresponding to the last occurring Watch Dog error. If due to bad programming watchdog error of more than one channel happen simultaneously then this register contains the timestamp value of the highest channel. For each channel, whether Timestamp A or B is captured depends on the value of PS_MSGA_CHn[TIMESTAMP_A_B_SEL] register bit. **Note:** These bits are automatically cleared by the hardware when PS_WDGTSSR is read by the application.

Note: All bits of PS_WDGTSSR register are self clearing on read. This implies that in case this register is read by the application before the corresponding PS_WDGTSSR[F_WD_ERR_STATUS[CHn]] has had a chance to be sent in the next upcoming PSI5 message, then this bit would be sent as a "0" in this upcoming PSI5 message.

61.3.4.13 PSI5-S ECU to sensor direct command write register (PS_DIRCMD)

Address: 0x012C Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	DIRCMD_BYTE3								DIRCMD_BYTE2							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	DIRCMD_BYTE1								DIRCMD_BYTE0							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1556. PSI5-S ECU to sensor direct command write register (PS_DIRCMD)

Table 1575. PS_DIRCMD field descriptions

Field	Description
0:7 DIRCMD_BYTE3	Byte3 of the Direct Command The Direct Command is a special ECU to Sensor command that is directly written by the CPU for transmission to the UART **Note:** These bits can also be written in the PS_NORMAL mode.These bits can only be written when the PS_GLSR[DIRCMD_RDY] is "1"
8:15 DIRCMD_BYTE2	Byte2 of the Direct Command The Direct Command is a special ECU to Sensor command that is directly written by the CPU for transmission to the UART **Note:** These bits can also be written in the PS_NORMAL mode.These bits can only be written when the PS_GLSR[DIRCMD_RDY] is "1"
16:23 DIRCMD_BYTE1	Byte1 of the Direct Command The Direct Command is a special ECU to Sensor command that is directly written by the CPU for transmission to the UART **Note:** These bits can also be written in the PS_NORMAL mode.These bits can only be written when the PS_GLSR[DIRCMD_RDY] is "1"
24:31 DIRCMD_BYTE0	Byte0 of the Direct Command The Direct Command is a special ECU to Sensor command that is directly written by the CPU for transmission to the UART. **Note:** These bits can also be written in the PS_NORMAL mode.These bits can only be written when the PS_GLSR[DIRCMD_RDY] is "1"

Note: A "w1c" should be done on the PS_GLSR[DIRCMD_RDY] so that the command written on the PS_DIRCMD register is queued for transmission on the UART TX line. As soon as any ongoing transmission on the UART TX is finished, the PS_DIRCMD starts being transmitted

on the UART Tx line. When PS_GCR[TDFBM] = 0, then the bits on the UART Tx line are shifted in the following sequence "DIRCMD_BYTE0[0], DIRCMD_BYTE0[1]....DIRCMD_BYTE3[0]....DIRCMD_BYTE3[7]" with DIRCMD_BYTE0[0] being shifted first.

61.3.4.14 PSI5-S channel 0 message configuration register A (PS_MSGA_CH0)

Address: 0x016C Access: User read/write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	F0_BYTE			0	0	TIME_STAMP_A_B_SEL	0	0	TSBUF_EN	0	0
W													TSBUF_CLR			
Reset	0	0	0	0	0	0	0	0	0	1	0	1	0	0	0	1

Figure 1557. PSI5-S channel 0 message configuration register A (PS_MSGA_CH0)

Table 1576. PS_MSGA_CH0 field descriptions

Field	Description
21:23 F0_BYTE	Number of UART bytes comprising one UART packet. Applicable only to special messages having CID = 0 in their Message Header. Legal values are 0 or between 3 and 6. Value of 0 means that the slot is not enabled and a message with the corresponding frame ID is not expected to occur. Refer to note in *Section 61.3.4.18: PSI5-S channel message configuration register B (PS_MSGB_CHn)* that explains the recalculation of Fn_bytes.
26 TIME_STAMP_A_B_SEL	Whether the Timestamp corresponding to timestamp counter A or B, is appended to messages targeted for Channel0. 0 Timestamp counter A 1 Timestamp counter B
28 TSBUF_CLR	This bit is used to clear the local timestamp buffer for the respective channel.Once cleared the buffer remains clear till the event to capture time stamp is triggered. 1 Timestamp buffer is cleared. This bit is one shot and is always read as "0". **Note:** This bit is also writable in PS_NORMAL mode.
29 TSBUF_EN	This bit controls whether timestamp buffer maintains a value "0" or is refreshed on a timestamp "capture event" 0 Timestamp buffer maintains value "0". 1 Timestamp buffer responds to a "capture event" and captures the respective timestamp value.

61.3.4.15 PSI5-S channel 0 message configuration register B (PS_MSGB_CH0)

Address: 0x0170 Access: User read/write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	F0_payload				
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0

Figure 1558. PSI5-S channel 0 message configuration register B (PS_MSGB_CH0)

Table 1577. PS_MSGB_CH0 field descriptions

Field	Description
27:31 F0_payload	Number of bits in payload region of the message (only for channels having CID as “0” in the message header). This channel is always asynchronous. Legal values are between 8 and 28. Any value programmed as less than 8 is defaulted to 8 and any value programmed as greater than 28 is defaulted to 28.This updation occurs in the NORMAL mode and when read back the register will be read with the updated value.

61.3.4.16 PSI5-S mailbox status register channel 0 (PS_MBOX_SR_CH0)

Address: 0x0178 Access: User read/write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	F1_ERR	F1_OV	F1_READ	F0_ERR	F0_OV	F0_READ
W											w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1559. PSI5-S mailbox status register channel 0 (PS_MBOX_SR_CH0)

Table 1578. PS_MBOX_SR_CH0 field descriptions

Field	Description
26 F1_ERR	1 Indicates that the message that has just been read from the PS_MRU_BUF2 and is to be stored in (MBOX0, LOC1) by the DMA, has one or more _ERR bits set. 0 Indicates that the message that has just been read from the PS_MRU_BUF2 and is to be stored in (MBOX0, LOC1) by the DMA, has ALL the "_ERR" bits as "0". This bit is the "OR" of the contents of the PS_MBOX_SR_IRQ register. It is updated when an unrecoverable message (with the CID = 0 and FID=1) is read from the MRU_BUF2. It is a w1c and software should clear it whenever it reads the message from the MBOX0, LOC1. This bit is also writable in ps_normal mode. **Note:** Also refer to *Figure 1584: IRQ[n] interrupts*, for bit field details.
27 F1_OV	1 Indicates that there is an unread message stored in MBOX0, LOC1 which will be overwritten with the new message, that has just been read from the PS_MRU_BUF2 by the DMA. 0 No overwrite of unread message in MBOX0, LOC1 It is a w1c and software should clear it whenever it reads the message from the MBOX0, LOC1. This bit is also writable in ps_normal mode.
28 F1_READ	MBOX0, LOC1 is reserved for any unrecoverable message (XCRC_ERR has occurred or it is an "illegal message"). It can correspond to any channel from 1 to 7. 1 It indicates that the message that has just been read from PS_MRU_BUF2, will be stored in MBOX0, LOC1 by the DMA. 0 No unread message in MBOX0.LOC1 It is a w1c and software should clear it whenever it reads the message from the MBOX0, LOC1. This bit is also writable in ps_normal mode.
29 F0_ERR	1 Indicates that the message that has just been read from the PS_MRU_BUF2 and is to be stored in (MBOX0, LOC0) by the DMA, has one or more _ERR bits set. 0 Indicates that the message that has just been read from the PS_MRU_BUF2 and is to be stored in (MBOX0, LOC0) by the DMA, has ALL the "_ERR" bits as "0". This bit is the "OR" of the contents of the PS_MBOX_SR_IRQ register. It is updated when a transceiver message (with the CID = 0 and FID=0) is read from the MRU_BUF2. It is a w1c and software should clear it whenever it reads the message from the MBOX0, LOC0. This bit is also writable in ps_normal mode. **Note:** Also refer to *Figure 1584: IRQ[n] interrupts*, for bit field details.

Table 1578. PS_MBOX_SR_CH0 field descriptions (continued)

Field	Description
30 F0_OV	1 Indicates that there is an unread message stored in MBOX0, LOC0 which will be overwritten with the new message, that has just been read from the PS_MRU_BUF2 by the DMA. 0 No overwrite of unread message in MBOX0, LOC0. It is a w1c and software should clear it whenever it reads the message from the MBOX0, LOC0. This bit is also writable in ps_normal mode.
31 F0_READ	MBOX0, LOC0 is reserved for a transceiver message. 1 It indicates that the message that has just been read from PS_MRU_BUF2 will be stored in MBOX0, LOC0 by the DMA. 0 No unread message in MBOX0, LOC0. It is a w1c and software should clear it whenever it reads the message from the MBOX0, LOC0. This bit is also writable in ps_normal mode.

61.3.4.17 PSI5-S channel message configuration register A (PS_MSGA_CHn)

Address: 0x0190 + [n-1]*0x3C (n=1 to 7) Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	L_PC_EN	F5_BYTE			L_PC5	F4_BYTE			L_PC4	F3_BYTE			L_PC3	F2_BYTE		
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	L_PC2	F1_BYTE			L_PC1	F0_BYTE			L_PC0	MODE	TIME_STAMP_A_B_SEL	TMSG_TCMD	0	TSBUF_EN	G_PC	CH_EN
W													TSBUF_CLR			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1560. PSI5-S channel message configuration register A (PS_MSGA_CHn)

Table 1579. PS_MSGA_CHn field descriptions

Field	Description
0 L_PC_EN	Selects whether the parity/CRC selection for a PSI5 message is to be controlled globally on a channel basis (through the G_PC bits) or locally on a per frame basis (through the L_PC[frame] bits). 0 Parity/CRC option controlled globally through the G_PC bit 1 Parity/CRC option controlled locally through the L_PC[frame] bits individually for message of each frame.
1:3 F5_BYTE	Number of bytes in UART packet for message corresponding to frame 5. Legal values are 0 or between 3 and 6. Value of 0 means that the slot is not enabled (unconfigured) and a message with the corresponding frame ID is not expected to occur. In this case the value programmed in PS_MSGB_CHn[F5_payload] is immaterial. Refer to note in *Section 61.3.4.18: PSI5-S channel message configuration register B (PS_MSGB_CHn)* that explains the recalculation of Fn_bytes.
4 L_PC5	Effective when L_PC_EN == 1. Controls the local parity/CRC option for the message 0 Parity selected for message for frame5 1 CRC selected for PSI5 message for frame5
5:7 F4_BYTE	Number of bytes in UART packet for message corresponding to frame 4. Legal values are 0 or between 3 and 6. Value of 0 means that the slot is not enabled (unconfigured) and a message with the corresponding frame ID is not expected to occur. In this case the value programmed in PS_MSGB_CHn[F4_payload] is immaterial. Refer to note in *Section 61.3.4.18: PSI5-S channel message configuration register B (PS_MSGB_CHn)* that explains the recalculation of Fn_bytes.
8 L_PC4	Effective when L_PC_EN == 1. Controls the local parity/CRC option for the message 0 Parity selected for message for frame4 1 CRC selected for PSI5 message for frame4
9:11 F3_BYTE	Number of bytes in UART packet for message corresponding to frame 3. Legal values are 0 or between 3 and 6. Value of 0 means that the slot is not enabled (unconfigured) and a message with the corresponding frame ID is not expected to occur. In this case the value programmed in PS_MSGB_CHn[F3_payload] is immaterial. Refer to note in *Section 61.3.4.18: PSI5-S channel message configuration register B (PS_MSGB_CHn)* that explains the recalculation of Fn_bytes.
12 L_PC3	Effective when L_PC_EN == 1. Controls the local parity/CRC option for the message 0 Parity selected for message for frame3 1 CRC selected for PSI5 message for frame3.
13:15 F2_BYTE	Number of bytes in UART packet for message corresponding to frame 2. Legal values are 0 or between 3 and 6. Value of 0 means that the slot is not enabled (unconfigured) and a message with the corresponding frame ID is not expected to occur. In this case the value programmed in PS_MSGB_CHn[F2_payload] is immaterial. Refer to note in *Section 61.3.4.18: PSI5-S channel message configuration register B (PS_MSGB_CHn)* that explains the recalculation of Fn_bytes.
16L_PC2	Effective when L_PC_EN == 1. Controls the local parity/CRC option for the message 0 Parity selected for message for frame2 1 CRC selected for PSI5 message for frame2.

Table 1579. PS_MSGA_CHn field descriptions (continued)

Field	Description
17:19 F1_BYTE	Number of bytes in UART packet for message corresponding to frame 1. Legal values are 0 or between 3 and 6. Value of 0 means that the slot is not enabled (unconfigured) and a message with the corresponding frame ID is not expected to occur. In this case the value programmed in PS_MSGB_CHn[F1_payload] is immaterial. Refer to note in *Section 61.3.4.18: PSI5-S channel message configuration register B (PS_MSGB_CHn)* that explains the recalculation of Fn_bytes.
20 L_PC1	Effective when L_PC_EN == 1. Controls the local parity/CRC option for the message. 0 Parity selected for message for frame1 1 CRC selected for PSI5 message for frame1
21:23 F0_BYTE	Number of bytes in UART packet for message corresponding to frame 0. Legal values are 0 or between 3 and 6. Value of 0 means that the slot is not enabled (unconfigured) and a message with the corresponding frame ID is not expected to occur. In this case the value programmed in PS_MSGB_CHn[F0_payload] is immaterial. Refer to note in *Section 61.3.4.18: PSI5-S channel message configuration register B (PS_MSGB_CHn)* that explains the recalculation of Fn_bytes.
24 L_PC0	Effective when L_PC_EN == 1. Controls the local parity/CRC option for the message. 0 Parity selected for message for frame0 1 CRC selected for PSI5 message for frame0
25 MODE	This bit selects the operating modes between sync and async for the channel. 1 Asynchronous 0 Synchronous
26 TIME_STAMP_A_B_SEL	Decides whether timestamp counter A or B value is captured in the local timestamp buffer, whenever a “capture event” occurs. 0 Timestamp counter A 1 Timestamp counter B
27 TMSG_TCMD	Whether the timestamp is recorded when the message is received or when the channel command is moved from the TxFIFO to the UART Tx register (emulates a sync pulse) 1 Timestamp recorded based on message header byte received. 0 Timestamp recorded based on “emulated” sync pulse.
28 TSBUF_CLR	This bit is used to clear the local timestamp buffer for the respective channel.Once cleared the buffer remains clear till the event to capture time stamp is triggered. 1 Timestamp buffer is cleared. This bit is one shot and is always read as “0”. **Note:** This bit is also writable in PS_NORMAL mode.
29 TSBUF_EN	This bit controls whether timestamp buffer maintains a value “0” or is refreshed on a timestamp “capture event”. 0 Timestamp buffer maintains value “0”. 1 Timestamp buffer responds to a “capture event” and captures the respective timestamp value.

Table 1579. PS_MSGA_CHn field descriptions (continued)

Field	Description
30 G_PC	This bit globally selects the Parity or the CRC option for all the Frames of that particular channel provided L_PC_EN is set “0”. 0 Parity option selected globally for messages of all frames. 1 CRC option selected globally for messages of all frames.
31 CH_EN	PSI5 Channel Enable. This bit decides if the IP takes the data belonging to the specific channel as “legal” or “illegal”. 0 Concerned PSI5 channel is disabled. Any data occurring with a CID value of a disabled channel is treated as an “illegal message” and send to Channel 0 unrecoverable message area ($MBOX_0$, LOC_1). 1 Concerned PSI5 channel is enabled.Any data occurring with a CID value of an enabled channel can either be treated as “legal” or “illegal”. Refer to *Section 61.4.2.1: PSI5 message extraction* for the definition of legal and illegal messages. Legal messages are sent to $MBOX_{CID}$, LOC_{FID} while illegal messages are sent to $MBOX_0$, BOX_0, LOC_1.

61.3.4.18 PSI5-S channel message configuration register B (PS_MSGB_CHn)

Address: 0x0194 + [n-1]*0x3C (n=1 to 7) Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	F5_payload						F4_payload				F3_payload			
W																
Reset	0	0	0	1	0	0	0	0	1	0	0	0	0	1	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	F3_payload	F2_payload						F1_payload						F0_payload		
W																
Reset	0	0	1	0	0	0	0	1	0	0	0	0	1	0	0	0

Figure 1561. PSI5-S channel message configuration register B (PS_MSGB_CHn)

Table 1580. PS_MSGB_CHn field descriptions

Field	Description
2:6 F5_payload	Number of bits in the payload region of the PSI5 message corresponding to frame 5. Legal values are between 8 and 28. Any value programmed as less than 8 is defaulted to 8 and any value programmed as greater than 28 is defaulted to 28.This updation occurs in the NORMAL mode and when read back the register will be read with the updated value.
7:11 F4_payload	Number of bits in the payload region of the PSI5 message corresponding to frame 4. Legal values are between 8 and 28. Any value programmed as less than 8 is defaulted to 8 and any value programmed as greater than 28 is defaulted to 28.This updation occurs in the NORMAL mode and when read back the register will be read with the updated value.

Table 1580. PS_MSGB_CHn field descriptions (continued)

Field	Description
12;16 F3_payload	Number of bits in the payload region of the PSI5 message corresponding to frame 3. Legal values are between 8 and 28. Any value programmed as less than 8 is defaulted to 8 and any value programmed as greater than 28 is defaulted to 28.This updation occurs in the NORMAL mode and when read back the register will be read with the updated value.
17:21 F2_payload	Number of bits in the payload region of the PSI5 message corresponding to frame 2. Legal values are between 8 and 28. Any value programmed as less than 8 is defaulted to 8 and any value programmed as greater than 28 is defaulted to 28.This updation occurs in the NORMAL mode and when read back the register will be read with the updated value.
22:26 F1_payload	Number of bits in the payload region of the PSI5 message corresponding to frame 1. Legal values are between 8 and 28. Any value programmed as less than 8 is defaulted to 8 and any value programmed as greater than 28 is defaulted to 28.This updation occurs in the NORMAL mode and when read back the register will be read with the updated value.
27:31 F0_payload	Number of bits in the payload region of the PSI5 message corresponding to frame 0. Legal values are between 8 and 28. Any value programmed as less than 8 is defaulted to 8 and any value programmed as greater than 28 is defaulted to 28This updation occurs in the NORMAL mode and when read back the register will be read with the updated value.

Note: The IP expects legal values to be programmed for Fn_byte field. If this is not followed then the IP reprogrammed the registers with default values. Whenever the value of "Fn_byte" field becomes 1 or 2 it is defaulted to 3 and value of 7 is defaulted to 6. This updation happens once the IP enters the PS_NORMAL mode.

In addition to the above, the following checks are also performed.

By default the IP will expect that there is coherence between the expected number of bytes calculated through the "minUARTbytes" and the number of bytes in the "Fn_byte" field. Refer to Figure 1571: PSI5-S packet structure for definition of "minUARTbytes". If there is no coherence (that is the above two calculations do not match) then the IP does the following recalculation:

When Programmed Fn_byte > "minUARTbytes", then the IP will expect that the additional extra bytes would be stuffed with 0s from the transceiver. Rest of the checking will be done as described in Section 61.4.2.2: PSI5 "recoverable and unrecoverable" messages.

When Fn_byte < "minUARTbytes" then the "Fn_bytes" is internally "increased" such that the expanded "Fn_bytes" is now equal to minUARTbytes". Refer to Figure 1571: PSI5-S packet structure for definition of "minUARTbytes". Using this expanded "Fn_bytes" value the message that would be arriving is analyzed. This would mean that no data bits are lost. Rest of the checking will be done as described in Section 61.4.2.2: PSI5 "recoverable and unrecoverable" messages.

The "recalculated" Fn_byte field is automatically updated by the IP once the IP enters the NORMAL mode of operation. For the subsequent occurrence in the document, all references to "Fn_byte" would mean the recalculated value of "Fn_byte".

61.3.4.19 PSI5-S mailbox status register channel (PS_MBOX_SR_CHn)

Address: 0x019C +[n-1]*0x3C (n= 1 to 7) Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	F5_ERR	F5_OV
W															w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	F5_READ	F4_ERR	F4_OV	F4_READ	F3_ERR	F3_OV	F3_READ	F2_ERR	F2_OV	F2_READ	F1_ERR	F1_OV	F1_READ	F0_ERR	F0_OV	F0_READ
W	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1562. PSI5-S mailbox status register channel (PS_MBOX_SR_CHn)

Table 1581. PS_MBOX_SR_CHn field descriptions

Field	Description
14 F5_ERR	1 Indicates that the message that has just been read from the PS_MRU_BUF2 and is to be stored in (MBOX[n], LOC5) by the DMA, has one or more _ERR bits set. 0 Indicates that the message that has just been read from the PS_MRU_BUF2 and is to be stored in (MBOX[n], LOC5) by the DMA, has ALL the "_ERR" bits as "0". This bit is the "OR" of the contents of the PS_MBOX_SR_IRQ register. It is updated when the corresponding message (with the CID = n and FID=5) is read from the MRU_BUF2 by the DMA. It is a w1c and software should clear it whenever it reads the message from the MBOX[n], LOC5. This bit is also writeable in ps_normal mode. **Note:** Also refer to *Figure 1584: IRQ[n] interrupts*, for bit field details.
15 F5_OV	1 Indicates that there is an unread message stored in MBOX[n], LOC5 which will be overwritten with the new message, that has just been read from the PS_MRU_BUF2 by the DMA. 0 No overwrite of unread message in MBOX[n], LOC5. It is a w1c and software should clear it whenever it reads the message from the MBOX[n], LOC5. This bit is also writable in ps_normal mode.
16 F5_READ	1 It indicates that the message that has just been read from PS_MRU_BUF2, will be stored in MBOX[n], LOC5. 0 No unread message in MBOX[n], LOC5. It is a w1c and software should clear it whenever it reads the message from the MBOX[n], LOC5. This bit is also writable in ps_normal mode.

Table 1581. PS_MBOX_SR_CHn field descriptions (continued)

Field	Description
17 F4_ERR	1 Indicates that the message that has just been read from the PS_MRU_BUF2 and is to be stored in (MBOX[n], LOC4) by the DMA, has one or more _ERR bits set. 0 Indicates that the message that has just been read from the PS_MRU_BUF2 and is to be stored in (MBOX[n], LOC4) by the DMA, has ALL the "_ERR" bits as "0". This bit is the "OR" of the contents of the PS_MBOX_SR_IRQ register. It is updated when the corresponding message (with the CID = n and FID=4) is read from the MRU_BUF2 by the DMA. It is a w1c and software should clear it whenever it reads the message from the MBOX[n], LOC4. This bit is also writable in ps_normal mode. **Note:** Also refer to *Figure 1584: IRQ[n] interrupts*, for bit field details.
18 F4_OV	1 Indicates that there is an unread message stored in MBOX[n], LOC4 which will be overwritten with the new message, that has just been read from the PS_MRU_BUF2 by the DMA. 0 No overwrite of unread message in MBOX[n], LOC4. It is a w1c and software should clear it whenever it reads the message from the MBOX[n], LOC4. This bit is also writable in ps_normal mode.
19 F4_READ	1 It indicates that the message that has just been read from PS_MRU_BUF2, will be stored in MBOX[n], LOC4. 0 No unread message in MBOX[n], LOC4. It is a w1c and software should clear it whenever it reads the message from the MBOX[n], LOC4. This bit is also writable in ps_normal mode.
20 F3_ERR	1 Indicates that the message that has just been read from the PS_MRU_BUF2 and is to be stored in (MBOX[n], LOC3) by the DMA, has one or more _ERR bits set. 0 Indicates that the message that has just been read from the PS_MRU_BUF2 and is to be stored in (MBOX[n], LOC3) by the DMA, has ALL the "_ERR" bits as "0". This bit is the "OR" of the contents of the PS_MBOX_SR_IRQ register. It is updated when the corresponding message (with the CID = n and FID=3) is read from the MRU_BUF2 by the DMA. It is a w1c and software should clear it whenever it reads the message from the MBOX[n], LOC3. This bit is also writable in ps_normal mode. **Note:** Also refer to *Figure 1584: IRQ[n] interrupts*, for bit field details.
21 F3_OV	1 Indicates that there is an unread message stored in MBOX[n], LOC3 which will be overwritten with the new message, that has just been read from the PS_MRU_BUF2 by the DMA. 0 No overwrite of unread message in MBOX[n], LOC3. It is a w1c and software should clear it whenever it reads the message from the MBOX[n], LOC3. This bit is also writable in ps_normal mode.
22 F3_READ	1 It indicates that the message that has just been read from PS_MRU_BUF2, will be stored in MBOX[n], LOC3 by the DMA. 0 No unread message in MBOX[n], LOC3. It is a w1c and software should clear it whenever it reads the message from the MBOX[n], LOC3. This bit is also writable in ps_normal mode.

Table 1581. PS_MBOX_SR_CHn field descriptions (continued)

Field	Description
23 F2_ERR	1 Indicates that the message that has just been read from the PS_MRU_BUF2 and is to be stored in (MBOX[n], LOC2) by the DMA, has one or more _ERR bits set. 0 Indicates that the message that has just been read from the PS_MRU_BUF2 and is to be stored in (MBOX[n], LOC2) by the DMA, has ALL the "_ERR" bits as "0". This bit is the "OR" of the contents of the PS_MBOX_SR_IRQ register. It is updated when the corresponding message (with the CID = n and FID=2) is read from the MRU_BUF2 by the DMA. It is a w1c and software should clear it whenever it reads the message from the MBOX[n], LOC2. This bit is also writable in ps_normal mode. **Note:** Also refer to *Figure 1584: IRQ[n] interrupts*, for bit fields details.
24 F2_OV	1 Indicates that there is an unread message stored in MBOX[n], LOC2 which will be overwritten with the new message, that has just been read from the PS_MRU_BUF2 by the DMA. 0 No overwrite of unread message in MBOX[n], LOC2. It is a w1c and software should clear it whenever it reads the message from the MBOX[n], LOC2. This bit is also writable in ps_normal mode.
25 F2_READ	1 It indicates that the message that has just been read from PS_MRU_BUF2, will be stored in MBOX[n], LOC2. 0 No unread message in MBOX[n], LOC2. It is a w1c and software should clear it whenever it reads the message from the MBOX[n], LOC2. This bit is also writable in ps_normal mode.
26 F1_ERR	1 Indicates that the message that has just been read from the PS_MRU_BUF2 and is to be stored in (MBOX[n], LOC1) by the DMA, has one or more _ERR bits set. 0 Indicates that the message that has just been read from the PS_MRU_BUF2 and is to be stored in (MBOX[n], LOC1) by the DMA, has ALL the "_ERR" bits as "0". This bit is the "OR" of the contents of the PS_MBOX_SR_IRQ register. It is updated when the corresponding message (with the CID = n and FID=1) is read from the MRU_BUF2 by the DMA. It is a w1c and software should clear it whenever it reads the message from the MBOX[n], LOC1. This bit is also writable in ps_normal mode. **Note:** Also refer to *Figure 1584: IRQ[n] interrupts*, for bit field details.
27 F1_OV	1 Indicates that there is an unread message stored in MBOX[n], LOC1 which will be overwritten with the new message, that has just been read from the PS_MRU_BUF2 by the DMA. 0 No overwrite of unread message in MBOX[n], LOC1. It is a w1c and software should clear it whenever it reads the message from the MBOX[n], LOC1. This bit is also writable in ps_normal mode.
28 F1_READ	1 It indicates that the message that has just been read from PS_MRU_BUF2, will be stored in MBOX[n], LOC1. 0 No unread message in MBOX[n], LOC1. It is a w1c and software should clear it whenever it reads the message from the MBOX[n], LOC1. This bit is also writable in ps_normal mode.

Table 1581. PS_MBOX_SR_CHn field descriptions (continued)

Field	Description
29 F0_ERR	1 Indicates that the message that has just been read from the PS_MRU_BUF2 and is to be stored in (MBOX[n], LOC0) by the DMA, has one or more _ERR bits set. 0 Indicates that the message that has just been read from the PS_MRU_BUF2 and is to be stored in (MBOX[n], LOC0) by the DMA, has ALL the "_ERR" bits as "0". This bit is the "OR" of the contents of the PS_MBOX_SR_IRQ register. It is updated when the corresponding message (with the CID = n and FID=0) is read from the MRU_BUF2 by the DMA. It is a w1c and software should clear it whenever it reads the message from the MBOX[n], LOC0. This bit is also writable in ps_normal mode. **Note:** Also refer to *Figure 1584: IRQ[n] interrupts*, for bit field details.
30 F0_OV	1 Indicates that there is an unread message stored in MBOX[n], LOC0 which will be overwritten with the new message, that has just been read from the PS_MRU_BUF2 by the DMA. 0 No overwrite of unread message in MBOX[n], LOC0. It is a w1c and software should clear it whenever it reads the message from the MBOX[n], LOC0. This bit is also writable in ps_normal mode.
31 F0_READ	1 It indicates that the message that has just been read from PS_MRU_BUF2, will be stored in MBOX[n], LOC0. 0 No unread message in MBOX[n], LOC0. It is a w1c and software should clear it whenever it reads the message from the MBOX[n], LOC0. This bit is also writable in ps_normal mode.

61.3.4.20 PSI5-S channel watchdog configuration register (PS_WD_CFGR_CHn)

Address: 0x01A0 + [n-1]*0x3C (n=1 to 7) Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	WDRST	WDCS	WDEN	0	WD_TO							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	WD_TO															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1563. PSI5-S channel watchdog configuration register (PS_WD_CFGR_CHn)

Table 1582. PS_WD_CFGR_CHn field descriptions

Field	Description
4 WDRST	Resets watchdog counter for the channel This bit will reset the watchdog counter and the counter will be reloaded with the value of WD_TO and automatically start counting again. **Note:** It is also writable in PS_NORMAL mode.
5 WDCS	This will select either GTM or the module clock for watchdog counter of the channel. See *Figure 1587: Clock distribution in PSI5-S*. 0 ipg_clk_ps_wd clock 1 gtm_trig is selected as the clock.
6 WDEN	The watchdog counter for each channel can be started, stopped. 1 watchdog counter started. 0 watchdog counter stopped. It maintains its value. **Note:** This bit is also writable in PS_NORMAL mode.
8:31 WD_TO	This indicates the watchdog time out value. The watchdog generates an overrun error, in case the watchdog is not refreshed before the watchdog counter reached the WD_TO value. The Watchdog is refreshed in a different manner depending on whether the PSI5-S channel is in asynchronous mode or synchronous mode as described below: In Asynchronous mode: As soon as the IP enters the PS_NORMAL mode the watchdog for each channel is loaded with the PSI5_WD_CFGR_CH[n][WD_TO] value, and in case the watchdog is enabled then the counter starts immediately. Whether the WD has to be “refreshed (re-loaded)” or “not to be refreshed” is analyzed when the XCRC is calculated. In case the XCRC is correct the watchdog is re-loaded (It is not STOPPED), else the WD counter keeps on running normally eventually leading to a watchdog overrun error. In case of watchdog overrun error the counter is refreshed. In Synchronous Mode: As soon as the IP enters the PS_NORMAL mode, the watchdog for each channel is loaded with the PSI5_WD_CFGR_CH[n][WD_TO] value and in case the watchdog is enabled then the counter starts immediately. Thereby it is re-loaded, either on the occurrence of a sync pulse or the watchdog overrun error. In case the event for this load is the Sync Pulse then it immediately starts counting again and keeps on counting till it is stopped when the “Packet_counter” reaches the count of “total number of configured slots”. The “Packet_counter” here is a counter that keeps track of the number of configured (enabled) slots. It is incremented on the reception of each UART packet with a “CID” and “FID” such that PS_MSGA_CHCID[FFID_byte] is not equal to “0”. The packet counter is however incremented ONLY if the XCRC of the received UART packet is correct else it is not incremented. In case the “Packet Counter” does not reach the “expected packet count” within the “WD timeout configured“, the WD overrun error is generated. In case of a WD overrun error, the counter is stopped and does not start till a Sync Pulse is encountered. In both the above modes, the watchdog can be always refreshed using a software reload. **Note:** The watchdog counter is enabled ONLY if the channel is enabled.

61.3.4.21 PSI5-S DDSR trigger offset register channel (PS_DDTRIG_OFFR_CHn)

Address: 0x01A4 + [n-1]*0x3C (n=1 to 7) Access: User read/write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	DDTRIG_OFFR															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1564. PSI5-S DDSR trigger offset register channel (PS_DDTRIG_OFFR_CHn)

Table 1583. PS_DDTRIG_OFFR_CHn field descriptions

Field	Description
16:31 DDTRIG_OFFR	These bits configure the delay between the “shift trigger” of the various DDSRs across different channels. The value in this register governs the delay after which the specific DDSR would be triggered once the Global Shift trigger (PS_GLCR[GL_DDSR_TRIG]) is written to “1” and the corresponding PS_E2SCR_CH[n][GL_TRIG_SEL] is set to “1” for that channel. There is an internal counter which is clocked on the “ipg_clk_ps_ddtrig” clock. The counters count up to the value as programmed in the PS_DDTRIG_PERR_CH[n] registers after which they are reset. At each reset a shift trigger is generated for the DDSR. The “start” of these counters is delayed based on the value programmed in the PS_DDTRIG_OFFR_CH[n] register.

61.3.4.22 PSI5-S DDSR trigger period register channel (PS_DDTRIG_PERR_CHn)

Address: 0x01A8 + [n-1]*0x3C (n=1 to 7) Access: User read/write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	DDTRIG_PERR															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1565. PSI5-S DDSR trigger period register channel (PS_DDTRIG_PERR_CHn)

Table 1584. PS_DDTRIG_PERR_CHn field descriptions

Field	Description
16:31 DDTRIG_PERR	These bits configure the “shift trigger” period for the DDSR of that specific channel. The value in this register governs the period with which the data for the DDSR of that particular channel would be shifted out. There is an internal counter which is clocked on the “ipg_clk_ps_ddtrig” clock. The counters count up to the value as programmed in the PS_DDTRIG_PERR_CH[n] registers after which they are reset. At each reset a shift trigger is generated for the DDSR. The “start” of these counters is delayed based on the value programmed in the PS_DDTRIG_OFFR_CH[n] register. **Note:** This register should be programmed to be greater than or equal to 2.

61.3.4.23 PSI5-S ECU to sensor control register (PS_E2SCR_CHn)

Address: 0x01AC + [n-1]*0x3C (n=1 to 7) Access: User read/write

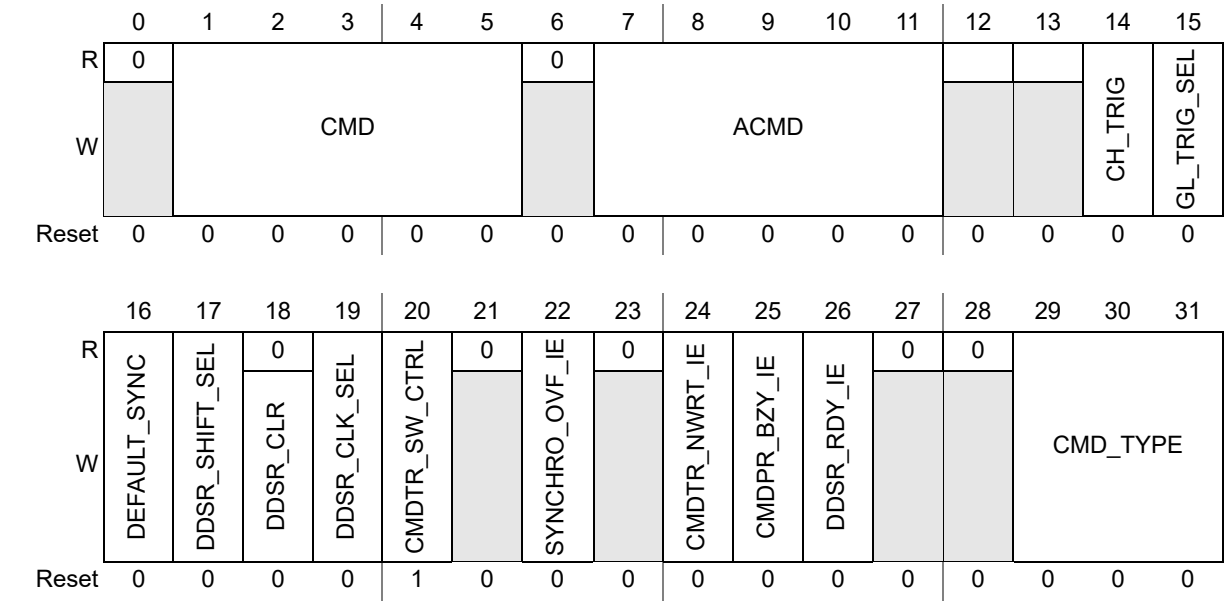


Figure 1566. PSI5-S ECU to sensor control register (PS_E2SCR_CHn)

Table 1585. PS_E2SCR_CHn field descriptions

Field	Description
1:5 CMD	Bits for the ECU-to-sensor “CMD” format. They refer to the command send to the UART Tx logic, when a “1” is shifted from the DDSR register.
7:11 ACMD	Bits for the ECU-to-sensor “ACMD” format. They refer to the command send to the UART Tx logic, when a “0” is shifted from the DDSR register.

Table 1585. PS_E2SCR_CHn field descriptions (continued)

Field	Description
14 CH_TRIG	This controls that the DDSR of the specific channel is triggered locally. This bit controls the shifting when PS_E2SCR_CHn[GL_TRIG_SEL] is 0. 1 Start the “shift trigger” of the DDSR. 0 Stop the DDSR from shifting the commands. The counters corresponding to DDTRIG_PERR and the DDTRIG_OFFR are reset. **Note:** This bit is also writable in PS_NORMAL mode.
15 GL_TRIG_SEL	This bit controls whether the DDSR shift trigger is a global trigger or a local channel trigger. 1 The DDSR shift trigger is a global trigger dependent on PS_GLCR[GL_DDSR_TRIG]. 0 The DDSR shift trigger is a local channel trigger dependent on the bit CH_TRIG.
16 DEFAULT_SYNC	Decides the value to which the DDSR should default to when PS_E2SSR_CHn[DDSR_RDY] == 1. When PS_E2SSR_CHn[DDSR_RDY] == 1 then the default shift (whether it is CMD or ACMD) is based on the value of this bit.
17 DDSR_SHIFT_SEL	DDSR Shift Selection This is used to select the shift trigger for the DDSR registers. 0 The shift trigger is from internal DDSR counters which are clocked from either the gtm_trig or the ipg_clk_ps_ddtrig. 1 The input signal gtm_trig directly acts as the shift trigger of the DDSR registers. See *Figure 1587: Clock distribution in PSI5-S* for more details.
18 DDSR_CLR	Used for rejecting the contents of the DDSR register. As soon as this bit is written to “1” the contents of DDSR are immediately rejected and defaulted to “0” after the completion of the current ACMD or the CMD transmission. PS_E2SSR_CHn[DDSR_RDY] goes to 1 and value of PS_E2SCR_CHn[DEFAULT_SYNC] starts getting shifted. This bit is one shot (W1S). **Note:** This bit is also writable in PS_NORMAL mode.
19 DDSR_CLK_SEL	Selects which clock clocks shift logic of the DDSR, for shifting the bits for command transmission. 0 ipg_clk_ps_ddtrig clock 1 Reserved
19 DDSR_CLK_SEL	Selects which clock clocks shift logic of the DDSR, for shifting the bits for command transmission. 0 ipg_clk_ps_ddtrig clock 1 gtm_trig is selected as the clock.

Table 1585. PS_E2SCR_CHn field descriptions (continued)

Field	Description
20 CMDTR_SW_CTRL	This bit decides that once the command is ready for transmission (indicated by the setting PS_E2SSR_CHn[CMDPR_BZY] to 1), then when should the actual transmission start: 0 The actual transmission of command remains suspended, even though the command is ready in the DDSR. 1 The command transmission starts once the PS_E2SSR_CHn[CMDPR_BZY] becomes 1, and the internal shift logic is available. **Note:** This bit is also writable in PS_NORMAL mode.
22 SYNCHRO_OVF_IE	Synchro overflow interrupt enable 1 Generate an interrupt when PS_E2SSR_CHn[SYNCHRO_OVF] is set to “1” 0 No interrupt is generated when PS_E2SSR_CHn[SYNCHRO_OVF] is set to “1”
24 CMDTR_NWRT_IE	Interrupt Enable for DDSR being attempted to be written when it is still not empty, that is PS_E2SSR_CHn[DDSR_RDY] == 0 yet an attempt to write to DDSR is made. 1 Enable the interrupt 0 Disable the interrupt **Note:** This bit is also writable in PS_NORMAL mode.
25 CMDPR_BZY_IE	Interrupt Enable for DDSR getting updated with the internally processed complete command (with CRC, stuff bits, start bits getting appended) 1 Enable the interrupt 0 Disable the interrupt **Note:** This bit is also writable in PS_NORMAL mode.

Table 1585. PS_E2SCR_CHn field descriptions (continued)

<table>
<tr><th>Field</th><th>Description</th></tr>
<tr><td>26
DDSR_RDY_IE</td><td>Interrupt Enable for DDSR getting empty and ready for receiving new command.
1 Enable the interrupt
0 Disable the interrupt
Note: This bit is also writable in PS_NORMAL mode.</td></tr>
<tr><td>29:31
CMD_TYPE</td><td>
<table>
<tr><th>CMD_TYPE</th><th>PSI5 Frame</th><th>Number of bits writable by the Application in the DDSR</th></tr>
<tr><td>“000” - 0</td><td>Short Frame with 31 “1s” as the start condition*</td><td>6 LSBs of DDSR_L</td></tr>
<tr><td>“001” - 1</td><td>Short Frame with 5 “0s” as the start condition*</td><td>6 LSBs of DDSR_L</td></tr>
<tr><td>“010” - 2</td><td>Long Frame with 31 “1s” as the start condition*</td><td>16 LSBs of DDSR_L</td></tr>
<tr><td>“011” - 3</td><td>Long Frame with 5 “0s” as the start condition*</td><td>16 LSBs of DDSR_L</td></tr>
<tr><td>“100” - 4</td><td>X-Long Frame with 31 “1s” as the start condition*</td><td>22 LSBs of DDSR_L</td></tr>
<tr><td>“101” - 5</td><td>X-Long Frame with 5 “0s” as the start condition*.</td><td>22 LSBs of DDSR_L</td></tr>
<tr><td>“110” - 6</td><td>XX-Long</td><td>24 LSBs of DDSR_L</td></tr>
<tr><td>“111” - 7</td><td>Illegal</td><td></td></tr>
</table>
* The start condition (31 “1s” or 5 “0s”) is automatically taken care by the hardware and should NOT be written to the registers as part of the command; neither is this visible in any register.
</td></tr>
</table>

61.3.4.24 PSI5-S ECU to sensor status register (PS_E2SSR_CHn)

Address: 0x01B0 + [n-1]*0x3C (n=1 to 7) Access: User read/write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	SYNCHRO_OVF	0	CMDTR_NWRT	CMDPR_BZY	DDSR_RDY	0	0	0	0	0
W							w1c		w1c	w1c	w1c					
Reset	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0

Figure 1567. PSI5-S ECU to sensor status register (PS_E2SSR_CHn)

Table 1586. PS_E2SSR_CHn field descriptions

Field	Description
22 SYNCHRO_OVF	This bit gets set whenever due to any reason (like too less DDTRIG period), sync pulse of a channel comes before the previous sync pulse of the same channel has actually shifted its data. It can be used to detect sync pulse overflow. If PS_E2SCR_CHn[SYNCHRO_OVF_IE] bit is set to “1” then this can also be used to generate an interrupt on the ipi_ps_e2s_ch[n] line where “n” is the specific channel number. 1 Synch Pulse overflow has occurred for the specific channel. 0 No synch pulse overflow has occurred in the specific channel.
24 CMDTR_NWRT	Status of Command written to the DDSR: 1 Command written to the DDSR is rejected as an attempt to write to it has been made when DDSR_RDY == 0. 0 Command written to the DDSR has been accepted for processing.

Table 1586. PS_E2SSR_CHn field descriptions (continued)

Field	Description
25 CMDPR_BZY	Command Processing Busy Status This bit along with the DDSR_RDY bit can be used to check the status of the command that is being processed in the DDSR. 1 DDSR is ready with the processed command complete with CRC/start bits/stuff bits.The transmission of the same would start if PS_E2SCR_CHn[CMDTR_SW_CTRL] == 1. For this status DDSR_RDY should also be read as “0”. 0 DDSR is busy with command processing. For this status DDSR_RDY should also be read as “0”. Note that when DDSR_RDY goes to “1” then this bit goes to “0”. So it represents the above status conditions ONLY when DDSR_RDY is “0”.
26 DDSR_RDY	Status for DDSR getting empty and ready for receiving new command. 1 DDSR is empty and ready for receiving a new command. 0 DDSR is not ready and is busy with either the command processing or the command transmission.

61.3.4.25 PSI5-S channel1 ECU to sensor downstream data shift register high (PS_DDSR_H_CHn)

Address: 0x01B4 + [n-1]*0x3C (n=1 to 7) Access: User read/write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	DDSR_H										
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1568. PSI5-S channel1 ECU to sensor downstream data shift register high (PS_DDSR_H_CHn)

Table 1587. PS_DDSR_H_CHn field descriptions

Field	Description
21:31 DDSR_H	Upper High bits of the 43 bit complete DDSR register. These bits are updated by the hardware and software cannot write to these bits. They can be read by the software to check for the processed command after the command is written to the DDSR by the application, the DDSR_RDY is cleared and the PS_E2SSR_CHn[CMDPR_BZY] goes to “1”.

61.3.4.26 PSI5-S channel1 ECU to sensor downstream data shift register low (PS_DDSR_L_CHn)

Address: 0x01B8 + [n-1]*0x3C (n=1 to 7) Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	DDSR_L1								DDSR_L0							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	DDSR_L0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1569. PSI5-S channel1 ECU to sensor downstream data shift register low (PS_DDSR_L_CHn)

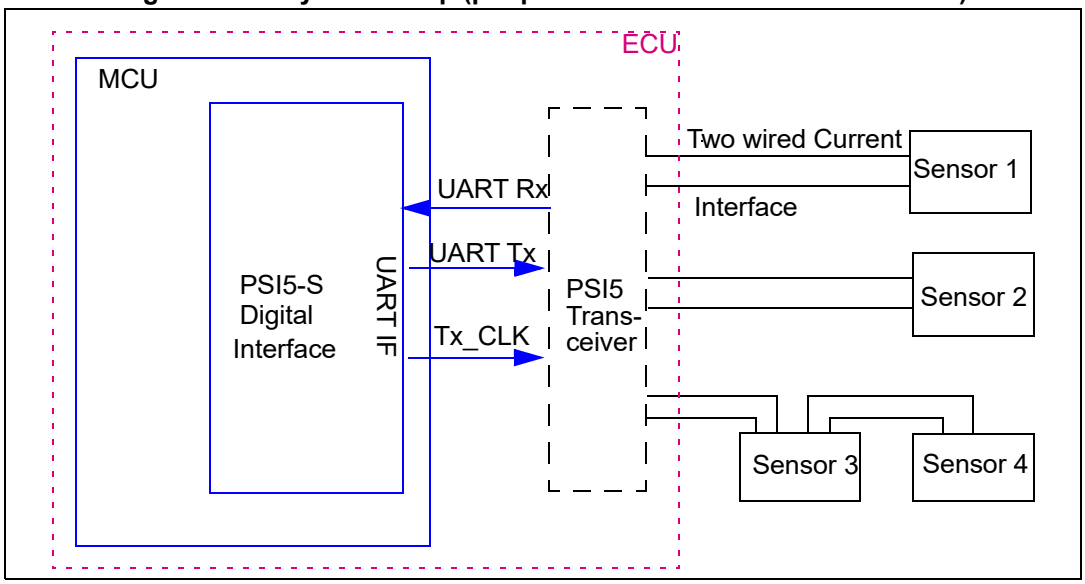
Table 1588. PS_DDSR_L_CHn field descriptions

Field	Description
0:7 DDSR_L1	8 bits of the DDSR_L that can only be read by the Application. These 8 bits are automatically updated by the hardware. They can be read by the software to check for the processed command after the command is written to the DDSR by the application, the DDSR_RDY is cleared and the PS_E2SSR_CHn[CMDPR_BZY] goes to “1”.
8:31 DDSR_L0	Lower 24 bits of the DDSR register that are written by the Application as well as updated by the hardware once the command processing is over. Though all the 24 bits can be written by the software, yet only those number of bits as governed by PS_E2SCR_CHn[CMD_TYPE] are actually taken for processing. This register can be read by the software to check for the processed command, after the command is written to the DDSR by the application, the DDSR_RDY is cleared and the PS_E2SSR_CHn[CMDPR_BZY] goes to “1”. **Note:** These bits are also writable in the PS_NORMAL mode.

61.4 Functional description

The PSI5 is a communication protocol for peripheral inertial, pressure, temperature, and position sensors. These sensors are connected to the ECU by just two wires, using the same lines for power supply and data transmission. The UART compatible transceiver (present externally) provides a pre-regulated voltage to the sensors and reads in the transmitted sensor data. *Figure 1570* shows a point-to-point connection for sensors 1 and 2 and bus configuration for sensors 3 and 4.

Figure 1570. System setup (peripheral sensor connected to device)



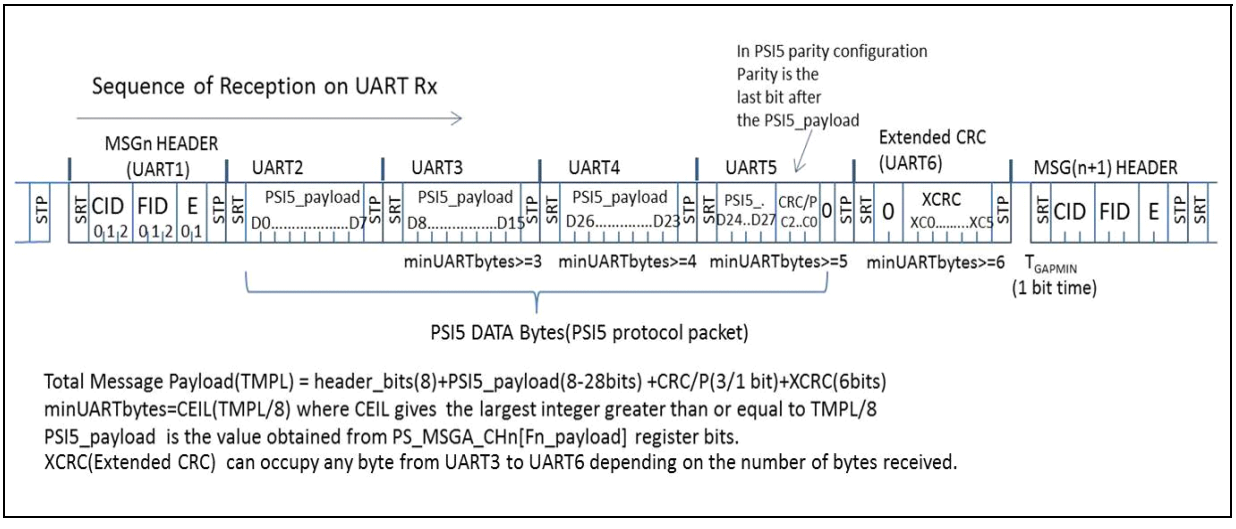
ECU (Electronic Control Unit) is comprised of PSI5-S digital interface and the UART compatible PSI5 transceiver (present externally). The PSI5-S digital interface communicates with the PSI5 Transceiver though the UART interface as shown in *Figure 1570*. When conducting the Sensor To ECU communication, the external transceiver converts the sensor data to the UART packet format before sending it to the PSI5-S module. The IP extracts the PSI5 Frames from the input UART Packet.

61.4.1 PSI5-S UART message structure

The transceiver sends a UART message packet consisting of between three to six UART bytes which contains information to allow rebuilding of the PSI5 message as well as the message itself and has a CRC which protects the complete packet (XCRC).

Each byte is made up of one start bit, eight data bits and one stop bit - no parity is included in the UART byte. *Figure 1571* shows the PSI5-S UART packet structure.

Figure 1571. PSI5-S packet structure



The UART packet is made up of the following:

- MSGnHEADER: This is the message header byte which includes the indication of which channel the PSI5 message was received on (3-bit CID), the slot on that channel in which the message was received (3-bit FID) and transceiver error indication bits (2-bit E).
- PSI5 DATA bytes - minimum one, maximum four, depending on the data payload region of the PSI5 message (8 - 28 bits) and whether parity (1-bit) or CRC (3-bit) is configured for the PSI5 message. In the above example shown in *Figure 1571* it is taken as 28 bits and 4 UART bytes are shown (UART2-UART5). Messages having CID and FID both as “0s” do not have the CRC/Parity field in them. Such messages only have the message header, payload and the XCRC fields.
- XCRC- 6-bits - This is the extended CRC which protects the UART packet. Note that any bits between the end of the PSI5 message payload and the XCRC are ‘stuffed’ with ‘0’s. This can also be seen in *Figure 1571*. For details about XCRC handling, refer to *Section 61.4.2.5*.
- Example to calculate the minUARTbytes (for messages NOT having CID and FID both as 0):
  - Fn_payload = 14
    with PSI5 CRC option: TMPL = 8+14+3+6=31, minUARTbytes=CEIL(3.8)=4
    These input data stream is interpreted as
    UART1(header), UART2(D[0:7]), UART3(D[8:13], CRC[2:1]),
    UART4(CRC[0],0,XCRC[0:5])
  - Fn-payload = 13
    with PSI5 Parity option: TMPL = 8+13+1+6=28, minUARTbytes=CEIL(3.5)=4
    These input data stream is interpreted as
    UART1(header), UART2(D[0:7]), UART3(D[8:12],Paritybit,00),
    UART4(00,XCRC[0:5]).

Note: The “0” stuffing between the parity bit and the XCRC. The XCRC is always in the last 6 bits of the last byte.

- Example to calculate the minUARTbytes (for messages having CID and FID both as 0):
  - F0_payload = 8
    TMPL = 8+8+0+6=22, minUARTbytes=CEIL(2.75)=3
    These input data stream is interpreted as
    UART1(header), UART2(D[0:7]), UART3(00,XCRC[0:5]).
  - F0_payload = 10
    TMPL = 8+10+0+6=24, minUARTbytes=CEIL(3)=3
    These input data stream is interpreted as
    UART1(header), UART2(D[0:7]), UART3(D[8:9],XCRC[0:5])

61.4.2 PSI5 received message handling

The PSI5 message (UART Type) received from sensors (through the external UART Transceiver) is handled internally to the PSI5-S IP by three separated parts:

- PSI5 message extraction:

  This decides when a UART packet has been received with the correct header byte, and then how to extract the information from this UART packet form a complete PSI5 message.
- PSI5 “recoverable” and “unrecoverable messages”:

  This differentiates the PSI5 messages received above, into “recoverable” and “unrecoverable” messages.
- PSI5 message storage and Mailbox transfer:

  This ensures the transfer of the PSI5 message to the correct location in the system RAM, once the PSI5 message has been correctly differentiated into “recoverable” and “unrecoverable” messages.

Following sections detail the above 3 parts.

61.4.2.1 PSI5 message extraction

Once a UART start bit is detected after the idle timeout (the idle time between message packets on the UART interface, that is the number of bit times between message packets in the range of at least 1 to 16, configurable in the UART module through the PS_UARTPTO register) has occurred, the UART bytes are subsequently sampled. They are being received by an internal state machine in the MRU, which checks for the formation of a complete UART packet, its coherence and extraction of the PSI5 message from the UART packet. Also, refer to *Figure 1572* for further understanding of this section.Each UART packet that is received can be considered as “recoverable” or “unrecoverable”. Demarcation between the two is explained subsequently. Following are the sequence of steps performed as each UART byte is being received:

1. There is a 6 byte UART message input buffer into which the UART bytes from the UART Rx register are written as they are received. A 3-bit UART message counter is incremented on the reception of each byte in UART message packet that is each UART byte received between idle time counter indications.
2. All the subsequent UART bytes in the UART packet are stored internally till any of the following events occurs:
   - UART Timeout that is bus idle time is detected.
   - Framing Error from the UART.
   - Number of UART bytes that are received as reported through the UART message counter described in *1.* above exceed 6.
   - STOP mode request mode is asserted. Refer to *Section 61.1.4.1*, for further details.
3. As soon as any of the events described in *2.* above occur, it is assumed that a complete UART packet has been received. The XCRC is then calculated immediately on the received UART packet. Refer to *Section 61.4.2.5*, for details about XCRC. If there is an XCRC error the message is assumed to be “unrecoverable” and it is sent to

$MBOX_0$, LOC_1.The timestamp of the message is captured as is described in *Section 61.4.4*.The watchdog error is captured as is described in *Section 61.4.5*.

4. If there is NO XCRC error, then the CID and FID are extracted from the header byte, and the configured length of the message is determined from the configuration in PSI5 Channel*n* Message Configuration Register A. Following steps are performed.
   - The FID and the CID from the header byte is transferred directly to the PSI5 Frame status word.
   - The transceiver error flags (E0, E1) are transferred directly to the PSI5 Frame status word.
   - The message is then analyzed to see whether it is “LEGAL” or “ILLEGAL.
   - Following are the conditions for a “LEGAL” message:
     When the CID and the FID of the received message are both “0s” and PS_MSGA_CH0[F0_byte] is NOT “0” OR
     When the data of an Asynchronous Enabled channel is received with FID = 0 and $PS_MSGA_CH_{CID}$[F0_byte] is NOT “0” OR
     When the data of a Synchronous Enabled channel is received with legal FIDs (0 to 5) and if the received FID is such that the corresponding $PS_MSGA_CH_{CID}[F_{FID}_byte]$ is NOT 0.
   - Any received message that does not satisfy the above condition is considered as “ILLEGAL”. Illegal Messages are also treated as “unrecoverable” messages and are sent to $MBOX_0$, LOC_1.
     Example can be a message with CID as 0 but FID as 1(illegal because Channel0 is asynchronous and can only have FID equal to 0 messages) or a message received for an enabled synchronous channel but having an FID equal to 7 (illegal because FID as 7 is not a legal Frame ID).
   - The timestamp of a Legal/Illegal message is captured as is described in *Section 61.4.4*. The watchdog error is captured as is described in *Section 61.4.5*.
   - The number of UART bytes received in the UART message packet is compared with the number specified in Message Configuration Register A for the channel indicated by CID and FID in the header byte. If the number of bytes indicated by the message counter does not match $PS_MSGA_CH_{CID}[F_{FID}_byte]$ field, then the status bit N_Err within the PSI5 Frame status word is set. The above “number of bytes received” check is not carried out for an “illegal message”.
5. The PSI5 message payload is checked using the PSI5 CRC or parity as per the configuration. Whether the CRC or parity check is correct or not, the PSI5 message payload is written to the PSI5 Sensor Data word and the CRC or parity is written to the PSI5 Frame Status word. Note that the messages which have CID and FID both as “0”, are not checked for CRC or parity error. The CRC/Parity error status bits remain “0” for such messages.This above check is NOT done for “illegal messages”.

61.4.2.2 PSI5 “recoverable and unrecoverable” messages

Based on *Section 61.4.2.1: PSI5 message extraction*, a message is considered “unrecoverable”, if the XCRC is incorrect or it is an “illegal message”, else the message is considered “recoverable”. “Unrecoverable” message indicates that an error has been detected during the assembling phase of the UART packet that is during the formation of the UART packet from the individual UART bytes. “Unrecoverable” messages are always stored in PSI5 mailbox $MBOX_0$, LOC_1 in system RAM. “Recoverable” messages are always stored in the Mailbox location pointed out by the CIDn and the corresponding FID fields.

Figure 1572 describes the treatment of the “Recoverable” and “Unrecoverable” messages. It also details the setting of the various error flags. The various cases for a new PSI5 message reception are:

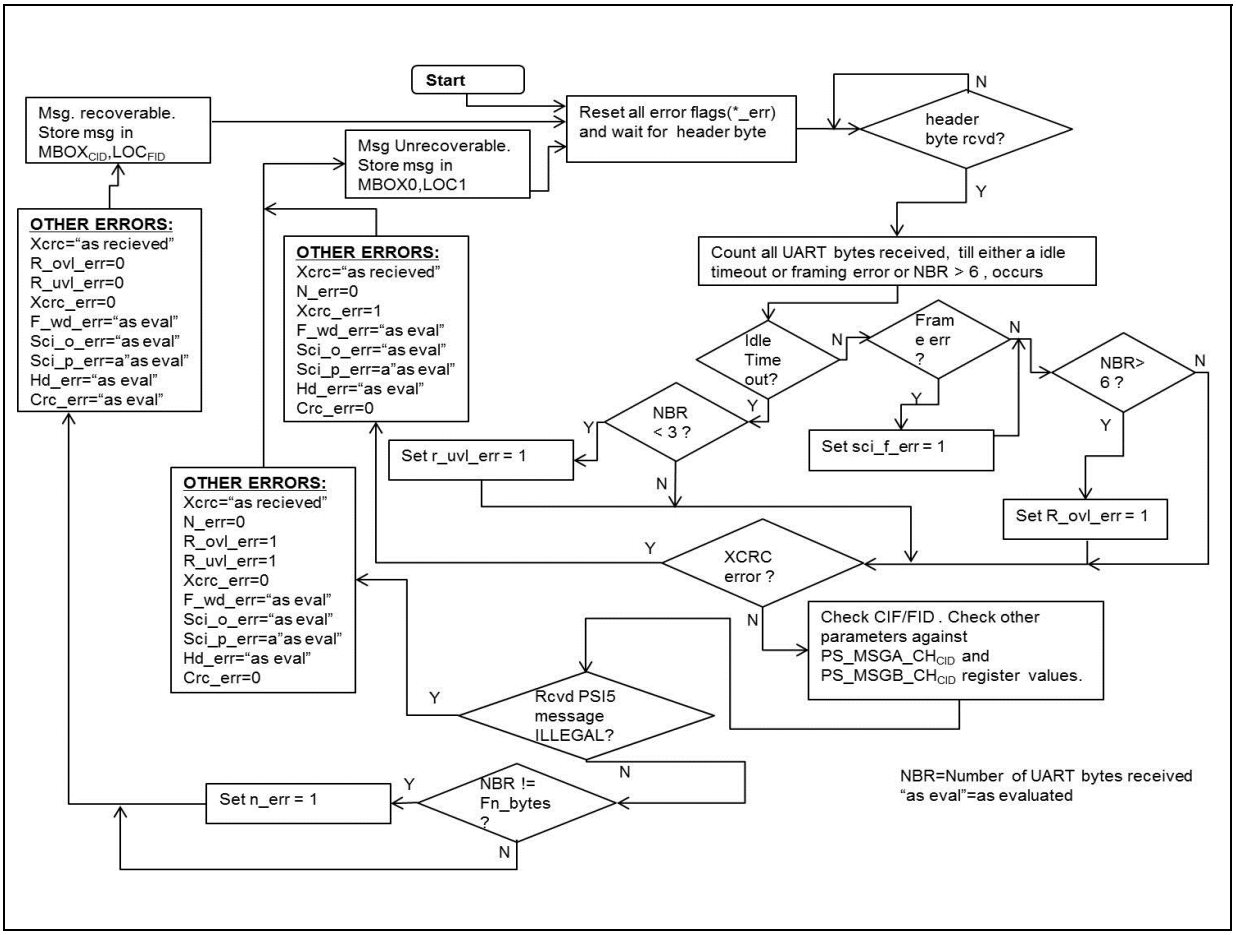
- Idle Timeout occurrence
- Framing Error Occurrence

  As soon as a framing error has occurred, all the UART packets sampled till that instance are assembled to form a PSI5 packet and are transferred to the MRU buffer. The state machine starts sampling the next UART byte only after the occurrence of an idle Time Out from the UART or after a reset or if the mode of the IP is changed (PS_NORMAL to PS_CONFIG/PS_DISABLE/UART_STANDALONE and then back to PS_NORMAL) or if a stop mode is entered and then exited.
- Number of UART bytes received exceed 6

  In this case any UART bytes received after the 6th byte, are totally rejected by the internal PSI5-S state machine. The state machine starts sampling the next UART byte only after the occurrence of an idle Time Out from the UART or after a reset or if the mode of the IP is changed (PS_NORMAL to PS_CONFIG/PS_DISABLE/UART_STANDALONE and then back to PS_NORMAL) or if a stop mode is entered and then exited.
- STOP mode entry

  Refer to *Section 61.1.4.1*. Stop mode entry causes the flushing of the internal PSI5 message buffer and this event is treated like new message completion. Exiting from the stop mode resets the state machine and it becomes ready to respond to the next UART packet. The next UART packet is taken as the header byte of a new PSI5 message.The stop mode, as an event for a new PSI5 message formation is not shown in *Figure 1572*.

Figure 1572. PSI5-S message handling and error flags



Once the message has been correctly deciphered as “Recoverable” or “Unrecoverable” and the various error flags have been set then the Message is finally transferred to the MRU buf2. Once the corresponding message has been read from the MRU buf2 the bits in the “PS_MBOX_SR_CH[n]” register are set appropriately. This aids the application to correctly read the concerned message from the Mailbox. *Section 61.4.2.3* describes the PSI5 Message Storage and detection in detail.

61.4.2.3 PSI5 message storage and mailbox transfer

Once the message has been correctly deciphered as “recoverable” or “unrecoverable” and the various status bits are set, it is then transferred to the MRU output buffer. The MRU output buffer is a 2 stage buffer differentiated as MRU Buf1 and MRU Buf2. The MRU buf1 is composed of 3, 32 bit registers while the MRU buf2 is composed of 4, 32bit registers. *Figure 1573* shows the structure of MRU buf1. It is composed of three internal registers. These registers are not in the addressable space of the IP. When the internal state machine has correctly decoded the PSI5 message, it is transferred to MRU buffer1. It serves to hold the PSI5 message until it is transferred to MRU buf2.

Figure 1573. MRU buf1 structure

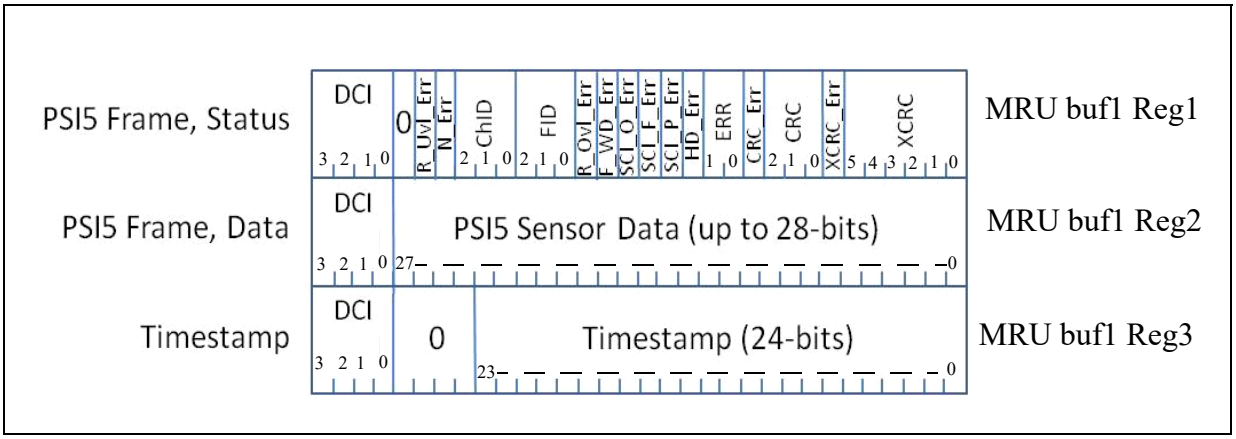
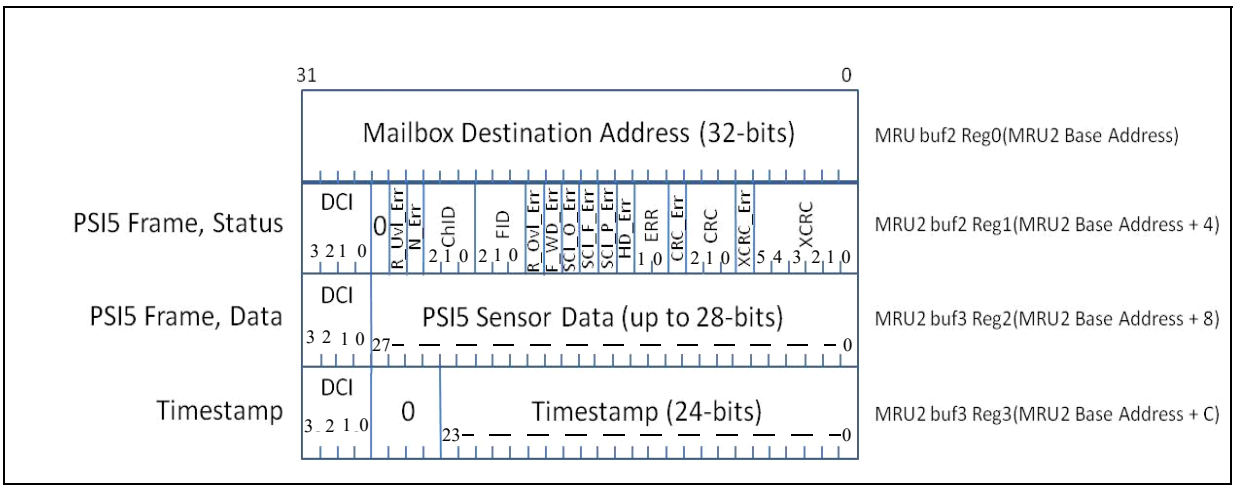


Figure 1574 shows the structure of MRU buf2. MRU buf2 has one extra location as compared to MRU buf1. This extra location (MRU buf2 Reg0) contains the address of the Mailbox location, where the message contained in the rest of the three locations of MRU buffer2 is to be placed. This specific MRU buf2 structure aids the DMA to perform linked scatter-gather data transfer from the IP. The extra location in the MRU buf2 is automatically updated by the hardware and provides the destination address for the message. This destination is the Mailbox Address at which the message (stored in MRU buf2 (Reg1-Reg3) is to be transferred by the DMA. This destination address is calculated by the IP based on ChID in the message header (or ChID0 in the case of an unrecoverable message), the FID in the message header and Base address of the mailbox in system RAM (provided by CH_BASE_ADDR) register.

Figure 1574. MRU buf2 structure



MRU buf2 is in the addressable space of the IP and is read by the DMA when the corresponding DMA request is asserted. The PSI5 message is transferred from the MRU buf1 to the MRU buf2 only when the MRU buf2 has been read completely by the DMA, that is all the 4 locations of the MRU buffer2 have been read by the DMA, for the corresponding PSI5 message.

Following are the sequence of operations that are performed, before the message in MRU buf2, gets ready to be accessed by the DMA:

1. The IP decodes the message as detailed in *Section 61.4.2.1: PSI5 message extraction* and *Section 61.4.2.2: PSI5 "recoverable and unrecoverable" messages*.
2. The corresponding message is available in MRU buf1 along with the various status bits latched internally in the IP. Externally, no interrupt is generated and neither are any status bit updated, when the message is transferred to MRU buf1.When the message is transferred to MRU buf1 a 3 bit DCI (data consistency indicator is added). The DCI is a 3-bit counter that is counted on a reception of each new message in MRU buf1.
3. The IP checks whether MRU buf2 is empty. If it is empty, then the MRU buf1 "Reg1 to Reg3" contents are transferred to the MRU buf2 "Reg1 to Reg3". The corresponding interrupts (except the Message Read from MRU) are also asserted based on the Interrupt Enable settings. Based on the PSI_CH_BASE_ADDR, the CID, FID values and whether an "unrecoverable message" has been received, the IP calculates the contents of MRU buf2 Reg0 register.
   - If CID == 0 then the "MRU buf2 Reg0" contains the Mailbox Location corresponding to MBOX0, LOC0.
   - If "unrecoverable message" is received then "MRU buf2 Reg0" contains the Mailbox Location corresponding to MBOX0, LOC1.
   - In all other cases "MRU buf2 Reg0" contains the Mailbox location corresponding to the $MBOX_{CID}$, LOC_{FID}.
4. Once the calculations in "*3.*" are done, the IP asserts the DMA request for Access 1. In DMA Access 1, the DMA reads the first location (MRU buf2 Reg0) of the MRU buf2. This single 32-bit access writes the content of MRU buf2 Reg0 into the destination address for the DMA channel performing the second access (Access 2). Access 2 is initiated by the DMA once the Access1 transfer is complete. Access 2 is composed of 3, 32-bit accesses of MRU buf2 and it writes the reconstructed PSI5 message along with its status and timestamp to the mailbox, starting at the destination address provided by Access 1.In Access2 the DMA reads the MRU buf2 Reg1, Reg2 and Reg3 in that order.
5. The application can read the PS_MBOX_SR_CH[n] register to get an indication to which mailbox location (or locations) the DMA would transfer the message. Note that "DMA transfer complete" bit/interrupt of the DMA indicates to the application, that the message read by the DMA from the IP has actually been written to the Mailbox.
6. The application should clear the corresponding "Fn_read" bits in the PS_MBOX_SR_CH[n] register once the application has read the corresponding message from the Mailbox. If a PSI5 message gets written to a Mailbox location, with "Fn_read" bit still set at the time at which the DMA has started the Acces1, then the corresponding "Fn_OV" bits are set in the PS_MBOX_SR_CH[n] register.
7. Reference to "*3.*" above, if the MRU buf2 has a pending message and the MRU buf1 also has a message that is ready to be transferred to the MRU buf2 and then a new message gets ready to be written to MRU buf1, then the previous message in MRU buf1 gets overwritten by the new message. The MRU overwrite interrupt is also generated with the setting of the corresponding register bit in the PS_GLSR register. PS_GLSR register contains the CID and the FID of the message that got overwritten by the new message in MRU buf1.
8. When writing an unrecoverable message to the PSI5 mailbox CID0, with the XCRC_ERR bit set, the timestamp that will be used will be the one which is captured at the time XCRC_ERR bit is set. When the unrecoverable message is due to an "illegal message", then the timestamp used would be the one which is captured on the

reception of any message that is always captured at the end of the header byte of every message.

9. In Asynchronous mode, FID will always be configured to be “0” and the module will use the PSI5 Mailbox for the channel (identified by CID in header) as a ring buffer, incrementing the destination address for each message received and saving up to the last six messages received. The bits of the PS_MBOX_SR_CH[n] register will still indicate the status of each location of the Ring Buffer. The software has to still clear the six bits of this register when it reads the Mailbox location else the overwrite bits would be set in the PS_MBOX_SR_CH[n] register. The message identification mechanism allows software to identify the latest message received.

61.4.2.4 PSI5-S data payload CRC recalculation

Error detection is realized by a single bit even parity (recommended for 10 or fewer bits) or a 3-bit CRC (recommended for long data words). The transmission error detection is selectable (by frame configuration register) as:

- 1-bit even parity
- 3-bit CRC

The generator polynomial of the CRC is $g(x)=1+x+x^3$ with a binary CRC initialization value of 111. The transmitter extends the data bits by three 0s (as MSBs). This augmented data word is fed (LSB first) into the shift registers of the CRC check. Start bits are ignored in this check. When the last zero of the augmentation is calculated on the input adder and the result is shifted, the buffer registers contain the CRC checksum. These three check bits are transmitted in reverse order (MSB first: C2, C1, C0).

Figure 1575. CRC calculation scheme

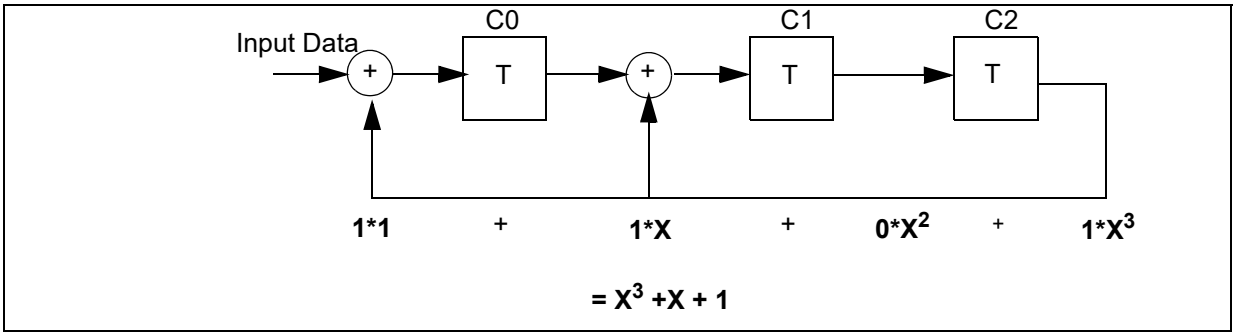


Table 1589. CRC Calculation (at transmitter side)

Sensor Data (LSB first)		0	1	1	1	0	1	1	1	0	0	0	0	0
C0	1	1	0	1	1	0	0	0	0	1	1	1	0	0
C1	1	0	0	0	1	1	1	1	1	1	0	0	1	0
C2	1	1	0	0	0	1	1	1	1	1	1	0	0	1

Table 1590. Transmitted and Received Sensor Data

Transmitted Sensor Data	S0	S1	D0	D1	D2	D3	D4	D5	D6	D7	D8	D9	C2	C1	C0
	0	0	0	1	1	1	0	1	1	1	0	0	1	0	0

Table 1590. Transmitted and Received Sensor Data (continued)

Received Sensor Data	S0	S1	D0	D1	D2	D3	D4	D5	D6	D7	D8	D9	C2	C1	C0
	0	0	0	1	1	1	0	1	1	1	0	0	1	0	0

Table 1591. CRC Re-Calculation (At Receiver Side)

Sensor Data (LSB first)		0	1	1	1	0	1	1	1	0	0	1	0	0
C0	1	1	0	1	1	0	0	0	0	1	1	0	0	0
C1	1	0	0	0	1	1	1	1	1	1	0	0	0	0
C2	1	1	0	0	0	1	1	1	1	1	1	0	0	0

Syndrome = 000 → Transmission successful

The correct transmission is checked by the PSI5 receiver by comparing the syndrome (content of the shift register after reception of both data bits and CRC bits) with the zero vector. For error free reception, all shift registers have to be zero.

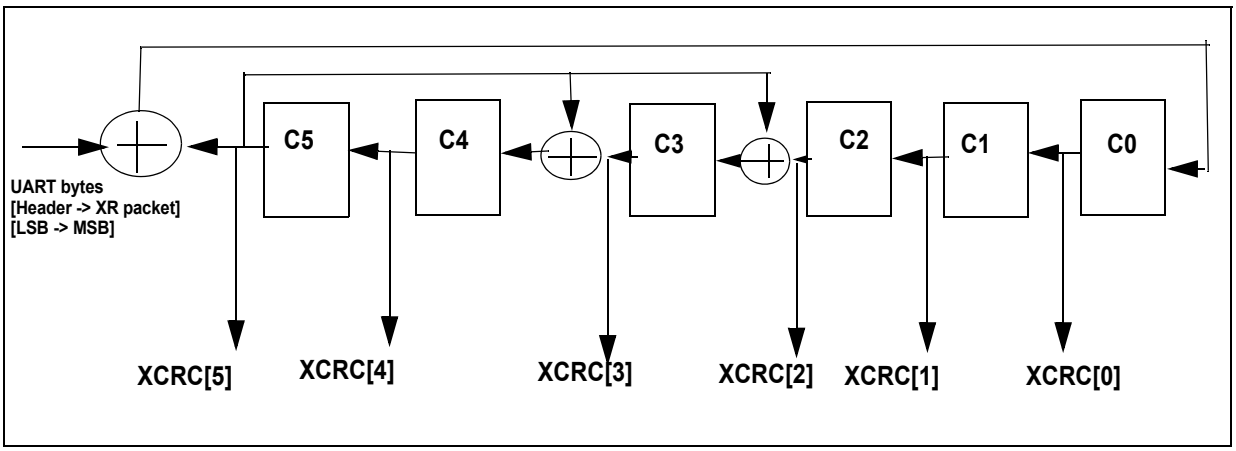
61.4.2.5 PSI5-S XCRC handling

The XCRC calculation is triggered on any of the following events (which also indicate the reception of a new UART packet):

- UART idle timeout
- number of UART bytes exceed 6
- UART Framing error
- STOP mode assertion (refer to *Section 61.1.4.1*)

When any of the above events occur, the internal XCRC calculator becomes active. Each UART byte that is received, passes through the XCRC calculator on a bit by bit basis, starting from the header byte to the expected XCRC byte. The XCRC is assumed to be the last 6 bits of the last UART byte that is received before the occurence of any of the above events. *Figure 1576* shows the XCRC calculation algorithm.

Figure 1576. XCRC calculation



The UART bytes are shifted starting from the received header to data packets to the XCRC packet. Each of these UART bytes is shifted LSB first into this XCRC calculator. Once all the

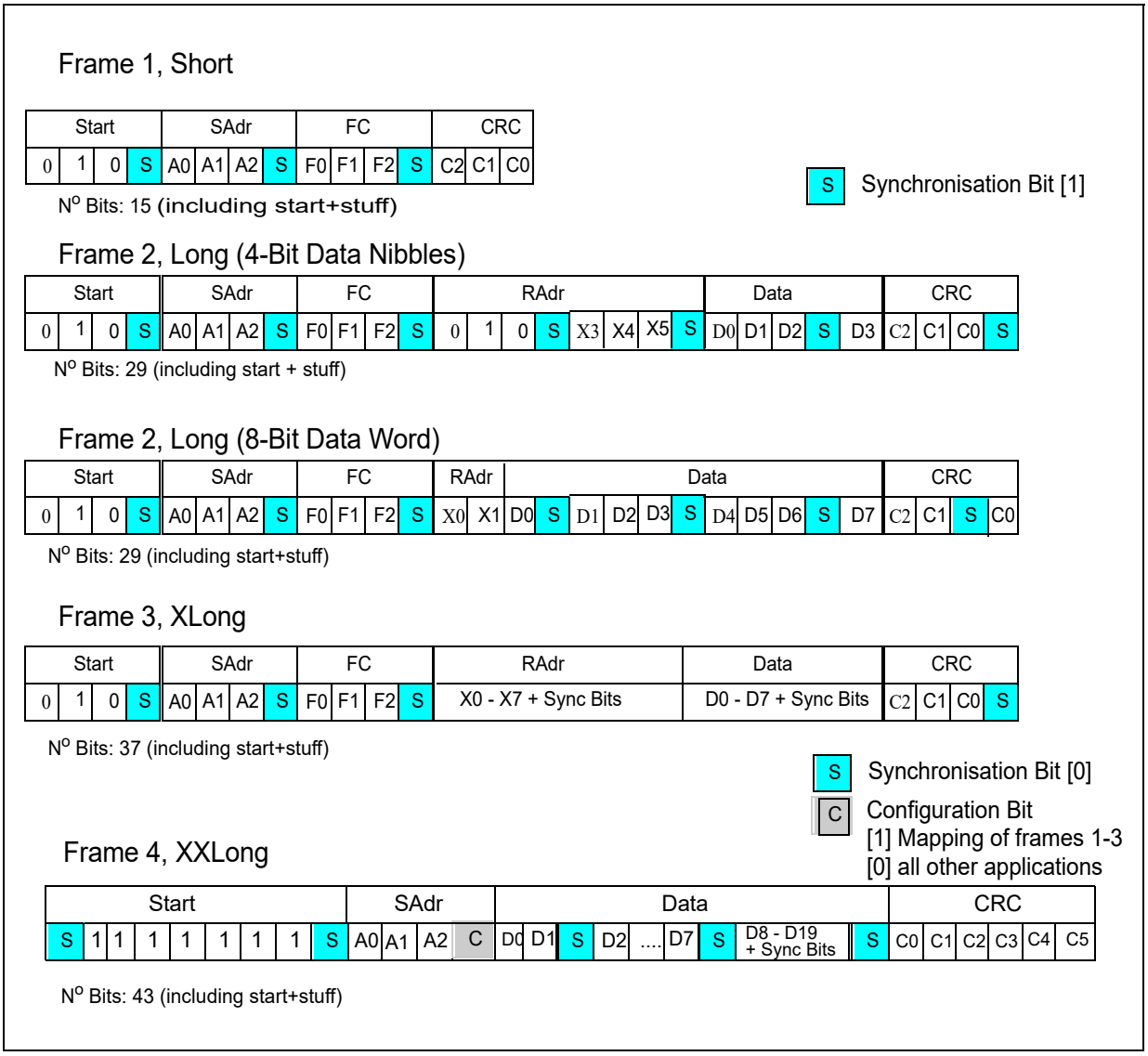
bytes as denoted by the Fn bytes have been received and the Fnth packet (XCRC) has been shifted completely through this XCRC calculator and if XCRC[5:0] is equal to “0” then it denotes that the XCRC check has passed else there is an XCRC error and the XCRC_ERR bit is set.

The CRC polynomial used is “x6+x4+x3+1” with a binary initialization value of XCRC[5:0] = “010101”.

61.4.3 ECU-to-sensor communication

The IP send the commands for the ECU to Sensor communication in terms of UART bytes. It expects the external transceiver to correctly interpret the commands and then generate the appropriate logic 0 or logic 1 on the PSI5 bus. Following are the various PSI5-S ECU to Sensor commands supported by the IP, as shown in *Figure 1577*.

Figure 1577. Standard frame formats supported



- Short: It consists of the following data stream (Order of transmission from IP is Left to Right. Thus C0 comes out Last) -- “010”, S, “A0,A1,A2”, S, “F0,F1,F2”, S, “C2,C1,C0” where “010” is the start sequence “S” is the stuff bit(1), A0-A2 is the sensor address, F0-F2 are the functional codes and “C2-C0” is the CRC. Refer to *Figure 1577* for the frame structure.
  - The CRC polynomial used is $x3 + x + 1$ with a binary initialization value of “111(C2:C0)”.
  - The transmitter extends the data bits by three zeros (as MSBs).
  - The CRC calculator is fed LSB first. Thus for CRC calculation the sequence given is [A0...A2, F0...F2, ”000”] with “A0” being given first and “0” as the last. When the last zero of the augmentation is calculated on the input adder and the result is shifted, the CRC calculation has been achieved.
  - When mapping this sequence to the DDSR register bits for calculating the CRC; then it becomes (PS_DDSR[0], PS_DDSR[1]...PS_DDSR[5], 000) with PS_DDSR[0] being fed first in the CRC calculator.
- Long (4bit Data Nibble): It consists of the following data stream (Order of transmission from IP is Left to Right. Thus C0 comes out Last) -- “010”, S, “A0,A1,A2”, S, “F0,F1,F2”, S, “X0,X1,X2”, S, “X3,X4,X5, S, “D0,D1,D2”, S,“D3,C2,C1”, S, “C0” where “010” is the start sequence “S” is the stuff bit(1), A0-A2 is the sensor address, F0-F2 are the functional codes, “X0-X5” is the Range Address, “D0-D3” is the data and “C2-C0” is the CRC. Refer to *Figure 1577* for the frame structure.
  - The CRC polynomial used is $x3 + x + 1$ with a binary initialization value of “111(C2:C0)”.
  - The transmitter extends the data bits by three zeros (as MSBs).
  - The CRC calculator is fed LSB first. Thus for CRC calculation the sequence given is [A0...A2, F0...F2, X0...X5, D0...D3, ”000”] with “A0” being given first and “0” as the last. When the last zero of the augmentation is calculated on the input adder and the result is shifted, the CRC calculation has been achieved.
  - When mapping this sequence to the DDSR register bits for calculating the CRC; then it becomes (PS_DDSR[0], PS_DDSR[1]....PS_DDSR[15], 000) with PS_DDSR[0] being fed first in the CRC calculator.
- Long (8bit Data Word): It consists of the following data stream (Order of transmission from IP is Left to Right. Thus C0 comes out Last) Data Stream - “010”, S, “A0,A1,A2”, S, “F0,F1,F2”, S, “X0,X1,D0”, S, “D1,D2,D3, S, “D4,D4,D6”, S,“D7,C2,C1”, S, “C0” where “010” is the start sequence “S” is the stuff bit(1), A0-A2 is the sensor address, F0-F2 are the functional codes, “X0-X1” is the Range Address, “D0-D7” is the data and “C2-C0” is the CRC. Refer to *Figure 1577* for the frame structure.
  - The CRC polynomial used is $x3 + x + 1$ with a binary initialization value of “111(C2:C0)”.
  - The transmitter extends the data bits by three zeros (as MSBs).
  - The CRC calculator is fed LSB first. Thus for CRC calculation the sequence given is [A0...A2, F0...F2, X0...X1, D0...D7, ”000”] with “A0” being given first and “0” as

the last. When the last zero of the augmentation is calculated on the input adder and the result is shifted, the CRC calculation has been achieved.

– When mapping this sequence to the DDSR register bits for calculating the CRC; then it becomes (PS_DDSR[0], PS_DDSR[1]....PS_DDSR[15], 000) with PS_DDSR[0] being fed first in the CRC calculator.

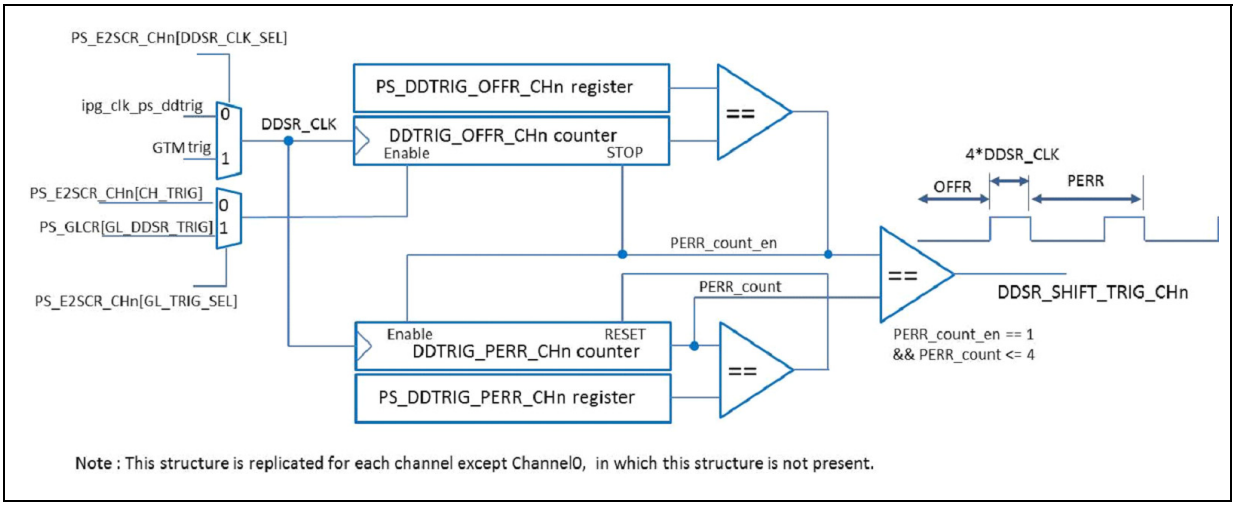
- X-Long: It consists of the following data stream (Order of transmission from IP is Left to Right. Thus C0 comes out Last) Data Stream - “010”, S, “A0,A1,A2”, S, “F0,F1,F2”, S, “X0,X1,X2”, S, “X3,X4,X5”, S, “X6,X7,D0”, S, “D1,D2,D3”, S, “D4,D5,D6”, S,“D7,C2,C1”, S, “C0” where “010” is the start sequence “S” is the stuff bit(1), A0-A2 is the sensor address, F0-F2 are the functional codes, “X0-X5” is the Range Address, “D0-D3” is the data and “C2-C0” is the CRC. Refer to *Figure 1577* for the frame structure.
  – The CRC polynomial used is $x3 + x + 1$ with a binary initialization value of “111(C2:C0)”.
  – The transmitter extends the data bits by three zeros (as MSBs).
  – The CRC calculator is fed LSB first. Thus for CRC calculation the sequence given is [A0...A2, F0...F2, X0...X7, D0...D7, ”000”] with “A0” being given first and “0” as the last. When the last zero of the augmentation is calculated on the input adder and the result is shifted, the CRC calculation has been achieved.
  – When mapping this sequence to the DDSR register bits for calculating the CRC; then it becomes (PS_DDSR[0], PS_DDSR[1]...PS_DDSR[21], 000) with PS_DDSR[0] being fed first in the CRC calculator.
- XX-Long: It consists of the following data stream (Order of transmission from IP is Left to Right. Thus C5 comes Last) Data Stream - “011111110”, “A0,A1,A2”, “C”, “D0,D1”, S, “D2,D3,D4” D5,D6,D7, S, “D8,D9,D10,D11,D12,D13”, S,“D14,D15,D16,D17,D18,D19”, S, “C0,C1,C2,C3,C4,C5” where “011111110” is the start sequence “S” is the stuff bit(0), A0-A2 is the sensor address, “C” is the configuration bit, “D0-D19” is the data and “C0-C5” is the CRC. Refer to *Figure 1577* for the frame structure.
  – The CRC polynomial used is $x6 + x4 + x3 + 1$ with a binary CRC initialization value 010101(C5:C0).
  – The transmitter extends the data bits by six zeros (as MSBs).
  – The CRC calculator is fed LSB first. Thus for CRC calculation the sequence given is [A0...A2, D0...D19, ”000000”] with “A0” being given first and “0” as the last. When the last zero of the augmentation is calculated on the input adder and the result is shifted, the CRC calculation has been achieved.
  – When mapping this sequence to the DDSR register bits for calculating the CRC; then it becomes (PS_DDSR[0], PS_DDSR[1]....PS_DDSR[23], 000000) with PS_DDSR[0] being fed first in the CRC calculator.

61.4.3.1 PSI5-S ECU to sensor command handling

The ECU-to-Sensor commands can be written to the PSI5 IP in two ways:

1. Direct writing for the special Transceiver commands: This can be used by the software to directly write the transceiver/diagnostic commands without using the internal sync pulse generators. Following procedure should be adopted for this:
   a) Application configures the PS_GLCR[DIRCMD_LEN] register for deciding the length of the direct command that is to be written. The length can be 1, 2 or 4 bytes.
   b) Application will check for the status flag PS_GLSR[DIRCMD_RDY]. A “1” here would mean that the PS_DIRCMD register is empty and ready to accept the Direct write command. Trying to write to PS_DIRCMD while DIRCMD_RDY is “0”, will be rejected without any information. If the PS_GLCR[IE_DIRCMD_RDY] bit is set to “1” then an interrupt will also be generated on the “ipi_ps_glbl” interrupt line, whenever the PS_GLSR[DIRCMD_RDY] bit goes to “1”.
   c) The application then writes the Direct Command to the PS_DIRCMD register. Depending on the PS_GLCR[DIRCMD_LEN] register, only the specific bytes are considered for transmission. Other bytes are ignored.
   d) A “w1c” to the PS_GLSR[DIRCMD_RDY] bit would queue the above direct command for UART transmission with the highest priority; and thereafter as soon as any pending current transmission is complete, the internal state machine would start the transfer of this direct command to the UART Tx line. All other channels would internally buffer their pending DDSR transmission and resume only once the above transmission is over.
   e) Once the Direct transmission is complete the PS_GLSR[DIRCMD_RDY] status bit would go to “1” and if the corresponding interrupt is enabled in the PS_GLCR register, then an interrupt will also be generated on the “ipi_ps_glbl” interrupt line.
2. Writing to the DDSR (Downstream Data Shift Register): In this case the application writes the basic command to the DDSR_L and the IP will add the start bits/start condition/CRC/stuff bits before sending this command to the UART Tx FIFO. Following are the details:
   – There is one DDSR per channel. The default value of DDSR is dependent on the PS_E2SCR_CHn[CMD_TYPE] value. DDSR is composed of two sub registers DDSR_H (Higher) and DDSR_L (Lower). The application can only write to DDSR_L. The number of bits that can be written in the DDSR_L is dependent on the value of PS_E2SCR_CHn[CMD_TYPE] bit values. Note that Channel0 does not have any DDSR, as it does not support any ECU-to-Sensor communication.
   – The “shift trigger” of the DDSR is governed by the values programmed in the PS_DDTRIG_OFFR_CH[n] and the PS_DDTRIG_PERR_CH[n] register and the frequency of ipg_clk_ps_ddtrig clock.
   – Once the PS_DDTRIG_OFFR_CH[n] and the PS_DDTRIG_PERR_CH[n are correctly programmed and each channel is programmed to respond to the selected shift trigger (local trigger or global trigger is decided via the bit PS_E2SCR_CHn[GL_TRIG_SEL]) the software has to write to the Global DDSR trigger bit, PS_GLCR[GL_DDSR_TRIG] or the PS_E2CR[CH_TRIG] bit, so as to synchronously start the DDSR triggers of each channel or locally start the DDSR shift triggers of each channel. The DDSR trigger logic gets the trigger signal immediately but it responds after a delay as programmed in the PS_DDTRIG_OFFR_CH[n] register. The shift trigger period is governed by the value programed in the PS_DDTRIG_PERR_CH[n] register. Refer to *Figure 1578* for the structure of the logic that generates the internal DDSR shift triggers.

Figure 1578. DDSR internal shift trigger generator



- The number of bits of DDSR_L that can be written by the application, is dependent on the value of PS_E2SCR_CHn[CMD_TYPE] value.
- When PS_E2SSR_CHn[DDSR_RDY] == 1, then it indicates that the DDSR is empty and is ready to accept a new command. If the PS_E2SCR_CHn[DDSR_RDY_IE] == 1 then this results in the generation of interrupt on the “ipi_ps_e2s_ch[n]” line.
- Attempting to write to DDSR while PS_E2SSR_CHn[DDSR_RDY] == 0 results in the rejection of the new command and setting of the PS_E2SSR_CHn[CMD_NWRT] bit. If the PS_E2SCR_CHn[CMDTR_NWRT_IE] == 1 then this results in the generation of interrupt on the “ipi_ps_e2s_ch[n]” line.
- The application then writes the command to the PS_DDSR_L and does a “w1c” to the PS_E2SCR_CHn[DDSR_RDY]. This indicates to the internal logic that the command is now ready for further processing.The clock that is used for shifting the contents of the DDSR can be configured through the PS_E2SCR_CHn[DDSR_CLK_SEL] register.
- Depending on the value of PS_E2SCR_CHn[CMD_TYPE], the internal logic adds the start bits/start condition/CRC/bit stuffing and updates the DDSR with the complete command, ready for transmission.
- Once this happens the PS_E2SSR_CHn[CMDPR_BZY] becomes “1”. If the PS_E2SCR_CHn[CMDPR_BZY_IE] == 1 then this results in the generation of interrupt on the “ipi_ps_e2s_ch[n]” line. If PS_E2SCR_CHn[CMDTR_SW_CTRL] == 0 then the application can check the final value of the command in DDSR, that would actually be transmitted.Once the checking is complete it can put the PS_E2SCR_CHn[CMDTR_SW_CTRL] to “1”, to start the actual transmission. By default PS_E2SCR_CHn[CMDTR_SW_CTRL] is set to “1”. Note that the start condition is not a part of the DDSR and its transmission is dependent on the value of the PS_E2SCR_CHn[CMD_TYPE] and is automatically taken care by the IP.
- Each bit of the command in DDSR, if it is “1”, results in PS_E2SCR_CHn[CMD] being transferred to the UARTTX buffer or if it is “0” results in PS_E2SCR_CHn[ACMD] being transmitted to the UART-TX buffer. The “CMD” and the “ACMD” governs as to how the transceiver has to respond when it receives these values. These will specify to the transceiver whether it should send

a short or long SYNC pulse on the PSI5 bus when the application wishes to write to the sensors.

– Once the complete command has been transmitted, the PS_E2SSR_CHn[DDSR_RDY] goes to “1” while PS_E2SSR_CHn[CMDPR_BZY] goes to “0.

– Once the PS_E2SSR_CHn[DDSR_RDY] goes to “1” then default “1s” or default “0s” are transmitted, depending on the value of PS_E2SCR_CHn[CMD_TYPE].

– When PS_E2SSR_CHn[DDSR_RDY] == 1, then it is possible to abort the current transmission by doing a “W1C” of the PS_E2SCR_CHn[DDSR_CLR] bit. As soon as this is done the DDSR is reset to “0”, PS_E2SSR_CHn[DDSR_RDY] goes to “1” while PS_E2SSR_CHn[CMDPR_BZY] goes to “0”.Once the PS_E2SSR_CHn[DDSR_RDY] goes to “1” then default “1s” or default “0s” are transmitted, depending on the value of PS_E2SCR_CHn[CMD_TYPE]. Note that doing this “W1C” does not have any effect on the CMD/ACMD transmission that is already under progress in the UART.

Figure 1579. PS_DDSR sensor command (Ex: XXLong Frame)

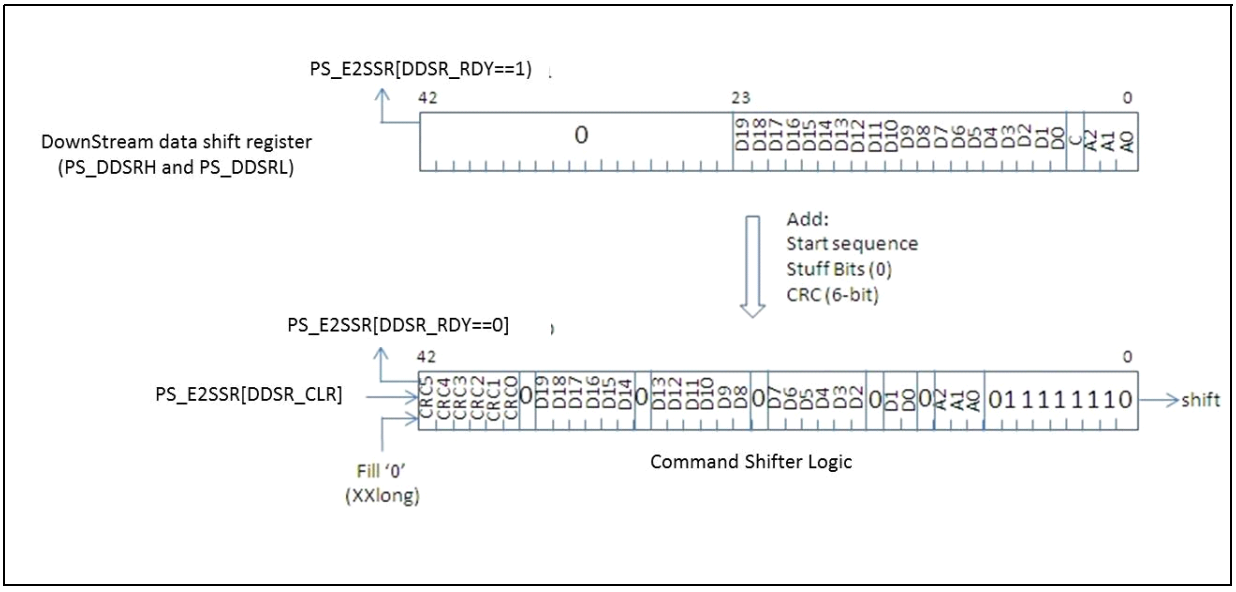


Figure 1580. PS_DDSR sensor command (Ex: PSI5 LONG format)

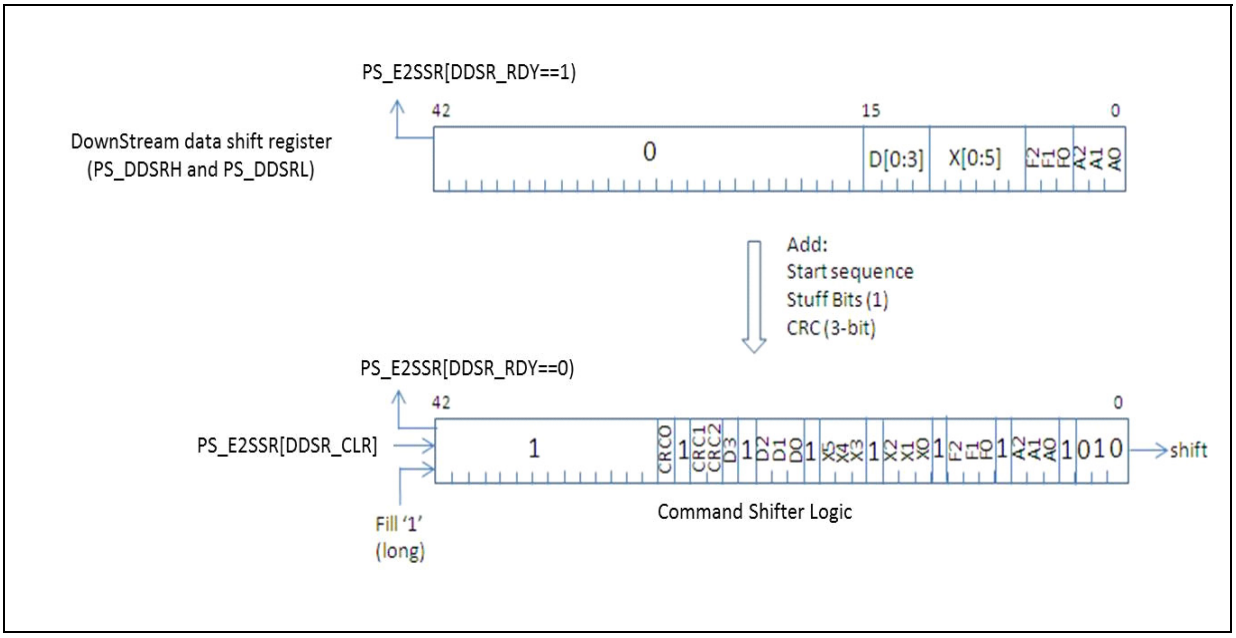
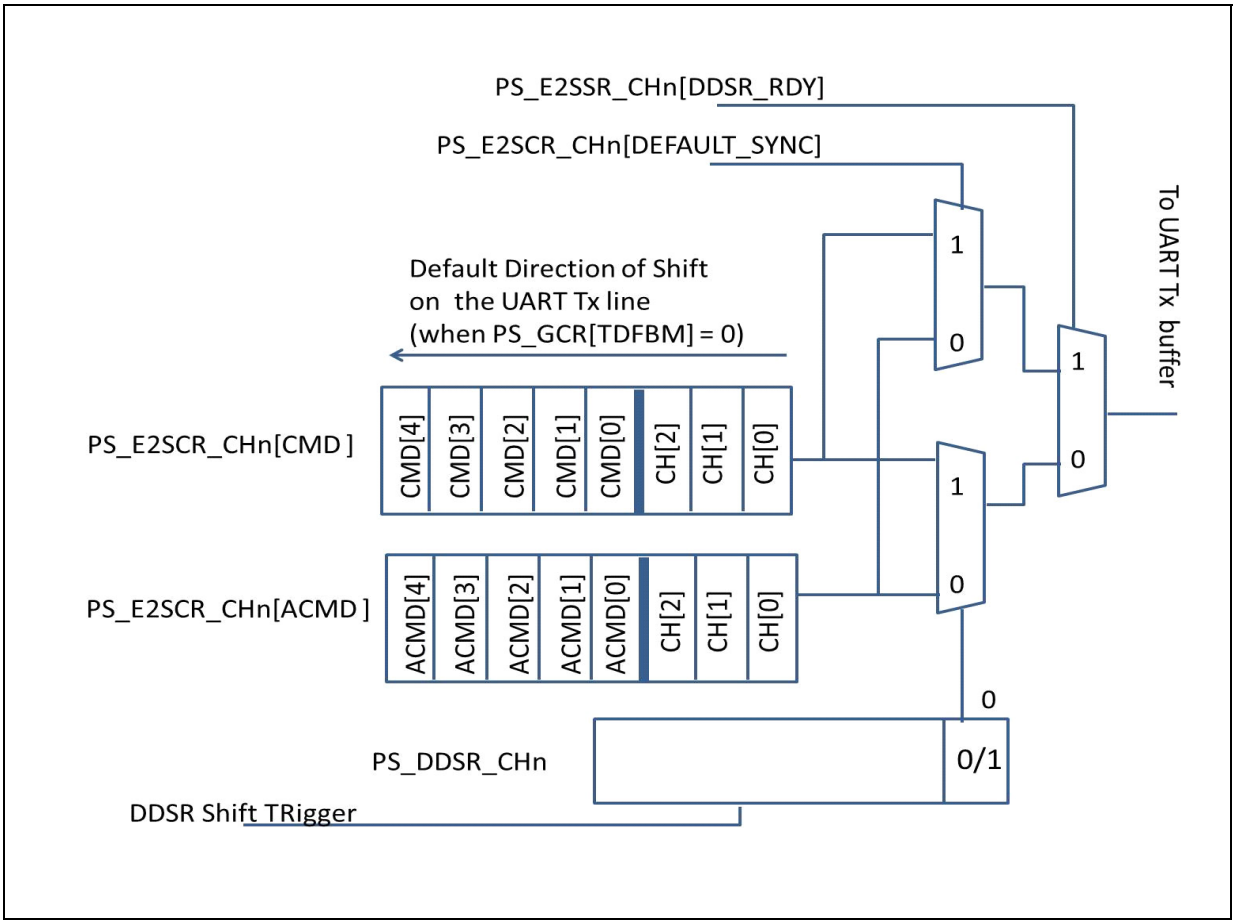


Figure 1581. CMD and ACMD



- The logic for processing the DDSR contents is shared among all the 7 channels. It has to be ensured by the software to correctly program the PS_DDTRIG_OFFR_CH[n] and the PS_DDTRIG_PERR_CH[n] register contents so that the none of the bits in the command get missed. Also since "*1.*" is possible, it is up to the application to ensure that the UART TX_BUFFER does not remain busy to the extent that some particular DDSR bit does not get the chance to be shifted out before it gets overwritten by some new DDSR bit.
- The approximate maximum time, beginning from when PS_E2SSR_CHn[DDSR_RDY] is made "0", to the time when the command is available for transmission is less than 1us, provided no other channel has an active command processing ongoing.If commands are written to multiple DDSR registers one after the other, then the time after which the particular channel will again get an access will increase. Note that the irrespective of the sequence in which the PS_E2SSR_CHn[DDSR_RDY] of the channels are cleared, the channels are always searched in a round robin fashion for the status of the respective DDSR_RDY bits. When the internal sequencer find the respective DDSR_RDY as "0" it starts the processing of the corresponding DDSR. The sequencer always goes as Channel1 -> Channel2 -> Channel3...Channel7 -> Channel1...
- The logic that shifts each bit of the DDSRs of each channel is shared among all the channels. When the shift trigger of a particular DDSR is received and if the UART TX Buffer is empty then the particular command bit is transmitted to the UART TX by the appropriate CMD/ACMD. It is possible that at one point of time, CMD/ACMD of more than one channel are pending to be transmitted to the UART TX. In this case the channel with a higher channel number will be given a priority. Thus if channel2 and channel1, both have a pending CMD/ACMD shift, then channel2 would be the first to get an access. Each pending channel would queue up for transmission. It can be ensured by the software that the internal shift triggers generated through the PS_DDTRIG_PERR_CHn and PS_DDTRIG_OFFR_CHn registers are correctly spaced so that all the channels get immediate access to transmit their DDSR bits.
- If a command bit that is desired to be shifted encounters more than one shift triggers (because it did not get shifted on the previous shift trigger) then the PS_E2SSR_CHn[SYNCHRO_OVF] bit gets set, to indicate synchronization overflow. An interrupt can also be generated if the PS_E2SCR_CHn[SYNCHRO_OVF_IE] bit is set to 1. In general, if the if PS_DDTRIG_PERR is programmed to be greater than 500us (which is a practical PSI5 sync pulse period) then it would be ensured that the bits are correctly transmitted even if all the channels simultaneously transmit their command bits.

61.4.4 Timestamp

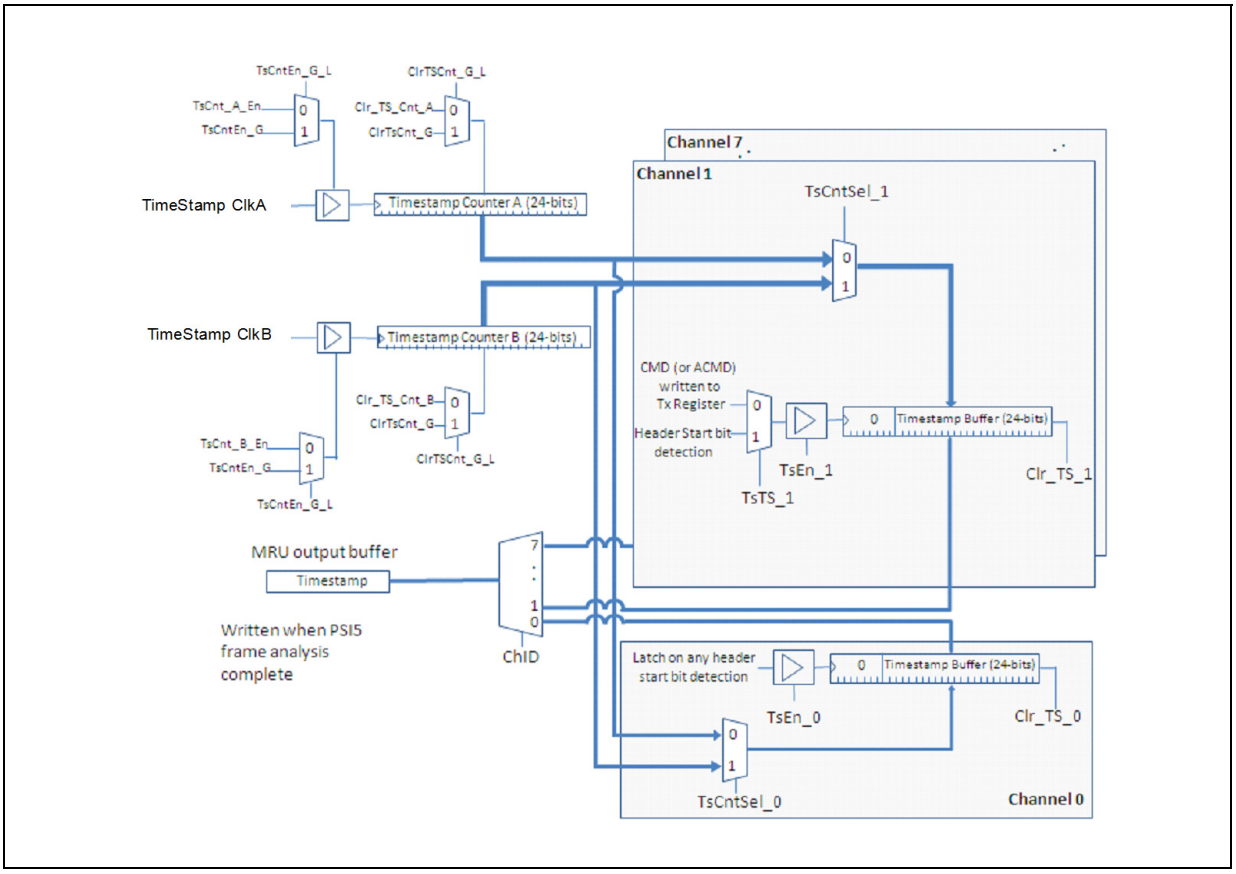
The IP provides for two global 24-bit timestamp counters. Each can be clocked by either the module clock or the GTM trigger. The two timestamp counters are detailed as Timestamp Counter A and B (refer to *Figure 1582*). Following are the details:

1. Each channel has an internal Timestamp buffer that can be configured to capture either the TS counter A or B when it encounters the "capture event". Provided there is no XCRC error in the message, then this Timestamp buffer is appended to the message

when it is written to MRU buf1. When there is an XCRC error in the message, then Channel0 timestamp is appended (See "7.").

2. The "captured event" for the timestamp buffer of each channel can either be when a "a valid message header is detected" or when "a Command (or Alternate Command) is moved from the UART Tx Buffer to the UART Tx shift register" corresponding to the channel for which this command is being shifted to the UART
3. The Timestamp Buffer can be enabled or disabled through the PS_MSGA_CHn[TSBUF_EN] bit. When the Timestamp buffer is disabled then it no longer responds to "capture events" and always contains the value "0".
4. It is possible to reset each Timestamp buffer by using the PS_MSGA_CHn[TSBUF_CLR] bit. As soon as this bit is written to "1" the Timestamp buffer is reset to all "0s" and it remains in this state till it encounters a "capture event".
5. In the case when the reception of message is chosen as the trigger, the time stamp will be captured immediately after the header byte of the UART message packet has been received.
6. In the case when the move of CMD or ACMD from the internal UART Tx Buffer to the UART Tx shift register in the UART module is chosen as the trigger, the timestamp for the intended channel (indicated in the CIDn fields of CMD or ACMD) is valid for all frames received on a channel between moves of CMD or ACMD from the UART Tx Buffer to the UART Tx Register (that is for a PSI5 SYNC period).
7. When a XCRC error is detected in any message then the timestamp that is appended with the message is the Channel0 timestamp which is captured at the time when XCRC_ERR has been detected.
8. For "illegal messages", the timestamp value that is captured is the Channel0 Timestamp value, which is captured whenever "a valid message header for that illegal message is detected". Note that the definition of a "legal" or a "illegal" message applies to a PSI5 message only when the concerned UART packet has no XCRC error.
9. In addition to the timestamp buffers described in "1.", there is a separate timestamp capture that occurs in case of a watchdog error. This timestamp is captured in the PS_WDGTSSR[WDGTS_STATUS] bits. The Timestamp A or B (as programmed in the PS_MSGA_CHn[TIMESTAMP_A_B_SEL]) is captured as soon as the watchdog overrun error occurs for that channel. Refer to *Section 61.3.4.12: PSI5-S watchdog error status and watchdog timestamp status register (PS_WDGTSSR)*, for more details.

Figure 1582. Timestamp capture



61.4.5 PSI5-S watchdog operation

The IP provides a watchdog to keep track of a number of reception parameters. The watchdog counter is a 24 bit down counter. Each channel, except Channel0, contains a watchdog counter. The refresh event for the watchdog is automatically chosen depending on whether the channels mode is synchronous or asynchronous. The watchdog timeout value can be configured in the “WD_TO” bits in the PS_WD_CFGR_CH[n] register. The refresh event for the watchdog are different in the synchronous and asynchronous modes as mentioned below:

- In Asynchronous mode:

  As soon as the IP enters the PS_NORMAL mode the watchdog for each channel is loaded with the PSI5_WD_CFGR_CH[n][WD_TO] value, and in case the watchdog is enabled then the counter starts immediately. Whether the WD has to be “refreshed (re-loaded)” or “not to be refreshed” is analyzed when the XCRC is calculated. In case the XCRC is correct the watchdog is re-loaded (It is not STOPPED), else the WD counter keeps on running normally eventually leading to a watchdog overrun error. In case of watchdog overrun error the counter is refreshed.

- In Synchronous Mode:

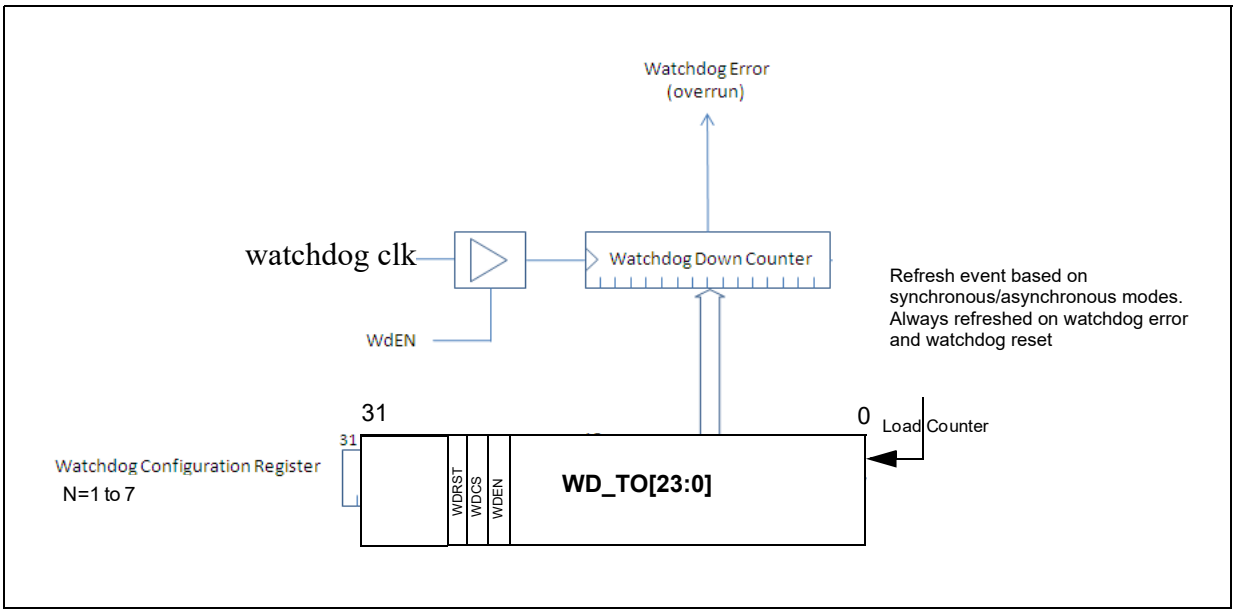
  As soon as the IP enters the PS_NORMAL mode, the watchdog for each channel is loaded with the PSI5_WD_CFGR_CH[n][WD_TO] value and in case the watchdog is enabled then the counter starts immediately. Thereby it is re-loaded, either on the occurrence of a sync pulse or the watchdog overrun error. In case the event for this

load is the Sync Pulse then it immediately starts counting again and keeps on counting till it is stopped when the “Packet_counter” reaches the count of “total number of configured slots”. The “Packet_counter” here is a counter that keeps track of the number of configured (enabled) slots. It is incremented on the reception of each UART packet with a “CID” and “FID” such that PS_MSGA_CHCID[FID_byte] is not equal to “0”. The packet counter is however incremented ONLY if the XCRC of the received UART packet is correct else it is not incremented. In case the “Packet Counter” does not reach the “expected packet count” within the “WD timeout configured“, the WD overrun error is generated. In case of a WD overrun error, the counter is stopped and does not start till a Sync Pulse is encountered.

In both the above modes, the watchdog is always refreshed on the occurrence of Watchdog overrun error (F_WD_ERR) or a software reload. Also refer to *Section 61.3.4.20: PSI5-S channel watchdog configuration register (PS_WD_CFGR_CHn)*.

The watchdog can be started in the NORMAL mode by writing a “1” to the WDEN bit in the PSI5_WD_CFGR_CH[n]. If the watchdog counter is not refreshed by its event and the watchdog counter becomes equal to “0”; the F_WD_ERR flag is set in the PSI5_IRQ_ERR_SR register and the same is also reflected in the PS_MRU_BUF2_REG1 register. An interrupt can also be generated on any of the “ipi_ps_irq[irq_n]” lines depending on the configuration. Note that writing a “0” to WDEN bit disables the watchdog checking mechanism.*Figure 1583* shows the watchdog counter.

Figure 1583. Watchdog counter



As soon as the watchdog overrun error for a particular channel occurs, the status of the watchdog error is immediately latched in the register described in *Section 61.3.4.12: PSI5-S watchdog error status and watchdog timestamp status register (PS_WDGTSSR)*. This watchdog error status (PS_WDGTSSR[F_WD_ERR_STATUS[CHn]]) is also sent to the MRU registers (as F_WD_ERR) as shown in the PSI5 frame status word (shown in *Figure 1573* and *Figure 1574*). However this information is sent in the next upcoming PSI5 message packet that occurs for that particular channel, provided this upcoming PSI5 message has no XCRC error. In case the upcoming UART packet has an XCRC error then the Frame watchdog error field in this PSI5 message packet is the “OR” of the F_WD_ERR

status of all the channels that is OR of all the fields of PS_WDGTSSR[F_WD_ERR_STATUS] bits.

61.4.6 Interrupt handling

Following are the various interrupt signals that represent the various events:

1. PSI5-S general interrupt request (Status flags in PS_MBOX_SR_IRQ and PS_ERR_SR_IRQ)/ipi_ps_irq[irq_n] where irq_n = 0 to 7 and denotes the irq number.

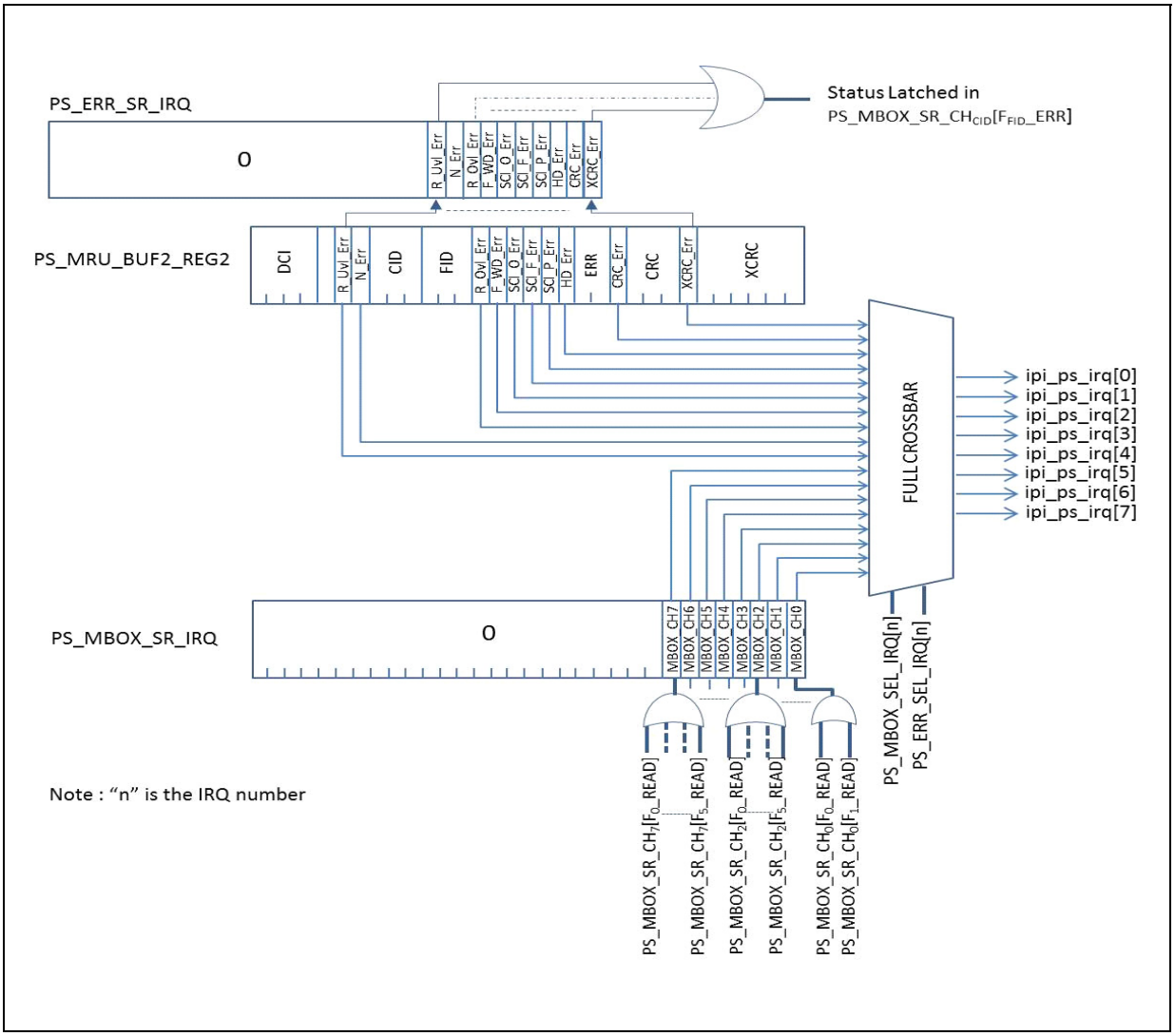
   There are 8 such interrupts in the IP. Each of these “irq” lines can be configured to respond to any or all of the following events:

   – At least one message of the specific channel has been read by the DMA from the MRU buffer (The interrupt is configurable to respond to any channel through the corresponding PS_MBOX_SEL_IRQ[irq_n] register)
   – When there is a XCRC error in a new packet that is transferred to MRU buffer2 OR
   – When there is a CRC or parity error in a new packet that is transferred to MRU buffer2 OR
   – When the watchdog Error has been detected in the message just written to MRU buf2 OR
   – When the message header has been detected with transceiver error flags OR
   – When the PSI5 state machine detects in the UART, the message parity error, the overflow error or the framing error and the corresponding frame is transferred to MRU buffer 2
   – When the “R_OVL_ERR” error occurs in the new packet that is transferred to MRU buffer 2 OR
   – When the “R_UVL_ERR” error occurs in the new packet that is transferred to MRU buffer 2 OR
   – When the number of bytes received does not match the expected Fn bytes (N_ERR == 1)

   Each of the “ipi_ps_irq[irq_n]” lines can be configured to be asserted on any of the above events by correctly programming the registers PS_MBOX_SEL_IRQ[irq_n] and PS_ERR_SEL_IRQ[irq_n].

   This interrupt is cleared when the corresponding bits in PS_MBOX_SR_IRQ and PS_ERR_SR_IRQ are cleared. It is assumed that one event would not be mapped on more than one “irq” lines. If the same event is mapped on more than one “irq” lines then clearing the corresponding status bit in the PS_MBOX_SR_IRQ or PS_ERR_SR_IRQ will clear all the interrupt requests related to that same event.*Figure 1584* shows the interrupt structure for IRQ interrupts.

Figure 1584. IRQ[n] interrupts

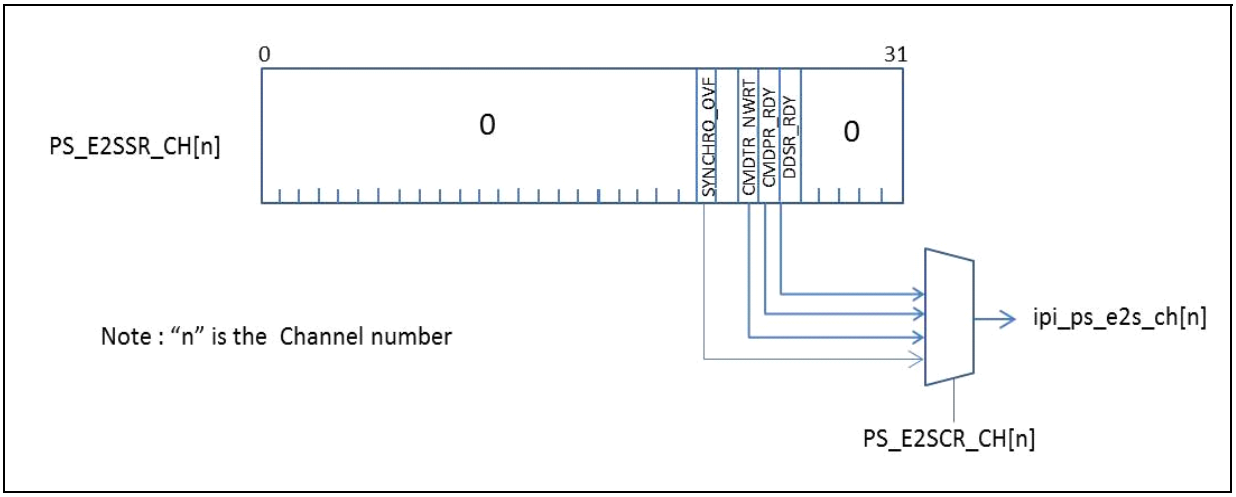


2. PSI5-S ECU-to-Sensor interrupt (Status Flags in PSI5_E2SSR)/ipi_ps_e2s_ch[n] where n = 1 to 7.

   This interrupt is used for the ECU to Sensor Communications and is generated in the following cases:

   – When the DDSR is empty and is ready to accept a new command new from the application. In this case the PS_E2SSR_CHn[DDSR_RDY] bit is set to "1".
   – When the DDSR is still busy (PS_E2SSR_CHn[DDSR_RDY] bit is set to "0") and the application attempts to write a new command to the DDSR. The new command is rejected and the PS_E2SSR_CHn[CMD_NWRT] bit is set to "1".
   – When the command that is written to the DDSR has finished calculating the CRC/stuff bits/start bits and is ready for transmission the PS_E2SSR_CHn[CMDPR_BZY] goes to "1" leading to the generation of this interrupt.
   – When there is a synchronization overflow in a particular channel (the same data bit encounters 2 sync pulses without getting a chance to shift), then the bit PS_E2SSR_CHn[SYNCHRO_OVF] is set to "1".

Figure 1585. ECU to sensor interrupts



3. Global interrupt (Status Flags in PS_GLSR) /ipi_ps_glbl.

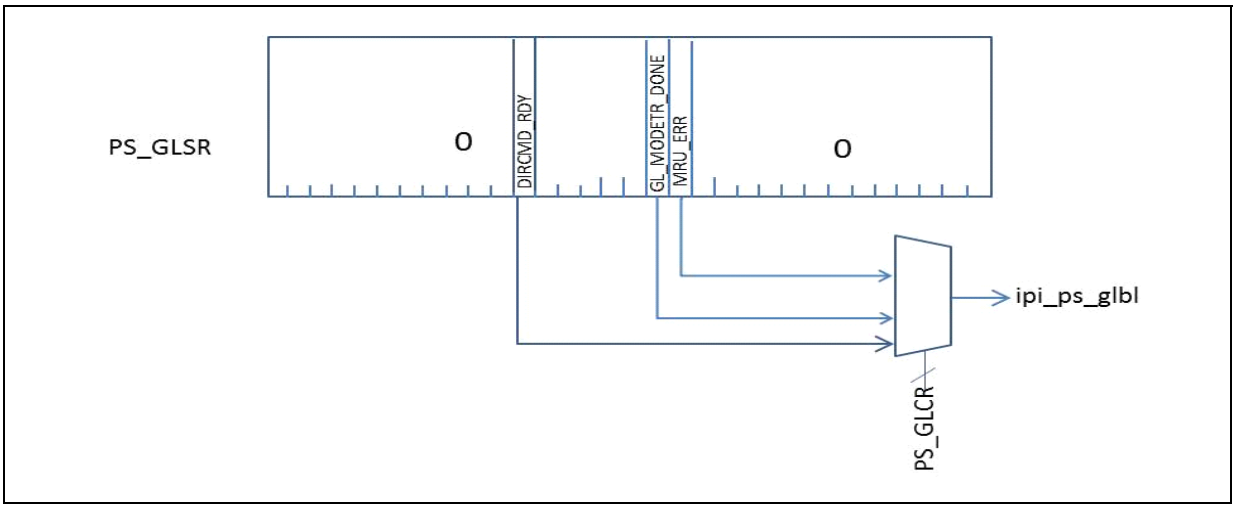
   This interrupt is a single interrupt for the whole IP and is generated in the following cases:

   – When the DMA has not completed reading the MRU buffer2; simultaneously there is a pending data in MRU buffer1 that needs to be offloaded to the MRU buffer2 and a new message comes, that overwrites the contents of MRU buffer 1. The channel ID and the Frame ID of the overwritten message can be read from the PS_GLSR registers.

   – When there is any change in the "PSI5-S" global modes of the IP. Thus it is asserted during the following transitions UART_STDALONE to PS_DISABLE, PS_DISABLE to PS_CONFIG, PS_DISABLE to PS_NORMAL and PS_CONFIG to PS_NORMAL. Assertion of this interrupt indicates that the desired mode has been accomplished by the IP.

   – When the "DIRCMD_RDY" bit in the PS_GLSR is set to "1", indicating that PS_DIRCMD register is ready to accept a new Direct Command write from the CPU.

Figure 1586. Global interrupts



61.5 PSI5-S clock and reset

Figure 1587. Clock distribution in PSI5-S

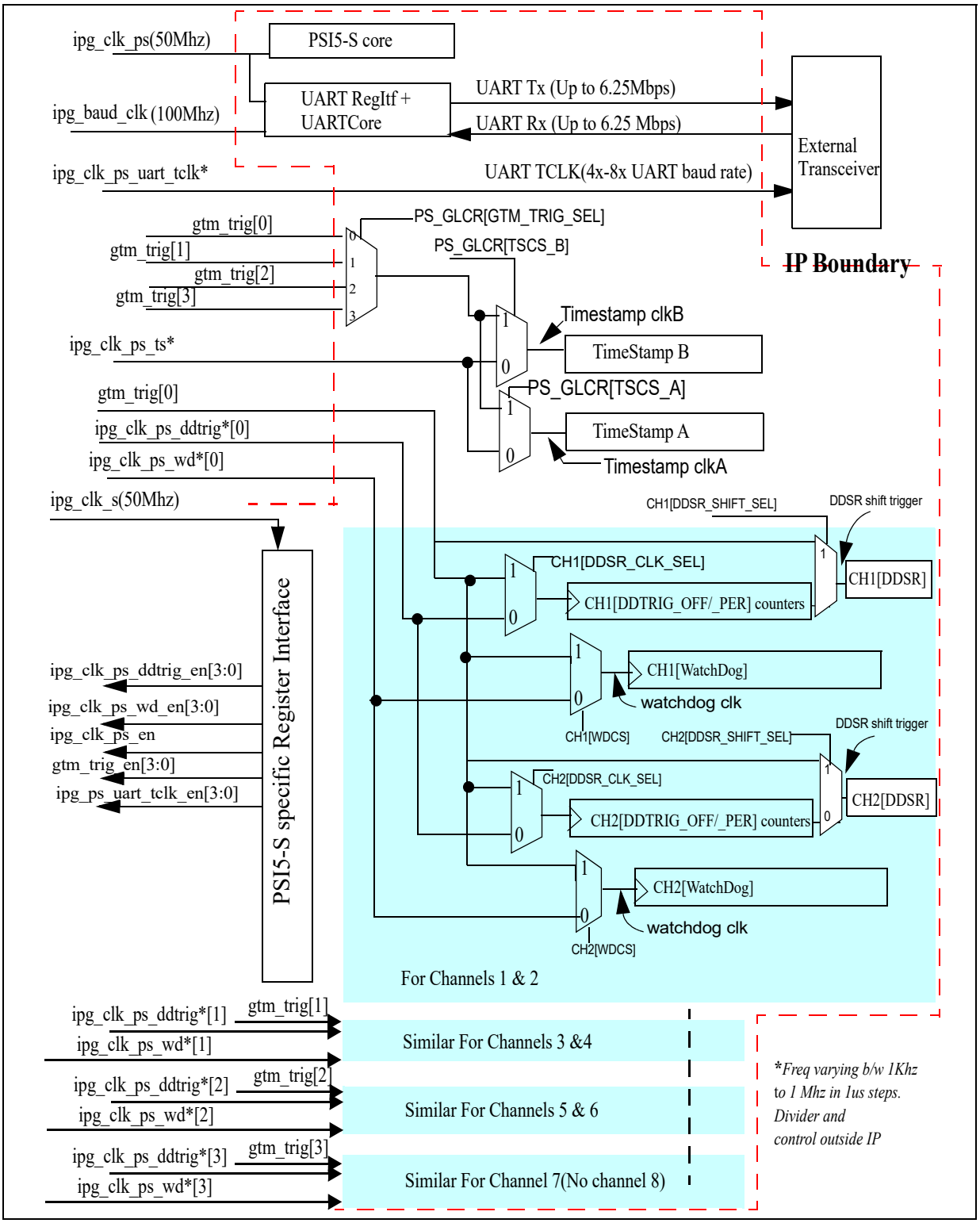


Figure 1587 shows the clocking scheme for the PSI5-S IP. It also shows the distribution of various clocks on the basis of channels.

The divider and the control of clock “ipg_clk_ps_ts”, “ipg_clk_ps_wd” and “ipg_clk_ps_ddtrig” are outside the IP. The various clock enables are also controlled by the IP. When the IP enters the disable mode it deasserts all the clock enables from the IP. However the clock enables are deasserted only when once the PS_DISABLE mode entry interrupt has been received and the PS_GLSR[GL_MODETR_DONE_EN] bit has been cleared.

For the safe operation of the UART module, the following relation should be satisfied between “ipg_baud_clk” and “ipg_clk_ps”:

ipg_baud_clk > *ipg_clk_ps* > ipg_baud_clk/3

62 Deserial Serial Peripheral Interface (DSPI)

62.1 Introduction

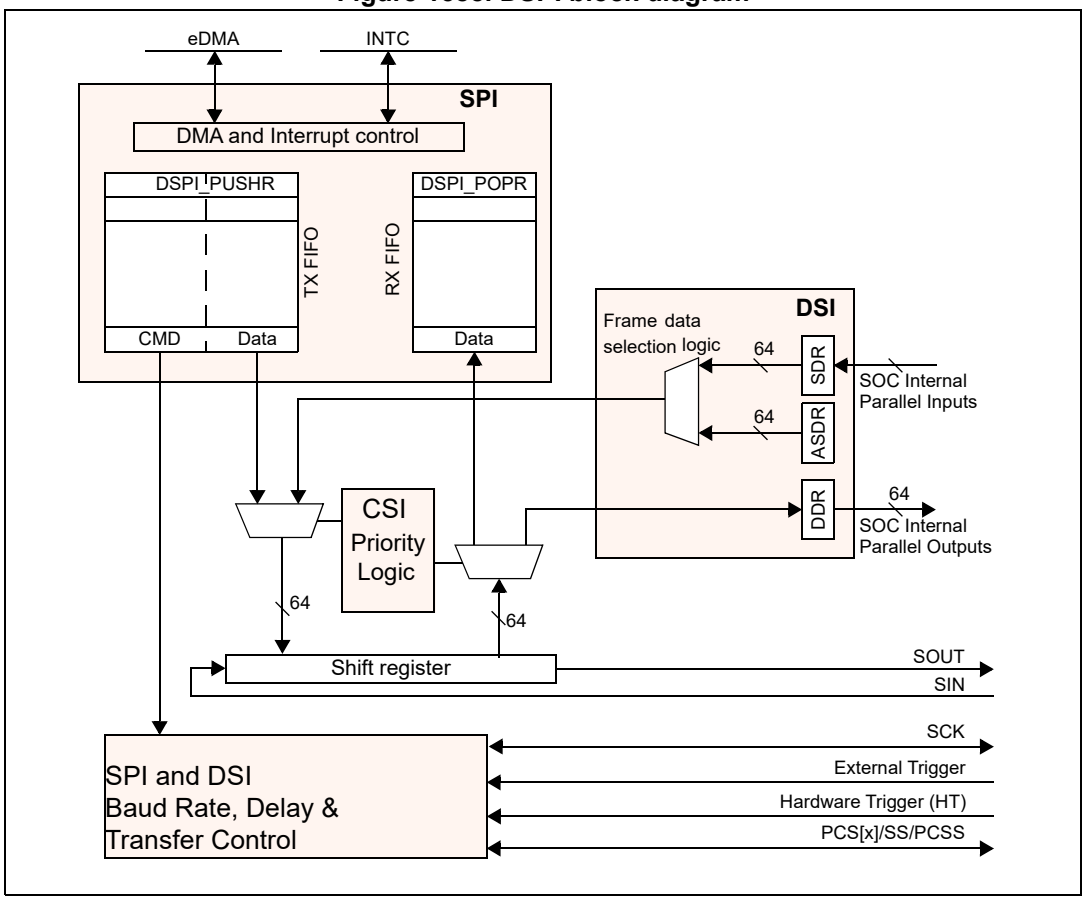
The Deserial Serial Peripheral Interface (DSPI) module provides a synchronous serial bus for communication between an MCU and an external peripheral device. The module supports MCU pin count reduction through serialization and deserialization of MCU internal signals transmitted over the SPI serial link.

Note: For the chip-specific implementation details of the instances of this module, refer to the Device Configuration chapter.

62.1.1 Block diagram

Figure 1588 provides the block diagram of this module.

Figure 1588. DSPI block diagram



62.1.2 Features

The DSPI supports the following features:

- Full-duplex, four-wire synchronous transfers.
- Master and Slave modes: data streaming operation in slave mode with continuous slave selection.
- Buffered transmit operation with 4-entry TX FIFO buffer.
- Buffered receive operation with 4-entry RX FIFO buffer.
- Asynchronous clocking scheme for Register and Protocol interfaces.
- Individual TX and RX FIFOs disabling for low-latency updates to SPI queues.
- Visibility inside TX and RX FIFOs for debugging.
- Programmable transfer attributes on a per-frame basis:
  - 8 transfer attribute registers along with 8 extended transfer attribute registers
  - Serial clock with programmable polarity and phase
  - Various programmable delays
  - programmable serial frame size of 4 to 64 bits, expandable by software control. SPI frames longer than 32 bits are supported using the continuous selection format
  - Continuously held chip select capability
  - Parity control
- 8 Peripheral Chip Selects, expandable to 256 with external demultiplexer.
- Deglitching support for up to 128 Peripheral Chip Selects with external demultiplexer.
- DMA support for adding entries to TX FIFO and removing entries from RX FIFO:
  - TX FIFO is not full (TFFF)
  - RX FIFO is not empty (RFDF)
  - CMD FIFO is not full (CMDFFF)
- Interrupt conditions:
  - End Of Queue reached (EOQF)
  - TX FIFO is not full (TFFF)
  - CMD FIFO is not full (CMDFFF)
  - Transfer of current frame Complete (TCF)
  - Transfers due from current Command frame Complete (CMDTCF)
  - Transfer of current SPI frame Complete (SPITCF)
  - Attempt to transmit with an empty Transmit FIFO (TFUF)
  - RX FIFO is not empty (RFDF)
  - Frame received while Receive FIFO is full (RFOF)
  - SPI Parity Error (SPEF)
  - Data present in TX FIFO while CMD FIFO is empty (TFIWF)
- Global interrupt request line.
- Modified SPI transfer formats for communication with slower peripheral devices.
- Power-saving architectural features: support for stop mode

The DSPI also supports pin reduction through serialization and deserialization if enabled for the module.

- Two sources of serialized data:
  - DSPI memory-mapped register
  - Parallel input signals
  - Programmable selection of source data on bit basis.
- Deserialized data provided as:
  - Parallel output signals
  - Bits in a memory-mapped register
- Interrupt conditions:
  - Deserialized data matches preprogrammed pattern (DDIF)
  - Transfer of current DSI frame complete (DSITCF)
  - DSI parity error (DPEF)
- DMA request support for following conditions:
  - Deserialized data matches preprogrammed pattern (DDIF)
- Transfer initiation conditions:
  - Continuous
  - Edge sensitive hardware trigger
  - Change in data
- Pin serialization/deserialization with interleaved SPI frames for control and diagnostics.
- Continuous serial communications clock.

DSPI supports a combination of SPI and DSI modes of operation (Combined Serial Interface (CSI)) for the downstream Micro Second Channel in either Timed Serial Bus (TSB) configuration or Interleaved Frames Configuration (configurable by software). Both TSB and Interleaved TSB modes have the following common features:

- Transmission of frames is performed at frame boundaries.
- Frames from SPI and DSI are identifiable by a bit transmitted at the start of each frame.
- Separate interrupts for frame completion from SPI and DSI.

62.1.3 DSPI configurations

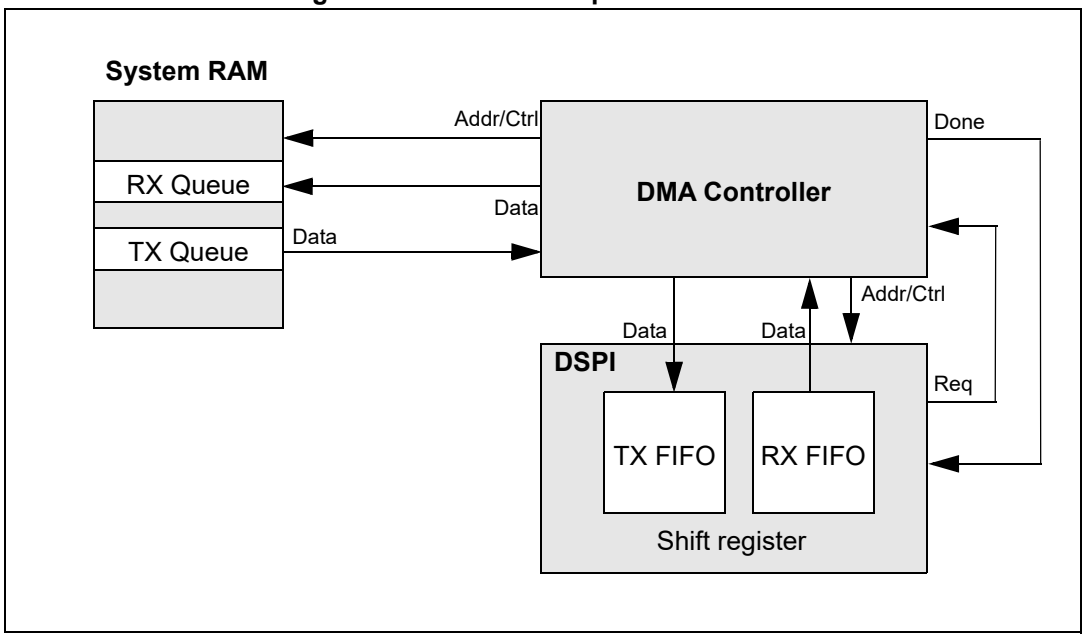
The DSPI module can operate in SPI configuration as well as in DSI, and CSI configuration. Refer to the relevant section in the “Communication interfaces” section of the “Device Configuration” chapter.

62.1.3.1 SPI configuration

The SPI configuration allows the DSPI to send and receive serial data. This configuration allows the DSPI to operate as a basic SPI block with internal FIFOs supporting external queues operation. Transmit data and received data reside in separate FIFOs. The host CPU or a DMA controller read the received data from the receive FIFO and write transmit data to the transmit FIFO.

For queued operations, the SPI queues can reside in system RAM which is external to the DSPI. Data transfers between the queues and the DSPI FIFOs are accomplished by a DMA controller or host CPU. *Figure 1589* shows a system example with DMA, DSPI and external queues in system RAM.

Figure 1589. DSPI with queues and DMA



62.1.3.2 DSI configuration

In the DSI configuration the DSPI serializes up to 64 parallel input signals or register bits. The DSPI also deserializes the received data to parallel output signals or to a memory-mapped register. The data is transferred using a SPI-like protocol.

TSB mode provides the Micro Second downstream Channel support (MSC), serializing from 4 to 64 parallel input signals or register bits. Refer to *Section 62.5.9: Timed Serial Bus (TSB)*.

62.1.3.3 CSI configuration

There are three configurations available in this mode.

- The normal CSI configuration is a combination of the SPI and DSI configurations. In this configuration the DSPI interleaves DSI data frames with SPI data frames. Interleaving is done on the frame boundaries.
- In the TSB configuration, transmission of SPI data has higher priority than DSI data.
- In the Interleaved TSB (ITSB) configuration, the frames from SPI and DSI are interleaved without priority. On every trigger, frames from DSI are sent when there are no frames in the SPI or the previous transmission was a frame from SPI. ITSB is detailed in *Section 62.5.10: Interleaved TSB (ITSB) Mode*.

62.1.4 Modes of operation

The DSPI modes of operation that can be divided into two categories:

- Module-specific modes
  - Master mode
  - Slave mode
  - Module Disable mode
- MCU-specific modes
  - External Stop mode
  - Debug mode

The DSPI enters module-specific modes when the host writes a DSPI register. The MCU-specific modes are controlled by signals external to the DSPI. The MCU-specific modes are modes that an MCU may enter in parallel to the DSPI block-specific modes.

62.1.4.1 Master mode

Master mode allows the DSPI to initiate and control serial communication. In this mode, the SCK signal and the PCS[x] signals are controlled by the DSPI and configured as outputs.

62.1.4.2 Slave mode

Slave mode allows the DSPI to communicate with SPI/DSI bus masters In this mode, the DSPI responds to externally controlled serial transfers. The SCK signal and the PCS[0]/SS signals are configured as inputs and driven by a SPI bus master.

62.1.4.3 Module Disable mode

The Module Disable mode can be used for MCU power management. The clock to the non-memory-mapped logic in the DSPI can be stopped while in Module Disable mode.

62.1.4.4 External Stop mode

External Stop mode is used for MCU power management; the DSPI supports the Peripheral Bus Stop mode mechanism.

Upon a request to enter External Stop mode:

- In Master mode, the DSPI block acknowledges the request and completes the transfer in progress. When the DSPI reaches the frame boundary, it signals that the protocol clock to the DSPI module may be shut off.
- In Slave mode, a request to enter External Stop mode must be run after the last clock cycle or Low-power mode is not entered.

62.1.4.5 Debug mode

Debug mode is used for system development and debugging. The MCR[FRZ] bit controls DSPI behavior in the debug mode. If the bit is set, the DSPI stops all serial transfers, when the MCU is in Debug mode. If the bit is cleared, the MCU Debug mode has no effect on the DSPI.

62.2 DSPI signal description

This section provides the DSPI signals description.

Table 1592 lists the signals that may connect off chip depending on device implementation.

Table 1592. DSPI signal description

Signal	Description	I/O
PCS0/ SS	Master mode: Peripheral Chip Select 0 output Slave mode: Slave Select input	I/O
PCS[3:1]	Master mode: Peripheral Chip Select 1–3 Slave mode: Unused	O
PCS4	Master mode: Peripheral Chip Select 4	O
PCS5/ PCSS	Master mode: Peripheral Chip Select 5 / Peripheral Chip Select Strobe Slave mode: Unused	O
PCS[7:6]	Master mode: Peripheral Chip Select 6–7 Slave mode: Unused	O
SIN	Serial Data In	I
SOUT	Serial Data Out	O
SCK	Master mode: Serial Clock (output) Slave mode: Serial Clock (input)	I/O
HT	Hardware Trigger	I

62.2.1 PCS0/$\overline{SS}$ — Peripheral Chip Select/Slave Select

In master mode, the PCS0 signal is a Peripheral Chip Select output that selects which slave device the current transmission is intended for.

In slave mode, the active low $\overline{SS}$ signal is a Slave Select input signal that allows a SPI master to select the DSPI as the target for transmission.

62.2.2 PCS1 – PCS3 — Peripheral Chip Selects 1 – 3

PCS1 – PCS3 are Peripheral Chip Select output signals in master mode.

In slave mode, these signals are unused.

62.2.3 PCS4 — Peripheral Chip Select 4

In master mode, PCS4 is a Peripheral Chip Select output signal.

62.2.4 PCS5/$\overline{PCSS}$ — Peripheral Chip Select 5/Peripheral Chip Select Strobe

PCS5 is a Peripheral Chip Select output signal. When the DSPI is in master mode and the MCR[PCSSE] bit is cleared, this signal selects which slave device the current transfer is intended for.

When the DSPI is in master mode and the MCR[PCSSE] bit is set, the $\overline{\text{PCSS}}$ signal acts as a strobe to an external peripheral chip select demultiplexer, which decodes the PCS0 – PCS4 and PCS6 – PCS7 signals, preventing glitches on the demultiplexer outputs.

This signal is not used in slave mode.

62.2.5 PCS[6] – PCS[7] — Peripheral Chip Selects 6 – 7

PCS[6] – PCS[7] are Peripheral Chip Select output signals in master mode. In slave mode, these signals are not used.

62.2.6 SIN — Serial Input

SIN is a serial data input signal.

62.2.7 SOUT — Serial Output

SOUT is a serial data output signal.

62.2.8 SCK — Serial Clock

SCK is a serial communication clock signal. In master mode, the DSPI generates the SCK. In slave mode, SCK is an input from an external bus master.

62.2.9 HT — Hardware Trigger

In master mode while in DSI or CSI Configurations, the HT signal initiates a data transfer when the DSICR0[TRRE] bit is set and a rising or falling edge is detected on HT. Which edge to trigger on is determined by the DSICR0[TPOL] bit.

62.3 Memory map/register definition

Register accesses to memory addresses that are reserved or undefined result in a transfer error. Write access to the DSPI_POPR also results in a transfer error.

Table 1593. DSPI memory map

Offset (hex)	Register name	Width (in bits)	Access	Reset value	Location
0000_0000	DSPI Module Configuration Register (DSPI_MCR)	32	R/W	0000_4001h	*Section 62.3.1*
0000_0004	Hardware Configuration Register (DSPI_HCR)	32	R	0133_3000h	*Section 62.3.2*
0000_0008	DSPI Transfer Count Register (DSPI_TCR)	32	R/W	0000_0000h	*Section 62.3.3*
0000_000C	DSPI Clock and Transfer Attributes Register 0 (In Master Mode) (DSPI_CTAR0)	32	R/W	7800_0000h	*Section 62.3.4*
0000_000C	DSPI Clock and Transfer Attributes Register 0 (In Slave Mode) (DSPI_CTAR0_SLAVE)	32	R/W	7800_0000h	*Section 62.3.5*
0000_0010	DSPI Clock and Transfer Attributes Register 1 (In Master Mode) (DSPI_CTAR1)	32	R/W	7800_0000h	*Section 62.3.4*

Table 1593. DSPI memory map (continued)

Offset (hex)	Register name	Width (in bits)	Access	Reset value	Location
0000_0010	DSPI Clock and Transfer Attributes Register 1 (In Slave Mode) (DSPI_CTAR1_SLAVE)	32	R/W	7800_0000h	*Section 62.3.5*
0000_0014	DSPI Clock and Transfer Attributes Register 2 (In Master Mode) (DSPI_CTAR2)	32	R/W	7800_0000h	*Section 62.3.4*
0000_0018	DSPI Clock and Transfer Attributes Register 3 (In Master Mode) (DSPI_CTAR3)	32	R/W	7800_0000h	*Section 62.3.4*
0000_001C	DSPI Clock and Transfer Attributes Register 4 (In Master Mode) (DSPI_CTAR4)	32	R/W	7800_0000h	*Section 62.3.4*
0000_0020	DSPI Clock and Transfer Attributes Register 5 (In Master Mode) (DSPI_CTAR5)	32	R/W	7800_0000h	*Section 62.3.4*
0000_0024	DSPI Clock and Transfer Attributes Register 6 (In Master Mode) (DSPI_CTAR6)	32	R/W	7800_0000h	*Section 62.3.4*
0000_0028	DSPI Clock and Transfer Attributes Register 7 (In Master Mode) (DSPI_CTAR7)	32	R/W	7800_0000h	*Section 62.3.4*
0000_002C	DSPI Status Register (DSPI_SR)	32	R/W	Refer to the register description	*Section 62.3.6*
0000_0030	DSPI DMA/Interrupt Request Select and Enable Register (DSPI_RSER)	32	R/W	0000_0000h	*Section 62.3.7*
0000_0034	DSPI PUSH FIFO Register In Master Mode (DSPI_PUSHR)	32	R/W	0000_0000h	*Section 62.3.8*
0000_0034	DSPI PUSH FIFO Register In Slave Mode (DSPI_PUSHR_SLAVE)	32	R/W	0000_0000h	*Section 62.3.9*
0000_0038	DSPI POP FIFO Register (DSPI_POPR)	32	R	0000_0000h	*Section 62.3.10*
0000_003C	DSPI Transmit FIFO Register 0 (DSPI_TXFR0)	32	R	0000_0000h	*Section 62.3.11*
0000_0040	DSPI Transmit FIFO Register 1 (DSPI_TXFR1)	32	R	0000_0000h	*Section 62.3.11*
0000_0044	DSPI Transmit FIFO Register 2 (DSPI_TXFR2)	32	R	0000_0000h	*Section 62.3.11*
0000_0048	DSPI Transmit FIFO Register 3 (DSPI_TXFR3)	32	R	0000_0000h	*Section 62.3.11*
0000_007C	DSPI Receive FIFO Register 0 (DSPI_RXFR0)	32	R	0000_0000h	*Section 62.3.12*
0000_0080	DSPI Receive FIFO Register 1 (DSPI_RXFR1)	32	R	0000_0000h	*Section 62.3.12*
0000_0084	DSPI Receive FIFO Register 2 (DSPI_RXFR2)	32	R	0000_0000h	*Section 62.3.12*
0000_0088	DSPI Receive FIFO Register 3 (DSPI_RXFR3)	32	R	0000_0000h	*Section 62.3.12*

Table 1593. DSPI memory map (continued)

Offset (hex)	Register name	Width (in bits)	Access	Reset value	Location
0000_00BC	DSPI DSI Configuration Register 0 (DSPI_DSICR0)	32	R/W	0000_0000h	*Section 62.3.13*
0000_00C0	DSPI DSI Serialization Data Register 0 (DSPI_SDR0)	32	R	0000_0000h	*Section 62.3.14*
0000_00C4	DSPI DSI Alternate Serialization Data Register 0 (DSPI_ASDR0)	32	R/W	0000_0000h	*Section 62.3.15*
0000_00C8	DSPI DSI Transmit Comparison Register 0 (DSPI_COMPR0)	32	R	0000_0000h	*Section 62.3.16*
0000_00CC	DSPI DSI Deserialization Data Register 0 (DSPI_DDR0)	32	R	0000_0000h	*Section 62.3.17*
0000_00D0	DSPI DSI Configuration Register 1 (DSPI_DSICR1)	32	R/W	0000_0000h	*Section 62.3.18*
0000_00D4	DSPI DSI Serialization Source Select Register 0 (DSPI_SSR0)	32	R/W	0000_0000h	*Section 62.3.19*
0000_00D8	DSPI DSI Parallel Input Select Register 0 (DSPI_PISR0)	32	R/W	0000_0000h	*Section 62.3.20*
0000_00DC	DSPI DSI Parallel Input Select Register 1 (DSPI_PISR1)	32	R/W	0000_0000h	*Section 62.3.21*
0000_00E0	DSPI DSI Parallel Input Select Register 2 (DSPI_PISR2)	32	R/W	0000_0000h	*Section 62.3.22*
0000_00E4	DSPI DSI Parallel Input Select Register 3 (DSPI_PISR3)	32	R/W	0000_0000h	*Section 62.3.23*
0000_00E8	DSPI DSI Deserialized Data Interrupt Mask Register 0 (DSPI_DIMR0)	32	R/W	0000_0000h	*Section 62.3.24*
0000_00EC	DSPI DSI Deserialized Data Polarity Interrupt Register 0 (DSPI_DPIR0)	32	R/W	0000_0000h	*Section 62.3.25*
0000_00F0	DSPI DSI Serialization Data Register 1 (DSPI_SDR1)	32	R	0000_0000h	*Section 62.3.26*
0000_00F4	DSPI DSI Alternate Serialization Data Register 1 (DSPI_ASDR1)	32	R/W	0000_0000h	*Section 62.3.27*
0000_00F8	DSPI DSI Transmit Comparison Register 1 (DSPI_COMPR1)	32	R	0000_0000h	*Section 62.3.28*
0000_00FC	DSPI DSI Deserialization Data Register 1 (DSPI_DDR1)	32	R	0000_0000h	*Section 62.3.29*
0000_0100	DSPI DSI Serialization Source Select Register 1 (DSPI_SSR1)	32	R/W	0000_0000h	*Section 62.3.30*
0000_0104	DSPI DSI Parallel Input Select Register 4 (DSPI_PISR4)	32	R/W	0000_0000h	*Section 62.3.31*
0000_0108	DSPI DSI Parallel Input Select Register 5 (DSPI_PISR5)	32	R/W	0000_0000h	*Section 62.3.32*

Table 1593. DSPI memory map (continued)

Offset (hex)	Register name	Width (in bits)	Access	Reset value	Location
0000_010C	DSPI DSI Parallel Input Select Register 6 (DSPI_PISR6)	32	R/W	0000_0000h	*Section 62.3.33*
0000_0110	DSPI DSI Parallel Input Select Register 7 (DSPI_PISR7)	32	R/W	0000_0000h	*Section 62.3.34*
0000_0114	DSPI DSI Deserialized Data Interrupt Mask Register 1 (DSPI_DIMR1)	32	R/W	0000_0000h	*Section 62.3.35*
0000_0118	DSPI DSI Deserialized Data Polarity Interrupt Register 1 (DSPI_DPIR1)	32	R/W	0000_0000h	*Section 62.3.36*
0000_011C	DSPI Clock and Transfer Attributes Register Extended 0 (DSPI_CTARE0)	32	R/W	0000_0001h	*Section 62.3.37*
0000_0120	DSPI Clock and Transfer Attributes Register Extended 1 (DSPI_CTARE1)	32	R/W	0000_0001h	*Section 62.3.37*
0000_0124	DSPI Clock and Transfer Attributes Register Extended 2 (DSPI_CTARE2)	32	R/W	0000_0001h	*Section 62.3.37*
0000_0128	DSPI Clock and Transfer Attributes Register Extended 3 (DSPI_CTARE3)	32	R/W	0000_0001h	*Section 62.3.37*
0000_012C	DSPI Clock and Transfer Attributes Register Extended 4 (DSPI_CTARE4)	32	R/W	0000_0001h	*Section 62.3.37*
0000_0130	DSPI Clock and Transfer Attributes Register Extended 5 (DSPI_CTARE5)	32	R/W	0000_0001h	*Section 62.3.37*
0000_0134	DSPI Clock and Transfer Attributes Register Extended 6 (DSPI_CTARE6)	32	R/W	0000_0001h	*Section 62.3.37*
0000_0138	DSPI Clock and Transfer Attributes Register Extended 7 (DSPI_CTARE7)	32	R/W	0000_0001h	*Section 62.3.37*
0000_013C	DSPI Status Register Extended (DSPI_SREX)	32	R	0000_0000h	*Section 62.3.38*

62.3.1 DSPI Module Configuration Register (DSPI_MCR)

The DSPI_MCR contains bits to configure various attributes associated with DSPI operations. The HALT and MDIS bits can be changed at any time, but they only take effect on the next frame boundary. Only the HALT and MDIS bits in the MCR can be changed while the DSPI is in the Running state.

Address: 0x0000 Access: User read/write

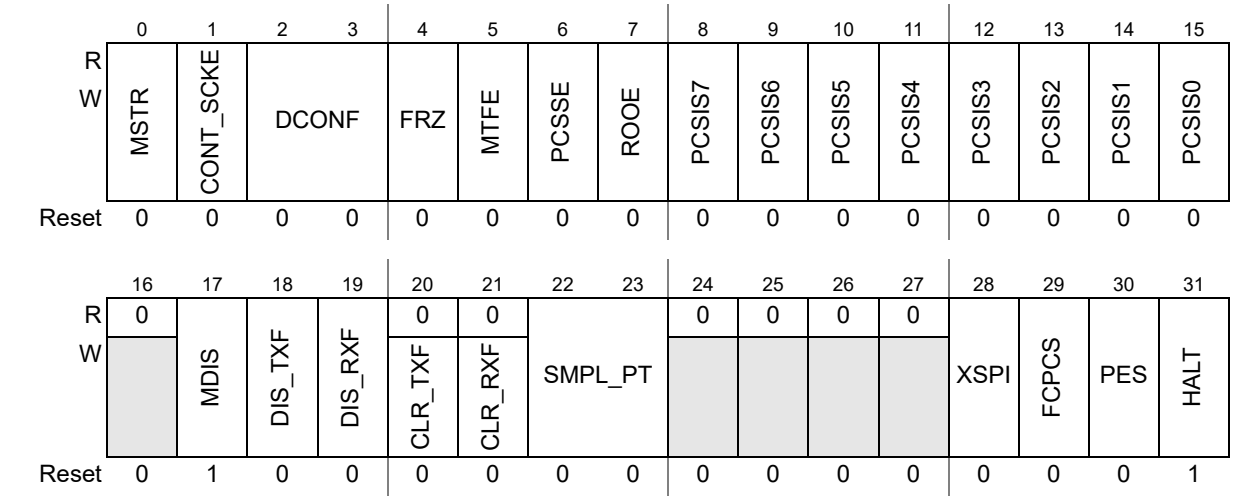


Figure 1590. DSPI Module Configuration Register (DSPI_MCR)

Table 1594. DSPI_MCR field descriptions

Field	Description
0 MSTR	Master/Slave Mode Select Configures the DSPI for either master mode or slave mode. 0 DSPI is in slave mode. 1 DSPI is in master mode.
1 CONT_SCKE	Continuous SCK Enable Enables the Serial Communication Clock (SCK) to run continuously. 0 Continuous SCK disabled. 1 Continuous SCK enabled.
2:3 DCONF	DSPI Configuration Selects the DSPI configuration. 00 SPI 01 DSI 10 CSI 11 Reserved
4 FRZ	Freeze Enables the DSPI transfers to be stopped on the next frame boundary when the device enters Debug mode. 0 Do not halt serial transfers in debug mode. 1 Halt serial transfers in debug mode.
5 MTFE	Modified Timing Format Enable Enables a modified transfer format to be used. MTFE should not be set in ITSB Mode as Microsecond Channel upstream operation is not supported. 0 Modified SPI transfer format disabled. 1 Modified SPI transfer format enabled.

Table 1594. DSPI_MCR field descriptions (continued)

Field	Description
6 PCSSE	Peripheral Chip Select Strobe Enable Enables the PCS[5]/ $\overline{\text{PCSS}}$ to operate as a PCS Strobe output signal. 0 PCS[5]/PCSS is used as the Peripheral Chip Select[5] signal (PCS[5]). 1 PCS[5]/PCSS is used as an active-low PCS Strobe signal (PCSS).
7 ROOE	Receive FIFO Overflow Overwrite Enable In the RX FIFO overflow condition, configures the DSPI to ignore the incoming serial data or overwrite existing data. If the RX FIFO is full and new data is received, the data from the transfer generating the overflow is ignored or shifted into the shift register. 0 Incoming data is ignored. 1 Incoming data is shifted into the shift register.
8:15 PCSIS[7:0]	Peripheral Chip Select x Inactive State Determines the inactive state of PCSx when DSPI is in Master Mode. This field has no effect when DSPI is in Slave Mode. The Slave Select input to DSPI in slave mode is always Active Low. 0 The inactive state of PCSx is low. 1 The inactive state of PCSx is high.
17 MDIS	Module Disable Allows the clock to be stopped to the non-memory-mapped logic in the DSPI, putting the DSPI in a software controlled power-saving state. The reset value of the MDIS bit is 1. MDIS should be set to ‘0’ in Slave Mode as a slave does not control master transactions. 0 Enable DSPI clocks. 1 Allow external logic to disable DSPI clocks.
18 DIS_TXF	Disable Transmit FIFO When the TX FIFO is disabled, the transmit part of the DSPI (TXFIFO and CMD FIFO) operates as a simplified double-buffered SPI. This bit can only be written when the MDIS bit is cleared. 0 Tx FIFO is enabled. 1 Tx FIFO is disabled.
19 DIS_RXF	Disable Receive FIFO When the RX FIFO is disabled, the receive part of the DSPI operates as a simplified double-buffered SPI. This bit can only be written when the MDIS bit is cleared. 0 Rx FIFO is enabled. 1 Rx FIFO is disabled.
20 CLR_TXF	Clear TX FIFO Flushes TX FIFO and CMD FIFO. Writing a ‘1’ to CLR_TXF clears the TX FIFO and CMD FIFO Counters. The CLR_TXF bit is always read as zero. 0 Do not clear the Tx FIFO and CMD FIFO counters. 1 Clears the Tx FIFO and CMD FIFO counters.

Table 1594. DSPI_MCR field descriptions (continued)

Field	Description
21 CLR_RXF	Clear RX FIFO Flushes RX FIFO. Writing a '1' to CLR_RXF clears the RX Counter. The CLR_RXF bit is always read as zero. 0 Do not clear the Rx FIFO counter. 1 Clear the Rx FIFO counter.
22:23 SMPL_PT	Sample Point Controls when the DSPI master samples SIN in Modified Transfer Format. This field is valid only when CPHA bit in CTAR is 0. 00 0 protocol clocks between SCK edge and SIN sample 01 1 protocol clock between SCK edge and SIN sample 10 2 protocol clocks between SCK edge and SIN sample 11 Reserved
28 XSPI	Extended SPI Mode Enables use of CTARE (Command and Transfer Attribute Register Extended) registers, which allow the user to send up to 32-bit SPI frames. Command Cycling is also enabled, allowing the user to send multiple Data Frames in a single Command Frame. When MCR[DIS_TXF] is asserted, Extended SPI Mode cannot be used to transmit SPI frames over 16 bits in size. 0 Normal SPI Mode. Up to 16-bit Frames. Command Cycling is not available. 1 Extended SPI Mode. Up to 32-bit SPI Frames. Command Cycling Enabled.
29 FCPCS	Fast Continuous PCS Mode. This bit enables the masking of "After SCK (t_{ASC})" and "PCS to SCK (t_{CSC})" delays when operating in Continuous PCS mode. The individual delay masks are selected via PUSHR[PE_MASC] and PUSHR{PP_MCSC]. The firmware should select appropriate masks when providing continuous frames via the PUSHR. This masking is not available if Continuous SCK mode is enabled. 0 Normal or Slow Continuous PCS mode. Masking of delays is disabled. 1 Fast Continuous PCS mode. Delays masked via control bits in PUSHR.
30 PES	Parity Error Stop Controls SPI operation when a parity error is detected in a received SPI frame. 0 SPI frame transmission continues. 1 SPI frame transmission stops.
31 HALT	Halt Starts and stops DSPI transfers. 0 Start transfers. 1 Stop transfers. **Note:** HALT bit starts/stops the internal state machine. All the other MCR bits must be set to their expected value before releasing HALT bit. If one MCR bit needs to be changed in running state, HALT bit must be set first.

62.3.2 Hardware Configuration Register (DSPI_HCR)

Hardware Configuration Register provides particular implementation details about the module, that is number of Receive and Transmit FIFO entries, number of CTAR registers and if DSI features are implemented in the module or not. It is read only register. A write to this register will generate a Transfer Error.

Note: The reset bits in the HCR are set by the configuration parameters.

Address: 0x0004 Access: User read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	DSI	PISR	DSI64	PISR_EX	XSPI	CTAR			TXFR				RXFR			
W																
Reset	0	0	0	0	0	1	1	1	0	0	1	1	0	0	1	1

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	CMDFR				MSTR_MODE	SLV_MODE	TSB_EN	ITSB_EN	0	0	0	0	0	0	0	0
W																
Reset	0	0	1	1	1	1	0	0	0	0	0	0	0	0	0	0

Figure 1591. Hardware Configuration Register (DSPI_HCR)

Table 1595. DSPI_HCR field descriptions

Field	Description
0 DSI	DSI features are implemented for the module. 0 DSI features are not implemented, DSI registers do not exist. 1 DSI features are implemented
1 PISR	PISR0-3 and parallel inputs frame positions selection logic (DSI Muxing Logic) are implemented for the module. 0 PISR0-3 registers are not implemented. 1 PISR0-3 registers are implemented.
2 DSI64	DSI features in 64 bit mode are implemented for the module. 0 DSI features for 64 bit mode are not implemented; related registers do not exist. 1 DSI features for 64 bit mode are implemented
3 PISR_EX	PISR4-7 and parallel inputs frame positions selection logic (DSI 64 bit Muxing Logic) are implemented for the module. 0 PISR4-7 registers are not implemented. 1 PISR4-7 registers are implemented.
4 XSPI	Extended SPI Mode feature is implemented for the module. 0 Extended SPI Mode is not implemented. 1 Extended SPI Mode is implemented.
5:7 CTAR	Maximum implemented CTAR register number.
8:11 TXFR	Maximum implemented TXFR register number.
12:15 RXFR	Maximum implemented RXFR register number.

Table 1595. DSPI_HCR field descriptions (continued)

Field	Description
16:19 CMDFR	Maximum implemented command FIFO number.
20 MSTR_MODE	Master Mode implementation is enabled
21 SLV_MODE	Slave Mode implementation is enabled
22 TSB_EN	TSB Mode implementation is enabled
23 ITSB_EN	Interleaved TSB Mode implementation is enabled

62.3.3 DSPI Transfer Count Register (DSPI_TCR)

DSPI_TCR has an SPI transfer counter to assist in queue management.

Do not write the TCR when the DSPI is in the Running state.

Address: 0x0008 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	SPI_TCNT															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

Figure 1592. DSPI Transfer Count Register (DSPI_TCR)

Table 1596. DSPI_TCR field descriptions

Field	Description
0:15 SPI_TCNT	SPI Transfer Counter The SPI_TCNT field increments every time the last bit of a SPI frame is transmitted. SPI_TCNT is reset to zero at the beginning of the frame when the CTCNT field is set in the executing SPI command. A value written to SPI_TCNT presets the counter to that value. The Transfer Counter wraps around; incrementing past 65535 resets the counter to zero.

62.3.4 DSPI Clock and Transfer Attributes Register (In Master Mode) (DSPI_CTAR*n*)

CTARs are used to define different transfer attributes:

- In master mode, the CTARs define combinations of transfer attributes such as frame size, clock phase and polarity, data bit ordering, baud rate, and various delays.
- In slave mode, a subset of the bitfields in CTAR0 and CTAR1 set the slave transfer attributes.

Do not write to the CTARs while the DSPI is in the Running state.

When the DSPI is configured as:

- SPI master – the CTAS field in the command portion of the TX FIFO entry selects which CTAR is used.
- SPI bus slave – the CTAR0 register is used.
- DSI master – the DSICTAS field in the DSPI DSI Configuration Register 0 (DSICR0) selects which CTAR is used.
- DSI bus slave – the CTAR1 register is used.

In CSI Configuration, the transfer attributes are based on whether the current frame is:

- SPI data – follow the protocol described for SPI Configuration,
- DSI data – follow the protocol described for DSI Configuration.

CSI Configuration is valid only in master mode.

TSB mode sets some limitations on transfer attributes:

- Clock phase is forced to be CPHA = 1 and the CPHA bit setting has no effect.
- PCS lines are driven at the driving edge of the SCK clock together with SOUT, so PCS assertion and negation delays control is unavailable and PCSSCK, PASC, CSSCK and ASC fields have no effect.
- Delay after transfer can be set from 1 to 64 serial clocks via PDT and DT fields.

Address: 0x000C + *n**0x4 (*n*=0 to 7) Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	DBR	FMSZ				CPOL	CPHA	LSBFE	PCSSCK		PASC		PDT		PBR	
Reset	0	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	CSSCK				ASC				DT				BR			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1593. DSPI Clock and Transfer Attributes Register (In Master Mode) (DSPI_CTAR*n*)

Table 1597. DSPI_CTAR*n* field descriptions

Field	Description
0 DBR	Double Baud Rate. (Master mode only) Doubles the baud rate of the Serial Communications Clock (SCK). It halves the Baud Rate division ratio, supporting faster frequencies and odd division ratios for the Serial Communications Clock (SCK). When DBR is set, the duty cycle of the Serial Communications Clock (SCK) depends on the value in the Baud Rate Prescaler and the Clock Phase bit as listed in *Table 1598*. Refer to the BR field description in this table for details on how to compute the baud rate. Refer to *Table 1598* for DSPI SCK duty cycle values. 0 The baud rate is computed normally with a 50/50 duty cycle. 1 The baud rate is doubled with the duty cycle depending on the Baud Rate Prescaler.
1:4 FMSZ	Frame Size The number of bits transferred per frame is equal to the FMSZ field value plus 1. The minimum valid value for the number of bits to be transmitted for any mode is 4.There is a constraint on the minimum allowable Frame size, which depends on the register Read/Write clock to the Protocol Clock Ratio. The minimum Frame size (FMSZ + 1) is decided by the following equation. Upper Ceiling must be applied for non-integer values. Min. Frame Size = ((4 x f_p) + (3 x f_r)) / (n x f_r) f_p = Protocol Clock Frequency f_r = Register Read/Write Clock Frequency n = Multiple of protocol clock required to get Baud Clock - Given by (PBR x BR) / (1 + DBR) The minimum frame size can never be less than 4. There is no constraint on the maximum programmable frame size. Note that 'f_p' can be equal to, less than or greater than 'f_r' purely based on user requirement. When the DSPI operates in TSB mode, the FMSZ field value plus 1 is equal to the data frame bit number, where control of the PCS assertion switches from the DSICR0 to the DSICR1 register. If TSB Mode is operated when DSI is used in 64-bit mode, then DSICR0[FMSZ4] also comes into picture while switching PCS assertion control.
5 CPOL	Clock Polarity. (Master and Slave mode) Selects the inactive state of the Serial Communications Clock (SCK). Devices must have identical clock polarities for successful communication between serial devices. When the Continuous Selection Format is selected, switching between clock polarities without stopping the DSPI can cause errors in the transfer due to the peripheral device interpreting the switch of clock polarity as a valid clock edge. 0 The inactive state value of SCK is low. 1 The inactive state value of SCK is high.

Table 1597. DSPI_CTAR*n* field descriptions (continued)

Field	Description
6 CPHA	Clock Phase or TSB mode. (Master and Slave mode) Selects which edge of SCK causes data to change and which edge causes data to be captured. Devices must have identical clock phase settings for successful communication between serial devices. In Continuous SCK or TSB modes, this bit is ignored and the transfers are done as if the CPHA bit is set to 1. 0 Data is captured on the leading edge of SCK and changed on the following edge. 1 Data is changed on the leading edge of SCK and captured on the following edge.
7 LSBFE	LSB First. Specifies whether the LSB or MSB of the frame is transferred first. In TSB mode, this bit should comply with the MSC specification. 0 Data is transferred MSB first. 1 Data is transferred LSB first.
8:9 PCSSCK	PCS to SCK Delay Prescaler. Selects the prescaler value for the delay between assertion of PCS and the first edge of the SCK. Refer to the CSSCK field description in this table for calculating the PCS to SCK Delay. Refer to *Section 62.5.5.2: PCS to SCK delay (tCSC)* for more details. In TSB mode, PCSSCK has no effect. 00 PCS to SCK Prescaler value is 1. 01 PCS to SCK Prescaler value is 3. 10 PCS to SCK Prescaler value is 5. 11 PCS to SCK Prescaler value is 7.
10:11 PASC	After SCK Delay Prescaler. Selects the prescaler value for the delay between the last edge of SCK and the negation of PCS. Refer to the ASC field description in this table for calculating the After SCK Delay. Refer to *Section 62.5.5.3: After SCK delay (tASC)* for more details. In TSB mode, PASC has no effect. 00 Delay after Transfer Prescaler value is 1. 01 Delay after Transfer Prescaler value is 3. 10 Delay after Transfer Prescaler value is 5. 11 Delay after Transfer Prescaler value is 7.
12:13 PDT	Delay after Transfer Prescaler. (Master mode only) Selects the prescaler value for the delay between the negation of the PCS signal at the end of a frame and the assertion of PCS at the beginning of the next frame. In TSB mode, the PDT field defines two MSB bits of the Delay after Transfer. Refer to the DT field description for calculating the Delay after Transfer. Refer to *Section 62.5.5.4: Delay after transfer (tDT)* for more details. 00 Delay after Transfer Prescaler value is 1. 01 Delay after Transfer Prescaler value is 3. 10 Delay after Transfer Prescaler value is 5. 11 Delay after Transfer Prescaler value is 7.

Table 1597. DSPI_CTAR*n* field descriptions (continued)

Field	Description
14:15 PBR	Baud Rate Prescaler. (Master mode only) Selects the prescaler value for the baud rate. The baud rate is the frequency of the SCK. The protocol clock is divided by the prescaler value before the baud rate selection takes place. Refer to the BR field description in this table for calculating the baud rate. 00 Baud Rate Prescaler value is 2. 01 Baud Rate Prescaler value is 3. 10 Baud Rate Prescaler value is 5. 11 Baud Rate Prescaler value is 7.
16:19 CSSCK	PCS to SCK Delay Scaler. (Master mode only) Selects the scaler value for the PCS to SCK delay. PCS to SCK Delay is the delay between the assertion of PCS and the first edge of the SCK. The delay is a multiple of the protocol clock period calculated by: $t_{CSC} = (1/f_P) \times PCSSCK \times CSSCK$ *Table 1599* lists the delay scaler values. Refer to *Section 62.5.5.2: PCS to SCK delay (tCSC)* for more details. In TSB mode, the field has no effect.
20:23 ASC	After SCK Delay Scaler. (Master mode only) Selects the scaler value for the After SCK Delay. The After SCK Delay is the delay between the last edge of SCK and the negation of PCS. The delay is a multiple of the protocol clock period calculated by: $t_{ASC} = (1/f_P) \times PASC \times ASC$ *Table 1599* lists the delay scaler values. Refer to *Section 62.5.5.3: After SCK delay (tASC)* for more details.
24:27 DT	Delay After Transfer Scaler (Master mode only) Selects the Delay after Transfer Scaler. The Delay after Transfer is the time between the negation of the PCS signal at the end of a frame and the assertion of PCS at the beginning of the next frame. In the Continuous Serial Communications Clock operation, the DT value is fixed to one SCK clock period. The Delay after Transfer is a multiple of the protocol clock period calculated by: $t_{DT} = (1/f_P) \times PDT \times DT$ *Table 1599* lists the delay scaler values. In the TSB mode, the Delay after Transfer is equal to a number formed by concatenation of the PDT and DT fields plus 1 of the SCK clock periods. Refer to *Section 62.5.5.2: PCS to SCK delay (tCSC)* for more details.
28:31 BR	Baud Rate Scaler. (Master mode only) Selects the scaler value for the baud rate. The prescaled protocol clock is divided by the Baud Rate Scaler to generate the frequency of the SCK. The baud rate is calculated by: SCK baud rate = $(f_P/PBR) \times [(1+DBR)/BR]$ *Table 1600* lists the baud rate scaler values.

Table 1598. DSPI SCK duty cycle

DBR	CPHA	PBR	SCK duty cycle
0	any	any	50/50
1	0	00	50/50
1	0	01	33/66
1	0	10	40/60
1	0	11	43/57
1	1	00	50/50
1	1	01	66/33
1	1	10	60/40
1	1	11	57/43

Table 1599. Delay scaler encoding

Field value	Delay scaler value
0000	2
0001	4
0010	8
0011	16
0100	32
0101	64
0110	128
0111	256
1000	512
1001	1024
1010	2048
1011	4096
1100	8192
1101	16384
1110	32768
1111	65536

Table 1600. DSPI baud rate scaler

CTAR*n*[BR]	Baud rate scaler value
0000	2
0001	4
0010	6

Table 1600. DSPI baud rate scaler (continued)

CTAR*n*[BR]	Baud rate scaler value
0011	8
0100	16
0101	32
0110	64
0111	128
1000	256
1001	512
1010	1024
1011	2048
1100	4096
1101	8192
1110	16384
1111	32768

62.3.5 DSPI Clock and Transfer Attributes Register (In Slave Mode) (DSPI_CTAR*n*_SLAVE)

When the DSPI is configured as:

- SPI master – the CTAS field in the command portion of the TX FIFO entry selects which CTAR is used.
- SPI bus slave – the CTAR0 register is used.
- DSI master – the DSICTAS field in the DSPI DSI Configuration Register 0 (DSICR0) selects which CTAR is used.
- DSI bus slave – the CTAR1 register is used.

In CSI Configuration, the transfer attributes are based on whether the current frame is:

- SPI data – follow the protocol described for SPI Configuration,
- DSI data – follow the protocol described for DSI Configuration.

CSI Configuration is valid only in master mode.

TSB mode sets some limitations on transfer attributes:

- Clock phase is forced to be CPHA = 1 and the CPHA bit setting has no effect.
- PCS lines are driven at the driving edge of the SCK clock together with SOUT, so PCS assertion and negation delays control is unavailable and PCSSCK, PASC, CSSCK and ASC fields have no effect.
- Delay after transfer can be set from 1 to 64 serial clocks via PDT and DT fields.

Address: 0x000C + *n**0x4 (*n*=0 to 1) Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	FMSZ					CPOL	CPHA	PE	PP	FMSZ5	0	0	0	0	0	0
W																
Reset	0	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0

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

Figure 1594. DSPI Clock and Transfer Attributes Register (In Slave Mode) (DSPI_CTAR*n*_SLAVE)

Table 1601. DSPI_CTAR*n*_SLAVE field descriptions

Field	Description
0:4 FMSZ	Frame Size The number of bits transferred per frame is equal to the concatenated field {CTAR*n*_SLAVE[FMSZ5], CTAR*n*_SLAVE[FMSZ]} value plus 1. The minimum valid FMSZ field value is 3 (when FMSZ6 is 0).
5 CPOL	Clock Polarity Selects the inactive state of the Serial Communications Clock (SCK). 0 The inactive state value of SCK is low. 1 The inactive state value of SCK is high.
6 CPHA	Clock Phase or TSB mode. (Master and Slave mode) Selects which edge of SCK causes data to change and which edge causes data to be captured. Devices must have identical clock phase settings for successful communication between serial devices. In Continuous SCK or TSB modes, the bit value is ignored and the transfers are done as if the CPHA bit is set to 1. 0 Data is captured on the leading edge of SCK and changed on the following edge. 1 Data is changed on the leading edge of SCK and captured on the following edge.
7 PE	Parity Enable Enables parity bit transmission and reception for the frame. 0 No parity bit included/checked. 1 Parity bit is transmitted instead of last data bit in frame, parity checked for received frame.
8 PP	Parity Polarity Controls polarity of the parity bit transmitted and checked. 0 Even Parity: the number of 1 bits in the transmitted frame is even. The SR[SPEF] bit is set if the number of 1 bits is odd in the received frame. 1 Odd Parity: the number of 1 bits in the transmitted frame is odd. The SR[SPEF] bit is set if the number of 1 bits is even in the received frame.
9 FMSZ5	MSB of Frame Size when DSI is used in 64-bit Mode The number of bits transferred per frame is equal to the concatenated field {CTAR*n*_SLAVE[FMSZ5], CTAR*n*_SLAVE[FMSZ]} value plus 1.

62.3.6 DSPI Status Register (DSPI_SR)

The status register contains DSPI status bits and interrupt/DMA request event flag bits.

Software can clear most of the flag bits in the SR by writing a '1' to them (w1c). Writing a 0 to a flag bit has no effect. This register may not be writable in module disable mode due to the use of power saving mechanisms.

Address: 0x002C Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	TCF	TXRXS	SPITCF	EOQF	TFUF	DSITCF	TFFF	BSYF	CMDTCF	DPEF	SPEF	DDIF	RFOF	TFIWF	RFDF	CMDFFF
W	w1c	w1c	w1c	w1c	w1c	w1c	w1c		w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	1

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	TXCTR				TXNXTPTR				RXCTR				POPNXTPTR			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1595. DSPI Status Register (DSPI_SR)

Table 1602. DSPI_SR field descriptions

Field	Description
0 TCF	Transfer Complete Flag. Indicates that all bits in a frame have been shifted out. 0 Transfer not complete. 1 Transfer complete.
1 TXRXS	TX and RX Status. Reflects the run status of the DSPI. 0 Transmit and receive operations are disabled (DSPI is in stopped state). 1 Transmit and receive operations are enabled (DSPI is in running state).
2 SPITCF	SPI Frame Transfer Complete Flag. Similar to TCF but only asserted on SPI frame transmission completion in CSI Mode This includes ITSB and 'TSB in CSI' modes, but not SPI or DSI Modes. 0 SPI frame transfer is not complete. 1 SPI frame transfer is complete.
3 EOQF	End of Queue Flag. Indicates that the last entry in a queue has been transmitted when the DSPI is in master mode. The EOQF bit is set when the TX FIFO entry has the EOQ bit set in the command halfword and the end of the transfer is reached. When the EOQF bit is set, the TXRXS bit is automatically cleared. 0 EOQ is not set in the executing command. 1 EOQ is set in the executing SPI command.

Table 1602. DSPI_SR field descriptions (continued)

Field	Description
4 TFUF	Transmit FIFO Underflow Flag. Indicates an underflow condition in the TX FIFO. Transmit underflow is detected only for DSPI blocks in slave mode and SPI configuration. TFUF is set when the TX FIFO of a DSPI operating in SPI slave mode is empty and an external SPI master initiates a transfer. 0 No Tx FIFO underflow. 1 Tx FIFO underflow has occurred.
5 DSITCF	DSI Frame Transfer Complete Flag. Similar to TCF but only asserted on DSI frame transmission completion in CSI Mode. This includes ITSB and 'TSB in CSI' modes, but not SPI or DSI Modes. 0 DSI frame transfer is not complete. 1 DSI frame transfer is complete.
6 TFFF	Transmit FIFO Fill Flag. Provides a method for the DSPI to request more entries to be added to the TX FIFO. The TFFF bit is set while the TX FIFO is not full. The TFFF bit can be cleared by writing '1' to it or by acknowledgement from the DMA controller to the TX FIFO full request. 0 Tx FIFO is full. 1 Tx FIFO is not full.
7 BSYF	Busy Flag. This bit is valid only when DSPI_MCR[XSPI] is enabled. Indicates that the current Command Frame is being used for transmitting multiple data frames. This bit is not set for the last Data Frame of a Cyclic command Transfer or when DSPI_CTARE[DTCP] = 1. Refer to *Section 62.5.2.5: Command First In First Out (CMD FIFO) buffering mechanism*. This bit is valid only when DSPI_MCR[XSPI] is enabled. 0 No Cyclic Command Transfer in Progress. 1 Cyclic Command Transfer is in progress. Current Data Frame is not the last data frame for current cyclic command transfer.
8 CMDTCF	Command Transfer Complete Flag. Indicates that the last Data frame for the current Cyclic Command has been transmitted. Hence this bit is set only for the last Data Frame of a Cyclic Command Transfer or when DSPI_CTARE[DTCP] = 1. CMDTCF remains set until it is cleared by writing a '1' to it. 0 Data Transfer by current Command not complete. 1 Data Transfer by current Command complete.
9 DPEF	DSI Parity Error Flag. Indicates that a DSI frame with parity error has been received. 0 No parity error. 1 Parity error has occurred.

Table 1602. DSPI_SR field descriptions (continued)

Field	Description
10 SPEF	SPI Parity Error Flag. Indicates that a SPI frame with parity error has been received. 0 No parity error. 1 Parity error has occurred.
11 DDIF	DSI Data Received with Active Bits. Indicates that a DSI frame has been received with bits selected by DIMR with active polarity, defined by the DPIR. 0 No DSI data with active bits was received. 1 DSI data with active bits was received.
12 RFOF	Receive FIFO Overflow Flag. Indicates an overflow condition in the RX FIFO. The bit is set when the RX FIFO and shift register are full and a transfer is initiated. 0 No Rx FIFO overflow. 1 Rx FIFO overflow has occurred.
13 TFIWF	Transmit FIFO Invalid Write Flag. Indicates Data Write on TX FIFO while CMD FIFO is empty. Without a Command, the Data entries present in TXFIFO are invalid. 0 No Invalid Data present in TX FIFO 1 Invalid Data present in TX FIFO since CMD FIFO is empty
14 RFDF	Receive FIFO Drain Flag. Provides a method for the DSPI to request that entries be removed from the RX FIFO. The bit is set while the RX FIFO is not empty. The RFDF bit can be cleared by writing 1 to it or by acknowledgement from the DMA controller when the RX FIFO is empty. 0 Rx FIFO is empty. 1 Rx FIFO is not empty.
15 CMDFFF	Command FIFO Fill Flag. Provides a method for the DSPI to request more entries to be added to the CMD FIFO. The CMDFFF bit is set while the CMD FIFO is not full. The CMDFFF is cleared by writing a ‘1’ to it or by acknowledgement from the DMA controller to the CMD FIFO full request. 0 CMD FIFO is full. 1 CMD FIFO is not full.
16:19 TXCTR	TX FIFO Counter. Indicates the number of valid entries in the TX FIFO. The TXCTR is incremented every time the data part of PUSHR is written. The TXCTR is decremented every time the SPI data is transferred to the shift register.
20:23 TXNXTPTR	Transmit Next Pointer. Indicates which TX FIFO Entry is transmitted during the next transfer. The TXNXTPTR field is updated every time SPI data is transferred from the TX FIFO to the shift register.

Table 1602. DSPI_SR field descriptions (continued)

Field	Description
24:27 RXCTR	RX FIFO Counter. Indicates the number of entries in the RX FIFO. The RXCTR is decremented every time the POPR is read. The RXCTR is incremented every time data is transferred from the shift register to the RX FIFO.
28:31 POPNXTPTR	Pop Next Pointer. Contains a pointer to the RX FIFO entry to be returned when the POPR is read. The POPNXTPTR is updated when the POPR is read.

62.3.7 DSPI DMA/Interrupt Request Select and Enable Register (DSPI_RSER)

The DSPI_RSER controls DMA and interrupt requests.

Do not write to the RSER while the DSPI is in the Running state.

Address: 0x0030 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	TCF_RE	CMDFFF_RE	SPITCF_RE	EOQF_RE	TFUF_RE	DSITCF_RE	TFFF_RE	TFFF_DIRS	CMDTCF_RE	DPEF_RE	SPEF_RE	DDIF_RE	RFOF_RE	TFIWF_RE	RFDF_RE	RFDF_DIRS
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	CMDFFF_DIRS	DDIF_DIRS	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1596. DSPI DMA/Interrupt Request Select and Enable Register (DSPI_RSER)

Table 1603. DSPI_RSER field descriptions

Field	Description
0 TCF_RE	Transmission Complete Request Enable. Enables TCF flag in the Status register to generate an interrupt request. 0 Disabled. 1 Enabled.
1 CMDFFF_RE	Command FIFO Fill Flag Request Enable. Enables the CMDFFF flag in the Status register to generate a request. DSPI_RSER[CMDFFF_DIRS] toggles either an interrupt or a DMA request. 0 Disabled. 1 Enabled.

Table 1603. DSPI_RSER field descriptions (continued)

Field	Description
2 SPITCF_RE	SPI Frame Transmission Complete Request Enable. Enables SPITCF flag in the Status register to generate an interrupt request. 0 Disabled. 1 Enabled.
3 EOQF_RE	DSPI Finished Request Enable Enables the EOQF flag in the status register to generate an interrupt request. 0 Disabled. 1 Enabled.
4 TFUF_RE	Transmit FIFO Underflow Request Enable Enables the TFUF flag in the status register to generate an interrupt request. 0 Disabled. 1 Enabled.
5 DSITCF_RE	DSI Frame Transmission Complete Request Enable. Enables DSITCF flag in the status register to generate an interrupt request. 0 Disabled. 1 Enabled.
6 TFFF_RE	Transmit FIFO Fill Request Enable Enables the TFFF flag in the status register to generate a request. DSPI_RSER[TFFF_DIRS] toggles between either an interrupt or a DMA request. 0 Disabled. 1 Enabled.
7 TFFF_DIRS	Transmit FIFO Fill DMA or Interrupt Request Select Selects between generating a DMA request or an interrupt request. DSPI_SR[TFFF] flag and DSPI_RSER[TFFF_RE] must be set. 0 Interrupt requests. 1 DMA requests.
8 CMDTCF_RE	Command Transmission Complete Request Enable. The CMDTCF_RE bit enables CMDTCF flag in the status register to generate an interrupt request. 0 Disabled. 1 Enabled.
9 DPEF_RE	DSI Parity Error Request Enable Enables the DPEF flag in the status register to generate an interrupt request. 0 Disabled. 1 Enabled.
10 SPEF_RE	SPI Parity Error Request Enable Enables the SPEF flag in the status register to generate an interrupt request. 0 Disabled. 1 Enabled.
11 DDIF_RE	DSI data received with active bits Request Enable Enables the DDIF flag in the status register to generate an interrupt requests. 0 Disabled. 1 Enabled.

Table 1603. DSPI_RSER field descriptions (continued)

Field	Description
12 RFOF_RE	Receive FIFO Overflow Request Enable Enables the RFOF flag in the status register to generate an interrupt request. 0 Disabled. 1 Enabled.
13 TFIWF_RE	Transmit FIFO Invalid Write Request Enable Enables the TFIWF flag in the status register to generate an interrupt request. 0 Disabled. 1 Enabled.
14 RFDF_RE	Receive FIFO Drain Request Enable Enables the RFDF flag in the status register to generate a request. DSPIRSER[RFDF_DIRS] toggles either an interrupt or a DMA request. 0 Disabled 1 Enabled
15 RFDF_DIRS	Receive FIFO Drain DMA or Interrupt Request Select. Selects between generating a DMA request or an interrupt request. DSPI_SR[RFDF] flag and DSPI_RSER[RFDF_RE] must be set. 0 Interrupt request. 1 DMA request.
16 CMDFFF_DIRS	Command FIFO FIll DMA or Interrupt Request Select. Selects between generating a DMA request or an interrupt request. DSPI_SR[CMDFFF] flag and DSPI_RSER[CMDFFF_RE] must be set. 0 Interrupt request. 1 DMA request.
17 DDIF_DIRS	DSI data received with active bits - DMA or Interrupt Request Select. Selects between generating a DMA request or an interrupt request. DSPI_SR[DDIF] flag and DSPI_RSER[DDIF_RE] must be set. 0 Interrupt request. 1 DMA request.

62.3.8 DSPI PUSH FIFO Register In Master Mode (DSPI_PUSHR)

PUSHR provides the means to write to the TX FIFO and CMD FIFO.

Data written to this register is transferred to:

- The TX FIFO for 8- or 16-bit writes to the Data field of PUSHR.
- The CMD FIFO for writes to the Command field of PUSHR.

In master mode, the register provides 16-bit command to the CMD FIFO and 16-bit data to the TX FIFO.

In slave mode, CMD FIFO is unused and the 16-bit Command field of PUSHR is reserved.

When Extended SPI Mode is not enabled (MCR[XSPI] = 0):

- TX FIFO and CMD FIFO must be filled simultaneously.
- Writes must be given to both Data and Command fields of PUSHR for every PUSHR operation.
- TX FIFO and CMD FIFO can be considered a single 32-bit FIFO.

When Extended SPI Mode is enabled (MCR[XSPI] = 1):

- TX FIFO and CMD FIFO can be written independently.
- A PUSHR Read Operation returns the topmost TX FIFO and CMD FIFO entries concatenated.

When DSPI Module is disabled, any writes to this register do not update the FIFO. Reads during Module disable mode return the last PUSHR write performed when Module was enabled.

Address: 0x0034 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	CONT	CTAS			EOQ	CTCNT	PE_MASC	PP_MCSC	PCS							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	TXDATA															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1597. DSPI PUSH FIFO Register In Master Mode (DSPI_PUSHR)

Table 1604. DSPI_PUSHR field descriptions

Field	Description
0 CONT	Continuous Peripheral Chip Select Enable. (SPI master mode) Continuous Selection Format enables the selected PCS signals to remain asserted between transfers. PUSHR[EOQ] and PUSHR[CONT] bits cannot be asserted together in any Command Frame as EOQ signifies the last frame in a queue and the Last frame in a Continuous Transfer must have PUSHR[CONT] as 0. 0 Return PCSn signals to their inactive state between transfers. 1 Keep PCSn signals asserted between transfers.
1:3 CTAS	Clock and Transfer Attributes Select. Selects which CTAR (and corresponding CTARE register) to use in master mode to specify the transfer attributes for the associated SPI frame. In SPI slave mode, CTAR0 is used. Refer to the Device Configuration chapter to determine how many CTARs this device has. Do not program a value in this field for a register that is not present. 000 CTAR0/CTARE0 001 CTAR1/CTARE1 010 CTAR2/CTARE2 011 CTAR3/CTARE3 100 CTAR4/CTARE4 101 CTAR5/CTARE5 110 CTAR6/CTARE6 111 CTAR7/CTARE7

Table 1604. DSPI_PUSHR field descriptions (continued)

Field	Description
4 EOQ	End Of Queue Host software uses this bit to signal to the DSPI that the current SPI transfer is the last in a queue. DSPI_SR[EOQF] is set at the end of the transfer. 0 The SPI data is not the last data to transfer. 1 The SPI data is the last data to transfer.
5 CTCNT	Clear Transfer Counter. Clears the SPI_TCNT field in the Transfer Count register. The SPI_TCNT field is cleared before the DSPI starts transmitting the current SPI frame. Hence the current SPI frame causes this counter to increment by 1. 0 Do not clear the TCR[SPI_TCNT] field. 1 Clear the TCR[SPI_TCNT] field.
6 PE_MASC	Parity Enable or Mask t_{ASC} delay in the current frame PE: This bit enables parity bit transmission and parity reception check for the SPI frame. MASC: The current frame has the “after SCK” delay masked if this bit is asserted. Refer to *Section 62.5.6.6: Fast Continuous Selection Format* for more details. This bit is used as Mask t_{ASC} in the Fast Continuous PCS mode when MCR[FCPCS] is set. 0 PE: No parity bit included/checked. MASC: t_{ASC} delay is not masked and the current frame has the after SCK delay. 1 PE: Parity bit is transmitted instead of the last data bit in the frame; parity is checked for the received frame. MASC: t_{ASC} delay is masked in the current frame.
7 PP_MCSC	Parity Polarity or Mask t_{CSC} delay in the next frame PP: controls the polarity of the parity bit transmitted and checked. MCSC: The next frame has the “PCS to SCK” delay masked if this bit is asserted. Refer to *Section 62.5.6.6: Fast Continuous Selection Format* for more details. This bit is used as Mask t_{CSC} in the Fast Continuous PCS mode when MCR[FCPCS] is set. 0 PP: Even Parity: the number of 1 bits in the transmitted frame is even. The SR[SPEF] bit is set if the number of 1 bits is odd in the received frame. MCSC: t_{CSC} delay is not masked and the next frame has the PCS to SCK delay. 1 PP: Odd Parity: the number of 1 bits in the transmitted frame is odd. The SR[SPEF] bit is set if the number of 1 bits is even in the received frame. MCSC: t_{CSC} delay is masked in the next frame.
8:15 PCS	Select which PCS signals are to be asserted for the transfer. Refer to the chip configuration chapter for the number of PCS signals used in this MCU. 0 Negate the PCS[x] signal. 1 Assert the PCS[x] signal.
16:31 TXDATA	Transmit Data Holds SPI data to be transferred according to the associated SPI command

62.3.9 DSPI PUSH FIFO Register In Slave Mode (DSPI_PUSHR_SLAVE)

PUSHR provides the means to write to the TX FIFO.

Data written to this register is transferred to:

- The TX FIFO for 8- or 16-bit writes to the Data field of PUSHR.

In master mode, the register provides 16-bit command to the CMD FIFO and 16-bit data to the TX FIFO.

In slave mode, CMD FIFO is unused and the 16-bit Command field of PUSHR is reserved.

In Slave mode, up to 32-bit SPI frames may be queued for transmission/reception by enabling MCR[XSPI] mode.

Address: 0x0034 Access: User read/write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	TXDATA															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1598. DSPI PUSH FIFO Register In Slave Mode (DSPI_PUSHR_SLAVE)

Table 1605. DSPI_PUSHR_SLAVE field descriptions

Field	Description
16:31 TXDATA	Transmit Data Holds SPI data to be transferred.

62.3.10 DSPI POP FIFO Register (DSPI_POPR)

POPR is used to read the RX FIFO.

8- or 16-bit read accesses to the POPR have the same effect on the RX FIFO as 32-bit read accesses. A write to this register generates a Transfer Error.

Address: 0x0038 Access: User read-only

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	RXDATA															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	RXDATA															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1599. DSPI POP FIFO Register (DSPI_POPR)

Table 1606. DSPI_POPR field descriptions

Field	Description
0:31 RXDATA	Received Data Contains the SPI data from the RX FIFO entry to which the Pop Next Data Pointer points.

62.3.11 DSPI Transmit FIFO Registers (DSPI_TXFR*n*)

TXFR*n* provide visibility into the TX FIFO for debugging purposes.

Each read-only register is an entry in the TX FIFO. Reading the TXFR*n* registers does not alter the state of the TX FIFO.

Reading DSPI_TXFR*n* registers is invalid for a DSPI Master when used in Extended SPI (XSPI) mode.

Address: 0x003C + *n**0x4 (*n* = 0 to 3) Access: User read-only

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	TXCMD_TXDATA															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	TXDATA															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1600. DSPI Transmit FIFO Registers (DSPI_TXFR*n*)

Table 1607. DSPI_TXFR*n* field descriptions

Field	Description
0:15 TXCMD_TXDATA	Transmit Command or Transmit Data In master mode, the TXCMD field sets the transfer attributes for the SPI data. In slave mode, the TXDATA contains 16 MSB bits of the SPI data to be shifted out.
16:31 TXDATA	Transmit Data Contains the SPI data to be shifted out.

62.3.12 DSPI Receive FIFO Registers (DSPI_RXFR*n*)

RXFR*n* provide visibility into the RX FIFO for debugging purposes.

Each read-only register is an entry in the RX FIFO. Reading the RXFR*n* registers does not alter the state of the RX FIFO.

MCR[MDIS] must be 0 when RXFR is read.

Address: 0x007C + *n**0x4 (*n* = 0 to 3) Access: User read-only

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	RXDATA															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	RXDATA															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1601. DSPI Receive FIFO Registers (DSPI_RXFR*n*)

Table 1608. DSPI_RXFR*n* field descriptions

Field	Description
0:31 RXDATA	Receive Data Contains the received SPI data.

62.3.13 DSPI DSI Configuration Register 0 (DSPI_DSICR0)

DSICR0 selects various attributes associated with DSI and CSI Configurations.

Do not write to the DSICR0 while the DSPI is in the Running state.

Address: 0x00BC Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	FMSZ4	0	0	0	0	0	0	FMSZ5	0	ITSB	TSBC	TXSS	TPOL (1)	0	CID
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	DCONT	DSICTAS			DMS	PES	PE	PP	DPCSx							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1602. DSPI DSI Configuration Register 0 (DSPI_DSICR0)

1. This bit is reserved if the device does not have a hardware trigger input signal (HT).

Table 1609. DSPI_DSICR0 field descriptions

Field	Description
1 FMSZ4	MSB of the frame size in master mode when DSI is used in 32-bit mode. If the bit is set, 16 is added to the frame size, as defined by the CTAR*n*[FMSZ] field. The CTAR*n* register is selected by the DSICTAS field.
8 FMSZ5	MSB of the frame size in master mode when DSI is used in 64-bit mode. If the bit is set, 32 is added to the frame size, as defined by the concatenation of {DSICR0[FMSZ4], CTAR*n*[FMSZ]} fields. The CTAR*n* register is selected by the DSICTAS field. Hence the final concatenated frame size using this bit is {DSICR0[FMSZ5], DSICR0[FMSZ4], CTAR*n*[FMSZ]}. This field is only valid when DSICR1[DSI64] is enabled.
10 ITSB	Interleaved TSB mode. Enables the Interleaved TSB mode of operation. When enabled, there is no priority among frames from the SPI/DSI. DSI frames are sent when either the previous transmission was an SPI frame or the TX_FIFO is empty. Frames are transmitted on every trigger whose source is selected by the TRG bit setting. ITSB mode requires DSICR0[TSBC] set and should be written only when MCR[HALT] is asserted. If ITSB bit is set without setting DSICR0[TSBC], DSPI operates in Normal Mode depending on MCR[DCONF] bit. Refer to *Section 62.5.10: Interleaved TSB (ITSB) Mode* for more details. 0 Disabled. 1 Enabled.
11 TSBC	Timed Serial Bus Configuration. Enables the Timed Serial Bus Configuration, which allows up to 64-bit data to be used. It also allows t_{DT} to be programmable. 0 Disabled. 1 Enabled.
12 TXSS	Transmit Data Source Select. Selects the source of data to be serialized. The source can be either: a) data from host software written to the DSPI DSI Alternate Serialization Data Register (ASDR1/0), or b) parallel input pin states latched into the DSPI DSI Serialization Data Register (SDR1/0). 0 SDR source. 1 ASDR source.
13 TPOL	Trigger Polarity Selects the active edge of the hardware trigger input signal (HT), initiating DSI frames transfer. 0 Falling edge initiates a transfer. 1 Rising edge initiates a transfer.
15 CID	Change In Data Transfer Enable When the CID bit is set, DSI frames are initiated when the current DSI data differs from the previous DSI data shifted out. Do not enable CID when DCONT is enabled. 0 Disabled. 1 Enabled.

Table 1609. DSPI_DSICR0 field descriptions (continued)

Field	Description
16 DCONT	DSI Continuous Peripheral Chip Select Enable (DSI Master mode only) Enables the PCS signals to remain asserted between transfers. When the TSBC bit is set, DCONT has no effect. Do not enable DCONT when CID is enabled. 0 PCS signals return to inactive. 1 PCS signals remain asserted.
17:19 DSICTAS	DSI Clock and Transfer Attributes Select (DSI Master mode only) Selects the CTAR which provides transfer attributes for DSI frames. In DSI slave mode, CTAR1 is always selected.
20 DMS	Data Match Stop Stops DSI frame transmissions if DDIF flag is set in the status register. 0 Disabled. 1 Enabled.
21 PES	Parity Error Stop Stops DSI operation if a parity error occurred in the received DSI frame. 0 Disabled. 1 Enabled.
22 PE	Parity Enable Enables parity bit transmission and parity reception check for the DSI frames. 0 No parity bit included/checked. 1 Parity bit is transmitted instead of the last data bit in the frame; parity is checked for the received frame.
23 PP	Parity Polarity Controls the polarity of the parity bit transmitted and checked. 0 Even Parity: the number of '1' bits in the transmitted frame is even. The SR[DPEF] bit is set if in the received frame number of '1' bits is odd. 1 Odd Parity: the number of 1 bits in the transmitted frame is odd. The SR[DPEF] bit is set if in the received frame number of 1 bits is even.
24:31 DPCSx	DSI Peripheral Chip Select 0–7 Selects which of the PCS signals to assert during a DSI master mode transfer. 0 Negate PCS[x]. 1 Assert PCS[x].

62.3.14 DSPI DSI Serialization Data Register 0 (DSPI_SDR0)

Read-only SDR0 contains the states of the 32 LSB parallel input signals.

The states of these signals are latched into the SDR0 on the rising edge of every protocol clock.

When the TXSS bit in the DSICR0 is cleared, the data in the SDR0 is used as the source of the DSI frames.

When DSICR1[DSI64E] is set, SDR1 is the MSB half and SDR0 is the LSB half of the 64-bit SDR.

Address: 0x00C0 Access: User read-only

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	SER_DATA															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	SER_DATA															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1603. DSPI DSI Serialization Data Register 0 (DSPI_SDR0)

Table 1610. DSPI_SDR0 field descriptions

Field	Description
0:31 SER_DATA	Serialized Data Contains the signal states of the 32 LSB parallel input signals.

62.3.15 DSPI DSI Alternate Serialization Data Register 0 (DSPI_ASDR0)

ASDR0 is used by host software to write the 32 LSB of the data to be serialized.

When the TXSS bit in the DSICR0 is set, the data in the ASDR0 is the source of the DSI frames.

Writes to the ASDR0 take effect on the next frame boundary.

When DSICR1[DSI64E] is set, ASDR1 is the MSB half and ASDR0 is the LSB half of the 64-bit ASDR.

Address: 0x00C4 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	ASER_DATA															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	ASER_DATA															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1604. DSPI DSI Alternate Serialization Data Register 0 (DSPI_ASDR0)

Table 1611. DSPI_ASDR0 field descriptions

Field	Description
0:31 ASER_DATA	Alternate Serialized Data Holds the alternate 32 LSB of data to be serialized.

62.3.16 DSPI DSI Transmit Comparison Register 0 (DSPI_COMPR0)

Read-only COMPR0 holds a copy of the 32 LSB of last transmitted DSI data.

DSI data is transferred to this register as it is loaded into the TX Shift Register.

When DSICR1[DSI64E] is set, COMPR1 is the MSB half and COMPR0 is the LSB half of the 64-bit COMPR.

Address: 0x00C8 Access: User read-only

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	COMP_DATA															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	COMP_DATA															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1605. DSPI DSI Transmit Comparison Register 0 (DSPI_COMPR0)

Table 1612. DSPI_COMPR0 field descriptions

Field	Description
0:31 COMP_DATA	Compare Data Holds the 32 LSB of last serialized DSI data.

62.3.17 DSPI DSI Deserialization Data Register 0 (DSPI_DDR0)

Read-only DDR0 holds the 32 LSB signal states for the parallel output signals.

Host software can read data from incoming DSI frames.

When DSICR1[DSI64E] is set, DDR1 is the MSB half and DDR0 is the LSB half of the 64-bit DDR.

Address: 0x00CC Access: User read-only

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	DESER_DATA															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	DESER_DATA															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1606. DSPI DSI Deserialization Data Register 0 (DSPI_DDR0)

Table 1613. DSPI_DDR0 field descriptions

Field	Description
0:31 DESER_DATA	Deserialized Data Holds 32 LSB of deserialized data presented as signal states to the parallel output signals.

62.3.18 DSPI DSI Configuration Register 1 (DSPI_DSICR1)

DSICR1 selects various attributes associated with TSB (and ITSB) Configuration.

Do not write to the DSICR1 while the DSPI is in the Running state.

Address: 0x00D0 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	TSBCNT[5:0]						0	0	0	0	0	DSI64E	DSE1	DSE0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	TRGPRD								DPCS1_x							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1607. DSPI DSI Configuration Register 1 (DSPI_DSICR1)

Table 1614. DSPI_DSICR1 field descriptions

Field	Description
2:7 TSBCNT[5:0]	Timed Serial Bus Operation Count TSBCNT defines the length of the data frame When TSBC is set. The TSBCNT field valid value is from 3 to 63. The TSBCNT field selects the number of data bits to be shifted out during a transfer in TSB mode. Number of data bits in data frame = TSBCNT + 1.
13 DSI64E	DSI 64-bit Mode Enable Allows serialization and deserialization of Data frames up to 64 bits in size. The extended registers DSPI_SDR1, DSPI_ASDR1, DSPI_COMPR1, DSPI_DDR1, DSPI_SSR1, DSPI_PISR4-7, DSPI_DIMR1, DSPI_DPIR1 are valid in this mode. 0 Disabled. DSI Mode operates in the default 32-bit configuration 1 Enabled. DSI Mode operates in 64-bit configuration

Table 1614. DSPI_DSICR1 field descriptions (continued)

Field	Description
14 DSE1	Data Select Enable 1 When DSICR0[TBSC] bit is set, the DSE1 bit enables insertion of the zero bit (Data Select) in the middle of the data frame. The insertion bit position is defined by the FMSZ field of CTAR*n* register, selected by the DSICR0[DSICTAS] field. If DSICR1[DSI64] is enabled, then the insertion bit position is defined by concatenation of {DSICR0[FMSZ4], CTAR[FMSZ]} fields. The TSBCNT field value must be greater than the FMSZ field value plus one for proper operation of the DSE1 bit. 0 No Zero bit is inserted in the middle of the data frame. 1 Zero bit is inserted at the middle of the data frame. Total number of bits in the data frame is increased by 1.
15 DSE0	Data Select Enable 0 When DSICR0[TSBC] bit is set, the DSE0 bit controls insertion of the zero bit (Data Select) in the beginning of the Data frame. 0 No Zero bit is inserted in the beginning of the frame. 1 Zero bit is inserted at the beginning of the data frame. Total number of bits in the data frame is increased by 1.
16:23 TRGPRD	Internal Trigger Period for the ITSB mode. When DSICR0[TRG] bit is cleared and ITSB Mode is enabled, this field determines the trigger period for the internal trigger in number of baud rate clock cycles. The trigger period should be calculated as: Trigger Period = 64 bits (DSI frame maximum size) + 2 selection bits (maximum) + t_{PF} (passive frame time) When DSICR1[DSI64E] is disabled, DSI frame maximum size is 32. Refer CTAR[FMSZ] field for minimum frame size allowed. Refer to Interleaved TSB (ITSB) Mode for more details. 0 – 4 These values are invalid since minimum Command frame size is 5. (that is 1 selection bit + minimum frame size which is 4) 5 – 255 Number of baud rate clock cycles to be used as the trigger period. The trigger period implemented is equal to TRGPRD + 1.
24:31 DPCS1_x	DSI Peripheral Chip Select 0–7 These bits define the PCSs to assert for the second part of the DSI frame when operating in TSB (and ITSB) configuration with dual receiver. The DPCS1 bits select which of the PCS signals to assert during the second part of the DSI frame. The DPCS1 bits control the assertions of the PCS signals only in TSB and ITSB mode. 0 Negate PCS[x]. 1 Assert PCS[x].

62.3.19 DSPI DSI Serialization Source Select Register 0 (DSPI_SSR0)

SSR0 is used to create a combined frame for transmission that contains bits from ASDR0 and from SDR0.

Each bit in the SSR0 register selects a corresponding bit to be serialized.

When DSICR0[TXSS] is set, the SSR0 register value has no effect.

When DSICR1[DSI64E] is set, SSR1 is the MSB half and SSR0 is the LSB half of the 64-bit SSR.

Address: 0x00D4 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	SS															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	SS															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1608. DSPI DSI Serialization Source Select Register 0 (DSPI_SSR0)

Table 1615. DSPI_SSR0 field descriptions

Field	Description
0:31 SS	Source Select Select the serialization source for the 32 LSB of DSI frame. Each SS bit selects data for a corresponded bit in the transmitted frame. 0 The bit in the transmitted frame is taken from the parallel input pin. 1 The bit in the transmitted frame is taken from the ASDR0 register

62.3.20 DSPI DSI Parallel Input Select Register 0 (DSPI_PISR0)

PISR0–3 are used to select each data bit for the transmitted frame from 16 parallel input pins. Each Input Pin Select (IPS) field controls one bit in the transmitted frame. Each register contains control fields for 8 bits in the frame. The select field value is defined as a 4-bit signed integer number. The selected parallel input pin number is defined as a difference between the sum of the field number and the field value.

For example, if IPS16 is equal to the binary number 1111 (minus 1 decimal), then bit 16 in the frame is taken from parallel input pin number 15. When the IPS0 is equal to -1, then bit 0 in the frame is taken from parallel input 31 (bit 0 is taken from parallel input 63 if DSICR1[DSI64E] is enabled). When the IPS0 is equal to +1, then bit 0 in the frame is taken from parallel input 1 and so on.

Note that PISR0–3 only preselect the parallel input pins. The final selection to the transmitted frame is made with the DSPI_SSR1/0 register bits or the DSICR0[TXSS] bit.

When DSICR1[DSI64E] is enabled, PISR4-7 registers are used to handle the extra 32 bits (32 MSB pins of the 64 parallel input pins).

Address: 0x00D8 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	IPS7				IPS6				IPS5				IPS4			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	IPS3				IPS2				IPS1				IPS0			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1609. DSPI DSI Parallel Input Select Register 0 (DSPI_PISR0)

Table 1616. DSPI_PISR0 field descriptions

Field	Description
0:3 IPS7	Input Pin Select 7 Selects the parallel input pin for transmitted frame bit 7.
4:7 IPS6	Input Pin Select 6 Selects the parallel input pin for transmitted frame bit 6.
8:11 IPS5	Input Pin Select 5 Selects the parallel input pin for transmitted frame bit 5.
12:15 IPS4	Input Pin Select 5 Selects the parallel input pin for transmitted frame bit 4.
16:19 IPS3	Input Pin Select 3 Selects the parallel input pin for transmitted frame bit 3.
20:23 IPS2	Input Pin Select 2 Selects the parallel input pin for transmitted frame bit 2.
24:27 IPS1	Input Pin Select 1 Selects the parallel input pin for transmitted frame bit 1.
28:31 IPS0	Input Pin Select 0 Selects the parallel input pin for transmitted frame bit 0.

62.3.21 DSPI DSI Parallel Input Select Register 1 (DSPI_PISR1)

Address: Base + 0x00DC Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	IPS15				IPS14				IPS13				IPS12			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	IPS11				IPS10				IPS9				IPS8			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1610. DSPI DSI Parallel Input Select Register 1 (DSPI_PISR1)

Table 1617. DSPI_PISR1 field descriptions

Field	Description
0:3 IPS15	Input Pin Select 15 Selects the parallel input pin for transmitted frame bit 15.
4:7 IPS14	Input Pin Select 14 Selects the parallel input pin for transmitted frame bit 14.
8:11 IPS13	Input Pin Select 13 Selects the parallel input pin for transmitted frame bit 13.
12:15 IPS12	Input Pin Select 12 Selects the parallel input pin for transmitted frame bit 12.
16:19 IPS11	Input Pin Select 11 Selects the parallel input pin for transmitted frame bit 11.
20:23 IPS10	Input Pin Select 10 Selects the parallel input pin for transmitted frame bit 10.
24:27 IPS9	Input Pin Select 9 Selects the parallel input pin for transmitted frame bit 9.
28:31 IPS8	Input Pin Select 8 Selects the parallel input pin for transmitted frame bit 8.

62.3.22 DSPI DSI Parallel Input Select Register 2 (DSPI_PISR2)

Address: 0x00E0 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	IPS23				IPS22				IPS21				IPS20			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	IPS19				IPS18				IPS17				IPS16			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1611. DSPI DSI Parallel Input Select Register 2 (DSPI_PISR2)

Table 1618. DSPI_PISR2 field descriptions

Field	Description
0:3 IPS23	Input Pin Select 23 Selects the parallel input pin for transmitted frame bit 23.
4:7 IPS22	Input Pin Select 22 Selects the parallel input pin for transmitted frame bit 22.
8:11 IPS21	Input Pin Select 21 Selects the parallel input pin for transmitted frame bit 21.
12:15 IPS20	Input Pin Select 20 Selects the parallel input pin for transmitted frame bit 20.
16:19 IPS19	Input Pin Select 19 Selects the parallel input pin for transmitted frame bit 19.
20:23 IPS18	Input Pin Select 18 Selects the parallel input pin for transmitted frame bit 18.
24:27 IPS17	Input Pin Select 17 Selects the parallel input pin for transmitted frame bit 17.
28:31 IPS16	Input Pin Select 16 Selects the parallel input pin for transmitted frame bit 16.

62.3.23 DSPI DSI Parallel Input Select Register 3 (DSPI_PISR3)

Address: 0x00E4 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	IPS31				IPS30				IPS29				IPS28			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	IPS27				IPS26				IPS25				IPS24			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1612. DSPI DSI Parallel Input Select Register 3 (DSPI_PISR3)

Table 1619. DSPI_PISR3 field descriptions

Field	Description
0:3 IPS31	Input Pin Select 31 Selects the parallel input pin for transmitted frame bit 31.
4:7 IPS30	Input Pin Select 30 Selects the parallel input pin for transmitted frame bit 30.
8:11 IPS29	Input Pin Select 29 Selects the parallel input pin for transmitted frame bit 29.
12:15 IPS28	Input Pin Select 28 Selects the parallel input pin for transmitted frame bit 28.
16:19 IPS27	Input Pin Select 27 Selects the parallel input pin for transmitted frame bit 27.
20:23 IPS26	Input Pin Select 26 Selects the parallel input pin for transmitted frame bit 26.
24:27 IPS25	Input Pin Select 25 Selects the parallel input pin for transmitted frame bit 25.
28:31 IPS24	Input Pin Select 24 Selects the parallel input pin for transmitted frame bit 24.

62.3.24 DSPI DSI Deserialized Data Interrupt Mask Register 0 (DSPI_DIMR0)

DIMR0 selects bits in the 32 LSB of received DSI frame to be checked to generate the DDI interrupt.

When DSICR1[DSI64E] is set, DIMR1 is the MSB half and DIMR0 is the LSB half of the 64-bit DIMR.

Address: 0x00E8 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	MASK															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	MASK															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1613. DSPI DSI Deserialized Data Interrupt Mask Register 0 (DSPI_DIMR0)

Table 1620. DSPI_DIMR0 field descriptions

Field	Description
0:31 MASK	Mask 0 The bit in the received DSI frame does not produce a DDI interrupt. 1 The bit in the received DSI frame produces a DDI interrupt if the data bit matches the configured polarity.

62.3.25 DSPI DSI Deserialized Data Polarity Interrupt Register 0 (DSPI_DPIR0)

DPIR0 defines which data bit value in the 32 LSB of received DSI frame generates the DDI interrupt.

When DSICR1[DSI64E] is set, DPIR1 is the MSB half and DPIR0 is the LSB half of the 64-bit DPIR.

Address: 0x00EC Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	DP															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	DP															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1614. DSPI DSI Deserialized Data Polarity Interrupt Register 0 (DSPI_DPIR0)

Table 1621. DSPI_DPIR0 field descriptions

Field	Description
0:31 DP	Data Polarity 0 If the received bit is 0, the SR[DDIF] bit is set. 1 If the received bit is 1, the SR[DDIF] bit is set.

62.3.26 DSPI DSI Serialization Data Register 1 (DSPI_SDR1)

Read-only SDR1 contains the states of the 32 MSB parallel input signals.

The states of these signals are latched into the SDR1 on the rising edge of every protocol clock.

When DSICR0[TXSS] is cleared, the data in SDR1 and SDR0 is the source of the DSI frames.

The concatenation of {DSPI_SDR1, DSPI_SDR0} provides the 64-bit data to be serialized.

This register is valid only when DSICR1[DSI64E] is set.

Address: 0x00F0 Access: User read-only

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	SER_DATA															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	SER_DATA															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1615. DSPI DSI Serialization Data Register 1 (DSPI_SDR1)

Table 1622. DSPI_SDR1 field descriptions

Field	Description
0:31 SER_DATA	Serialized Data

62.3.27 DSPI DSI Alternate Serialization Data Register 1 (DSPI_ASDR1)

ASDR1 is used by host software to write the 32 MSB of the 64-bit data to be serialized.

When DSICR0[TXSS] is set, the data in ASDR1 and ASDR0 is the source of the DSI frames.

Writes to the ASDR1 take effect on the next frame boundary.

The concatenation of {DSPI_ASDR1, DSPI_ASDR0} provides the 64-bit alternate serialization data.

This register is valid only when DSICR1[DSI64E] is enabled.

Address: 0x00F4 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	ASER_DATA															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	ASER_DATA															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1616. DSPI DSI Alternate Serialization Data Register 1 (DSPI_ASDR1)

Table 1623. DSPI_ASDR1 field descriptions

Field	Description
0:31 ASER_DATA	Alternate Serialized Data

62.3.28 DSPI DSI Transmit Comparison Register 1 (DSPI_COMPR1)

Read-only COMPR1 holds a copy of the 32 MSB of the last transmitted 64-bit DSI data.

The upper 32 bits of the 64-bit DSI data is transferred to this register as it is loaded into the TX Shift register.

The concatenation of {DSPI_COMPR1, DSPI_COMPR0} provides the 64-bit transmit comparison data.

This register is valid only when DSICR1[DSI64E] is enabled.

Address: 0x00F8 Access: User read-only

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	COMP_DATA															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	COMP_DATA															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1617. DSPI DSI Transmit Comparison Register 1 (DSPI_COMPR1)

Table 1624. DSPI_COMPR1 field descriptions

Field	Description
0:31 COMP_DATA	Compare Data

62.3.29 DSPI DSI Deserialization Data Register 1 (DSPI_DDR1)

Read-only DDR1 holds the 32 MSB signal states for the 64 parallel output signals.

Host software can read data from incoming DSI frames.

The concatenation of {DSPI_DDR1, DSPI_DDR0} provides the 64-bit Deserialized data.

This register is valid only when DSICR1[DSI64E] is enabled.

Address: 0x00FC Access: User read-only

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	DESER_DATA															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	DESER_DATA															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1618. DSPI DSI Deserialization Data Register 1 (DSPI_DDR1)

Table 1625. DSPI_DDR1 field descriptions

Field	Description
0:31 DESER_DATA	Deserialized Data

62.3.30 DSPI DSI Serialization Source Select Register 1 (DSPI_SSR1)

SSR1 is used to create a combined frame for transmission that contains bits from ASDR1 and from SDR1.

Each bit in the SSR1 register selects a corresponding bit to be serialized.

When DSICR0[TXSS] is set, the SSR1 register value has no effect.

The concatenation of {DSPI_SSR1, DSPI_SSR0} provides the 64-bit Serialization Source Select value.

This register is valid only when DSICR1[DSI64E] is enabled.

Address: 0x0100 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	SS															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	SS															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1619. DSPI DSI Serialization Source Select Register (DSPI_SSR)

Table 1626. DSPI_SSR field descriptions

Field	Description
0:31 SS	Source Select 0 The bit in the transmitted frame is taken from the parallel input pin. 1 The bit in the transmitted frame is taken from the ASDR1 register

62.3.31 DSPI DSI Parallel Input Select Register 4 (DSPI_PISR4)

PISR4–7 are used to select each data bit for the transmitted frame from 16 parallel input pins. Each Input Pin Select (IPS) field controls one bit in the transmitted frame. Each register contains control fields for 8 bits in the frame. The select field value is defined as a 4-bit signed integer number. The selected parallel input pin number is defined as a difference between the sum of the field number and the field value.

For example, if IPS16 is equal to the binary number 1111 (minus 1 decimal), then bit 16 in the frame is taken from parallel input pin number 15. When the IPS0 is equal to -1, then bit 0 in the frame is taken from parallel input 31 (bit 0 is taken from parallel input 63 when DSICR1[DSI64E] is enabled). When the IPS0 is equal to +1, then bit 0 in the frame is taken from parallel input 1 and so on.

Note that PISR0–7 only preselect the parallel input pins. The final selection to the transmitted frame is made with the SSR0 and SSR1 register bits or the DSICR0[TXSS] bit.

When 64-bit DSI mode is enabled, PISR4-7 registers are used to handle the extra 32 bits (32 MSB pins of the 64 parallel input pins). If DSICR1[DSI64E] is disabled, then DSPI_PISR4-7 registers are invalid.

Address: 0x0104 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	IPS39				IPS38				IPS37				IPS36			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	IPS35				IPS34				IPS33				IPS32			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1620. DSPI DSI Parallel Input Select Register 4 (DSPI_PISR4)

Table 1627. DSPI_PISR4 field descriptions

Field	Description
0:3 IPS39	Input Pin Select 39 Selects the parallel input pin for transmitted frame bit 39.
4:7 IPS38	Input Pin Select 38 Selects the parallel input pin for transmitted frame bit 38.
8:11 IPS37	Input Pin Select 37 Selects the parallel input pin for transmitted frame bit 37.
12:15 IPS36	Input Pin Select 36 Selects the parallel input pin for transmitted frame bit 36.
16:19 IPS35	Input Pin Select 35 Selects the parallel input pin for transmitted frame bit 35.
20:23 IPS34	Input Pin Select 34 Selects the parallel input pin for transmitted frame bit 34.
24:27 IPS33	Input Pin Select 33 Selects the parallel input pin for transmitted frame bit 33.
28:31 IPS32	Input Pin Select 32 Selects the parallel input pin for transmitted frame bit 32.

62.3.32 DSPI DSI Parallel Input Select Register 5 (DSPI_PISR5)

Address: 0x0108 Access: User read/write

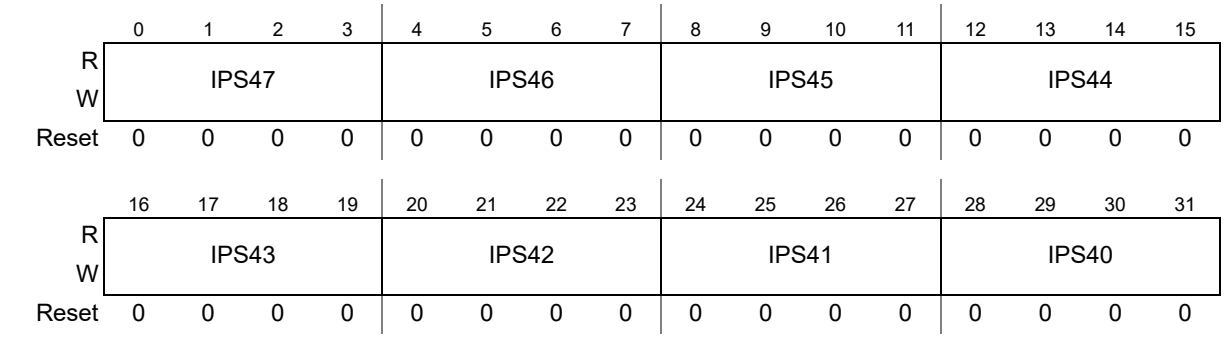


Figure 1621. DSPI DSI Parallel Input Select Register 5 (DSPI_PISR5)

Table 1628. DSPI_PISR5 field descriptions

Field	Description
0:3 IPS47	Input Pin Select 47 Selects the parallel input pin for transmitted frame bit 47.
4:7 IPS46	Input Pin Select 46 Selects the parallel input pin for transmitted frame bit 46.
8:11 IPS45	Input Pin Select 45 Selects the parallel input pin for transmitted frame bit 45.
12:15 IPS44	Input Pin Select 44 Selects the parallel input pin for transmitted frame bit 44.
16:19 IPS43	Input Pin Select 43 Selects the parallel input pin for transmitted frame bit 43.
20:23 IPS42	Input Pin Select 42 Selects the parallel input pin for transmitted frame bit 42.
24:27 IPS41	Input Pin Select 41 Selects the parallel input pin for transmitted frame bit 41.
28:31 IPS40	Input Pin Select 40 Selects the parallel input pin for transmitted frame bit 40.

62.3.33 DSPI DSI Parallel Input Select Register 6 (DSPI_PISR6)

Address: 0x010C Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	IPS55				IPS54				IPS53				IPS52			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	IPS51				IPS50				IPS49				IPS48			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1622. DSPI DSI Parallel Input Select Register 6 (DSPI_PISR6)

Table 1629. DSPI_PISR6 field descriptions

Field	Description
0:3 IPS55	Input Pin Select 55 Selects the parallel input pin for transmitted frame bit 55.
4:7 IPS54	Input Pin Select 54 Selects the parallel input pin for transmitted frame bit 54.
8:11 IPS53	Input Pin Select 53 Selects the parallel input pin for transmitted frame bit 53.
12:15 IPS52	Input Pin Select 52 Selects the parallel input pin for transmitted frame bit 52.
16:19 IPS51	Input Pin Select 51 Selects the parallel input pin for transmitted frame bit 51.
20:23 IPS50	Input Pin Select 50 Selects the parallel input pin for transmitted frame bit 50.
24:27 IPS49	Input Pin Select 49 Selects the parallel input pin for transmitted frame bit 49.
28:31 IPS48	Input Pin Select 48 Selects the parallel input pin for transmitted frame bit 48.

62.3.34 DSPI DSI Parallel Input Select Register 7 (DSPI_PISR7)

Address: 0x0110 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	IPS63				IPS62				IPS61				IPS60			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	IPS59				IPS58				IPS57				IPS56			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1623. DSPI DSI Parallel Input Select Register 7 (DSPI_PISR7)

Table 1630. DSPI_PISR7 field descriptions

Field	Description
0:3 IPS63	Input Pin Select 63 Selects the parallel input pin for transmitted frame bit 63
4:7 IPS62	Input Pin Select 62 Selects the parallel input pin for transmitted frame bit 62
8:11 IPS61	Input Pin Select 61 Selects the parallel input pin for transmitted frame bit 61
12:15 IPS60	Input Pin Select 60 Selects the parallel input pin for transmitted frame bit 60
16:19 IPS59	Input Pin Select 59 Selects the parallel input pin for transmitted frame bit 59
20:23 IPS58	Input Pin Select 58 Selects the parallel input pin for transmitted frame bit 58
24:27 IPS57	Input Pin Select 57 Selects the parallel input pin for transmitted frame bit 57
28:31 IPS56	Input Pin Select 56 Selects the parallel input pin for transmitted frame bit 56

62.3.35 DSPI DSI Deserialized Data Interrupt Mask Register 1 (DSPI_DIMR1)

DIMR1 selects bits from the 32 MSB in the received 64-bit DSI frame to be checked to generate the DDI interrupt.

The concatenation of {DSPI_DIMR1, DSPI_DIMR0} provides the 64-bit Deserialized Data Interrupt Mask value.

This register is valid only when DSICR1[DSI64E] is enabled.

Address: 0x0114 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	MASK															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	MASK															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1624. DSPI DSI Deserialized Data Interrupt Mask Register 1 (DSPI_DIMR1)

Table 1631. DSPI_DIMR1 field descriptions

Field	Description
0:31 MASK	Mask 0 The bit in the received DSI frame does not produce a DDI interrupt. 1 The bit in the received DSI frame can produce a DDI interrupt if the data bit matches the configured polarity.

62.3.36 DSPI DSI Deserialized Data Polarity Interrupt Register 1 (DSPI_DPIR1)

DPIR1 defines which of the 32 MSB value in the received 64-bit DSI frame generates the DDI interrupt.

The concatenation of {DSPI_DPIR1, DSPI_DPIR0} provides the 64-bit Deserialized Data Polarity value.

This register is valid only when DSICR1[DSI64E] is enabled.

Address: 0x0118 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	DP															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	DP															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1625. DSPI DSI Deserialized Data Polarity Interrupt Register 1 (DSPI_DPIR1)

Table 1632. DSPI_DPIR1 field descriptions

Field	Description
0:31 DP	Data Polarity 0 If the received bit is 0, the SR[DDIF] bit is set. 1 If the received bit is 1, the SR[DDIF] bit is set.

62.3.37 DSPI Clock and Transfer Attributes Register Extended (DSPI_CTARE*n*)

CTARE registers are used to define the extended transfer attributes for an SPI frame.

These registers are valid only when DSPI_MCR[XSPI] is set.

When the DSPI is configured as:

- an SPI master, the CTAS field in CMD FIFO entry selects which of the CTARE registers is used.
- an SPI bus slave, the CTARE0 register is used.

Address: 0x011C + *n**0x4 (*n* = 0 to 7) Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	FMSZE
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	DTCP										
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

Figure 1626. DSPI Clock and Transfer Attributes Register Extended (DSPI_CTARE*n*)

Table 1633. DSPI_CTARE*n* field descriptions

Field	Description
15 FMSZE	Frame Size Extended This field concatenated with DSPI_CTAR[FMSZ] defines the frame size of the SPI frames to be transmitted. Frame size is the concatenation of {DSPI_CTAR[FMSZ], DSPI_CTARE[FMSZE]} + 1. 0 Default Mode. Up to 16-bit SPI frames can be transferred. 1 Up to 32-bit SPI frames can be transferred. Each frame transfer is a result of 2 TX FIFO Pops.
21:31 DTCP	Data Transfer Count Preload This field defines the number of data frames (whose size is defined by CTARE[FMSZE] and CTAR[FMSZ]) to be transmitted using the Command frame that selected this DSPI_CTARE register. The value 0 is reserved and should not be written in this field. The default value of this field is 1.

62.3.38 DSPI Status Register Extended (DSPI_SREX)

The DSPI_SREX contains status fields that reflect the DSPI status and indicate the occurrence of events.

This register is not writable.

Address: 0x013C Access: User read-only

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	CMDCTR				CMDNXTPTR			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1627. DSPI Status Register Extended (DSPI_SREX)

Table 1634. DSPI_SREX field descriptions

Field	Description
24:27 CMDCTR	CMD FIFO Counter Indicates the number of entries in the CMD FIFO. The CMDCTR is incremented every time the command part of PUSHR is written. The CMDCTR is decremented every time a SPI command is executed (all data frames due to current command frame have been transmitted).
28:31 CMDNXTPTR	Command Next Pointer Indicates which CMD FIFO Entry is used during the next transfer. The CMDNXTPTR field is updated every time SPI data due to current command have been transmitted.

62.4 Register classification for safety

The registers of this module are classified as per safety requirements as follows:

Safety Critical: There are no safety critical registers in this module.

Safety Relevant: The register bits/fields in this category are safety relevant as any random error in these bits will lead to unwanted interference to CPU. The registers containing these

fields must be protected by periodic CRC or equivalent check implemented at chip level. The following register bits/fields are classified in this category:

- DSPI Module Configuration Register (DSPI_MCR)
- DSPI Transfer Count Register (DSPI_TCR)
- DSPI Clock and Transfer Attributes Register (In Master Mode) (DSPI_CTAR*n*)
- DSPI Clock and Transfer Attributes Extended Register (In Master Mode) (DSPI_CTARE*n*)
- DSPI DMA/Interrupt Request Select and Enable Register (DSPI_RSER)
- DSPI PUSH FIFO Register In Master Mode (DSPI_PUSHR) [0:15]
- DSPI DSI Configuration Register 0 (DSPI_DSICR0)
- DSPI DSI Configuration Register 1 (DSPI_DSICR1)

Safety Latent: The register bits/fields in this category should be protected by LBIST at chip level. The following register bits/fields are classified in this category:

- DSPI Status Register (DSPI_SR) [0:15] (for stuck at 0)
- DSPI Status Register Extended (DSPI_SREX) [0:8] (for stuck at 0)
- DSPI DSI Deserialized Data Interrupt Mask Register 0 (DSPI_DIMR0)
- DSPI DSI Deserialized Data Interrupt Mask Register 1 (DSPI_DIMR1)
- DSPI DSI Deserialized Data Polarity Interrupt Register 0 (DSPI_DPIR0)
- DSPI DSI Deserialized Data Polarity Interrupt Register 1 (DSPI_DPIR1)

Non-Safety: All other registers and bits fall in this category.

62.5 Functional description

The Deserial Serial Peripheral Interface (DSPI) block supports full-duplex, synchronous serial communications between MCUs and peripheral devices. The DSPI can also be used to reduce the number of pins required for I/O by serializing and deserializing up to 64 parallel input/output signals. All communications are done with SPI-like protocol.

The DSPI has the following configurations:

- SPI Configuration in which the DSPI operates as a basic SPI or a queued SPI.
- DSI Configuration in which the DSPI serializes and deserializes parallel input/output signals or bits from memory-mapped register.
- CSI Configuration in which the DSPI combines the functionality of the SPI and DSI configurations.

The DCONF field in the DSPI Module Configuration Register (MCR) determines the DSPI Configuration. Refer to *Section 62.3.1: DSPI Module Configuration Register (DSPI_MCR)* for the DSPI configuration values.

The CTAR*n* registers hold clock and transfer attributes. The SPI configuration allows CTAR selection on a frame by frame basis by setting a field in the SPI command. Extended SPI Mode (DSPI_MCR[XSPI]) further allows the usage of CTARE*n* (CTAR*n* Extended) registers to send multiple Data frames (of up to 32-bit frame size) using a single Command frame.

DSI configuration statically selects which CTAR to use. In CSI, Configuration priority logic determines if SPI data or DSI data is transferred and dictates which CTAR is used for the data transfer.

Refer to *Section 62.3.4: DSPI Clock and Transfer Attributes Register (In Master Mode) (DSPI_CTARn)* for information on CTARs fields.

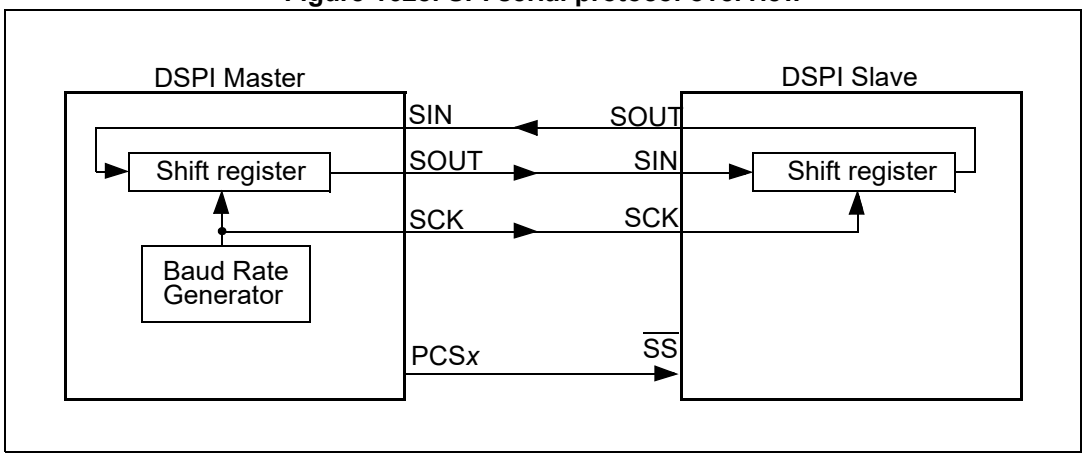
Refer to *Section 62.3.37: DSPI Clock and Transfer Attributes Register Extended (DSPI_CTAREn)* for information on CTAREs fields.

Typical master to slave connections are shown in *Figure 1628*.

When a data transfer operation is performed, data is serially shifted a predetermined number of bit positions. Because the modules are linked, data is exchanged between the master and the slave: the data that was in the master shift register is now in the shift register of the slave, and vice versa.

At the end of a transfer, the TCF bit in the Status register is set to indicate a completed transfer.

Figure 1628. SPI serial protocol overview



Generally, more than one slave device can be connected to the DSPI master.

Eight Peripheral Chip Select (PCS) signals of the DSPI masters can be used to select which of the slaves to communicate with. Refer to the chip configuration chapter for the number of PCS signals used in this MCU.

The three DSPI configurations share transfer protocol and timing properties which are described independently of the configuration in *Section 62.5.6: Transfer formats*.

The transfer rate and delay settings are described in *Section 62.5.5: DSPI baud rate and clock delay generation*.

62.5.1 Start and Stop of DSPI transfers

The DSPI has two operating states that are independent of DSPI configuration:

- STOPPED is the default DSPI state: no serial transfers are initiated in master mode and no transfers are responded to in slave mode. The STOPPED state is also a safe state for writing the various configuration registers of the DSPI without causing undetermined results.
- RUNNING is the state when serial transfers take place.

DSPI_SR[TXRXS] indicates what state the DSPI in (‘1’ = RUNNING state).

The DSPI is started (DSPI transitions to RUNNING) when all of the following conditions are true:

- SR[EOQF] bit is clear
- MCU is not in the debug mode or the MCR[FRZ] bit is clear
- MCR[HALT] bit is clear

The DSPI stops (transitions from RUNNING to STOPPED) after the current frame when any one of the following conditions exist:

- SR[EOQF] bit is set
- MCU in the debug mode and the MCR[FRZ] bit is set
- MCR[HALT] bit is set

State transitions from RUNNING to STOPPED occur on the next frame boundary if a transfer is in progress, or immediately if no transfers are in progress.

62.5.2 Serial Peripheral Interface (SPI) configuration

The SPI configuration transfers data serially using a shift register and a selection of programmable transfer attributes. The SPI frames can be 32 bits long.

The DSPI is in SPI Configuration when the DCONF field in the MCR is 0b00.

The host CPU or a DMA controller transfers the SPI data from memory external to DSPI RAM queues to a transmit FIFO (TX FIFO) buffer.

Received data is stored in entries in the Receive FIFO (RX FIFO) buffer. The host CPU or the DMA controller transfers the received data from the RX FIFO to memory external to the DSPI.

The FIFO buffers operation is described in:

- *Section 62.5.2.4: Transmit First In First Out (TX FIFO) buffering mechanism*,
- *Section 62.5.2.5: Command First In First Out (CMD FIFO) buffering mechanism*,
- *Section 62.5.2.6: Receive First In First Out (RX FIFO) Buffering Mechanism*.

The interrupt and DMA request conditions are described in *Section 62.5.12: Interrupts/DMA requests*.

The SPI Configuration supports two block-specific modes:

- Master mode: the DSPI initiates and controls the transfer according to the fields of the executing CMD FIFO entry.
- Slave mode: the DSPI only responds to transfers initiated by a bus master external to the DSPI and the SPI command field space is reserved.

62.5.2.1 Master mode

In SPI Master mode, the DSPI initiates the serial transfers by controlling the Serial Communications Clock (SCK) and the Peripheral Chip Select (PCS) signals.

The executing SPI CMD FIFO entry determines which CTARs will be used to set the transfer attributes and which PCS signals to assert. The command field also contains various bits that help with queue management and transfer protocol. See DSPI PUSH FIFO Register (PUSHR) for details on the SPI command fields.

The data in the executing TX FIFO entry is loaded into the shift register and shifted out on the Serial Out (SOUT) pin.

In SPI Master mode, each SPI frame to be transmitted has a command associated with it for transfer attribute control on a frame-by-frame basis.

In Extended SPI Master mode, multiple SPI frames can have a single command associated with them allowing for efficient SPI frame transfers requiring common transfer attributes. Extended SPI Mode allows for larger frame sizes of up to 32 bits.

62.5.2.2 Slave mode

In SPI slave mode, the DSPI responds to transfers initiated by a SPI bus master. The DSPI does not initiate transfers.

Certain transfer attributes such as clock polarity, clock phase and frame size must be set in CTAR0 and CTARE0 for successful communication with an SPI master. The data is shifted out with MSB first; shifting out of LSB is not supported in this mode.

62.5.2.3 FIFO Disable operation

The FIFO Disable mechanisms allow SPI transfers without using the TX FIFO, CMD FIFO or RX FIFO.

The DSPI operates as a double-buffered simplified SPI when the FIFOs are disabled.

The Transmit and Receive side of the FIFOs are disabled separately:

- MCR[DIS_TXF] disables TX FIFO and CMD FIFO.
  - TFFF, TFUF, CMDFFF, CMDCTR and TXCTR fields in the status register and the extended status register behave as if there is a one-entry FIFO.
  - The contents of the TXFR registers and TXNXTPTR and CMDNXTPTR are undefined.
- MCR[DIS_RXF] bit disables the RX FIFO.
  - The RFDF, RFOF and RXCTR fields in the status register behave as if there is a one-entry FIFO.
  - The contents of the RXFR registers and POPNXTPTR are undefined.

The FIFO Disable mechanisms are transparent to the user and to host software. Transmit data and commands are written to the PUSHR and received data is read from the POPR.

62.5.2.4 Transmit First In First Out (TX FIFO) buffering mechanism

The TX FIFO functions as a buffer of SPI data for transmission.

The TX FIFO holds 4 words, each consisting of SPI data that is added to the TX FIFO by writing to the Data field of DSPI PUSH FIFO Register (PUSHR).

The number of entries in the TX FIFO is device-specific.

TX FIFO entries can only be removed from the TX FIFO by being shifted out or by flushing the TX FIFO.

The TX FIFO Counter field (TXCTR) in the DSPI Status Register (SR) indicates the number of valid entries in the TX FIFO. TXCTR is updated every time an 8 or 16-bit write takes place to the Data field of DSPI_PUSHR or SPI data is transferred into the shift register from the TX FIFO.

The TXNXTPTR field indicates which TX FIFO Entry will be transmitted during the next transfer. This field is incremented every time SPI data is transferred from the TX FIFO to the

shift register. The maximum value of the field is equal to the maximum implemented TXFR register number and it rolls over after reaching the maximum.

The TXFR*n* registers are invalid in Extended SPI Mode as TX FIFO and CMD FIFO are used separately.

62.5.2.4.1 Filling the TX FIFO

Host software or other intelligent blocks can add (push) entries to the TX FIFO and CMD FIFO by writing to the PUSHR. When the TX FIFO is not full, the TX FIFO Fill Flag (TFFF) in the SR is set.When TX FIFO is full and the DMA controller indicates that a write to PUSHR is complete, the TFFF bit is cleared. Writing a '1' to the TFFF bit also clears it. The DSPI ignores attempts to push data to a full TX FIFO; the state of the TX FIFO does not change and no error condition is indicated.

The TFFF can generate a DMA request or an interrupt request. Refer to *Section 62.5.12.2: Transmit FIFO Fill interrupt or DMA request* for details.

62.5.2.4.2 Draining the TX FIFO

The TX FIFO entries are removed (drained) by shifting SPI data out through the shift register. Entries are transferred from the TX FIFO to the shift register and shifted out as long as there are valid entries in the TX FIFO. Every time an entry is transferred from the TX FIFO to the shift register, the TX FIFO Counter decrements by one.

When Extended SPI Mode (DSPI_MCR[XSPI]) is enabled and the frame size of SPI Data to be transmitted is more than 16 bits, then two Data entries are popped from TX FIFO simultaneously and transferred to the shift register. The first of the two popped entries forms the 16 LSB bits of the SPI frame. Such an operation also causes TX FIFO Counter to decrement by two.

At the end of a transfer, the TCF bit in the SR is set to indicate the completion of a transfer. The TX FIFO is flushed by writing a '1' to the CLR_TXF bit in MCR.

If an external bus master initiates a transfer with a DSPI slave while the slave's DSPI TX FIFO is empty, the Transmit FIFO Underflow Flag (TFUF) in the slave's SR is set. Refer to *Section 62.5.12.9: Transmit FIFO Underflow interrupt request* for details.

62.5.2.5 Command First In First Out (CMD FIFO) buffering mechanism

The CMD FIFO functions as a buffer of SPI command used for SPI data transmission.

The CMD FIFO holds four entries, each representing command fields. The number of entries in the CMD FIFO is device-specific.

SPI Command is added to the CMD FIFO by writing to the command field of DSPI PUSH FIFO Register (PUSHR). CMD FIFO entries can only be removed from the CMD FIFO by being shifted out (to help transmit SPI data) or by flushing the CMD FIFO.

When Extended SPI Mode (DSPI_MCR[XSPI]) is disabled:

- The TX FIFO and CMD FIFO must be filled together as every CMDFIFO entry has a corresponding single TXFIFO entry attached to it.

When Extended SPI Mode (DSPI_MCR[XSPI]) is enabled:

- The TX FIFO and CMD FIFO can be filled independently as every CMDFIFO entry can have multiple TXFIFO entries attached to it.
- The CTARE[DTCP] field decides the number of SPI Data Frames of size {FMSZE, FMSZ} to be transmitted using the current Command Entry. FMSZ and FMSZE fields are given in the CTAR/CTARE registers respectively, pointed by the CTAS field in the Command frame.
- The amount of time a command entry is in use is known as a Command Cycle. The busy flag DSPI_SR[BSYF] is asserted for the duration of the Command Cycle except for the last SPI frame in the Command Cycle.

The CMD FIFO counter field (CMDCTR) in the DSPI Status Register (SR) indicates the number of valid entries in the CMD FIFO. The TXCTR is updated every time an 8 or 16-bit write takes place on the lower half of DSPI_PUSHR or SPI data is transferred into the shift register from the TX FIFO.

The CMDNXTPTR field indicates which CMD FIFO entry is used during the next command cycle. This field is incremented every time the last SPI data in the command cycle is transferred from the TX FIFO to the shift register. The maximum value of the field is equal to the maximum implemented TXFR register number and it rolls over after reaching the maximum.

The TXFR*n* registers are invalid in Extended SPI Mode as TX FIFO and CMD FIFO are used separately.

62.5.2.6 Receive First In First Out (RX FIFO) Buffering Mechanism

The RX FIFO functions as a buffer for data received on the SIN pin.

The RX FIFO holds four received SPI data frames. The number of entries in the RX FIFO is device-specific.

SPI data is added to the RX FIFO on completion of a transfer when the received data in the shift register is transferred into the RX FIFO.

SPI data are removed (popped) from the RX FIFO by reading the DSPI POP FIFO Register (POPR). RX FIFO entries can only be removed from the RX FIFO by reading the POPR or by flushing the RX FIFO.

The RX FIFO Counter field (RXCTR) in the DSPI Status Register (SR) indicates the number of valid entries in the RX FIFO. The RXCTR is updated every time the POPR is read or SPI data is copied from the shift register to the RX FIFO.

The POPNXTPTR field in the status register points to the RX FIFO entry that is returned when the POPR is read. The POPNXTPTR contains the positive offset from RXFR0 in number of 32-bit registers. For example, POPNXTPTR = 2 means that the RXFR2 contains the received SPI data that will be returned when POPR is read. The POPNXTPTR field is incremented every time the POPR is read. The maximum value of the field is equal to the maximum implemented RXFR register number and it rolls over after reaching the maximum.

62.5.2.6.1 Filling the RX FIFO

The RX FIFO is filled with the received SPI data from the shift register.

While the RX FIFO is not full, SPI frames from the shift register are transferred to the RX FIFO. Every time a SPI frame is transferred to the RX FIFO the RX FIFO counter is incremented by one.

If the RX FIFO and shift register are full and a transfer is initiated, the RFOF bit in the status register is set to indicating an overflow condition. Consequently:

- If the DSPI_MCR[ROOE] bit is set, the incoming data is shifted in to the shift register.
- If the DSPI_MCR[ROOE] bit is cleared, the incoming data is ignored.

62.5.2.6.2 Draining the RX FIFO

Host CPU or a DMA can remove (pop) entries from the RX FIFO by reading the DSPI POP FIFO Register (POPR). A read of the POPR decrements the RX FIFO Counter by one.

Attempts to pop data from an empty RX FIFO are ignored and the RX FIFO Counter remains unchanged. The data read from the empty RX FIFO is undetermined.

When the RX FIFO is not empty, the RX FIFO Drain Flag (RFDF) in the status register is set. The RFDF bit is cleared when the RX_FIFO is empty and the DMA controller indicates that a read from POPR is complete, or by writing a ‘1’ to it.

62.5.3 Deserial Serial Interface (DSI) configuration

The DSI Configuration supports pin count reduction by serializing parallel input signals or register bits and shifting them out in a SPI-like protocol. The timing and transfer protocol is described in *Section 62.5.6: Transfer formats*. The received serial frames are converted to a parallel form (deserialized) and placed on the parallel output signals or in the DSPI_DDR. The various features of the DSI Configuration are set in DSPI DSI Configuration Registers (DSPI_DSICR1/0).

The DSI frames can be from 4 to 32 bits, or 64 bits when DSICR1[DSI64E] is enabled.

62.5.3.1 DSI master mode

In DSI master mode the DSPI initiates and controls the DSI transfers. The DSI master has four different conditions (described in *Section 62.5.3.5: DSI transfer initiation control*) that can initiate a transfer:

- Continuous.
- Change in data.
- Trigger signal.
- Trigger signal combined with a change in data.

Transfer attributes are set during initialization. DSICR0[DSICTAS] field determines which of the DSPI_CTARs will control the transfer attributes.

62.5.3.2 Slave Mode

In DSI slave mode the DSPI responds to transfers initiated by a SPI or DSI bus master. In this mode the DSPI does not initiate DSI transfers.

Certain transfer attributes such as clock polarity and phase must be set for successful communication with a DSI master. The DSI slave mode transfer attributes are set in the DSPI_CTAR1.

The data is shifted out with MSB first.

62.5.3.3 DSI serialization

The DSI configuration from 4 to 16 bits can be serialized using two different sources selected by DSICR0[TXSS]:

- DSPI DSI Serialization Data Register (DSPI_SDR1/0). The DSPI_SDR0 holds the latest 32 parallel input signal values which is sampled at every rising edge of the protocol clock.
  When DSICR1[DSI64E] is enabled, 64-bit DSI frames are supported and hence DSPI_SDR1 holds the 32 MSB parallel input signal values which is sampled at every rising edge of the protocol clock.
- DSPI DSI Alternate Serialization Data Register (DSPI_ASDR1/0). The DSPI_ASDR1/0 registers are written by host software and used as an alternate source of serialized data. DSPI_ASDR1 is only usable when DSICR1[DSI64E] is enabled.

The DSPI has to sample the (SDR/ASDR) inputs before it transmits a DSI frame. Due to the asynchronous clock structure within DSPI, the sampling of inputs come under Register Clock domain whereas DSI frame transmission logic lies in Protocol Clock domain. Hence the sampled data (from ASDR1/0 or SDR1/0) must be synchronized into the Protocol Domain before being loaded into the Shift register for transmission.

To prevent Clock Domain clocking issues, SDR/ASDR inputs should not change while they are being sampled into the Protocol Domain. This is assured in the following manner:

- For the ASDR0 and ASDR1 inputs:
  - The initial write performed on ASDR register causes the 32-bit ASDR input to be latched into a temporary register which is then locked (further latching is prevented) until this data gets synchronized into the Protocol Domain.
  - Once synchronized, the lock is removed and the next ASDR write triggers this process again. If a write is performed on ASDR while the lock is asserted, this is processed as soon as the previous synchronization process completes and the lock is de-asserted. If multiple writes are performed on ASDR when lock is asserted after the first write, only the last value is considered for synchronization once the lock is de-asserted.
- For the SDR0 and SDR1 inputs:
  - The process is the same as ASDR with the only difference being that SDR inputs are delayed internally by a single clock to detect a change in the SDR input pins.

Note: If SDR0 and SDR1 inputs are both changed at the same time instant, then the synchronization logic ensures that both SDR0 and SDR1 contribute to the same DSI frame.

Once the synchronization process for a particular Deserial input begins, it takes up to five Register clock periods and five Protocol clock periods before the synchronization process for the next Deserial input can begin. Hence the writes performed on ASDR0/1 and SDR0/1 are constrained with this timing.

The DSPI_PISR0–3 registers (DSPI_PISR0–7 registers if DSICR1[DSI64E] is enabled) allow changing of the relative position of the parallel input pins in the transmitted frame. Each transmitted frame bit can be selected from 16 adjacent parallel inputs by writing IPS*n* fields. The IPS*n* field is treated as a 4-bit integer number, representing numbers from -8 to 7. The parallel input pin number, selected by IPS*n* field is defined by the sum of the IPS*n* field number (*n*) and the IPS*n* field value. If the operation result is negative the number 32 should be added. If the result is higher than 32, 32 should be subtracted from the result. When DSICR1[DSI64E] is enabled, if the above operation result is negative, the number 64 should be added. If the result is higher than 64, 64 should be subtracted from the result.

For example, IPS0, set to minus 1 (binary 1111), preselects parallel input 31 to 0 position in the transmitted frame. (This preselects parallel input 63 to 0 position in the transmitted frame when DSICR1[DSI64E] is set).

IPS6, set to 3 (binary 0011), preselects parallel input 9 to be bit number 6 in the transmitted frame, while the value minus 2 (1110) preselects parallel input 4.

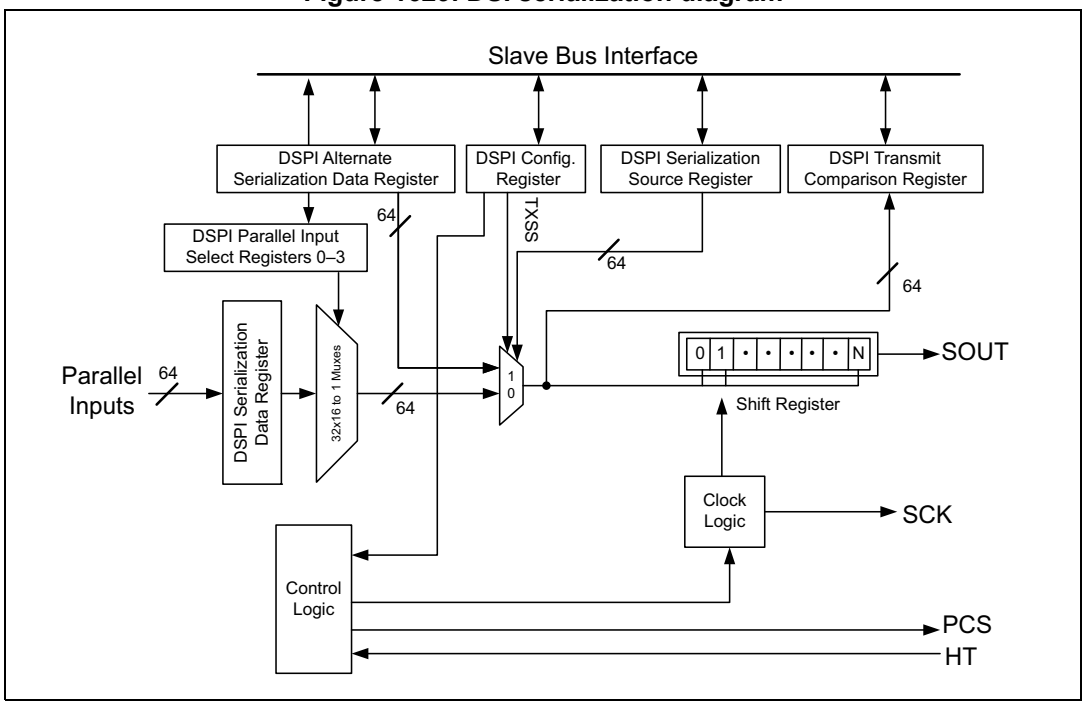
IPS31, set to minus 8 (binary 1000), preselects parallel input 23 to be bit number 31 in the transmitted frame.

The parallel input pin state to be transmitted should be selected by TXSS or DSPI_SSR1/0 registers and the frame size should be higher than the bit position in the preselected frame.

DSPI_SSR1/0 registers provide additional way of creating the frame for transmission. Each bit from this register is ORed with the TXSS bit and controls individual transmitted bit source. This way, the transmitted frame can have any combination of the DSPI_SDR1/0 and DSPI_ASDR1/0 bits. This feature allows control SPI based devices, requiring control and data fields in the frame. The control field may come from DSPI_ASDR1/0 registers set by the CPU, while data field can be generated by device peripheral modules, like PWM timers.

A copy of the last DSI frame shifted out of the Shift register is stored in the DSPI DSI Transmit Comparison Registers (DSPI_COMPR1/0). This register provides added visibility for debugging and it serves as a reference for transfer initiation control. *Figure 1629* shows the DSI serialization logic.

Figure 1629. DSI serialization diagram

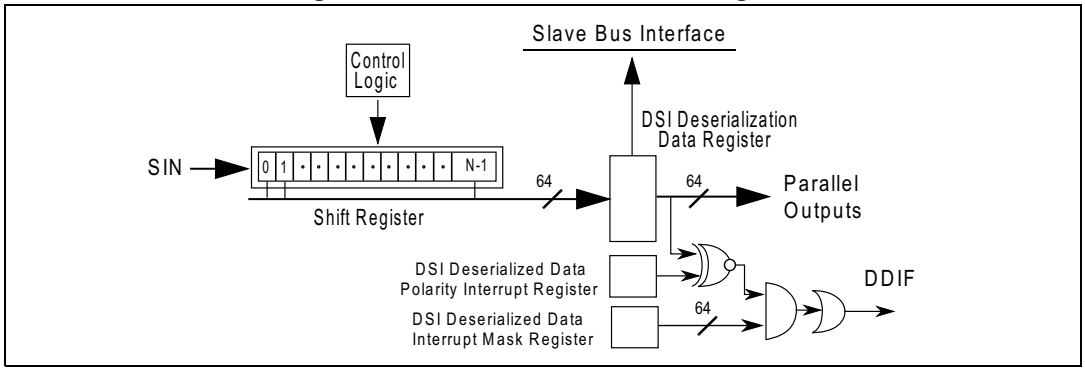


62.5.3.4 DSI deserialization

When all bits in a DSI frame have been shifted in, the frame is copied to the DSPI DSI Deserialization Data Register (DSPI_DDR1/0). This register presents the deserialized data as parallel output signal values. The DSPI_DDR1/0 is memory-mapped to allow host software to read the deserialized data directly.

The received data is bit-wise compared to the value of the DSI Deserialized Data Polarity Interrupt Register (DSPI_DPIR1/0), bit-wise ANDed with DSI Deserialized Interrupt Mask Register (DSPI_DIMR1/0) and the results ORed to produce DDIF flag in the DSPI_SR. Which in turn can cause DDIF interrupt request or DMA request based on RSER[DDIF_RE] and RSER[DDIF_DIRS] and/or stop DSI frame transmissions if DSICR0[DMS] bit is set. *Figure 1630* shows the DSI Deserialization logic.

Figure 1630. DSI deserialization diagram



62.5.3.5 DSI transfer initiation control

Data transfers for a master DSPI in DSI configuration are initiated by a condition. The transfer initiation conditions are selected by the TRRE and CID bits in the DSPI_DSICR0 register. *Table 1635* lists the four transfer initiation conditions.

Table 1635. DSI data transfer initiation control

DSPI_DSICR0 bits		Transfer initiation control
TRRE	CID	
0	0	Continuous
0	1	Change in Data
1	0	Triggered
1	1	Triggered or Change in Data

62.5.3.5.1 Continuous control

For continuous control, a new DSI frame shifts out when the previous transfer cycle has completed and the delay after transfer (t_{DT}) has elapsed.

62.5.3.5.2 Change in data control

For change in data control, a transfer is initiated when the data to be serialized has changed since the transfer of the last DSI frame. A copy of the previously transferred DSI data is stored in the DSPI_COMPR1/0. When the data selected for the transfer from the DSPI_SDR1/0 and DSPI_ASDR1/0 registers is different from the data in the DSPI_COMPR1/0 a new DSI frame is transmitted.

62.5.3.5.3 Triggered control

For Triggered Control, initiation of a transfer is controlled by the Hardware Trigger signal (HT). DSICR0[TPOL] bit selects the active edge of HT. For HT to have any affect, DSICR0[TRRE] bit must be set.

62.5.3.5.4 Change in data or triggered control

For change in data or triggered control, initiation of a transfer is controlled by the detection of a change in data to be serialized or by the HT signal.

It is not advisable to configure a DSPI slave with the combination DSICR0[DCONT] = 1, DSICR0[CID] = 1; since the following statement then holds true:

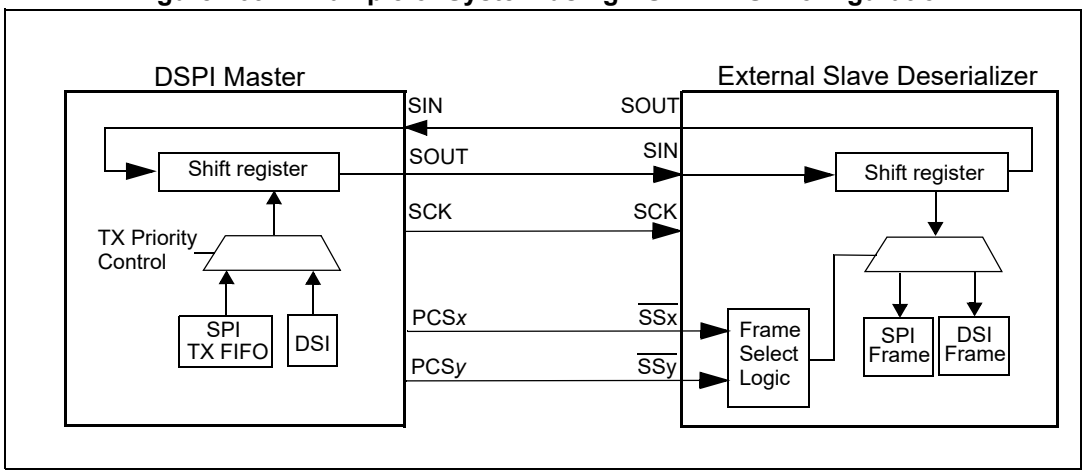
> Once slave select is asserted, there is a very small window (of up to three module clocks) after every SOF (Start of Frame) during which if a write on ASDR/SDR registers occurs, the next frame to be transmitted would be loaded into the shift register. Since it is not feasible that such a write is initiated and completed within this short span of time, the data transmitted by DSPI slave would lag by one data frame.

62.5.4 Combined Serial Interface (CSI) configuration

The CSI Configuration of the DSPI is used to support SPI and DSI functions on a frame by frame basis.

CSI Configuration allows interleaving of DSI data frames from the parallel input signals with SPI data frames from the TX FIFO. The data returned from the bus slave is either used to drive the parallel output signals or it is stored in the RX FIFO. The CSI Configuration allows serialized data and configuration or diagnostic data to be transferred to a slave device using only one serial link. The DSPI is in CSI Configuration when the DCONF field in the DSPI_MCR is 0b10. *Figure 1631* shows an example of how a DSPI can be used with a deserializing peripheral that supports SPI control for control and diagnostic frames.

Figure 1631. Example of System using DSPI in CSI Configuration



In CSI Configuration, the DSPI transfers DSI data based on *Section 62.5.3.5: DSI transfer initiation control*. When there are SPI commands in the TX FIFO, the SPI data has priority over the DSI frames. When the TX FIFO or CMD FIFO is empty, DSI transfer resumes.

Two peripheral chip select signals indicate whether DSI data or SPI data is transmitted. The user must configure the DSPI so that the two CTARs associated with DSI data and SPI data assert different peripheral chip select signals denoted in the figure as PCSx and PCSy.

The CSI Configuration is only supported in master mode.

Data returned from the external slave while a DSI frame is transferred is placed on the parallel output signals. Data returned from the external slave while a SPI frame is transferred is moved to the RX FIFO. The TX FIFO, CMD FIFO and RX FIFO are fully functional in CSI mode.

62.5.4.1 CSI serialization

Serialization in the CSI configuration is similar to serialization in DSI Configuration. The transfer attributes for SPI frames are determined by the DSPI_CTAR selected by the CTAS field in the SPI command halfword. The transfer attributes for the DSI frames are determined by the DSPI_CTAR selected by the DSICR0[DSICTAS] field.

The parallel inputs signal states are latched into the DSPI DSI Serialization Data Register (DSPI_SDR) on the rising edge of every protocol clock and serialized based on the transfer initiation control settings in the DSPI_DSICR1/0. When SPI frames are written to the TX FIFO (and when CMD FIFO is not empty), they have priority over DSI data and are transferred at the next frame boundary. A copy of the most recently transferred DSI frame is stored in the DSPI_COMPR1/0. The Transfer Priority Logic selects the source of the serialized data and asserts the appropriate PCS signal.

Separate frame completion interrupts are available to indicate frame transmission completion from SPI and DSI frames.

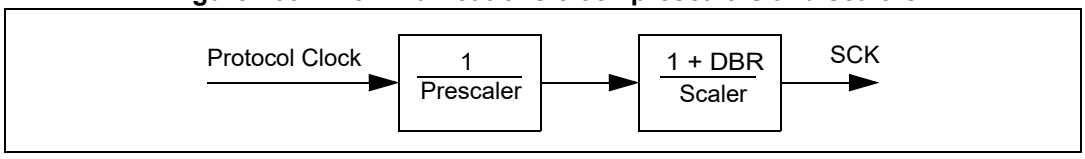
62.5.4.2 CSI deserialization

The deserialized frames in CSI Configuration goes into the DSPI_DDR1/0 or the RX FIFO based on the transfer priority logic. When DSI frames are transferred, the returned frames are deserialized and latched into the DSPI_DDR1/0. When SPI frames are transferred, the returned frames are deserialized and written to the RX FIFO.

62.5.5 DSPI baud rate and clock delay generation

The SCK frequency and the delay values for serial transfer are generated by dividing the protocol clock frequency by a prescaler and a scaler with the option for doubling the baud rate. *Figure 1632* shows conceptually how the SCK signal is generated.

Figure 1632. Communications clock prescalers and scalers



62.5.5.1 Baud rate generator

The baud rate is the frequency of the Serial Communication Clock (SCK). The protocol clock is divided by a prescaler (PBR) and scaler (BR) to produce SCK with the possibility of halving the scaler division. The DBR, PBR and BR fields in the CTARs select the frequency of SCK by the formula in the BR field description. *Table 1636* shows an example of how to compute the baud rate.

Table 1636. Baud rate computation example

f_P	PBR	Prescaler	BR	Scaler	DBR	Baud rate
100 MHz	0b00	2	0b0000	2	0	25 Mb/s
20 MHz	0b00	2	0b0000	2	1	10 Mb/s

Note: The clock frequencies mentioned in Table 1636 are given as an example. Refer to the Clocking chapter for the frequency used to drive this module in the device.

62.5.5.2 PCS to SCK delay (t_{CSC})

The PCS to SCK delay is the length of time from assertion of the PCS signal to the first SCK edge. The PCSSCK and CSSCK fields in the CTAR*n* registers select the PCS to SCK delay by the formula in the CSSCK field description. *Table 1637* shows an example of how to compute the PCS to SCK delay.

Table 1637. PCS to SCK delay computation example

f_P	PCSSCK	Prescaler	CSSCK	Scaler	PCS to SCK delay
100 MHz	0b01	3	0b0100	32	0.96 µs

PCSCSK and CSSCK fields have no effect in TSB configuration.

Note: The clock frequency mentioned in Table 1637 is given as an example. Refer to the Clocking chapter for the frequency used to drive this module in the device.

62.5.5.3 After SCK delay (t_{ASC})

The After SCK delay is the length of time between the last edge of SCK and the negation of PCS. The PASC and ASC fields in the CTAR*n* registers select the After SCK delay by the formula in the ASC field description. *Table 1638* shows an example of how to compute the After SCK delay.

Table 1638. After SCK delay computation example

f_P	PASC	Prescaler	ASC	Scaler	After SCK delay
100 MHz	0b01	3	0b0100	32	0.96 µs

PCASC and ASC fields have no effect in TSB configuration.

Note: The clock frequency mentioned in Table 1638 is given as an example. Refer to the Clocking chapter for the frequency used to drive this module in the device.

62.5.5.4 Delay after transfer (t_{DT})

The delay after transfer is the minimum time between negation of the PCS signal for a frame and the assertion of the PCS signal for the next frame. The PDT and DT fields in the CTAR*n* registers select the Delay after Transfer by the formula in the DT field description. *Table 1639* shows an example of how to compute the delay after transfer.

Table 1639. Delay after transfer computation example

f_P	PDT	Prescaler	DT	Scaler	Delay after transfer
100 MHz	0b01	3	0b1110	32768	0.98 ms

Note: The clock frequency mentioned in Table 1639 is given as an example. Refer to the Clocking chapter for the frequency used to drive this module in the device.

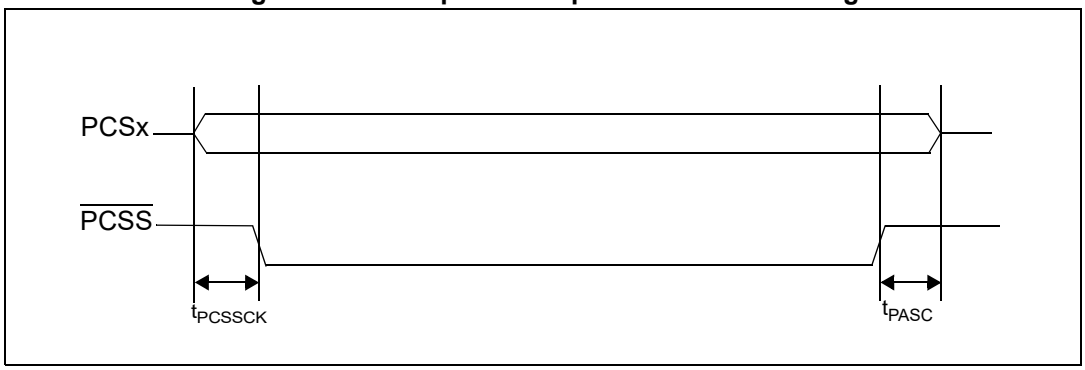
When in non-continuous clock mode the t_{DT} delay is configured according to the equation specified in the CTAR[DT] bitfield description. When in continuous clock mode and TSB is not enabled, the delay is fixed at 1 SCK period.

In TSB mode, the Delay after Transfer is equal to a number formed by concatenation of PDT and DT fields plus 1 of the SCK clock periods. Refer to detailed information on *Section 62.5.9: Timed Serial Bus (TSB)*.

62.5.5.5 Peripheral Chip Select Strobe Enable ($\overline{\text{PCSS}}$)

The PCSS signal provides a delay to allow the PCS signals to settle after a transition occurs, thereby avoiding glitches. When the DSPI is in master mode and the PCSSE bit is set in the MCR, $\overline{\text{PCSS}}$ provides a signal for an external demultiplexer to decode the PCS[0]–PCS[4]and PCS[6]–PCS[7] signals into as many as 128 glitch-free PCS signals. *Figure 1633* shows the timing of the $\overline{\text{PCSS}}$ signal relative to PCS signals.

Figure 1633. Peripheral Chip Select Strobe Timing



The delay between the assertion of the PCS signals and the assertion of PCSS is selected by the PCSSCK field in the CTAR based on the following formula:

Equation 83

$$t_{PCSSCK} = \frac{1}{f_P} \times PCSSCK$$

At the end of the transfer the delay between $\overline{\text{PCSS}}$ negation and PCS negation is selected by the PASC field in the CTAR based on the following formula:

Equation 84

$$t_{PASC} = \frac{1}{f_P} \times PASC$$

Table 1640 shows an example of how to compute the t_{PCSSCK} delay.

Table 1640. Peripheral Chip Select Strobe Assert Computation Example

f_P	PCSSCK	Prescaler	Delay before transfer
100 MHz	0b11	7	70.0 ns

Table 1641 shows an example of how to compute the t_{PASC} delay.

Table 1641. Peripheral Chip Select Strobe Negate Computation Example

f_P	PASC	Prescaler	Delay after transfer
100 MHz	0b11	7	70.0 ns

The $\overline{\text{PCSS}}$ signal is not supported when Continuous Serial Communication SCK or TSB mode are enabled.

Note: The clock frequency mentioned in the preceding Table 1640 and Table 1641 is given as an example. Refer to the Clocking chapter for the frequency used to drive this module in the device.

62.5.6 Transfer formats

The SPI serial communication is controlled by the Serial Communications Clock (SCK) signal and the PCS signals. The SCK signal provided by the master device synchronizes shifting and sampling of the data on the SIN and SOUT pins. The PCS signals serve as enable signals for the slave devices.

In master mode, the CPOL and CPHA bits in the Clock and Transfer Attributes Registers (CTAR*n*) select the polarity and phase of the serial clock, SCK.

- CPOL: selects the idle state polarity of the SCK.
- CPHA: selects if the data on SOUT is valid before or on the first SCK edge.

Even though the bus slave does not control the SCK signal, in slave mode these values must be identical to the master device settings to ensure proper transmission. In SPI slave mode, only CTAR0 is used.

In DSI slave mode, only CTAR1 is used.

The DSPI supports four different transfer formats:

- Classic SPI with CPHA = 0.
- Classic SPI with CPHA = 1.
- Modified Transfer format with CPHA = 0.
- Modified Transfer format with CPHA = 1.

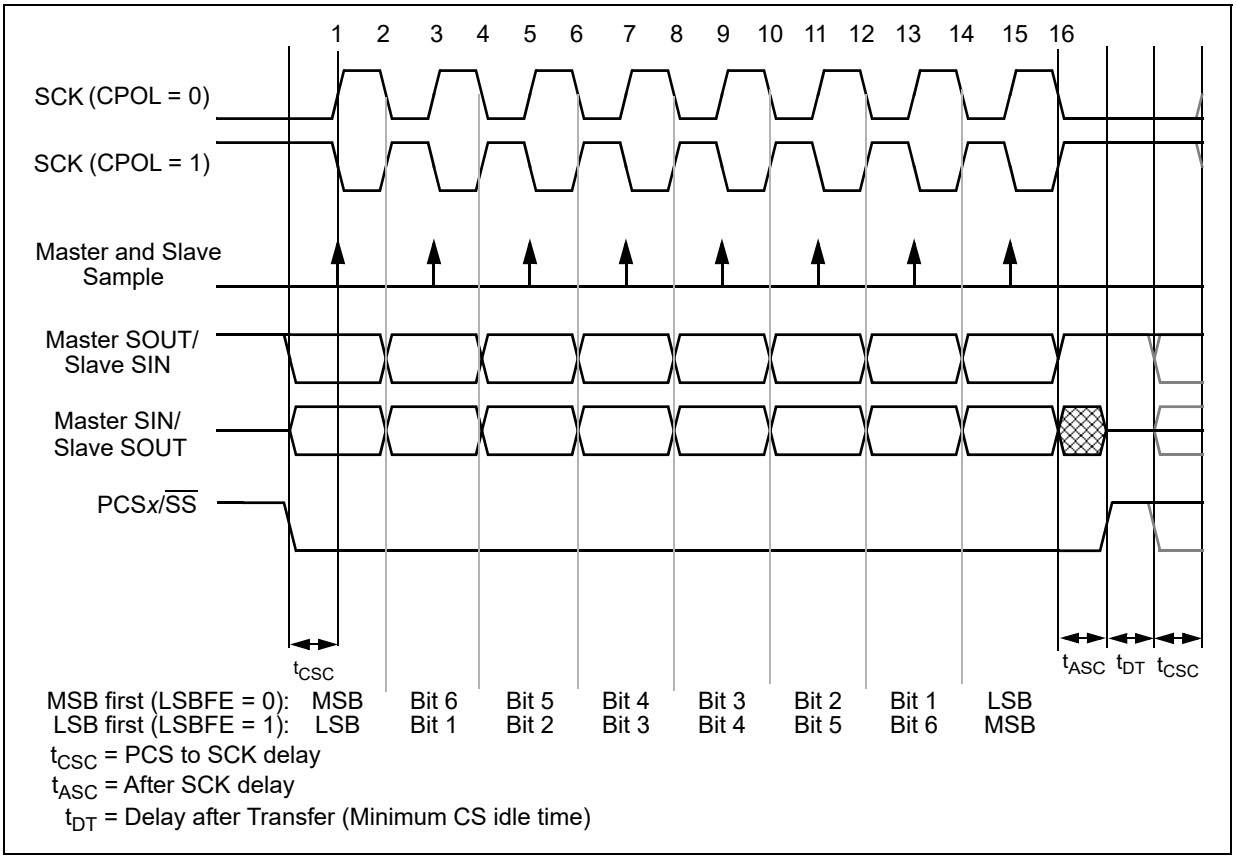
A modified transfer format is supported to allow for high-speed communication with peripherals that require longer setup times. The DSPI can sample the incoming data later than halfway through the cycle to give the peripheral more setup time. The MTFE bit in the MCR selects between Classic SPI Format and Modified Transfer Format.

In the SPI and DSI configurations, the DSPI provides the option of keeping the PCS signals asserted between frames. Refer to *Section 62.5.6.5: Continuous Selection Format* for details.

62.5.6.1 Classic SPI Transfer Format (CPHA = 0)

The transfer format shown in *Figure 1634* is used to communicate with peripheral SPI slave devices where the first data bit is available on the first clock edge. In this format, the master and slave sample their SIN pins on the odd-numbered SCK edges and change the data on their SOUT pins on the even-numbered SCK edges.

Figure 1634. DSPI transfer timing diagram (MTFE = 0, CPHA = 0, FMSZ = 8)



The master initiates the transfer by placing its first data bit on the SOUT pin and asserting the appropriate peripheral chip select signals to the slave device. The slave responds by placing its first data bit on its SOUT pin. After the t_{CSC} delay elapses, the master outputs the first edge of SCK. The master and slave devices use this edge to sample the first input data bit on their serial data input signals.

At the second edge of the SCK the master and slave devices place their second data bit on their serial data output signals.

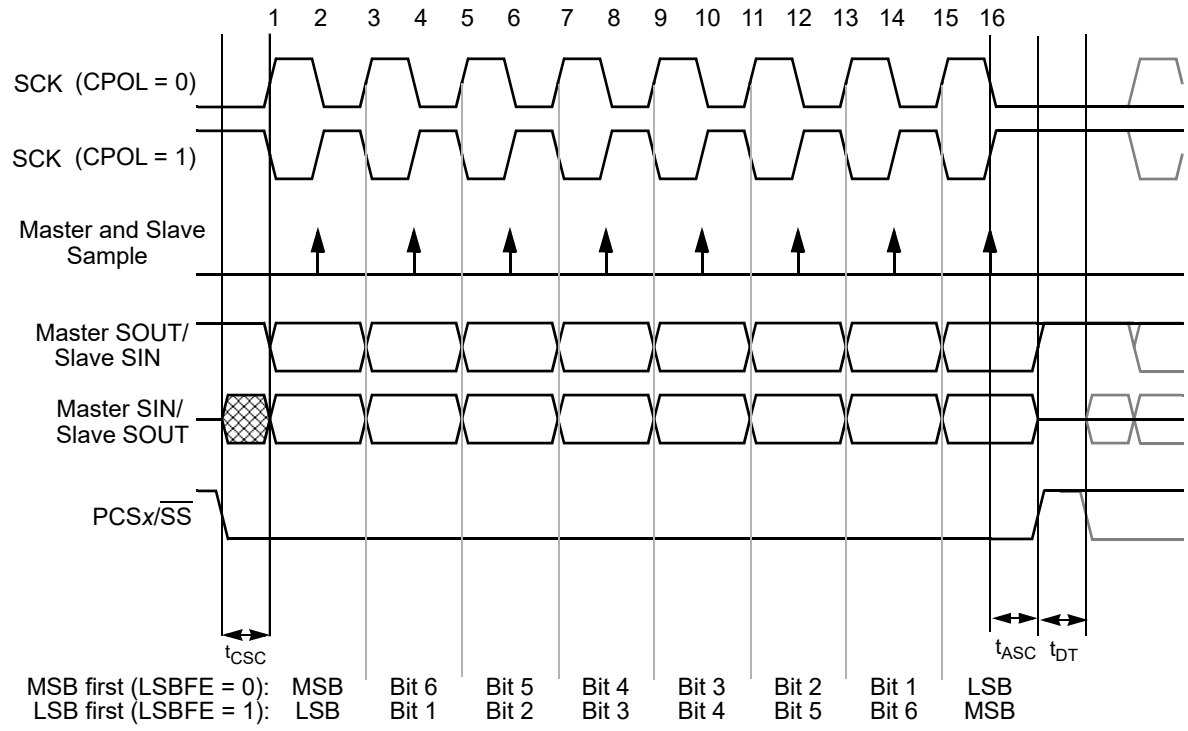
For the rest of the frame the master and the slave sample their SIN pins on the odd-numbered clock edges and changes the data on their SOUT pins on the even-numbered clock edges.

After the last clock edge occurs a delay of t_{ASC} is inserted before the master negates the PCS signals. A delay of t_{DT} is inserted before a new frame transfer can be initiated by the master.

62.5.6.2 Classic SPI Transfer Format (CPHA = 1)

This transfer format shown in *Figure 1635* is used to communicate with peripheral SPI slave devices that require the first SCK edge before the first data bit becomes available on the slave SOUT pin. In this format the master and slave devices change the data on their SOUT pins on the odd-numbered SCK edges and sample the data on their SIN pins on the even-numbered SCK edges.

Figure 1635. DSPI transfer timing diagram (MTFE = 0, CPHA = 1, FMSZ = 8)



t_{CSC} = PCS to SCK delay
t_{ASC} = After SCK delay
t_{DT} = Delay after Transfer (minimum CS negation time)

The master initiates the transfer by asserting the PCS signal to the slave.

After the t_{CSC} delay has elapsed, the master generates the first SCK edge and at the same time places valid data on the master SOUT pin. The slave responds to the first SCK edge by placing its first data bit on its slave SOUT pin.

At the second edge of the SCK the master and slave sample their SIN pins.

For the rest of the frame the master and the slave change the data on their SOUT pins on the odd-numbered clock edges and sample their SIN pins on the even-numbered clock edges.

After the last clock edge occurs a delay of t_{ASC} is inserted before the master negates the PCS signal. A delay of t_{DT} is inserted before a new frame transfer can be initiated by the master.

62.5.6.3 Modified SPI/DSI Transfer Format (MTFE = 1, CPHA = 0)

In this Modified Transfer Format both the master and the slave sample later in the SCK period than in Classic SPI mode to allow the logic to tolerate more delays in device pads and board traces. These delays become a more significant fraction of the SCK period as the SCK period decreases with increasing baud rates.

The master and the slave place data on the SOUT pins at the assertion of the PCS signal. After the PCS to SCK delay has elapsed the first SCK edge is generated. The slave samples the master SOUT signal on every odd numbered SCK edge. The DSPI in the slave mode when the MTFE bit is set also places new data on the slave SOUT on every odd numbered clock edge. Regular external slave, configured with CPHA=0 format drives its SOUT output at every even numbered SCK clock edge.

The DSPI master places its second data bit on the SOUT line one protocol clock after odd numbered SCK edge if the protocol clock frequency to SCK frequency ratio is higher than three. If this ratio is below four the master changes SOUT at odd numbered SCK edge. The point where the master samples the SIN is selected by the DSPI_MCR[SMPL_PT] field. The master sample point can be delayed by one or two protocol clock cycles. The SMPL_PT field should be set to 0 if the protocol to SCK frequency ratio is less than 4. However if this ratio is less than 4, the actually sample point is delayed by 1 protocol clock cycle automatically by the design.

The following timing diagrams illustrate the DSPI operation with MTFE = 1. Timing delays shown are:

- t_{CSC} – PCS to SCK assertion delay
- t_{ACS} – After SCK PCS negation delay
- t_{su_ms} – master SIN setup time
- t_{hd_ms} – master SIN hold time
- t_{vd_sl} – slave data output valid time, time between slave data output SCK driving edge and data becomes valid.
- t_{su_sl} – data setup time on slave data input
- t_{hd_sl} – data hold time on slave data input
- t_p – protocol clock period

Figure 1636 shows the modified transfer format for CPHA = 0 and f_p/f_{sck} = 4{s}. Only the condition where CPOL = 0 is illustrated. Solid triangles show the data sampling clock edges. The two possible slave behaviors are shown.

- Signal, marked “SOUT of Ext Slave”, presents regular SPI slave serial output.
- Signal, marked “SOUT of DSPI Slave”, presents DSPI in the slave mode with MTFE bit set.

Other MTFE = 1 diagrams show DSPI SIN input as being driven by a regular external SPI slave, configured according to DSPI master CPHA programming.

Note: In the following diagrams, f_p represents the protocol clock frequency from which the Baud frequency f_{sck} is derived.

Note: When MTFE=1 with continuous SCK enabled (MCR [CONT_SCKE] =1) in master mode, configure CTAR[LSBFE]=0 for correct operations while receiving unequal length frames. If PUSHR[CONT] is also set for back to back frame transfer, also configure the frame size of the first frame as less than or equal to the frame size of the next frame. In this scenario, make sure that for all received frames, the bits are read equal to their respective frame sizes and any extra bits during POP operation are masked.

Figure 1636. DSPI Modified Transfer Format (MTFE = 1, CPHA = 0, $f_{sck} = f_p/4$)

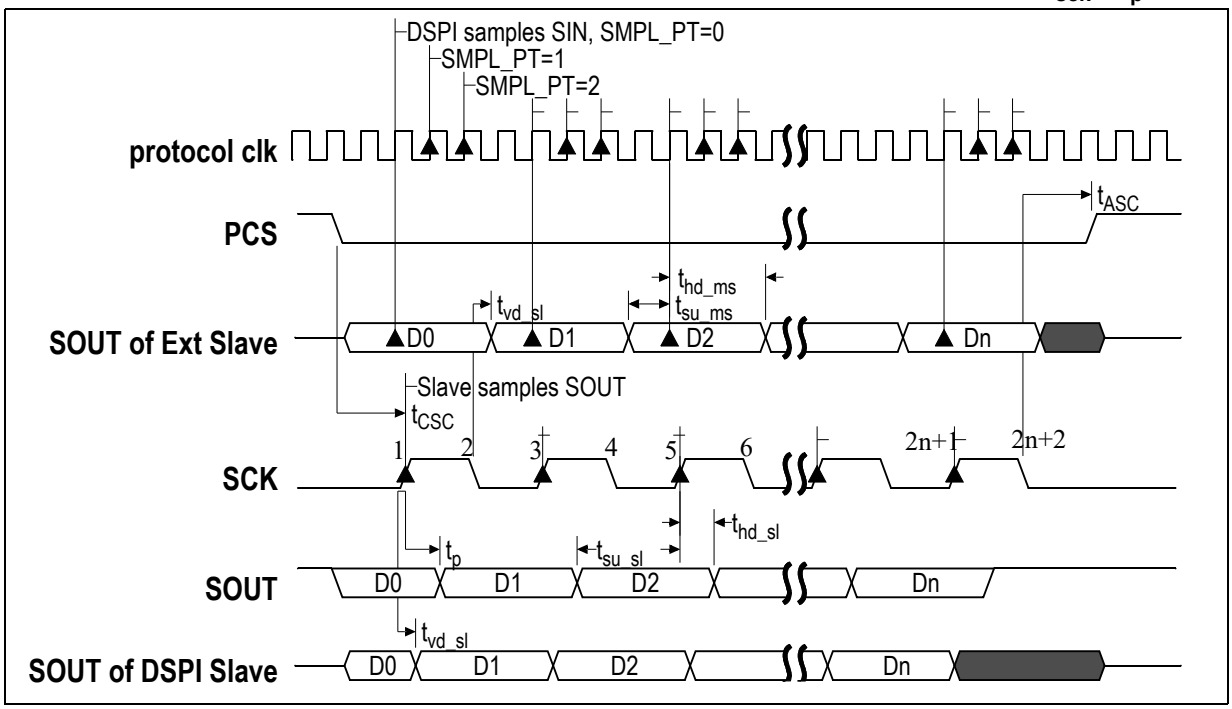


Figure 1637. DSPI Modified Transfer Format (MTFE = 1, CPHA = 0, $f_{sck} = f_p/2$)

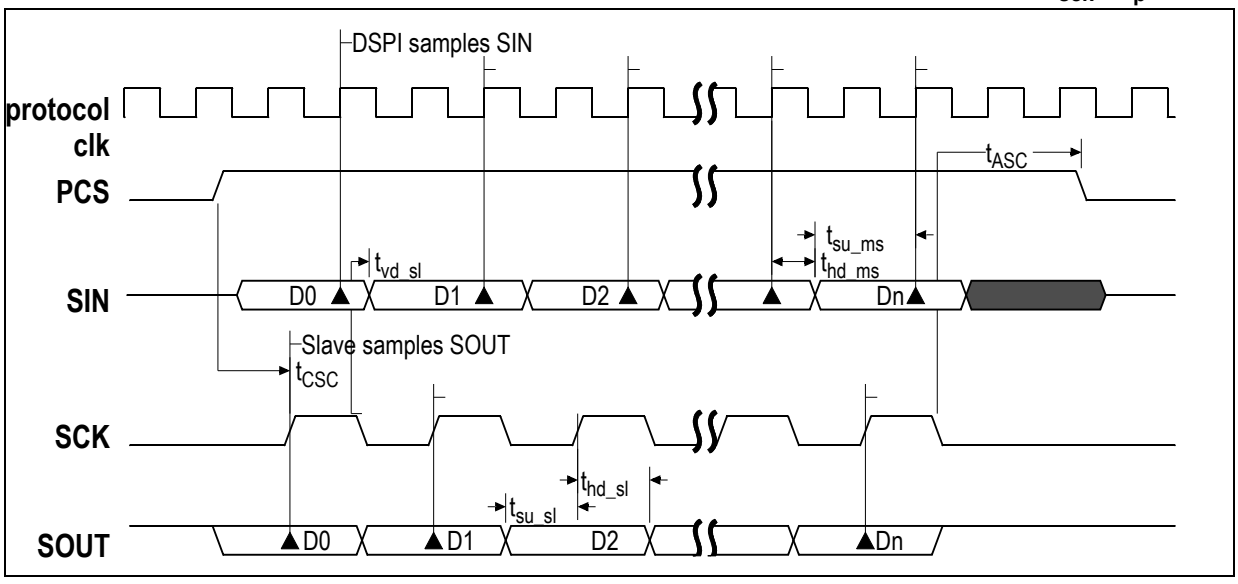
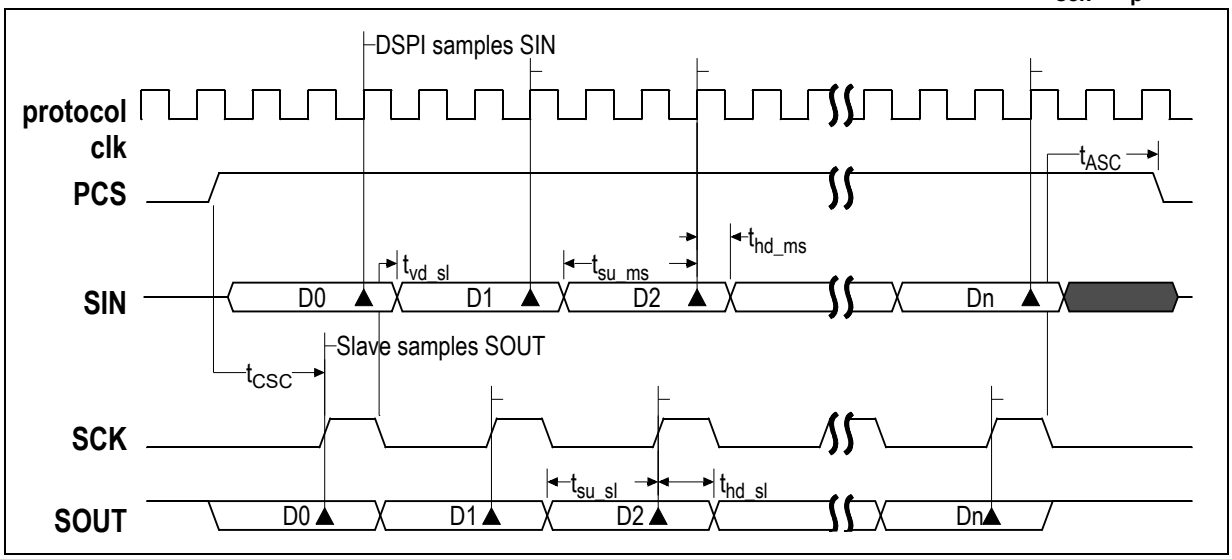


Figure 1638. DSPI Modified Transfer Format (MTFE = 1, CPHA = 0, $f_{sck} = f_p/3$)



62.5.6.4 Modified SPI/DSI Transfer Format (MTFE = 1, CPHA = 1)

The following figures show the Modified Transfer Format for CPHA = 1. Only the condition, where CPOL = 0 is shown. At the start of a transfer the DSPI asserts the PCS signal to the slave device. After the PCS to SCK delay has elapsed the master and the slave put data on their SOUT pins at the first edge of SCK. The slave samples the master SOUT signal on the even numbered edges of SCK. The master samples the slave SOUT signal on the odd numbered SCK edges starting with the third SCK edge. The slave samples the last bit on the last edge of the SCK. The master samples the last slave SOUT bit one half SCK cycle after the last edge of SCK. No clock edge will be visible on the master SCK pin during the sampling of the last bit. The SCK to PCS delay must be greater than or equal to half of the SCK period.

Figure 1639. DSPI Modified Transfer Format (MTFE = 1, CPHA = 1, $f_{sck} = f_p/3$)

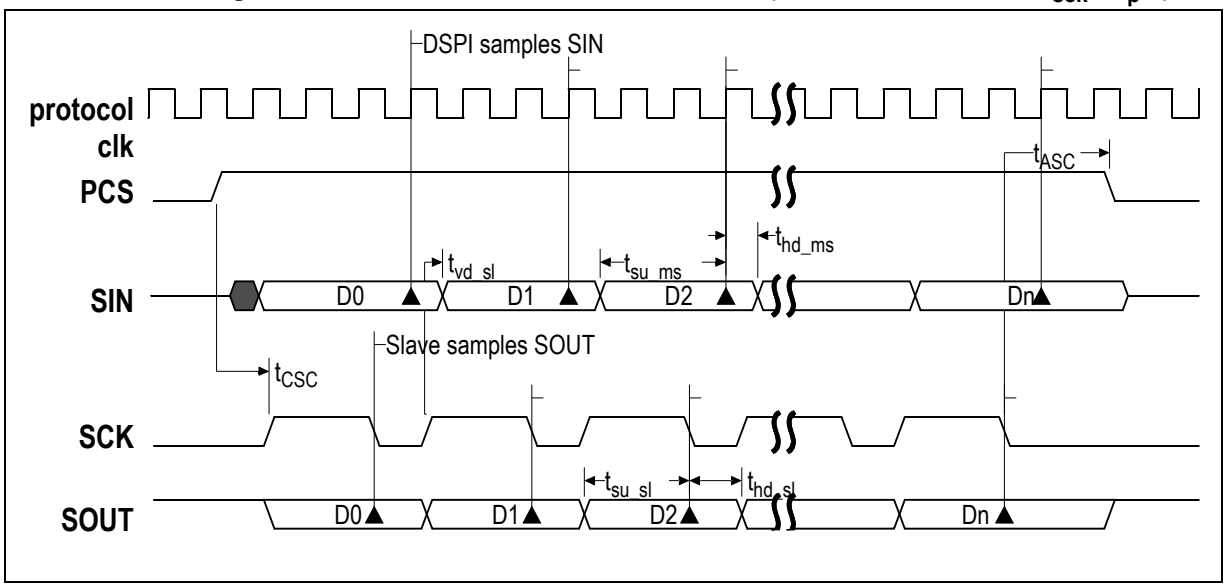


Figure 1640. DSPI Modified Transfer Format (MTFE = 1, CPHA = 1, $f_{sck} = f_p/4$)

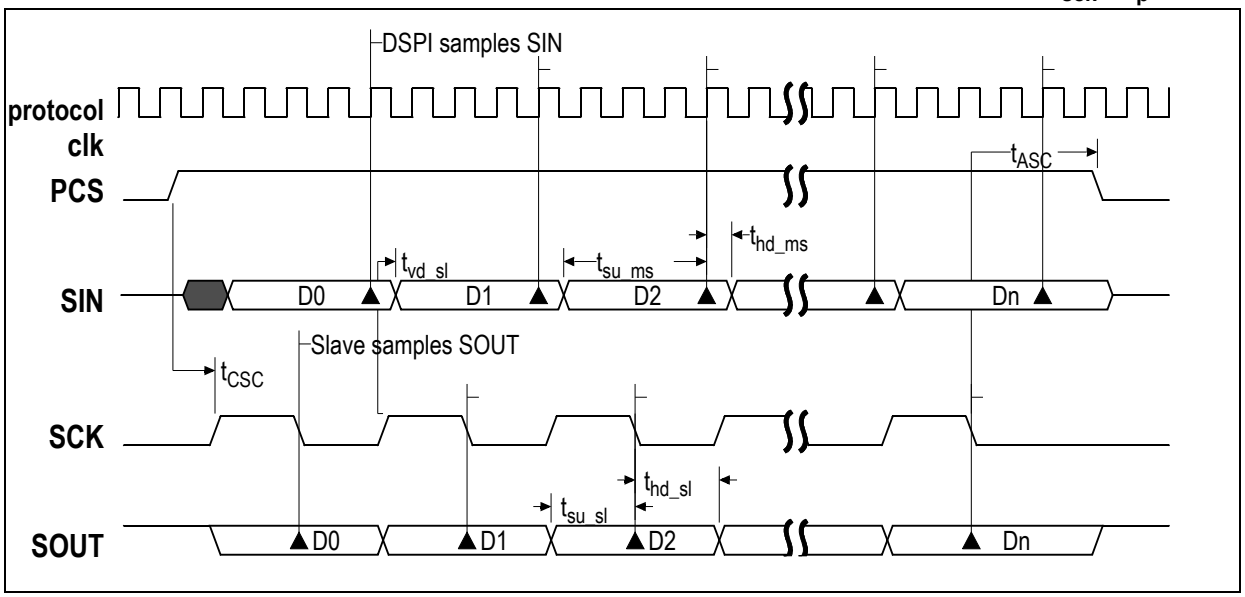
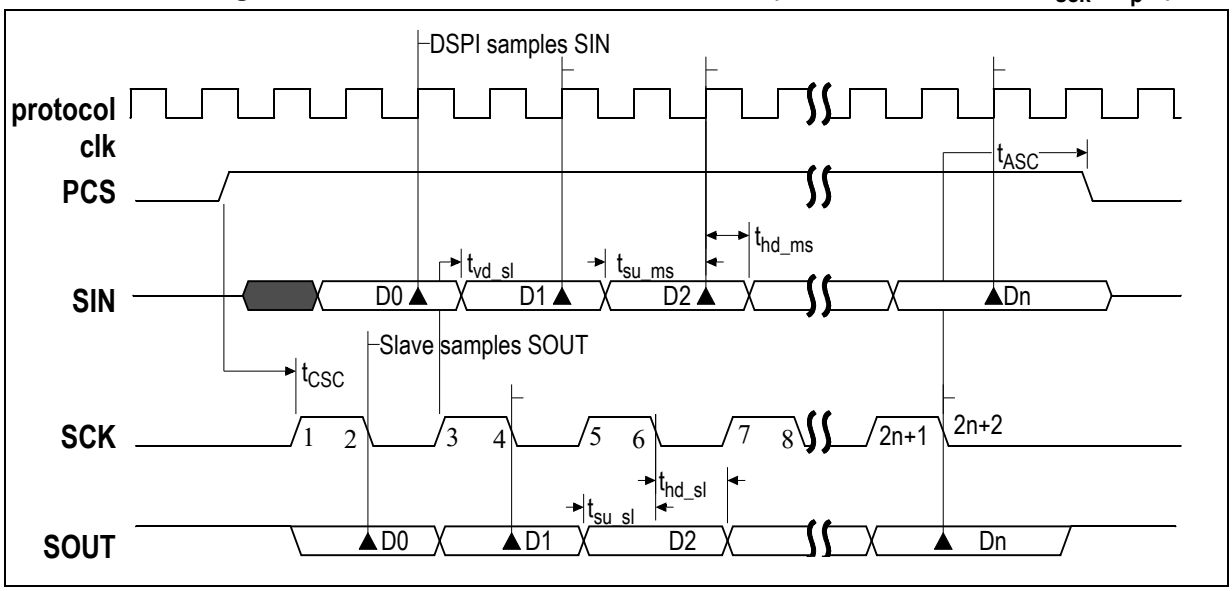


Figure 1641. DSPI Modified Transfer Format (MTFE = 1, CPHA = 1, $f_{sck} = f_p/2$)



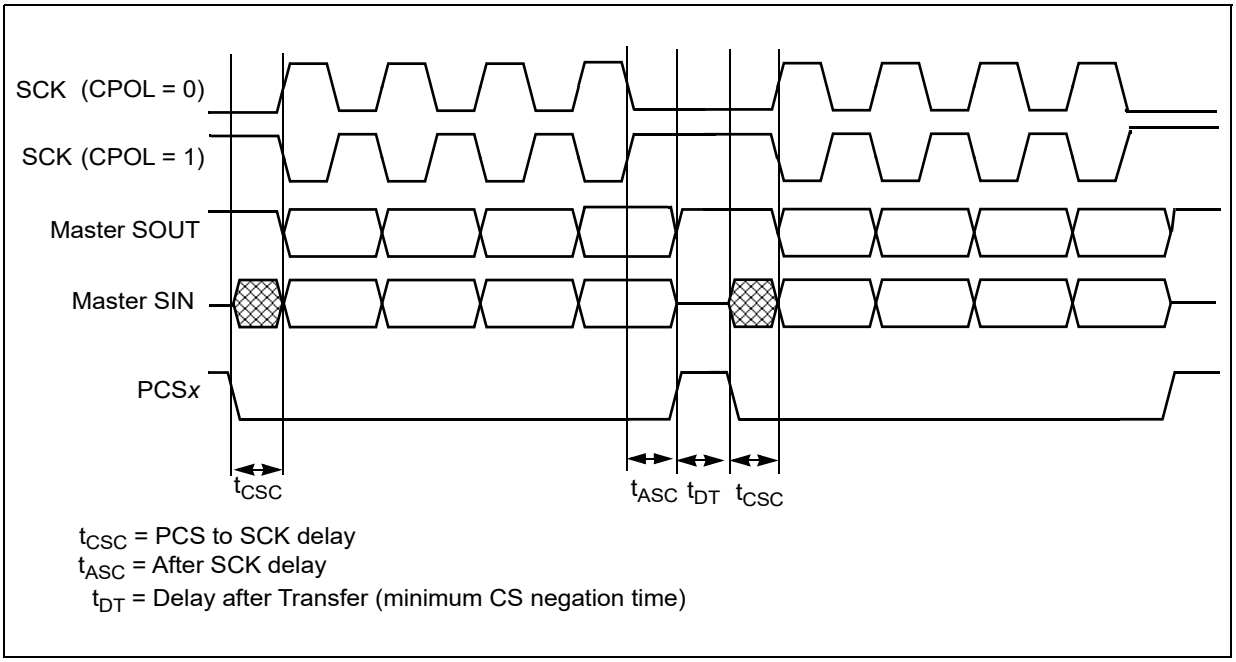
62.5.6.5 Continuous Selection Format

Some peripherals must be deselected between every transfer. Other peripherals must remain selected between several sequential serial transfers. The Continuous Selection Format provides the flexibility to handle both the following cases. The Continuous Selection Format is enabled for the SPI Configuration by setting the CONT bit in the SPI command. Continuous Selection is enabled for the DSI Configuration by setting the DCONT bit in the DSICR0.

The behavior of the PCS signals in the two configurations is identical so only SPI Configuration will be described. When the CONT bit = 0, the DSPI drives the asserted Chip Select signals to their idle states in between frames. The idle states of the Chip Select

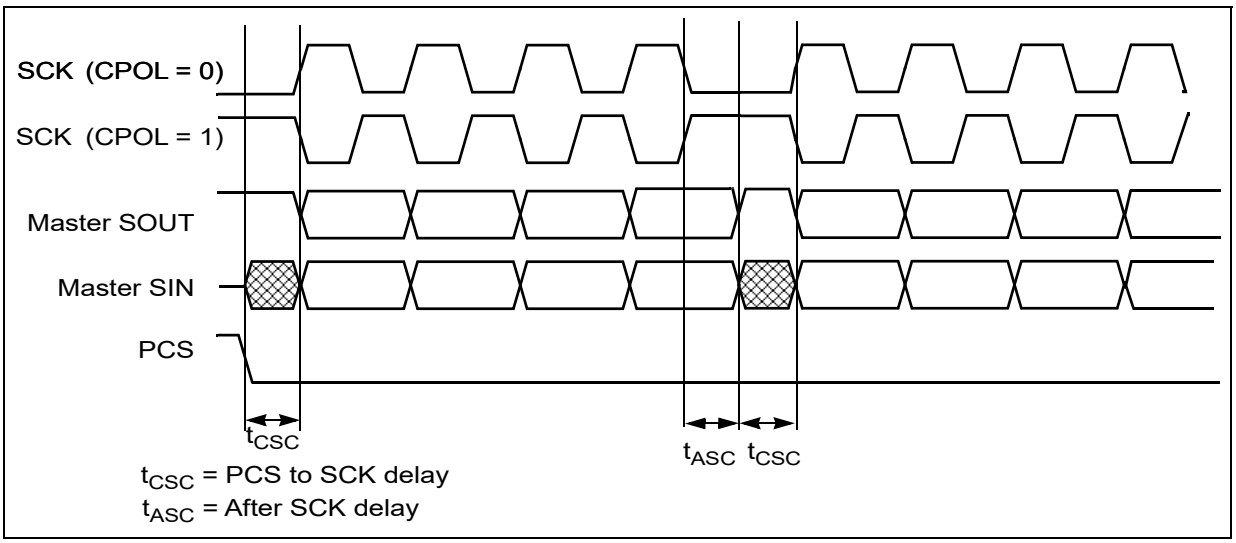
signals are selected by the PCSIS*n* bits in the MCR. The timing diagram in *Figure 1642* is for two four-bit transfers with CPHA = 1 and CONT = 0.

Figure 1642. Example of non-continuous format (CPHA = 1, CONT = 0)



When the CONT bit = 1, the PCS signal remains asserted for the duration of the two transfers. The Delay between Transfers (t_{DT}) is not inserted between the transfers. *Figure 1643* shows the timing diagram for two four-bit transfers with CPHA = 1 and CONT = 1.

Figure 1643. Example of continuous transfer (CPHA = 1, CONT = 1)



When using DSPI with continuous selection follow these rules:

- All transmit commands must have the same PCS*n* bits programming.
- The CTARs selected by transmit commands must be programed with the same transfer attributes. Only FMSZ field can be programed differently in these CTARs.
- When transmitting multiple frames in this mode, the user software must ensure that the last frame has the PUSHR[CONT] bit de-asserted (in master mode) and the user software must provide sufficient frames in the TX_FIFO to be sent out (in slave mode) and the master de-asserts the PCS*n* at end of transmission of last frame.
- The PUSHR[CONT]/DSICR0[DCONT] bits must be de-asserted before asserting MCR[HALT] bit (in master mode). This ensures that the PCS*n* signals are deasserted. Asserting MCR[HALT] bit during continuous transfer will cause the PCS*n* signals to remain asserted and hence Slave Device cannot transition from RUNNING to STOPPED state.

Note: You must fill the TXFIFO with the number of entries that will be concatenated together under one PCS assertion for both master and slave before the TXFIFO becomes empty.

Note: When operating in slave mode, ensure that when the last-entry in the TXFIFO is completely transmitted (that is the corresponding TCF flag is asserted and TXFIFO is empty), the slave is deselected for any further serial communication; otherwise, an underflow error occurs.

62.5.6.6 Fast Continuous Selection Format

The Fast Continuous Selection Format functions similar to the Continuous Selection Format except that the inter command delays, t_{ASC} and t_{CSC}, can be masked out and are not inserted by the hardware.

Note: The Fast Continuous Selection Format is available in the SPI configuration only and when Continuous Serial Communication Clock mode is disabled. Masking of delays is not allowed in DSI and CSI configurations or if the transfer is non-continuous.

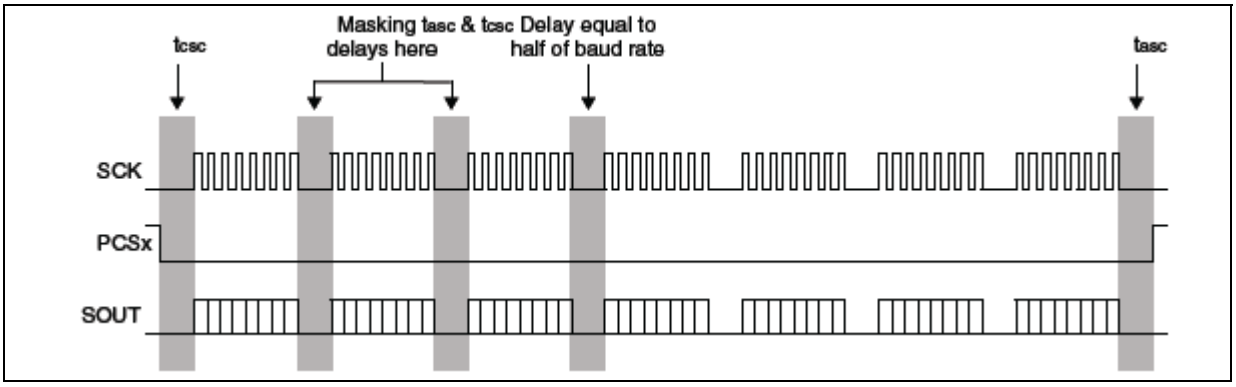
The Fast Continuous Selection Format is enabled by writing ‘1’ into FCPCS bit of the MCR. When this bit is asserted, MASC and MCSC bits of the PUSHR perform the function of mask bits for the transmit frame. These bits individually mask the t_{ASC} and t_{CSC} delays as programed by the user software. A normal Continuous Selection Format has these two delays for each frame that is transmitted with the CONT bit asserted. In order to avoid these delays and speed up the transfer process, the software can simply mask these delays while programming the command in the PUSHR.

While masking the delays, the software must follow the following masking rules, else correct operation is not guaranteed.

- MASC bit masks the “After SCK” delay for the current frame.
- MCSC bit masks the “PCS to SCK” delay for the next frame.
- “After SCK” (t_{ASC}) delay must not be masked when the current frame is the last frame in the continuous selection format.
- The “PCS to SCK” delay for the first frame in the continuous selection format cannot be masked.
- Masking of only t_{ASC} is not allowed. If t_{ASC} is masked then t_{CSC} must be masked too.
- Masking of both t_{ASC} and t_{CSC} delays is allowed. In this case, the delay between two frames is equal to half the baud rate set by the user software.
- Masking of only t_{CSC} is allowed. In this case, the delay between two frames is equal to the t_{ASC} time and thus the user software must ensure that the t_{ASC} time is greater than the baud rate.
- The user software must not mask these delays if the continuous selection format is not used and MCR[FCPCS] is asserted.
- Rules applicable to the Continuous Selection Format are applicable here, too.

Figure 1644 shows the timing for a Fast Continuous Selection Format transfer. Here seven frames are transferred with both t_{ASC} and t_{CSC} delays masked except for the last frame that terminated the transfer. The last frame has t_{ASC} delay at its end.

Figure 1644. Example of Fast Continuous Selection Format



In case any chip select is to be changed, then the fast continuous selection format should be terminated and then the chips selects should change and appropriate delays must be introduced.

62.5.7 Continuous Serial Communications Clock

The DSPI provides the option of generating a continuous SCK signal for slave peripherals that require a continuous clock.

Continuous SCK is enabled by setting the CONT_SCKE bit in the MCR. Enabling this bit generates the Continuous Serial Communications Clock regardless of the MCR[HALT] bit status. Continuous SCK is valid in all configurations.

Continuous SCK is only supported for CPHA = 1. Clearing CPHA is ignored if the CONT_SCKE bit is set. Continuous SCK is supported for Modified Transfer Format.

Clock and transfer attributes for the Continuous SCK mode are set according to the following rules:

- When the DSPI is in SPI configuration, CTAR0 is used initially. At the start of each SPI frame transfer, the CTAR specified by the CTAS for the frame is used.
- When the DSPI is in DSI configuration, the CTAR specified by the DSICTAS field is used at all times.
- When the DSPI is in CSI configuration, the CTAR selected by the DSICTAS field is used initially.
  - At the start of a SPI frame transfer, the CTAR specified by the CTAS value for the frame is used.
  - At the start of a DSI frame transfer, the CTAR specified by the DSICTAS field is used.
- In all configurations, the currently selected CTAR remains in use until the start of a frame with a different CTAR specified, or the Continuous SCK mode is terminated.

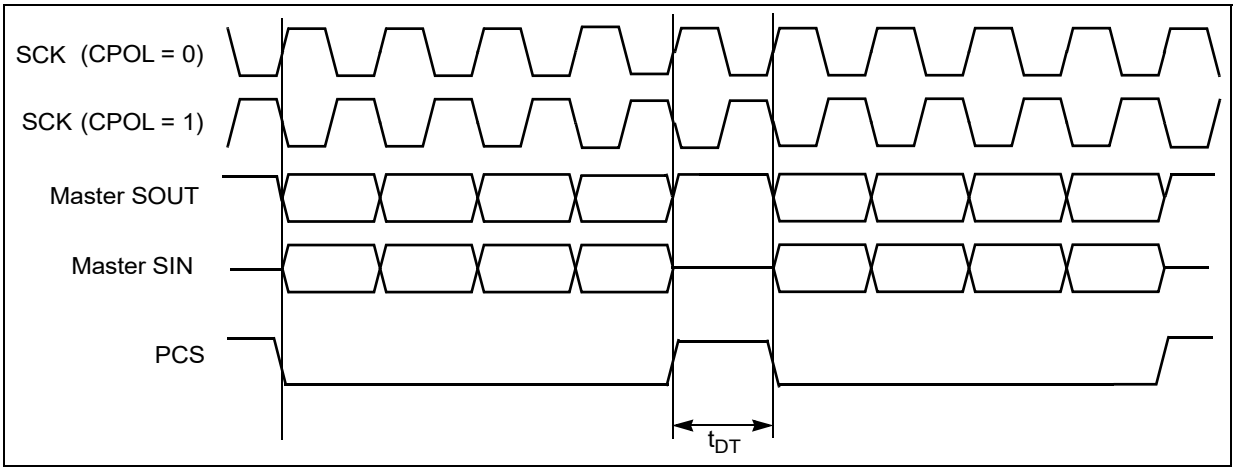
It is recommended to keep the baud rate the same while using the Continuous SCK. Switching clock polarity between frames while using Continuous SCK can cause errors in the transfer. Continuous SCK operation is not guaranteed if the DSPI is put into the External Stop mode or Module Disable mode.

Enabling Continuous SCK disables the PCS to SCK delay and the Delay after Transfer (t_{DT}) is fixed to one SCK cycle. When TSB configuration is enabled the t_{DT} is programmable from 1 to 64 SCK cycles.

Figure 1645 is the timing diagram for Continuous SCK format with Continuous Selection disabled.

Note: When in Continuous SCK mode, for the SPI transfer CTAR0 should always be used, and the TXFIFO must be cleared using the MCR[CLR_TXF] field before initiating transfer.

Figure 1645. Continuous SCK timing diagram (CONT = 0)



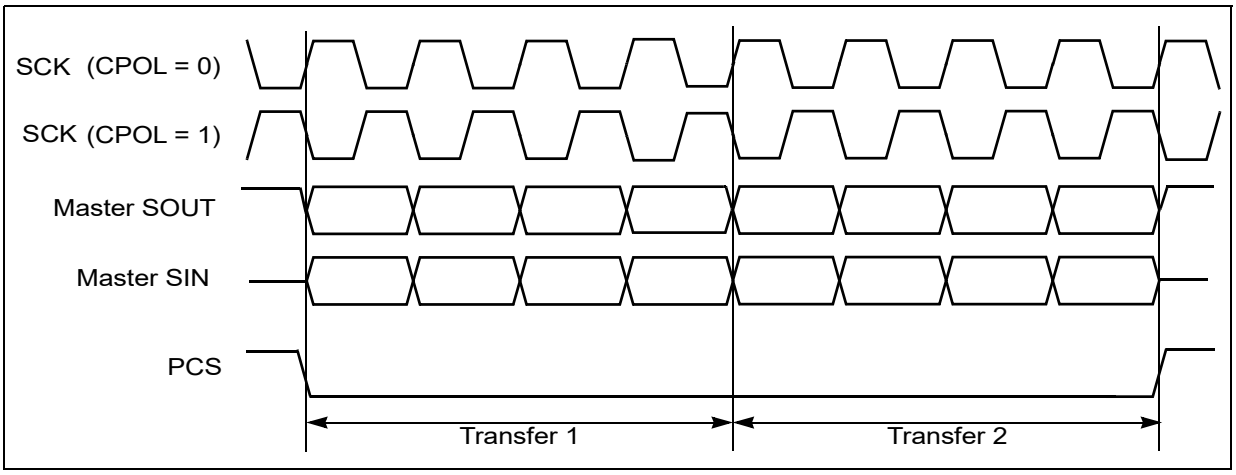
If the CONT bit in the TX FIFO entry is set or the DCONT in the DSICR0 is set, PCS remains asserted between the transfers. Under certain conditions, SCK can continue with

PCS asserted, but with no data being shifted out of SOUT (SOUT pulled high). This can cause the slave to receive incorrect data. Those conditions include:

- Continuous SCK with CONT bit set, but no data in the transmit FIFO.
- Continuous SCK with CONT bit set and entering STOPPED state (refer to *Section 62.5.1: Start and Stop of DSPI transfers*).
- Continuous SCK with CONT bit set and entering Stop mode or Module Disable mode.

Figure 1646 shows the timing diagram for Continuous SCK format with Continuous Selection enabled.

Figure 1646. Continuous SCK Timing Diagram (CONT = 1)



62.5.8 Slave mode operation constraints

Slave mode logic shift register is buffered. This allows data streaming operation, when the DSPI is permanently selected and data is shifted in with a constant rate.

The transmit data is transferred at second SCK clock edge of the each frame to the shift register if the $\overline{SS}$ signal is asserted and any time when transmit data is ready and SS signal is negated.

Received data is transferred to the receive buffer at last SCK edge of each frame, defined by frame size programed to the CTAR0/1 register. Then the data from the buffer is transferred to the RXFIFO or DDR1/0 register.

If the $\overline{SS}$ negates before that last SCK edge, the data from shift register is lost.

62.5.9 Timed Serial Bus (TSB)

The DSPI can be programed in Timed Serial Bus configuration by setting DSICR0[TSBC] bit. Refer to *Section 62.3.13: DSPI DSI Configuration Register 0 (DSPI_DSICR0)* for details.

TSB configuration provides the Micro Second Channel (MSC) downstream channel support.

The MSC upstream channel is not supported by the DSPI, but can be supported by any available Serial Communication Controller (SCI or UART) in the device.

To work in TSB mode the DSPI must be in master mode and in DSI (DCONF = 0b01) or CSI (DCONF = 0b10) configuration. Both Continuous and Non Continuous Serial Communication Clock (controlled by the DSPI_MCR[CONT_SCKE] bit) are supported in the TSB mode.

Figure 1647 shows the signals used in the TSB interface.

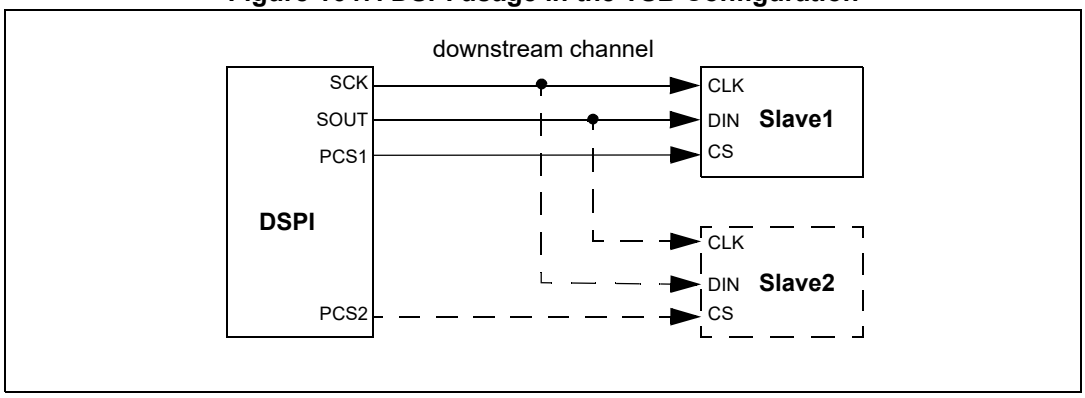
In the TSB configuration the DSPI is able to send from 4 to 66 bits MSC data frames (4 to 64 serialized data bits and up to 2 Data Selection zero bits). The serialized data bits source can be either:

- the DSPI DSI Alternate Serialization Data Register (DSPI_ASDR1/0), written by the host software,
- parallel input pin states latched into the DSPI DSI Serialization Data Register (DSPI_SDR1/0).

DSICR0[TXSS] bit or DSPI_SSR1/0 register bits define the source of the data.

The Least Significant Bits of the DSPI_ASDR0 or DSPI_SDR0 registers are selected to be serialized if the data frame is set to less than 32 bits.

Figure 1647. DSPI usage in the TSB Configuration



The PCS signals are driven together with SOUT. The t_{CSC} and t_{ASC} delays are not available. Delay after Transfer (DT) is set in SCK clock periods as a binary number formed by concatenation of the DSPI_CTAR*n* PDT and DT fields plus one, allowing to set DT from 1 to 64 serial clock periods. DT field provides least significant bits and PDT field provides most significant bits of the Delay after Transfer.

Figure 1648. TSB Downstream Frames

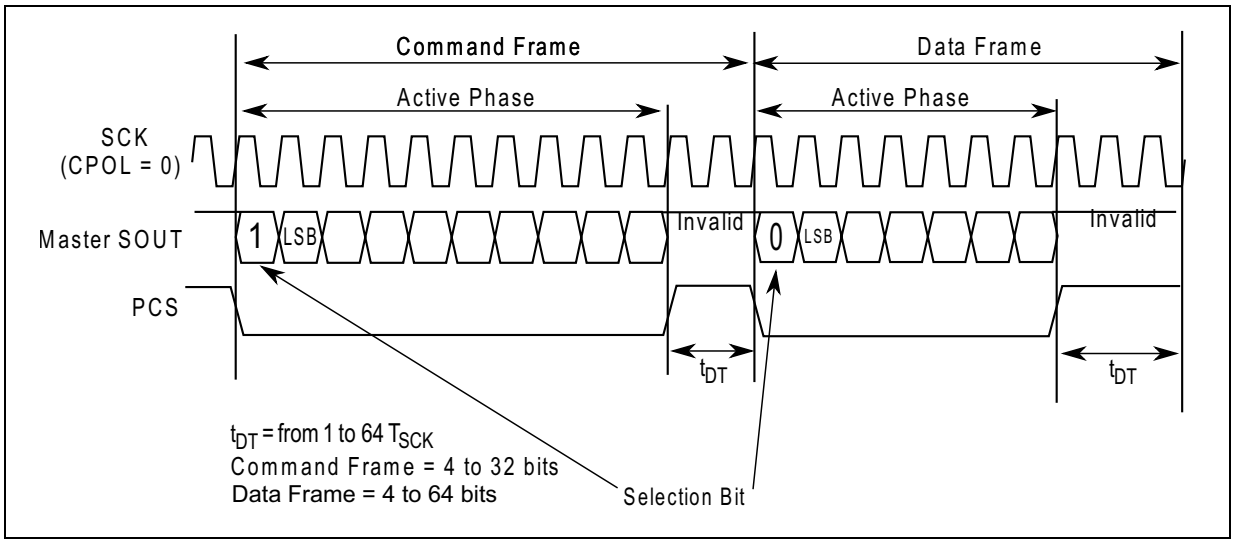


Figure 1648 shows two types of MSC downstream frames: command frame, and data frame.

The first transmitted bit, called the selection bit, determines the frame type:

- The selection bit '0' indicates a data frame
- The selection bit '1' indicates a command frame

Data frame may contain up to 2 selection bits to support two external slave devices, (so called dual receiver configuration) or no selection bits at all.

The command frame can be written by software, through SPI TX FIFO, using one or two FIFO entries with help of the CONT bit or MCR[XSPI]. The data frame consists of up to 64 bits from the SDR1/0 or ASDR1/0 registers and up to two zero selection bits. Number of data bits in the data frame is defined by the DSCICR1[TSBCNT] field.

The selection bit of the MSC command frames (1) can be implemented by software.

The selection bits in the data frames are enabled by DSPI_DSICR1 DSE0 and DSE1 bits. Each DSE*n* bit set increases the data frame size by one bit.

To comply with MSC specification, set DSPI_CTAR*n*[LSBFE] to transmit least significant bit first.

Regardless of the LSBFE bit setting, the data frame selection bits, if enabled, are always transmitted first, before the corresponding data subframes.

62.5.9.1 MSC Dual Receiver Support with PCS Switchover

When in TSB mode it is possible to switch the set of PCS signals that are driven during the first part of the frame to a different set of PCS signals during the second part of the frame. The bit, at which this switchover occurs, is defined by FMSZ field of the DSPI_CTAR*n* register, selected by DSICTAS field of the DSICR0 register. If DSICR1[DSI64E] is enabled, then the bit at which the PCS switch occurs, is defined by the concatenation of {DSICR0[FMSZ4], CTAR[FMSZ]} fields.

Note: When using Dual receiver support, it must be ensured that the number of bits to be transmitted to each slave must be at least 4.

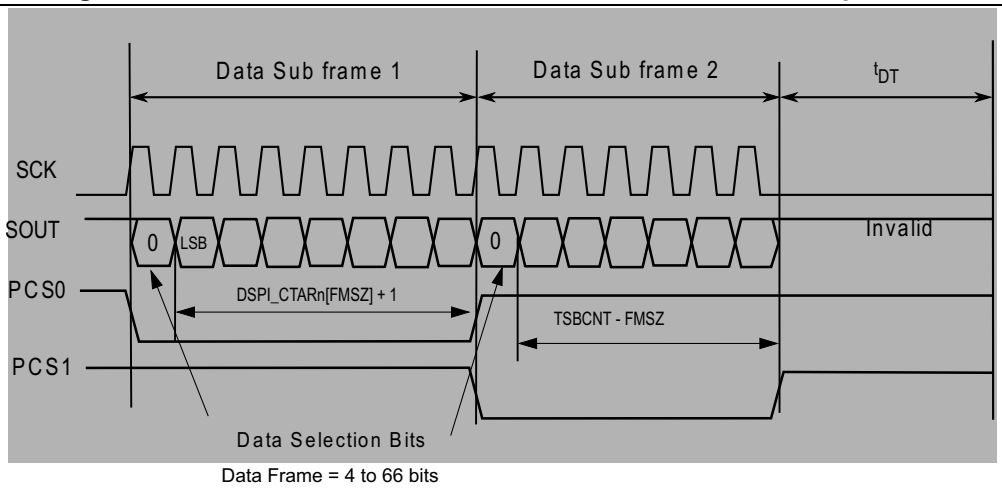
Number of the bits, not including the Data Selection Bit, in the first part of the frame is equal to value of the FMSZ field plus one. (This value changes to {FMSZ4,FMSZ} plus one, when DSICR1[DSI64E] is enabled). During this part of the frame the PCS signal levels are controlled by DSPI_DSICR0 DPCSn bits, after that by DSPI_DSICR1 DPCS1_n bits.

The PCS switchover occurs at the driving edge of the SCK clock output.

The second Data Selection Bit is inserted after the PCS switchover if enabled.

The Data Frame with PCS switchover is shown in *Figure 1649*.

Figure 1649. TSB Data Frame Format for MSC Dual Receiver Operation



62.5.10 Interleaved TSB (ITSB) Mode

This mode is similar to the TSB in all aspects except for the differences mentioned in *Table 1642*. In the Interleaved TSB or ITSB mode, frame transmission is governed by the following rules:

- Transmission of frames is executed on a periodic trigger that is generated internally to the DSPI. The trigger period is calculated as follows:
  - Trigger Period = 64 bits (maximum data frame size) + 2 selection bits (maximum bits in data frame) + t_{PT}

    t_{PT} is the passive frame time or dead time inserted to have constant trigger period.
  - The user software shall configure the trigger period and trigger source.
- If the previously transmitted frame was a SPI frame or the TX FIFO (or CMD FIFO) is empty, the frame transmitted on subsequent trigger will always be a DSI frame.
- If the previously transmitted frame was not a SPI frame and the TX_FIFO (and CMD FIFO) is not empty, the frame transmitted on subsequent trigger will always be a SPI frame.

This mode does not change the way DSI and SPI frame transmissions are done except that ITSB mode interleaves the source (SPI or DSI) in such a way to prevent back to back SPI frames. All features of SPI and DSI mode can be used except for the Continuous Frame Selection format (PUSHR[CONT] and DSICR0[DCONT] bits cannot be asserted).

MCR[MTFE] should be set to 0 during ITSB Mode of operation, since MicroSecond Channel upstream operation is not supported.

Table 1642 lists the differences between the TSB and ITSB modes.

Table 1642. Differences between TSB and ITSB modes

Mode attribute	TSB mode	ITSB mode
Frame priority	Priority available to frames from SPI. Back to back SPI frames are sent out.	No priority. Messages are interleaved such that there are no back-to-back SPI frames
Frame selection bit	Selection bit inserted for DSI frames only. Selection bit for SPI frames is inserted by software.	Selection bit for both DSI and SPI frames inserted automatically. Encoding of selection bit same as TSB mode (0 = DSI; 1 = SPI).
Frame transmission	Occasionally triggered transmission (that is trigger is received via hardware trigger input).	Periodically triggered transmission (that is frames are transmitted continuously on every trigger).
Frame gap	Gap between frames is configurable. Time between frames is t_{DT} which is configurable from 1 to 64 baud rate clock cycles.	Gap between frames is inserted automatically as the whole operation is based on a periodic trigger. If a frame has less than 64-bits to be transmitted, dead or passive time is automatically inserted by waiting for the next trigger pulse. Time t_{DT} is not used in this mode as it can be made part of the trigger period (as passive time).

In TSB and ITSB mode, separate frame completion interrupts are available to indicate frame transmission completion from SPI and DSI. MSC dual receiver support with PCS switchover is also supported in ITSB mode. Refer to *Section 62.5.9.1: MSC Dual Receiver Support with PCS Switchover* for more details.

The ITSB mode is suitable to implement the Downstream Channel for the MSC Bus because of the ITSB mode features listed in the table above.

62.5.10.1 Configuring DSPI for ITSB mode

The following steps give the recommended way to initialize the DSPI for the ITSB mode of operation:

- The DSPI should be put in HALT mode (by asserting the MCR[HALT] bit) before programming the registers.
- Set the DSICR0[TSBC] and DSICR0[ITSB] bits. The DSICR0[TRRE] and DSICR0[CID] bits should be cleared. DSICR0[DCONT] should not be used in ITSB mode. If DSICR0[ITSB] bit is set without setting DSICR0[TSBC], then the DSPI operates in the Normal Mode depending on MCR[DCONF] bit.
- Enable frame completion interrupts for either any frame completion (TCF) or separate frame completion (DSITCF and SPITCF) as required by application.
- When remainder configuration of DSPI is complete, clear the MCR[HALT] bit to allow DSPI to start operations.

Once configured and DSPI is brought out of HALT, the ITSB mode shall start sequencing the frames from either DSI or SPI as per the rules mentioned in *Section 62.5.10: Interleaved TSB (ITSB) Mode*.

62.5.10.2 Using ITSB mode for MSC downstream channel

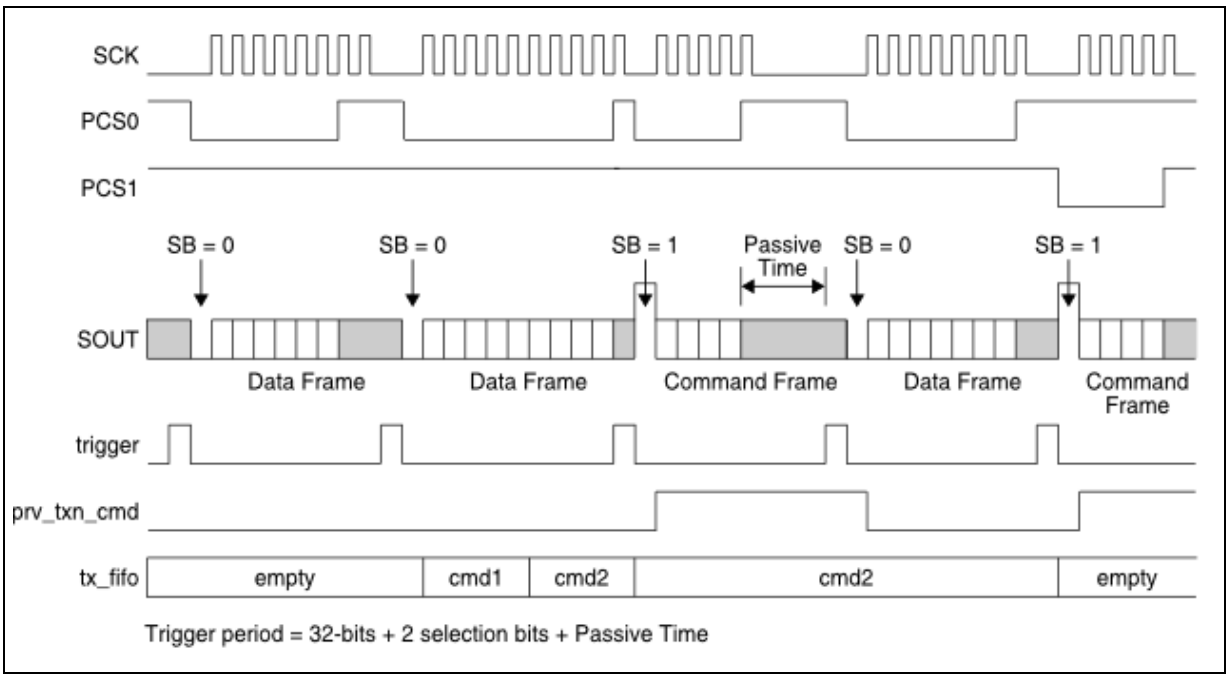
MSC Downstream Channel uses two types of frames: command frames and data frames. The Downstream operation of these frames require:

- Data frames can be sent back to back.
- Back to back command frames are not a legal scenario.
- If command frame has less bits than data frame, a gap is inserted to ensure next data frame starts at known reference time.
- In dual MSC ASIC, some bits of the data frame (which can be up to 64 bits) goes to first ASIC and remainder bits to second ASIC.
- In dual MSC ASIC, if a command frame is sent to one ASIC, it must not change the timing of the next data frame.

The ITSB mode can help implement the MSC Downstream Channel by allowing the command frames to be sent via the SPI interface and data frames to be sent via the DSI interface. Once the DSPI has been configured to work in ITSB mode (Refer to *Section 62.5.10.1: Configuring DSPI for ITSB mode*), the user software can program the DSI and SPI sides to send data and command frames respectively. The programming of frames for transmission via DSI or SPI is same as explained in *Section 62.5.4: Combined Serial Interface (CSI) configuration*.

Figure 1650 shows a sample MSC Downstream transmission using the ITSB mode.

Figure 1650. Sample MSC downstream transmission using ITSB mode



Brief description of operation includes:

- At the first trigger, since there are no command frames, a data frame is transmitted: to send a command frame at the start of operation, the user software must program a command frame in the TX_FIFO before enabling DSPI (MCR[HALT] = 0).
- The second trigger sets off the transmission of another data frame.
- Between the second and third trigger, two command frames are pushed into the TX_FIFO.
- At third trigger, a command frame is transmitted as the TX_FIFO is not empty: since only 16 bits or less are transmitted, no transmission happens till next trigger. This is marked as the “Passive Time”. The passive time is added in all frames after frame transmission completes and until next trigger pulse is detected.
- At fourth trigger, since the previous transmission was a command frame, a data frame is transmitted.
- At fifth trigger, since one command frame is pending in TX_FIFO and last transmitted frame was not a command frame, a command frame is transmitted.
- At sixth trigger (not shown), the process continues.

62.5.11 Parity generation and check

The DSPI module can generate and check parity in the serial frame. The parity bit replaces the last transmitted bit in the frame. The parity is calculated for all transmitted data bits in frame, not including the last data bit that would be transmitted. The parity generation/control is done on frame basis. The register fields setting frame size defines the total number of bits in the frame, including the parity bit. Thus, to transmit/receive the same number of data bits with parity check, increase the frame size by one versus the same data size frame without the parity check.

Parity can be selected as odd or even. Parity Errors in the received frame set Parity Error flags in the Status register. The Parity Error Interrupt Requests are generated if enabled. The DSPI module can be programed to stop SPI or/and DSI frame transmission in case of a frame reception with parity error.

62.5.11.1 Parity for SPI frames

When the DSPI is in the master mode the parity generation is controlled by PE and PP bits of the TX FIFO entries (PUSHR). Setting the PE bit enables parity generation for transmitted SPI frames and parity check for received frames. PP bit defines polarity of the parity bit.

When continuous PCS selection is used to transmit SPI data, two parity generation scenarios are available:

- Generate/check parity for the whole frame
- Generate/check parity for each sub-frame separately.

To generate/check parity for the whole frame set PE bit only in the last command/TX FIFO entry, forming this frame (with the PUSHR).

To generate/check parity for each sub-frame set PE bit in each command/TX FIFO entry, forming this frame.

If the parity error occurs for received SPI frame, the SR[SPEF] bit is set. If MCR[PES] bit is set, the DSPI stops SPI frames transmission. To resume SPI operation clear the SR[SPEF] or the MCR[PES] bits.

In slave mode, the parity is controlled by the PE and PP bits of the CTAR0 register similar to the master mode parity generation without continuous PCS selection.

62.5.11.2 Parity for DSI frames

When DSPI is in Master Mode, parity generation is controlled by PE and PP bits of the DSPI_DSICR0 register similar to the SPI frames. The parity is calculated and checked for each DSI frame. DSPI_DSICR0[DCONT] bit has no effect on parity generation. In slave mode, the parity is controlled by the PE and PP bits of the CTAR1 register.

If the parity error occurs for received DSI frame, the DSPI_SR[DPEF] bit is set. If DSPI_DSICR0[PES] bit is set, the DSPI stops DSI frames transmission. To resume DSI operation, clear the DSPI_SR[DPEF] or the DSPI_DSICR0[PES] bits.

62.5.12 Interrupts/DMA requests

Certain DSPI conditions can generate only interrupt requests and other DSPI conditions can generate either interrupt requests or DMA requests. *Table 1643* lists these conditions.

Table 1643. Interrupt and DMA request conditions

DSPI condition	Flag	Request type	
		Interrupt	DMA
End of Queue (EOQ)	EOQF	Yes	—
TX FIFO Fill	TFFF	Yes	Yes
CMD FIFO Fill	CMDFFF	Yes	Yes
TX FIFO Invalid Write	TFIWF	Yes	—
Transfer Complete	TCF	Yes	—
CMD Transfer Complete	CMDTCF	Yes	—
SPI Transfer Complete	SPITCF	Yes	—
DSI Transfer Complete	DSITCF	Yes	—
TX FIFO Underflow	TFUF	Yes	—
RX FIFO Drain	RFDF	Yes	Yes
RX FIFO Overflow	RFOF	Yes	—
SPI Parity Error	SPEF	Yes	—
DSI Parity Error	DPEF	Yes	—
DSI Deserialized Data Match	DDIF	Yes	Yes

Each condition has a flag bit in the DSPI Status Register (SR) and a Request Enable bit in the DSPI DMA/Interrupt Request Select and Enable Register (RSER). The TX FIFO Fill Flag (TFFF) and RX FIFO Drain Flag (RFDF) generate interrupt requests or DMA requests depending on the TFFF_DIRS and RFDF_DIRS bits in the RSER.

The DSPI module also provides a global interrupt request line, which is asserted when any of individual interrupt requests lines is asserted.

62.5.12.1 End of Queue interrupt request

The End of Queue Request indicates that the end of a transmit queue is reached. The End of Queue Request is generated when the EOQ bit in the executing SPI command is set and the EOQF_RE bit in the RSER is set.

When MCR[XSPI] is enabled and the EOQ bit in the executing SPI command is set, the End of Queue Request will only be generated once the last Data frame in the Command Cycle has been transmitted.

Note: This interrupt request is generated when the last bit of the SPI frame with EOQ bit set is transmitted.

62.5.12.2 Transmit FIFO Fill interrupt or DMA request

The Transmit FIFO Fill Request indicates that the TX FIFO is not full. The Transmit FIFO Fill Request is generated when the number of entries in the TX FIFO is less than the maximum number of possible entries, and the TFFF_RE bit in the RSER is set. The TFFF_DIRS bit in the RSER selects whether a DMA request or an interrupt request is generated.

Note: TFFF flag clears automatically when DMA is used to fill TXFIFO.

Note: To clear TFFF when not using DMA, follow these steps for every PUSH performed using CPU to fill TXFIFO:

Wait until TFFF = 1

Write data to PUSHR using CPU.

Clear TFFF by writing a '1' to its location. If TX FIFO is not full, this flag will not clear.

62.5.12.3 Command FIFO Fill interrupt or DMA request

The Command FIFO Fill Request indicates that the CMD FIFO is not full. The Command FIFO Fill Request is generated when the number of entries in the CMD FIFO is less than the maximum number of possible entries, and the CMDFFF_RE bit in the RSER is set. The CMDFFF_DIRS bit in the RSER selects whether a DMA request or an interrupt request is generated.

This Request is useful when MCR[XSPI] is enabled and hence TX FIFO and CMD FIFO can be filled independently. If MCR[XSPI] is disabled, then 'TX FIFO Fill Interrupt or DMA Request' will suffice to fill both FIFO's since both FIFO's must be filled simultaneously.

Note: CMDFFF flag clears automatically when DMA is used to fill CMD FIFO.

Note: To clear CMDFFF when not using DMA, follow these steps for every PUSH performed using CPU to fill CMD FIFO:

Wait until CMDFFF = 1

Write data to Command field of PUSHR using CPU.

Clear CMDFFF by writing a '1' to its location. If CMD FIFO is not full, this flag will not clear.

62.5.12.4 Transmit FIFO Invalid Write interrupt request

The Transfer Fifo Invalid Write Request is valid only when MCR[XSPI] is enabled. This Request indicates that Data exists in the TX FIFO while the CMD FIFO is empty. Since no Command Fields are associated with the Data present in TX FIFO, this data is considered invalid until a Command Entry becomes available. The Transfer Fifo Invalid Write Request is generated for the above condition when TFIWF_RE bit is set in the RSER.

62.5.12.5 Transfer Complete interrupt request

The Transfer Complete Request indicates the end of the transfer of a serial frame. The Transfer Complete Request is generated at the end of each frame transfer when the TCF_RE bit is set in the RSER.

62.5.12.6 Command Transfer Complete interrupt request

The Command Transfer Complete Request indicates the end of transfer of the last SPI frame in a Command Cycle. The Transfer Complete Request is generated for the above condition when the CMDTCF_RE bit is set in the RSER.

62.5.12.7 SPI Transfer Complete interrupt request

The SPI Transfer Complete Request indicates the end of transfer of an SPI frame in any Transmission Mode which uses DSPI in CSI Mode. The SPI Transfer Complete Request is generated when SR[SPITCF] flag asserts and SPITCF_RE bit is set in the RSER.

62.5.12.8 DSI Transfer Complete interrupt request

The DSI Transfer Complete Request indicates the end of transfer of a DSI frame in any Transmission Mode which uses DSPI in CSI Mode. The DSI Transfer Complete Request is generated when SR[DSITCF] flag asserts and DSITCF_RE bit is set in the RSER.

62.5.12.9 Transmit FIFO Underflow interrupt request

The Transmit FIFO Underflow request indicates that an underflow condition in the TX FIFO has occurred. The transmit underflow condition is detected only for the DSPI, operating in slave mode and SPI configuration. The TFUF bit is set when the TX FIFO of a DSPI is empty, and a transfer is initiated from an external SPI master. If the TFUF bit is set while the TFUF_RE bit in the RSER is set, an interrupt request is generated.

62.5.12.10 Receive FIFO Drain interrupt or DMA Request

The Receive FIFO Drain Request indicates that the RX FIFO is not empty. The Receive FIFO Drain Request is generated when the number of entries in the RX FIFO is not zero, and the RFDF_RE bit in the RSER is set. The RFDF_DIRS bit in the RSER selects whether a DMA request or an interrupt request is generated.

62.5.12.11 Receive FIFO Overflow Interrupt Request

The Receive FIFO Overflow Request indicates that an overflow condition in the RX FIFO has occurred. A Receive FIFO Overflow request is generated when RX FIFO and shift register are full and a transfer is initiated. The RFOF_RE bit in the RSER must be set for the interrupt request to be generated.

Depending on the state of the ROOE bit in the MCR, the data from the transfer that generated the overflow is either ignored or shifted in to the shift register. If the ROOE bit is set, the incoming data is shifted in to the shift register. If the ROOE bit is cleared, the incoming data is ignored.

62.5.12.12 SPI Frame Parity Error Interrupt Request

The SPI Frame Parity Error Flag indicates that a SPI frame with parity error had been received. The SPEF_RE bit in the RSER must be set for the interrupt request to be generated.

62.5.12.13 DSI Frame Parity Error Interrupt Request

The DSI Frame Parity Error Flag indicates that a DSI frame with parity error has been received. The DPEF_RE bit in the DSPI_RSER must be set for the interrupt request to be generated.

62.5.12.14 Deserialized Data Match Interrupt or DMA Request

The Deserialized Data Match Flag (DDIF) indicates that a DSI frame matches DSPI_DPIR1/0 data, masked with DSPI_DIMR1/0, has been received. This Request is generated if DDIF_RE bit in the DSPI_RSER is set. The DDIF_DIRS bit in the RSER selects whether a DMA request or an interrupt request is generated.

62.5.13 Power saving features

The DSPI supports following power-saving strategies:

- External Stop mode
- Module Disable mode – Clock gating of non-memory-mapped logic

Note: When in Slave Mode, the module can only be put in Low Power Modes (such as STOP and HALT) once the Master module has transitioned to the Low Power Mode as well.

62.5.13.1 Stop mode (External Stop Mode)

The DSPI supports the stop mode protocol. When a request is made to enter External Stop mode, the DSPI block acknowledges the request. If a serial transfer is in progress, the DSPI waits until it reaches the frame boundary before it is ready to have its clocks shut off. While the clocks are shut off, the DSPI memory-mapped logic is not accessible. This also puts the DSPI in STOPPED state. The SR[TXRXS] bit is cleared to indicate STOPPED state. The states of the interrupt and DMA request signals cannot be changed while in External Stop mode.

If a DSPI block is configured in Slave mode, External Stop mode requests must be run after the last clock cycle or Low-power mode is not entered.

62.5.13.2 Module disable mode

Module disable mode is a block-specific mode that the DSPI can enter to save power. The host CPU can initiate the module disable mode by setting the MDIS bit in the MCR. The module disable mode can also be initiated by hardware.

When the MDIS bit is set, the DSPI negates Clock Enable signal at the next frame boundary. Once the Clock Enable signal is negated, DSPI is said to have entered Module Disable Mode. This also puts the DSPI in STOPPED state. The SR[TXRXS] bit is cleared to indicate STOPPED state.

If implemented, the Clock Enable signal can stop the clock to the non-memory-mapped logic. When Clock Enable is negated, the DSPI is in a dormant state, but the memory-mapped registers are still accessible.

Certain read or write operations have a different effect when the DSPI is in the module disable mode:

- Reading the RX FIFO Pop register does not change the state of the RX FIFO.
- Writing to the FIFO Push register does not change the state of the TX FIFO or CMD FIFO.
- Clearing of the FIFOs has no effect.
- Changes to the DIS_TXF and DIS_RXF fields of the MCR have no effect.
- All status bits and register flags in the DSPI return the correct values when read, but writing to them has no effect.
- Writing to the TCR during module disable mode has no effect.
- Interrupt and DMA request signals cannot be cleared.

62.6 Initialization/application information

This section describes how to initialize the DSPI module.

62.6.1 Managing DSPI queues

The queues are not part of the DSPI, but the DSPI includes features in support of queue management. Queues are primarily supported in SPI Configuration.

1. When DSPI executes last command word from a queue, the EOQ bit in the command word is set to indicate to the DSPI that this is the last entry in the queue.
2. At the end of the transfer, corresponding to the command word with EOQ set is sampled, the EOQ flag (EOQF) in the SR is set.
3. The setting of the EOQF flag disables serial transmission and reception of data, putting the DSPI in the STOPPED state. The TXRXS bit is cleared to indicate the STOPPED state.
4. The DMA can continue to fill TX FIFO until it is full or step 5 occurs.
5. Disable DSPI DMA transfers by disabling the DMA enable request for the DMA channel assigned to TX FIFO (and CMD FIFO) and RX FIFO. This is done by clearing the corresponding DMA enable request bits in the DMA Controller.
6. Ensure all received data in RX FIFO has been transferred to memory receive queue by reading the RXCNT in SR or by checking RFDF in the SR after each read operation of the POPR.
7. Modify DMA descriptor of TX and RX channels for new queues
8. Flush TX FIFO (and CMD FIFO) by writing a ‘1’ to the CLR_TXF bit in the MCR. Flush RX FIFO by writing a ‘1’ to the CLR_RXF bit in the MCR.
9. Clear transfer count either by setting CTCNT bit in the command word of the first entry in the new queue or via CPU writing directly to SPI_TCNT field in the TCR.
10. Enable DMA channel by enabling the DMA enable request for the DMA channel assigned to the DSPI TX FIFO, (and CMD FIFO) and RX FIFO by setting the corresponding DMA set enable request bit.
11. Enable serial transmission and serial reception of data by clearing the EOQF bit.

62.6.2 Switching master and slave mode

When changing modes in the DSPI, follow the steps below to guarantee proper operation.

1. Halt the DSPI by setting MCR[HALT].
2. Clear the transmit and receive FIFOs by writing a ‘1’ to the CLR_TXF and CLR_RXF bits in MCR.
3. Set the appropriate mode in MCR[MSTR] and enable the DSPI by clearing MCR[HALT].

62.6.3 Initializing DSPI in Master/Slave Modes

Once the appropriate mode in MCR[MSTR] is configured, the DSPI is enabled by clearing MCR[HALT]. It should be ensured that DSPI Slave is enabled before enabling DSPI Master. This ensures the Slave is ready to be communicated with, before Master initializes communication.

62.6.4 Baud rate settings

Table 1644 shows the baud rate that is generated based on the combination of the baud rate prescaler PBR and the baud rate scaler BR in the CTARs. The values calculated assume a 100 MHz protocol frequency and the double baud rate DBR bit is clear.

Note: The protocol clock frequency mentioned above is given as an example in this chapter. Refer to the Clocking chapter for the frequency used to drive this module in the device.

Table 1644. Baud rate values (bps)

		Baud rate divider prescaler values (DSPI*n*_CTAR[PBR])			
		2	**3**	**5**	**7**
Baud Rate scaler values (DSPI*n*_CTAR[BR])	**2**	25.0 MHz	16.7 MHz	10.0 MHz	7.14 MHz
	4	12.5 MHz	8.33 MHz	5.00 MHz	3.57 MHz
	6	8.33 MHz	5.56 MHz	3.33 MHz	2.38 MHz
	8	6.25 MHz	4.17 MHz	2.50 MHz	1.79 MHz
	16	3.12 MHz	2.08 MHz	1.25 MHz	893 kHz
	32	1.56 MHz	1.04 MHz	625 kHz	446 kHz
	64	781 kHz	521 kHz	312 kHz	223 kHz
	128	391 kHz	260 kHz	156 kHz	112 kHz
	256	195 kHz	130 kHz	78.1 kHz	55.8 kHz
	512	97.7 kHz	65.1 kHz	39.1 kHz	27.9 kHz
	1024	48.8 kHz	32.6 kHz	19.5 kHz	14.0 kHz
	2048	24.4 kHz	16.3 kHz	9.77 kHz	6.98 kHz
	4096	12.2 kHz	8.14 kHz	4.88 kHz	3.49 kHz
	8192	6.10 kHz	4.07 kHz	2.44 kHz	1.74 kHz
	16384	3.05 kHz	2.04 kHz	1.22 kHz	872 Hz
	32768	1.53 kHz	1.02 kHz	610 Hz	436 Hz

62.6.5 Delay settings

Table 1645 shows the values for the Delay after Transfer (t_{DT}) and CS to SCK Delay (t_{CSC}) that can be generated based on the prescaler values and the scaler values set in the CTARs. The values calculated assume a 100 MHz protocol frequency.

Note: The protocol clock frequency mentioned above is given as an example in this chapter. Refer to the Clocking chapter for the frequency used to drive this module in the device.

This table does not apply for TSB Continuous mode.

Table 1645. Delay values

		Delay prescaler values (DSPI*n*_CTAR[PBR])			
		1	**3**	**5**	**7**
Delay scaler values	**2**	20.0 ns	60.0 ns	100.0 ns	140.0 ns
	4	40.0 ns	120.0 ns	200.0 ns	280.0 ns
	8	80.0 ns	240.0 ns	400.0 ns	560.0 ns
	16	160.0 ns	480.0 ns	800.0 ns	1.1 µs
	32	320.0 ns	960.0 ns	1.6 µs	2.2 µs
	64	640.0 ns	1.9 µs	3.2 µs	4.5 µs
	128	1.3 µs	3.8 µs	6.4 µs	9.0 µs
	256	2.6 µs	7.7 µs	12.8 µs	17.9 µs
	512	5.1 µs	15.4 µs	25.6 µs	35.8 µs
	1024	10.2 µs	30.7 µs	51.2 µs	71.7 µs
	2048	20.5 µs	61.4 µs	102.4 µs	143.4 µs
	4096	41.0 µs	122.9 µs	204.8 µs	286.7 µs
	8192	81.9 µs	245.8 µs	409.6 µs	573.4 µs
	16384	163.8 µs	491.5 µs	819.2 µs	1.1 ms
	32768	327.7 µs	983.0 µs	1.6 ms	2.3 ms
	65536	655.4 µs	2.0 ms	3.3 ms	4.6 ms

62.6.6 Calculation of FIFO pointer addresses

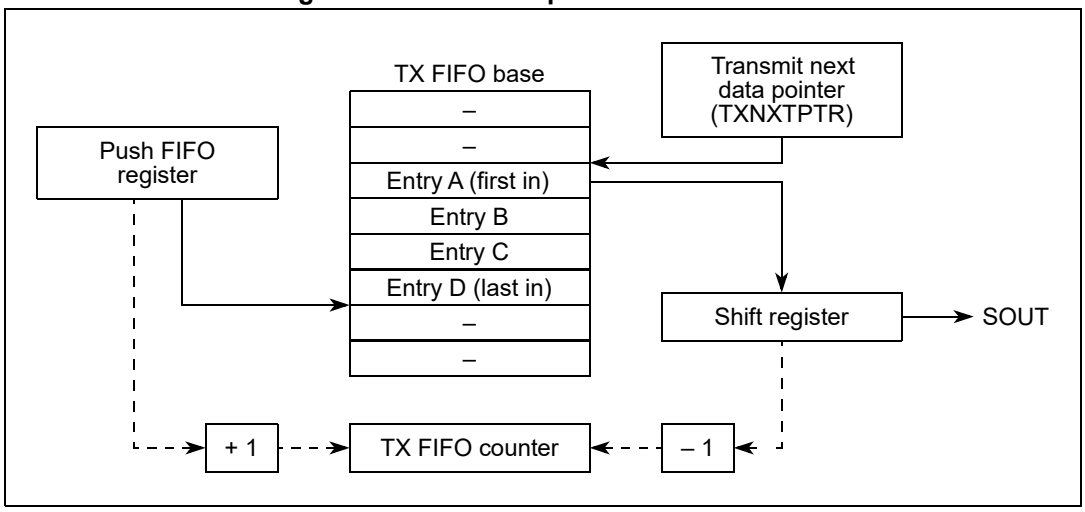
Complete visibility of the TX and CMD and RX FIFO contents is available through the FIFO registers, and valid entries can be identified through a memory-mapped pointer and a memory-mapped counter for each FIFO. The pointer to the first-in entry in each FIFO is memory-mapped.

- For TX FIFO, the first-in pointer is the Transmit Next Pointer (TXNXTPTR).
- For CMD FIFO, the first-in pointer is the Command Next Pointer (CMDNXTPTR).
- For RX FIFO, the first-in pointer is the Pop Next Pointer (POPNXTPTR).

Figure 1651 illustrates the concept of first-in and last-in FIFO entries along with the FIFO Counter. The TX FIFO is chosen for the illustration, but the concepts are applicable to the CMD FIFO and RX FIFO.

Refer to *Section 62.5.2.4: Transmit First In First Out (TX FIFO) buffering mechanism*, *Section 62.5.2.5: Command First In First Out (CMD FIFO) buffering mechanism* and *Section 62.5.2.6: Receive First In First Out (RX FIFO) Buffering Mechanism* for details on the FIFO operation.

Figure 1651. TX FIFO pointers and counter



62.6.6.1 Address calculation for the first-in entry and last-in entry in the TX FIFO

The memory address of the first-in entry in the TX FIFO is computed by the following equation:

Equation 85

$$First-inEntryAddress = TXFIFOBase + (4 \times TXNXTPTR)$$

The memory address of the last-in entry in the TX FIFO is computed by the following equation:

Equation 86

$$Last-inEntryAddress = TXFIFOBase + 4 \times (TXCTR + TXNXTPTR - 1) \bmod (TXFIFOdepth)$$

TX FIFO Base – Base address of TX FIFO

TXCTR – TX FIFO Counter

TXNXTPTR – Transmit Next Pointer

TX FIFO Depth – Transmit FIFO depth, implementation specific

62.6.6.2 Address calculation for the first-in entry and last-in entry in the CMD FIFO

The memory address of the first-in entry in the CMD FIFO is computed by the following equation:

Equation 87

$$First-inEntryAddress = CMDFIFOBase + (4 \times CMDNXTPTR)$$

The memory address of the last-in entry in the CMD FIFO is computed by the following equation:

Equation 88

$$Last-inEntryAddress = CMDFIFOBase + 4 \times (CMDCTR + CMDNXTPTR - 1) mod(CMDFIFOdepth)$$

CMD FIFO Base – Base address of CMD FIFO
CMDCTR – CMD FIFO Counter
CMDNXTPTR – Command Next Pointer
CMD FIFO Depth – Command FIFO depth, implementation specific

62.6.6.3 Address calculation for the first-in entry and last-in entry in the RX FIFO

The memory address of the first-in entry in the RX FIFO is computed by the following equation:

Equation 89

$$First-inEntryAddress = RXFIFOBase + (4 \times POPNXTPTR)$$

The memory address of the last-in entry in the RX FIFO is computed by the following equation:

Equation 90

$$Last-inEntryAddress = RXFIFOBase + 4 \times (RXCTR + POPNXTPTR - 1) mod(RXFIFOdepth)$$

RX FIFO Base – Base address of RX FIFO
RXCTR – RX FIFO counter
POPNXTPTR – Pop Next Pointer
RX FIFO Depth – Receive FIFO depth, implementation specific

63 LINFlexD

63.1 Introduction

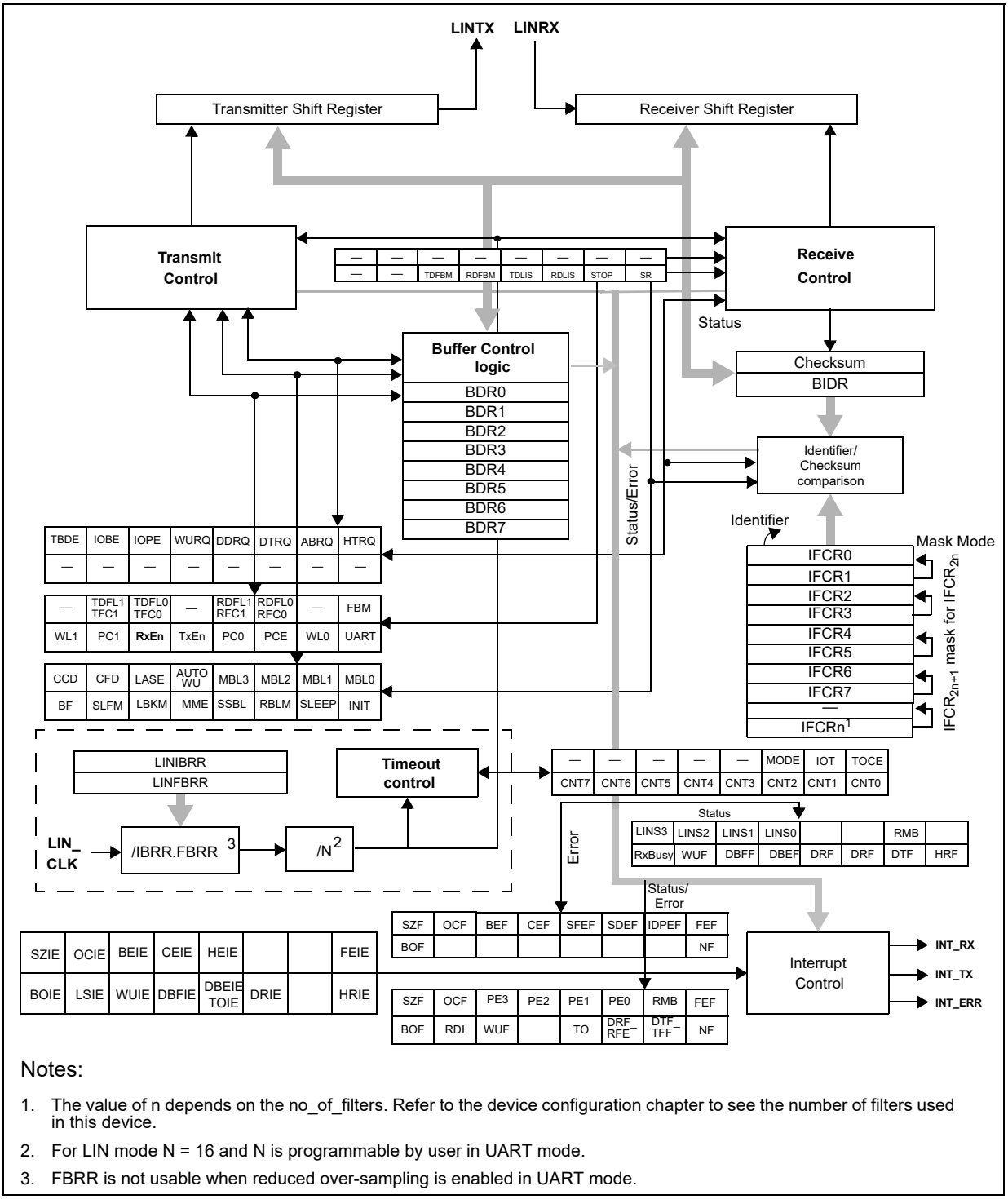
The LINFlexD controller is designed to manage a high number of LIN messages efficiently with a minimum of CPU load. To reduce the CPU loading in Master mode, LINFlexD autonomously handles the LIN messages once software has triggered the header transmission, until the next header transmission request in transmitter mode or until checksum reception in the receiver mode.

The LINFlexD supports LIN protocol version 1.3, 2.0, 2.1 and 2.2. It also consists of an 8-byte buffer for transmission/reception data.

The LINFlexD also provides support for some of the basic UART transfers of 8-bit, 9-bit, 16-bit, and 17-bit frames and also 12-bit data frame + parity reception in UART mode for MSC support.

The LINFlexD supports multi-channels and parametric DMA Tx/Rx interface in LIN/UART operating mode.

Figure 1652. Block diagram



Notes:

1. The value of n depends on the no_of_filters. Refer to the device configuration chapter to see the number of filters used in this device.
2. For LIN mode N = 16 and N is programmable by user in UART mode.
3. FBRR is not usable when reduced over-sampling is enabled in UART mode.

63.2 Main features

LINFlexD controller can operate in several modes, each of which has a distinct set of features which are described in the following sections. In addition, LINFlexD controller has several features common to all modes:

- Fractional baud rate generator
- 3 operating modes for power saving and configuration registers lock
  - Initialization
  - Normal
  - Sleep
- Test mode: Loop Back
- Maskable interrupts
- A maximum of 16 possible identifiers can be programmed into the identifier list

63.2.1 LIN mode features

- Supports LIN protocol version 1.3, 2.0, and 2.1
- Bit rates up to 20 Kbit/s (LIN protocol)
- Master/Slave mode
- Classic and Enhanced Checksum calculation and check
- Single 8-byte buffer or FIFO for Transmission/Reception
- Timeout management
- Identifier filters
- DMA interface
- Supports a maximum of 16 possible identifiers
- Master mode with autonomous message handling
- Extended frame mode for in-application programming purposes
- Wakeup event on dominant bit detection
- True LIN field state machine
- Advanced LIN error detection
- Header, response, and frame timeout
- Slave mode
  - Autonomous header handling
  - Autonomous transmit/receive data handling
- Identifier filters for autonomous message handling in Slave mode
- Separate clock for baud rate calculation (The relationship “(2/3)* LIN_CLK $\geq$ PBRIDGEx_CLK > 1/3*LIN_CLK” should be maintained)

63.2.2 UART mode features

- Full-duplex communication
- Baud rate is a function of baud clock, LINIBRR and LINFBRR registers (refer to *Section 63.3.2.8: Baud rate generation*)
- Separate clock for baud rate calculation (The relationship “(2/3)* LIN_CLK ≥ PBRIDGEx_CLK > 1/3*LIN_CLK” should be maintained)
- 15/16/7/8 bits data, parity
- 1/2/3 stop bits
- 12-bit + parity reception
- 4-byte buffer for reception, 4-byte buffer for transmission
- 12-bit counter for timeout management
- The maximum baud rate achievable is 25 Mbit/s.
- For bit rate (BR) <= LIN_CLK/16
  - 16 times oversampling
  - 3:1 majority voting
- For LIN_CLK/16 < BR <= LIN_CLK/8
  - OSR = 8
  - 3:1 majority voting
- For LIN_CLK/8 < BR <= LIN_CLK/4
  - OSR = 4,5,6
  - 1:1 voting

63.2.3 Standard features

DMA2x bus interface.

63.3 Functional description

63.3.1 LIN protocol

The LIN (Local Interconnect Network) is a serial communication protocol. A LIN cluster consists of one master task and several slave tasks. A master node contains the master task as well as a slave task - all other nodes contain a slave task only. The master node decides when and which frame is transferred on the bus. The slave task provides the data to be transported by the frame.

63.3.1.1 Frames

A frame consists of a header provided by the master task and a response provided by the slave task. The header consists of a break, a sync pattern, and an identifier. The break is followed by the sync pattern and the sync pattern is followed by the identifier. The slave task associated with the identifier provides the response. The response consists of a data field and checksum. The slave task designated to receive the data associated with the identifier receives the response and verifies the checksum.

Figure 1653. Frames

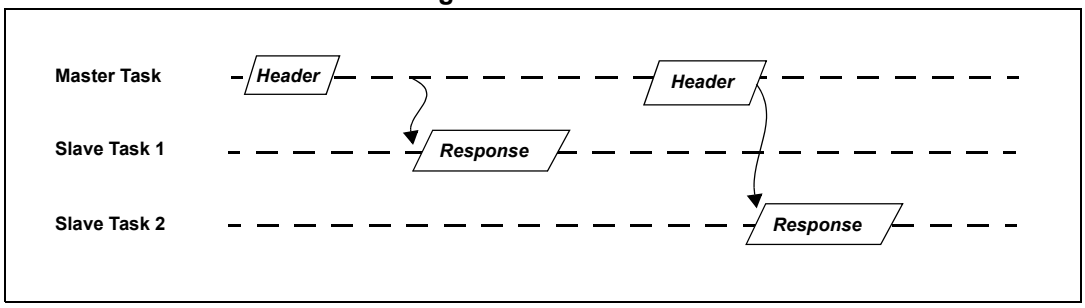
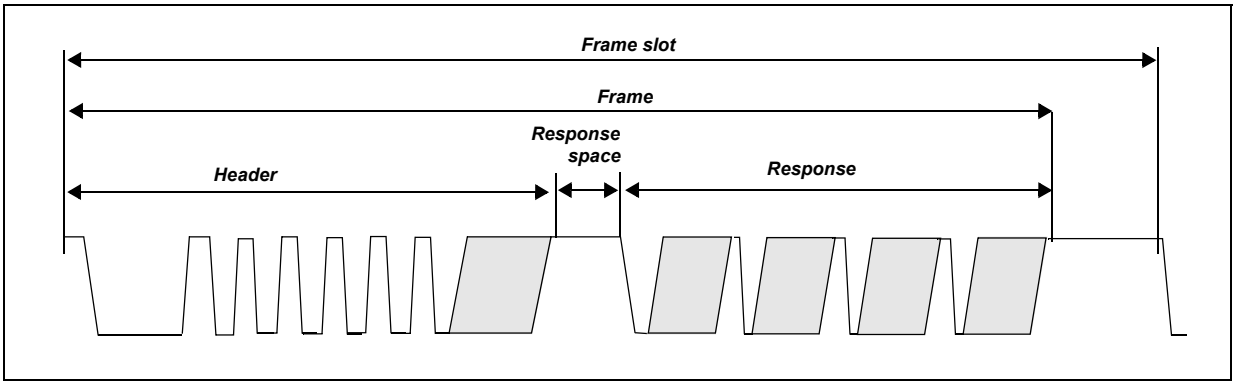


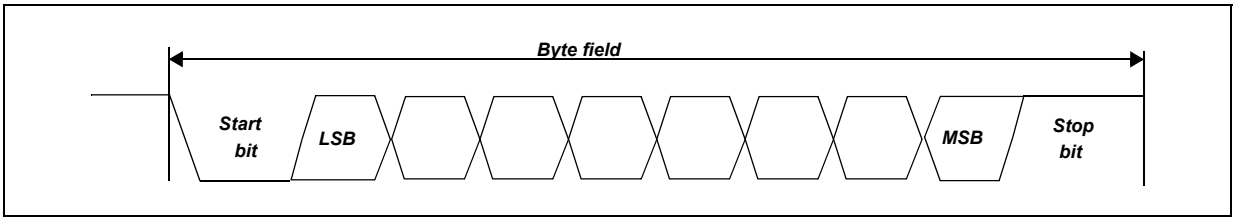
Figure 1654. Structure of LIN frame



63.3.1.2 Data field

Each byte is transmitted as shown in *Figure 1655*. The LSB of the data is sent first and the MSB is sent last. The start bit is encoded as a bit with value zero (dominant) and stop bit is encoded as bit value one (recessive).

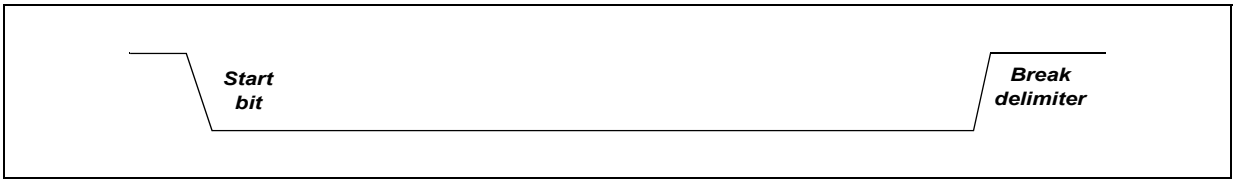
Figure 1655. Structure of byte field



63.3.1.3 Break

The break symbol is used to signal the beginning of a new frame. It is the only field that does not comply with the above figure. The break is always generated by the master and shall be at least 13 bits of dominant value including the start bit, followed by a break delimiter as shown in *Figure 1656*. The break delimiter must be of two-bit duration (to be compliant with LIN protocol 2.1).

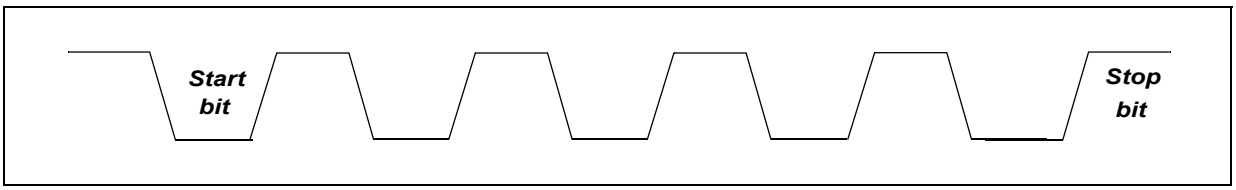
Figure 1656. Break field



63.3.1.4 Sync byte

Sync is a byte field with the data value of 0x55.

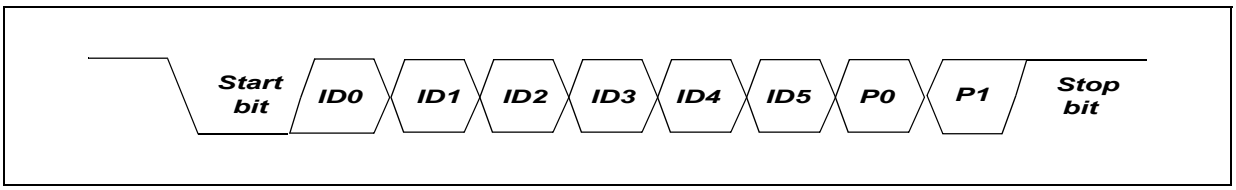
Figure 1657. Sync byte field



63.3.1.5 Identifier

The Identifier field consists of two sub fields, the identifier and the identifier parity. Bits 0 to 5 are the identifier and bits 6 and 7 indicate the parity.

Figure 1658. Identifier field



P0 = ID0 XOR ID1 XOR ID2 XOR ID4

P1 = NOT (ID1 XOR ID3 XOR ID4 XOR ID5)

63.3.1.6 Checksum

The last field of a frame is the checksum. The checksum contains the inverted 8-bit sum with carryover of all data bytes or all data bytes and the identifier. Checksum calculation over the data bytes only is called classic checksum and is used for communication with LIN 1.3 slaves. Checksum calculation over the data bytes and the protected identifier byte is called enhanced checksum, and it used for communication with LIN 2.0 slaves.

63.3.2 LINFlexD features

63.3.2.1 Operating modes

The LINFlexD has three operating modes: Initialization, Normal, and Sleep modes. After hardware reset, the LINFlexD enters sleep mode to reduce power consumption.

63.3.2.1.1 Initialization mode (INIT)

To enter this mode, software sets the INIT bit in the LINCR1. To exit the initialization mode, software should reset the INIT bit.

When in initialization mode, all message transfers to and from the LIN bus are stopped and the status of LIN bus output LINTX is recessive (high). If software invokes the initialization mode when a bus transfer is in progress, the transfer is aborted. The software should therefore check the LIN state before setting this bit.

To initialize the LINFlexD controller, software must:

1. Set up the baud rate registers
2. Reset the UART bit
3. Select the mode (Master or Slave)
4. Configure checksum control bits
5. Initialize the identifier list (Slave mode)

63.3.2.1.2 Normal mode (NM)

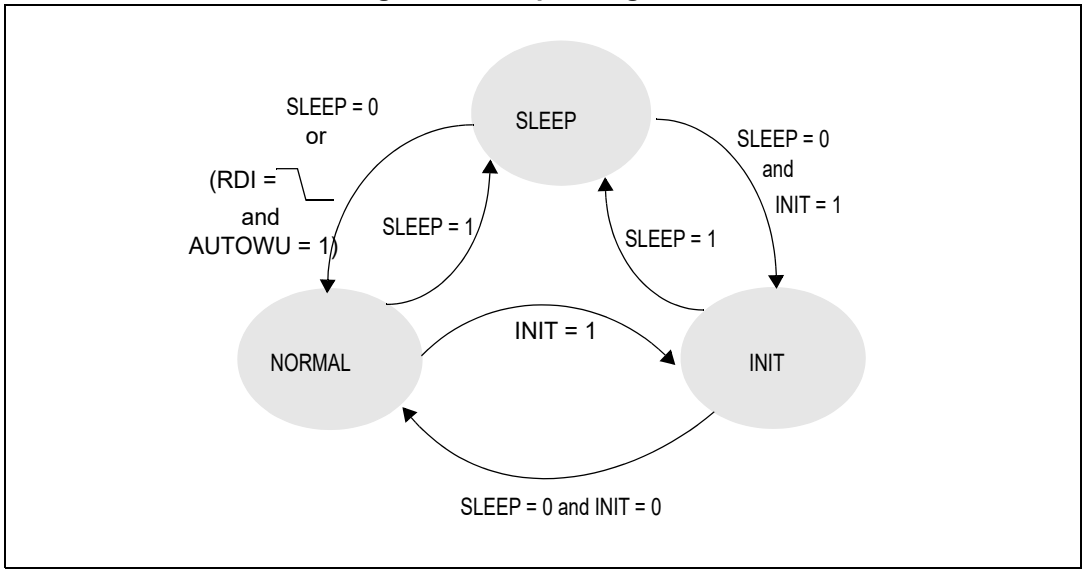
Once software has completed initialization of the LINFlexD controller, it can enter Normal mode by clearing the INIT bit.

63.3.2.1.3 Sleep mode (SM)

Sleep mode in LINFlexD reduces power consumption. This mode is entered by setting the SLEEP bit in LINCR1. In this mode the LINFlexD clock is stopped. LINFlexD can be woken from Sleep mode by clearing the SLEEP bit.

If software detects a wakeup pulse of 150 µs on the LIN Bus, it can request LINFlexD to wake up from Sleep mode. Refer to *Section 63.3.2.10: Wakeup management*.

Figure 1659. Operating modes



63.3.2.2 Test mode

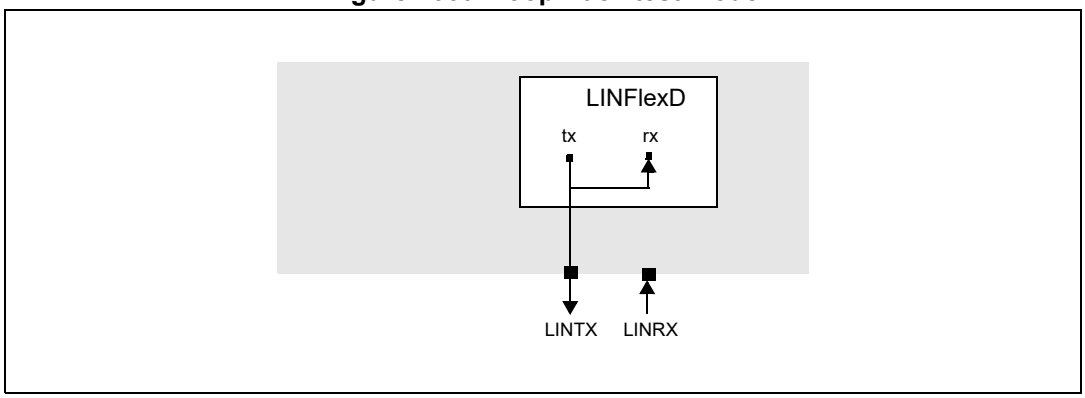
63.3.2.2.1 Loop Back mode

This mode is entered by setting the LBKM bit in LINCR1. In this mode, the LINFlexD receives the Identifier and Data transmitted by itself and writes the same in the BIDR and

Data buffers respectively. This mode is provided for selftest functions. Bit error is checked in this mode.

The LINFlexD ignores the LINRX signal. There is internal feedback from its TX output to its RX input. The TX pin can be disconnected from LINTX pin by SIUL2 setting.

Figure 1660. Loop Back test mode



63.3.2.3 Master mode

Master mode is selected by means of the MME bit in LINCR1 register.

63.3.2.3.1 Header transmission

According to the LIN protocol any communication on the LIN bus is triggered by the master sending a header. The header is transmitted by the master task of a node while the response is transmitted by the slave task of a node.

To transmit the header, first program the Identifier, data field length, message direction and checksum enable in the BIDR register; then set the HTRQ bit in LINCR2. Once header transmission starts, the user should not modify BIDR register bits until the current frame is complete. The transmitted ID is also received by the master node and copied to BIDR.

63.3.2.3.2 Data transmission

When the master node is the publisher of the data corresponding to the Identifier sent by the master, then the slave task of the node should send the data in the response part of the frame. Hence software must provide the data to the LINFlexD before the header transmission is requested. The data to be transmitted is stored in the message buffer BDR[0:7]. The number of bytes to be transmitted depends on the Data Field length in BIDR. The software uses the BIDR[CCS] bit to configure the checksum type (classic or enhanced) for each message.

If the response has been sent successfully, LINSR[DTF] is set. In case of error, the DTF flag is not set and the corresponding error flag is set in the LINESR (refer to error handling). It is possible to handle frames with a response size larger than eight bytes of data (extended frames). If the data field length in the BIDR is configured with a value higher than eight data bytes, LINSR[DBEF] is set once the first eight bytes have been transmitted. The application has to update the buffer BDR before resetting the DBEF bit. The transmission of the next bytes starts when the DBEF bit is reset. Once the last data byte (or the checksum byte) has been sent, the DTF flag is set.

The direction of the message buffer is decided by the DIR bit in the BIDR. The transmitted data is also received by the same node and copied to the buffer.

63.3.2.3.3 Data reception

To receive data from a slave node, the master sends a header with the corresponding Identifier. The data received from the slave is stored in the message buffer and the status of the message is stored in the LINSR.

If the response has been received successfully, the LINSR (DRF) bit is set. In case of error, the DRF flag is not set and the corresponding error flag is set in the LINESR (refer to error handling). It is possible to handle frames with a Response size larger than eight bytes of data (extended frames). If the data field length in the BIDR is configured with a value higher than eight data bytes, the LINSR (DBFF) bit is set once the first eight bytes have been received. The application has to read the buffer BDR before resetting the DBFF bit. Once the last data byte (or the checksum byte) has been received, the DRF flag is set.

63.3.2.3.4 Data Discard

If the user wants to discard the data after header transmission, then the DDRQ bit in LINCR2 should be set.

63.3.2.4 Slave mode

This mode is selected when the MME bit of the LINCR1 register is cleared.

63.3.2.4.1 Data transmission

On header reception, the HRF bit is set and an RX interrupt is generated. The software must then:

1. Read the received ID in the BIDR register
2. Fill the BDR[0:x] register
3. Program the CCS and DIR bits in the BIDR register
4. Specify the data field length DFL[5:0] bits in the BIDR register
5. Trigger the data transmission by setting the DTRQ bit

Note that the HRF bit should be reset only after the DTRQ bit is set. For the DTRQ to be effective, the HRF bit must be set. This is to ensure that DTRQ is not set randomly, but only after a header reception. It must be noted that you cannot set the DIR and DTRQ bits once RXbusy is asserted in LINSR.

Alternately, one or more Identifier filters are configured for transmission by setting the DIR bit and activated by setting the enable bits in IFER. When at least one Identifier filter is active and configured for transmission and the received ID matches the filter, a TX interrupt is generated. The software can use the index in the IFMI register to point directly to the corresponding data array in the RAM and copy this data to the BDR[0:7]. The use of a filter saves software the processing time required to read the ID value in the BIDR, match it, and configure the data field length and checksum type.

If the number of filters provided by LINFlexD are not sufficient for the application, mask mode can be used for the filters.

The transmitted data is also received by the same node and copied to the buffer.

63.3.2.4.2 Data reception

When LINFlexD is the subscriber of the data of the received identifier, then on header reception the HRF flag is set and an RX interrupt is generated. The software must read the received ID from the BIDR register and specify the data field length before the reception of the stop bit of the first data byte. When the checksum is received, an RX interrupt is generated for software to read the received data from BDR[0:7] and the RMB bit is set. Software must then release the data buffer by resetting the RMB bit in the LINSR.

When at least one identifier filter is active and configured for reception, an RX interrupt is generated only after the checksum reception. No interrupt request is generated on reception of the ID.

If the Identifier is filtered by software, you can discard the data by setting the DDRQ bit in the LINCR2, while HRF is set.

Note that for software filtering, software must decide the type of checksum (configure the CCS bit of the BIDR register) before reception of data. Otherwise the previous value of the CCS bit is considered while calculating checksum.

63.3.2.5 Errors

63.3.2.5.1 Header error

A header error is an error during header reception. The types are:

- Sync Del error (SDEF)
- Sync field error (SFEF)
- Identifier Parity error (IDPEF)

1. Sync Delimiter too short

The delimiter should be 1 for at least one bit time, otherwise it is considered short and consequently the receiver discards synchronization on the current header. Hence the frame is discarded. An interrupt is generated if the HEIE bit of LINIER is set.

2. Inconsistent Sync field

This error condition is monitored differently depending on whether autosynchronization is ON or OFF.

Case 1: Autosynchronization ON (LASE bit of LINCR1 = 1):

When Autosynchronization is enabled, the inconsistent Sync Field informs:

- That the deviation error on the Sync Field is outside the LIN specification, which allows up to 14% of period deviation between the slave and master oscillators,

  or

- An overflow has occurred during the Sync Field Measurement, which leads to an overflow of the divider registers.

Deviation error on the Sync Field

The deviation error is checked by comparing the current baud rate (relative to the slave oscillator) with the received LIN Sync Field (relative to the master oscillator). Two checks are performed in parallel:

1. The first check is based on a measurement between the first falling edge and the last falling edge of the Sync Field. Let's refer to this period deviation as D.

   If there is an inconsistent Sync Field error, it means that:

   D > 14.0625%

   If there is no error, it means that:

   D < 14.84375%

   If 14.0625% < D < 14.84375%, then the Sync Field could be either consistent or inconsistent depending on dephasing between the signal on the RDI line and the LIN_CLK.

2. The second check is based on the measurement of each bit time between both edges of the Sync Field. This checks that each of these bit times is large enough (more than 12 samples) compared to the bit time of the current baud rate.

Case 2: Autosynchronization OFF (LASE bit of LINCR1 = 0)

In this case the Sync character is received as a normal character. If the received character is 0x55, then the Sync Field is OK.

On any occurrence of an inconsistent Sync Field, the receiver immediately exits from Sync_Field state and the frame is consequently discarded. The SFEF bit of LINESR is set.

1. Identifier Parity error

There are two parity bits in the identifier field. These bits are checked against the hardware-calculated parity over the other six bits of identifier, according to the formula:

P0 = ID0 XOR ID1 XOR ID2 XOR ID4

P1 = NOT (ID1 XOR ID3 XOR ID4 XOR ID5)

This parity is checked after the ID is transferred to the BIDR register (after stop of ID has been detected properly) and upon parity mismatch, the receiver state machine exits from Identifier state immediately if the IOPE bit of LINCR2 is set.

63.3.2.5.2 Bit error

This error is flagged in transmission mode when the value read back from the bus is different from the value transmitted. Bit error checking on each bit is guaranteed if transceiver delay is less than one bit time minus 6 LIN_CLK cycles. Bit error is not checked during break field transmission.

Example 1

1-bit time at 20 Kbit/s = 50 µs

6 LIN_CLK cycles at 80 MHz = 75 ns

Thus, (1-bit time) – (6 LIN_CLK cycles) = 49.925 µs

Transmission of the frame is stopped after the corrupted bit if the IOBE bit in LINCR2 is set. If IOBE is reset, the transmitter continues to transmit in spite of the bit error. An interrupt is generated if the BEIER bit is set in LINIER.

Note: If the break delimiter is not detected by the master within one bit time after delimiter transmission due to transceiver delay or error on the bus, then a train of bit error interrupts are generated. In this case, the Identifier may not be replaced in the BIDR.

Similarly, if the start of a falling edge of data is not detected by the transmitter node within one bit time after start bit transmission, then a train of bit error interrupts are generated. In

this case also, data and checksum replacements in the BDR and CFR respectively are not guaranteed.

63.3.2.5.3 Framing error

This error is flagged when a dominant state is sampled on the stop bit of the current received character (sync field, identifier field, data field, checksum field). LINFlexD discards the current frame and returns to Idle state. An interrupt is generated if the FEIE bit is set in LINIER.

The byte that caused framing error is also shifted to the buffer but DRF or DBFF is never set in this case.

63.3.2.5.4 Checksum error

This error is flagged when the checksum computed by hardware does not match the received checksum.

LINFlexD discards the received frame and returns to Idle state. An interrupt is generated if the CEIE bit is set in LINIER.

63.3.2.5.5 Overrun error

Once the message buffer is full (RMB is set), the next valid message reception will lead to an overrun and the message will be lost. The hardware signals the overrun condition by setting the BOF bit. Which message is lost depends on the buffer lock function control bit RBLM.

If RBLM is cleared, the old message in the buffer is overwritten by the most recent message. If RBLM bit is set, the most recent message is discarded and the oldest message is available to the software. In the case of a slave, if the buffer is not released (RMB is not reset) before reception of the next Identifier and if RBLM is set, then the ID along with the data is discarded.

63.3.2.5.6 Timeout error

Figure 1661. Incomplete response (for example, missing checksum)

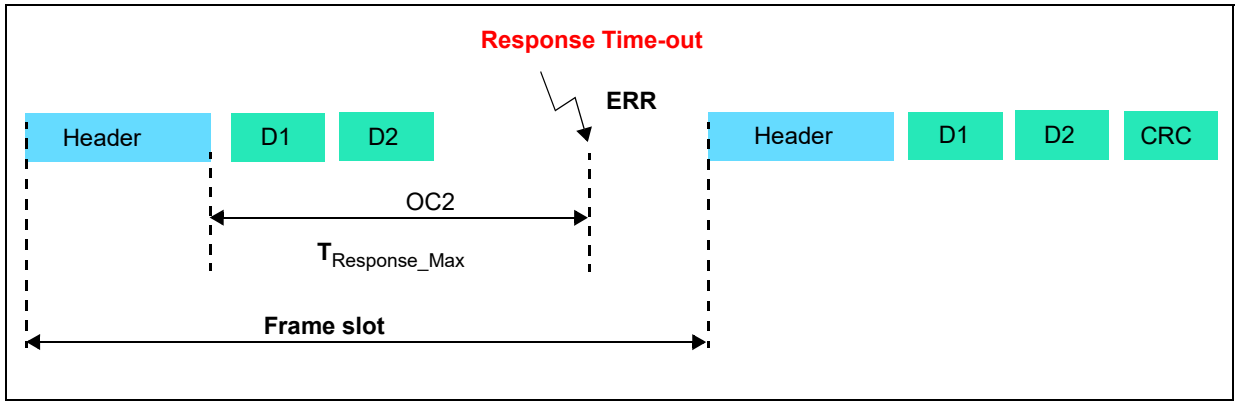
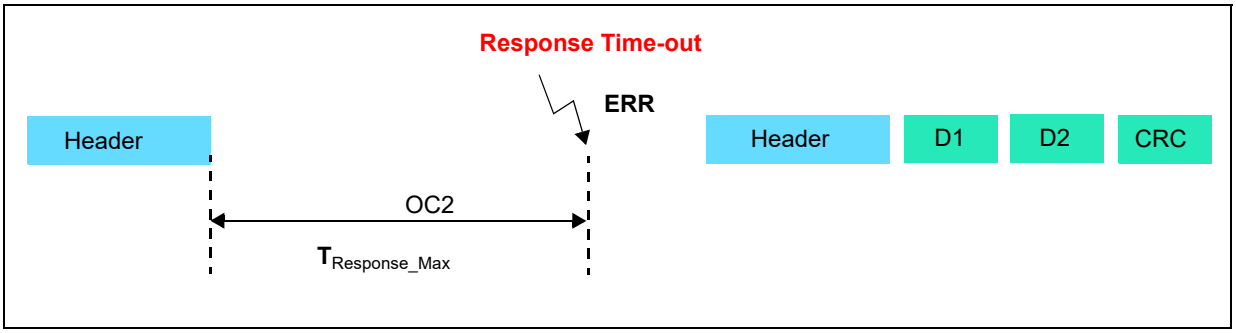


Figure 1662. No response



Header timeout mechanism

- Master mode
  - ocr_2 is loaded with Nominal_time_out-Correction_factor at the end of the stop bit of the Identifier field (where nominal_time_out = 1.4 × ((DFL + 2) × 10 bit time)).
  - The change with respect to earlier implementation is that this loading takes place at the end of the ID field (and not at the beginning of the data field). Moreover, the correct value, which depends on DFL, is loaded immediately. No further ocr_2 update is required.
  - In case of no response at all within this time (in other words, start of data is not received), timeout takes place when the counter reaches the ocr_2 value (no change compared to earlier implementation).
  - In case of an incomplete response, timeout also occurs when the counter reaches the ocr_2 value (no change compared to earlier implementation).
- Slave mode
  - **Case 1**: The received identifier is managed by a filter. The implementation can be similar to the master mode, because the DFL value is loaded by hardware. Thus, ocr_2 can be loaded with Nominal_time_out-Correction_factor at the end of the stop bit of the Identifier field (where nominal_time_out = 1.4 × ((DFL + 2) × 10 bit time)).
  - **Case 2**: The received identifier is not managed by a filter. As the DFL value needs to be updated by software after the identifier field has been received, the previous strategy cannot be used as the DFL value is not yet available. The earlier implementation loaded the maximum value = 1.4 × (9 × 10 bit time) and then updated it according to DFL after the first data byte reception. This was not acceptable if no data byte was received at all, as timeout expired too late. The proposed implementation is the following:

    At the end of the ID, ocr_2 is loaded with 36 (maximum possible response space)

    At the end of the first_data_byte it is reloaded again according to DFL

    Before reloading, LINFlexD checks the count_val. If count value is higher than the value to be reloaded, timeout takes place immediately and no reloading occurs.

Header timeout mechanism

- Master mode: As the header is generated by the LINFlexD, there are only two cases:
  - no error on the bus and timing is correct (nominal header length).
  - an error occurs on the bus and LINFlexD flags it in LINESR register (typically a bit error).

    Therefore there is no meaning of header timeout in master mode, so it is disabled.
- Slave mode
  - header_nominal = 13 + 2 + 10 + 10 = 35 Tbit
  - header_max = 1.4 × Header_nominal = 49 Tbit
  - taking into account a possible 14% clock deviation, header_max seen by LINFlexD is 49 × 1.14 = 56 Tbit.
  - If the counter starts after 11 Tbit, the HTO value is 56 – 11 = 45 Tbit

    The reset value of HTO is 45 and this register can only be programmed in slave mode.

    As response space is now included in the response, frame timeout is no longer needed and is removed completely. Indeed, header timeout and response timeout cover all cases.

    The timeout counter can be used to detect other timeouts. In this case, the MODE bit must be set and the output compare value can be updated in the LINTOCR register by software.

Stuck at zero timeout error

If the dominant pulse lasts for a time of at least 100 bits, the SZF bit in LINESR is set. If the same dominant pulse prolongs, the subsequent SZF setting will be 87 bit times apart (instead of 100 bit times).

63.3.2.5.7 Noise

During reception each bit is sampled 16 times and the value of the bit is obtained by taking the majority value of the 8th, 9th, and 10th samples. If any one of these three samples has a value different from the other two, this error is flagged.

When OSR = 8, majority of 2nd, 3rd, and 4th samples are taken into account to determine noise. Noise checking is disabled for all OSR less than eight.

This error is flagged when there is noise detected in the start bit (refer to *Section 63.4.2.4: LIN Error Status Register (LINESR)*).

63.3.2.6 Identifier filtering

In LIN protocol the identifier of a message is not associated with the address of a node but is related to the content of the message. A transmitter broadcasts its message to all the receivers. Based on the header received, the receiver decides whether to receive or transmit a response (depending on the identifier value). If the message does not target the node, it should be discarded.

To fulfill this requirement, the LINFlexD provides configurable filters in order to eliminate software intervention. This hardware filtering saves CPU resources that would otherwise be required to perform filtering by software.

There are a maximum of 16 filters (depending on the generic no_of_filters) in the LINFlexD, which can be programmed by the user only during Initialization mode. In order to activate a

filter the corresponding FACT bit in IFER needs to be set. There are two modes possible for each identifier depending on the corresponding IFM bit of IFMR.

63.3.2.6.1 Identifier list mode

If the n^{th} bit of IFMR is cleared, then filter number 2*n* and 2*n*+1 is in identifier list mode. In this mode, the maximum number of filters that can be configured for transmission/reception equals no_of_filters, depending on the FACT bit of IFER. In this mode the identifier received should match bit by bit to the ID field of IFCR2*n* or IFCR2*n*+1 (if the corresponding FACT bit is set).

63.3.2.6.2 Identifier mask mode

If the number of filters required is more than no_of_filters, then filters should be configured in Mask mode. If the n^{th} bit of IFMR is set, then filter number 2*n* is the filter and 2*n*+1 acts as a mask for it. The FACT bit for filter 2*n*+1 has no effect in this case. In this mode, if the x^{th} bit of the mask is set, then the x^{th} bit of the received identifier must match the x^{th} bit of filter.

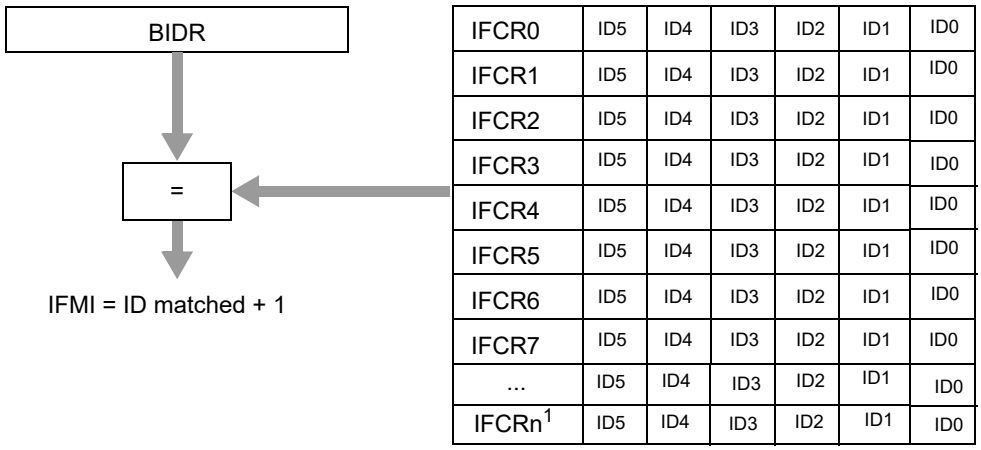
If there is a match of identifier with the *m*th filter in any mode, then *m*+1 is loaded in the IFMI register by hardware. No match condition is denoted by IFMI = 0.

Upon matching DFL (2:0), the CCS and DIR bits of the BIDR register are copied from the filter by hardware and from then on, the BIDR register is read-only until the end of the frame. Now if the DIR bit of BIDR is set, then a TXI interrupt is generated if the HRIE bit of LINIER is set. In this case, software uses the IFMI register to transfer the relevant data from the RAM area to BDR, and after a complete transfer the DTRQ bit of LINCR2 is set to start the transmission. If the DIR bit is cleared then an RXI interrupt is generated (provided DRIE bit of LINIER is set) when the checksum has been received and there is no checksum error.

In case of no filter match condition (IFMI = 0) and if the BF bit of LINCR1 is set, then an RXI is generated. Now it is the responsibility of software to configure BIDR and start transmission (by loading the BDR buffer and setting the DTRQ bit of LINCR2) or discard reception (by setting the DDRQ bit of LINCR2). If the BF bit is reset, then the receiver discards the received identifier and turns to Idle state in search of a new break.

Note: If one identifier matches with two filters (one is in the list and the other in Mask mode) then List mode prevails over Mask mode. In Mask mode, if two filters match with the identifier then the filter having the lower number prevails.

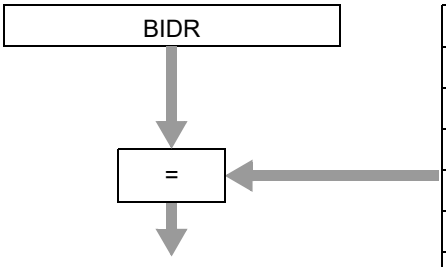
Figure 1663. Identifier List mode



IFCR0	ID5	ID4	ID3	ID2	ID1	ID0
IFCR1	ID5	ID4	ID3	ID2	ID1	ID0
IFCR2	ID5	ID4	ID3	ID2	ID1	ID0
IFCR3	ID5	ID4	ID3	ID2	ID1	ID0
IFCR4	ID5	ID4	ID3	ID2	ID1	ID0
IFCR5	ID5	ID4	ID3	ID2	ID1	ID0
IFCR6	ID5	ID4	ID3	ID2	ID1	ID0
IFCR7	ID5	ID4	ID3	ID2	ID1	ID0
...	ID5	ID4	ID3	ID2	ID1	ID0
IFCRn[1]	ID5	ID4	ID3	ID2	ID1	ID0

1. The value of n depends on the no_of_filters. Refer to the device configuration chapter to see the number of filters used in this device.

Figure 1664. Identifier Mask mode



IFCR0	ID5	ID4	ID3	ID2	ID1	ID0
IFCR1	0	0	0	1	1	1
IFCR2	ID5	ID4	ID3	ID2	ID1	ID0
IFCR3	1	1	1	1	0	0
IFCR4	ID5	ID4	ID3	ID2	ID1	ID0
IFCR5	1	1	1	1	0	1
IFCR6	ID5	ID4	ID3	ID2	ID1	ID0
IFCR7	1	1	1	1	1	0
...	...	...	...	...	...	...
IFCRn[1]	...	...	...	...	...	...

$IFCR_{2n+1}$ = Mask for $IFCR_{2n}$
1 = must match bits
0 = don't care bits

1. The value of n depends on the no_of_filters. Refer to the device configuration chapter to see the number of filters used in this device.

63.3.2.7 Start detection and delimiter detection in receiver

There is a 10-bit-shift-register sample register in the receiver that shifts signal data to the next less significant bit on each incoming sample.

Table 1646. 10-bit-shift-register sample register

Sample register bit number										Counter
9	8	7	6	5	4	3	2	1	0	
—	—	—	—	—	—	—	—	—	RT1	counter = 0
—	—	—	—	—	—	—	—	RT1	RT2	counter = 1
—	—	—	—	—	—	—	RT1	RT2	RT3	counter = 2
—	—	—	—	—	—	RT1	RT2	RT3	RT4	counter = 3
—	—	—	—	—	RT1	RT2	RT3	RT4	RT5	counter = 4
—	—	—	—	RT1	RT2	RT3	RT4	RT5	RT6	counter = 5
—	—	—	RT1	RT2	RT3	RT4	RT5	RT6	RT7	counter = 6
—	—	RT1	RT2	RT3	RT4	RT5	RT6	RT7	RT8	counter = 7
—	RT1	RT2	RT3	RT4	RT5	RT6	RT7	RT8	RT9	counter = 8
RT1	RT2	RT3	RT4	RT5	RT6	RT7	RT8	RT9	RT10	counter = 9
RT2	RT3	RT4	RT5	RT6	RT7	RT8	RT9	RT10	RT11	counter = 10
RT3	RT4	RT5	RT6	RT7	RT8	RT9	RT10	RT11	RT12	counter = 11
RT4	RT5	RT6	RT7	RT8	RT9	RT10	RT11	RT12	RT13	counter = 12
RT5	RT6	RT7	RT8	RT9	RT10	RT11	RT12	RT13	RT14	counter = 13
RT6	RT7	RT8	RT9	RT10	RT11	RT12	RT13	RT14	RT15	counter = 14
RT7	RT8	RT9	RT10	RT11	RT12	RT13	RT14	RT15	RT16	counter = 15
1	1	1	0	x(1)	0	x	0	x	0	Start detection
1	1	1	0	x	0	x	0	x	1	
1	1	1	0	x	0	x	1	x	0	
1	1	1	0	x	1	x	0	x	0	
0	0	0	1	x	1	x	1	x	1	Delimiter detection
0	0	0	1	x	1	x	1	x	0	
0	0	0	1	x	1	x	0	x	1	
0	0	0	1	x	0	x	1	x	1	

1. x: due to sampling during transition of the bit, the bit value might be sampled as a ‘0’ or ‘1’ depending on the rise/fall time. So ‘x’ refers to an indeterministic value.

63.3.2.7.1 Start detection mechanism

The following steps are followed for detecting a start bit:

1. Refer to *Table 1646* for the status of the sample register (shift register) when count = 6.
2. At this point, if RT1 (the first incoming sample) = 0 and the previous three samples already received are all ones, then it might be a start bit.
3. To make sure it is indeed a start bit, the incoming data samples in the sample register (4), sample register (2) and sample register (0) which correspond to RT3, RT5, and RT7 is verified using the steps mentioned below.
4. If the majority of these 3 samples are equal to ‘0’ (2 or more), then start bit is said to be detected.
5. These three samples RT3, RT5, and RT7 are called the “verification samples” which are checked at count = 6.
6. The sample register bits 9, 8, 7, and 6 are called “qualification samples” which are checked at count = 6.
7. At count = 9, if majority value of RT8, RT9, and RT10 is not equal to ‘0’ then Noise flag is set.

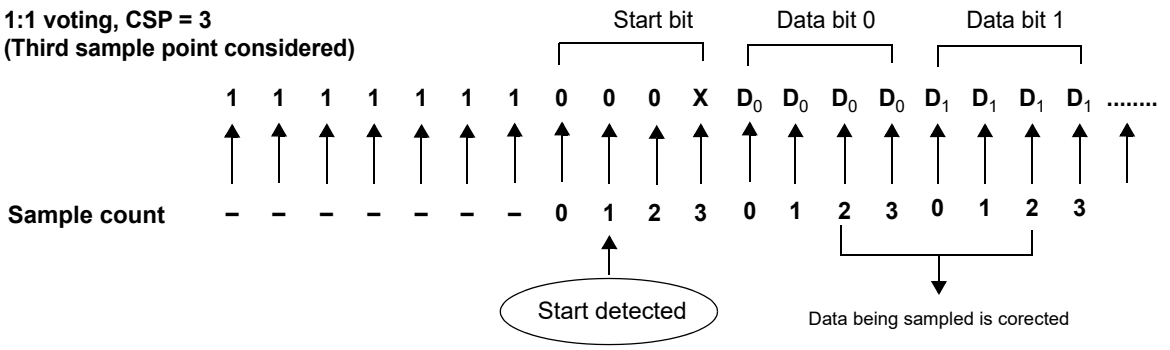
63.3.2.7.2 Delimiter detection mechanism

The following steps are followed for detecting a delimiter bit:

1. Refer to *Table 1646* for the status of the sample register (shift register) when count = 6.
2. At this point, if RT1(the first incoming sample) = 1 and the previous three samples already received are all zeros, then it might be a delimiter bit.
3. To make sure it is indeed a delimiter bit, the incoming data samples in the sample register (4), sample register (2) and sample register (0) which correspond to RT3, RT5, and RT7 is verified using the steps mentioned below.
4. If the majority of these 3 samples are equal to ‘1’ (2 or more), then delimiter bit is said to be detected.
5. These three samples RT3, RT5, and RT7 are called the “verification samples” which are checked at count = 6.
6. The sample register bits 9, 8, 7, and 6 are called “qualification samples” which are checked at count = 6.

Hence, a start is detected as soon as it is qualified with 1110 in the sample register and verified with at least two out of three predefined verification samples being zero. Similarly for delimiter detection, qualification samples are 0001 and at least two out of the three verification samples are one. The Noise Flag is set if the start is verified with only two valid samples.

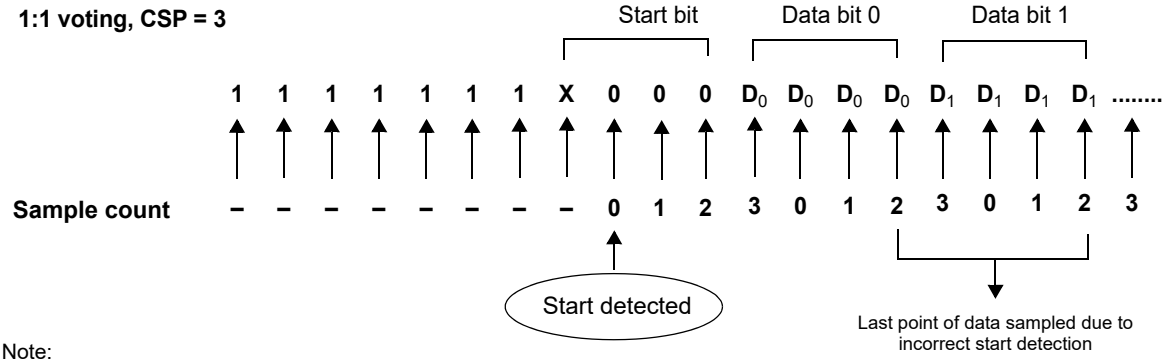
Figure 1665. Start detection and sampling for over sampling rate = 4 (Case 1)



Note:

1. D_0, D_1: Databits
2. X: Due to sampling during transition of the bit, value might be sampled as a ‘0’ or ‘1’ depending on the rise/fall time.Therefore ‘X’ refers to a indeterministic value.

Figure 1666. Start detection and sampling for over sampling rate = 4 (Case 2a)



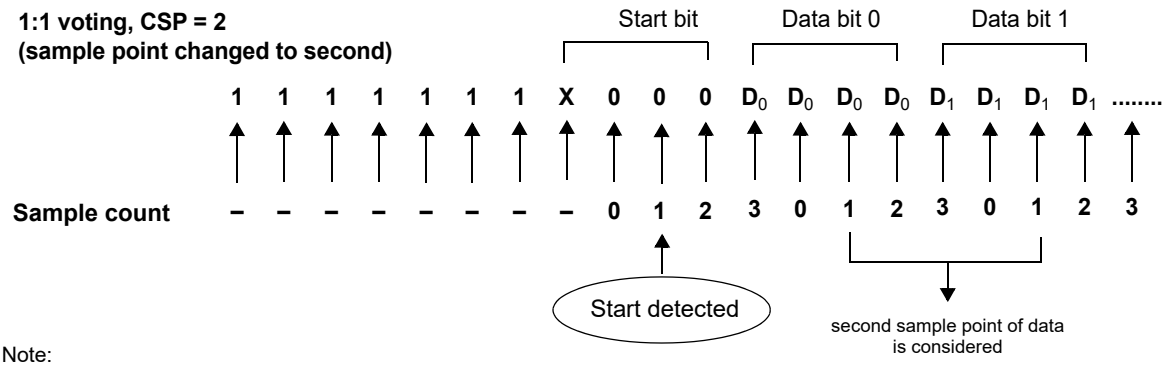
Note:

1. D_0, D_1: Databits
2. X: Due to sampling during transition of the bit, value might be sampled as a ‘0’ or ‘1’ depending on the rise/fall time. Therefore ‘X’ refers to a indeterministic value.

The choice of the correct sample points depends on the external Rx signal quality. During the application development process, the error information indicated by the parity error can help to find the best setting for an application-specific hardware signal.

To sample at the correct point therefore, CSP has to be changed to two resulting in the following situation:

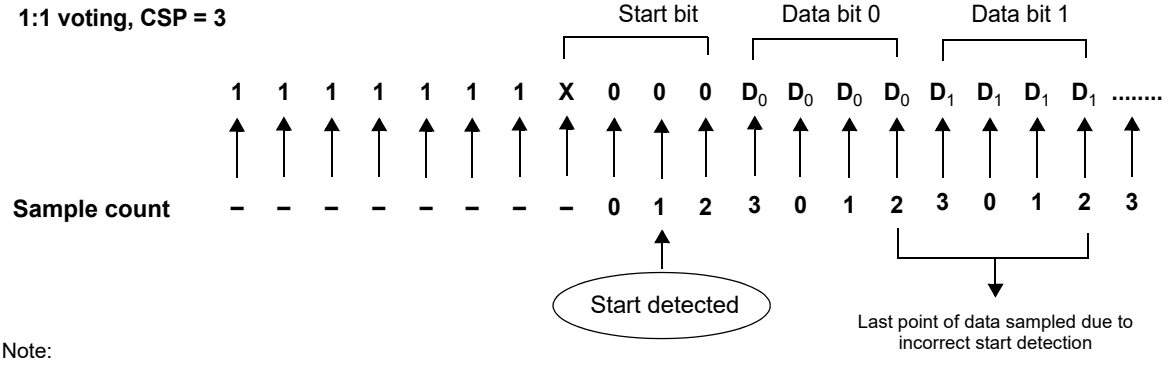
Figure 1667. Start detection and sampling for over sampling rate = 4 (Case 2b)



Note:

1. D_0, D_1: Databits
2. X: Due to sampling during transition of the bit, value might be sampled as a ‘0’ or ‘1’ depending on the rise/fall time. Therefore ‘X’ refers to a non deterministic value.

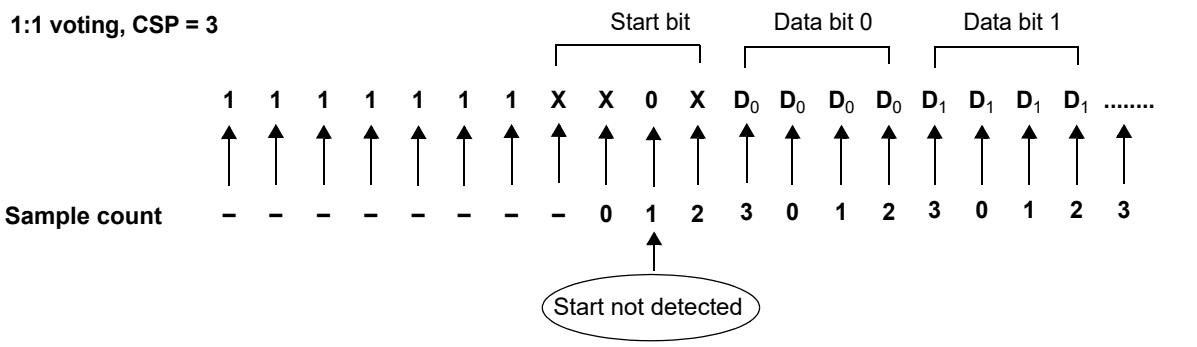
Figure 1668. Start detection and sampling for over sampling rate = 4 (Case 3)



Note:

1. D_0, D_1: Databits
2. X: Due to sampling during transition of the bit, value might be sampled as a ‘0’ or ‘1’ depending on the rise/fall time. Therefore ‘X’ refers to a non deterministic value.

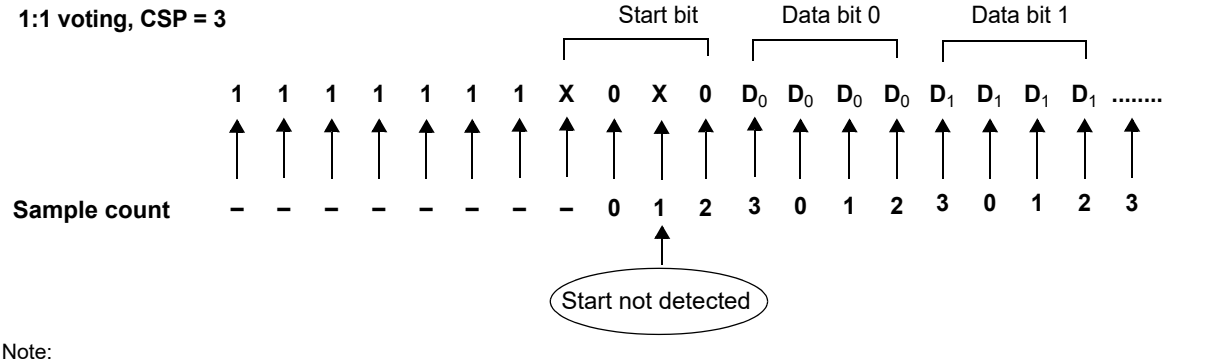
Figure 1669. Start detection and sampling for over sampling rate = 4 (Case 4)



Note:

1. D_0, D_1: Databits
2. X: Due to sampling during transition of the bit, value might be sampled as a ‘0’ or ‘1’ depending on the rise/fall time. Therefore ‘X’ refers to a non deterministic value.

Figure 1670. Start detection and sampling for over sampling rate = 4 (Case 5)



Note:

1. D_0, D_1: Databits
2. X: Due to sampling during transition of the bit, value might be sampled as a ‘0’ or ‘1’ depending on the rise/fall time. Therefore ‘X’ refers to a non deterministic value.

63.3.2.8 Baud rate generation

The LIN baud rate is programmed in two registers: the LIN Integer Baud Rate register and the LIN Fraction Baud Rate register. The baud rate registers can be programmed only during Initialization mode.

The baud rate is calculated with the following formula for both receiver and transmitter:

Equation 91 Tx = Rx = LIN_CLK / (16 × LDIV)

Where LIN_CLK is the frequency of the baud clock (ipg_baud_clk).

LDIV is an unsigned fixed point number. The mantissa is coded into 20 bits of LINIBRR and the fraction is coded on 4 bits of LINFBRR.

When reduced oversampling is enabled LINFBRR is not used and LDIV contains only the integer part of LINIBRR.

Example 2

When ROSE = 0 (For LIN and UART mode):

LDIV = 1041.67, LIN_CLK = 80 MHz

LINIBRR = 1041d = 411 h

LINFBRR = 11d = B h

Baud rate = 80 MHz/(16 x 1041,67) = 4.8 Kbit/s

Example 3

When ROSE = 1 (Only for UART mode):

LDIV = 10 d, LIN_CLK = 80 MHz

LINIBRR = 10 d, OSR = 4

Baud rate = LIN_CLK / (OSR × LDIV) = 80 MHz / (4 × 10) = 2 Mbit/s

Note: Due to design constraints the maximum baud rate achievable is 25 Mbit/s.
The relationship "(2/3)* LIN_CLK ≥ PBRIDGEx_CLK > 1/3*LIN_CLK" should be maintained)

63.3.2.9 Automatic resynchronization

To automatically adjust the baud rate based on measurement of the LIN sync field, write the nominal Prescaler value (nominal baud rate) in LINIBRR and LINFBRR, then set the LASE bit in LINCR1 to enable automatic synchronization.

When automatic synchronization is enabled, after each LIN Sync Del, the time duration between five falling edges on RDI is sampled on LIN_CLK.

63.3.2.10 Wakeup management

Any node in a sleeping LIN cluster may request a wakeup. The wakeup request is issued by forcing the bus to the dominant state for 250 µs to 5 ms. Every slave node should detect the wakeup request (a dominant pulse longer than 150 µs) and be ready to listen to bus commands within 100 ms, measured from the ending edge of the dominant pulse. The master also wakes on detecting a wakeup request and when the slaves are ready, starts sending frame headers to find out the cause of the wakeup. If the master does not issue a frame header within 150 ms from the wakeup request, then the node issuing a request may try issuing a new wakeup request.

In LINFlexD, a wakeup request can be generated by writing the wakeup character in BDR0 and setting the WURQ bit in LINCR2. On setting the WURQ bit, the character in BDR0 is transmitted. For LIN 2.0, character 0xF0 is sent as the wakeup character.

In LINFlexD, wakeup can be detected in two ways.

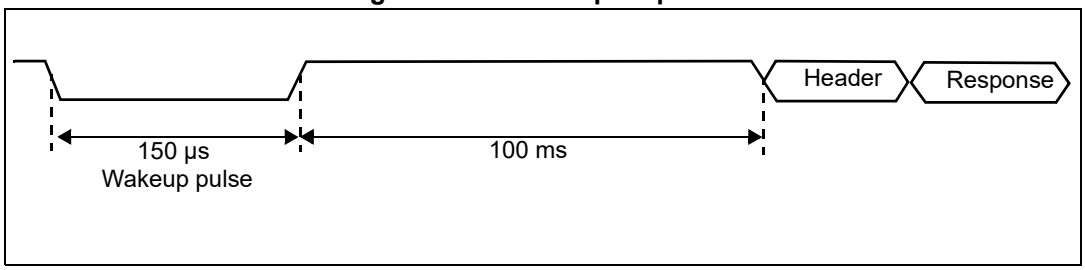
1. AUTOWU = 1

   On detecting a falling edge in sleep mode, the SLEEP bit is cleared by hardware, the WUF flag is set, and an interrupt is generated (if WUPIE is set). LINFlexD is now in normal mode and ready to receive frames.

2. AUTOWU = 0

   On detecting a falling edge the WUF flag is set and an interrupt is generated (if WUPIE is set). It is then up to the software to clear the SLEEP bit.

Figure 1671. Wakeup sequence



63.3.2.11 Use case and limitations

In LIN mode, when IP is requested to enter STOP mode, the SW has to first set the INIT bit to send the receiver to IDLE. This ensures that the IP acknowledges STOP request.

63.3.3 Timer

There is an 8-bit counter which has different behavior in different modes.

In Output Compare Mode (LINTCSR[MODE]= 1) : This counter is running even in Sleep and Initialization mode. The counter value which when matches the two software configurable output compare registers (LINOCR[OC1] or LINOCR[OC2]), generate output compare interrupt (LINESR[OCF]) provided TOCE bit in LINTCSR is set. Once an interrupt occurs subsequent interrupt is generated only when the application has soft reset the LIN IP through GCR [SR] bit to get reset the output compare flag (LINESR[OCF]).

In LIN Mode : The software has no control over the TOCE bit and output compare registers are utilized for generation of LIN timeout interrupts (header, frame, response times out). In this case, if LIN moves to Sleep or Init state then this counter remains in Reset state. LIN mode has no meaning if UART is enabled, hence the counter will remain in Reset state.

63.3.4 UART mode

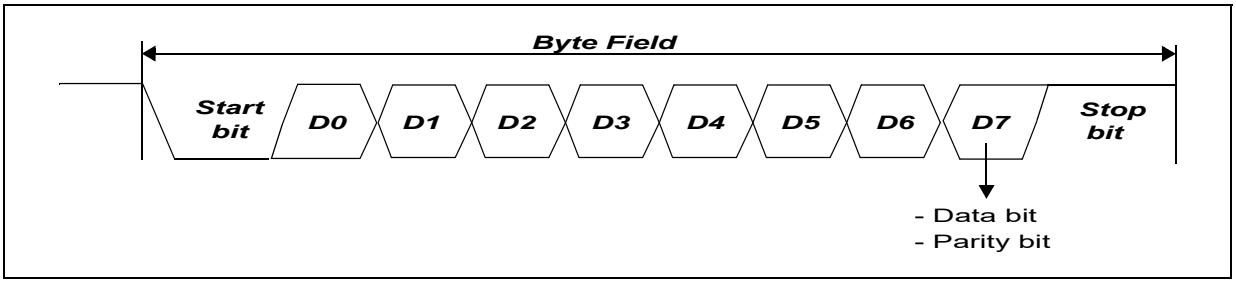
The main features in UART mode are:

- Full duplex communication
- 8-bit frames, 9-bit frames,16-bit frames,17-bit frames
- Even/Odd/0/1 Parity
- User programmable over sampling rate to obtain the baud rate of up to 25 Mbit/s

63.3.4.1 8-bit data frames

The eighth bit can be a data bit or a parity bit. Even/Odd/0/1 parity can be selected by the PC[1:0] bit in the same register. An even parity will be set if the modulo-2 sum of the seven data bits is one. An odd parity will be cleared in this case.

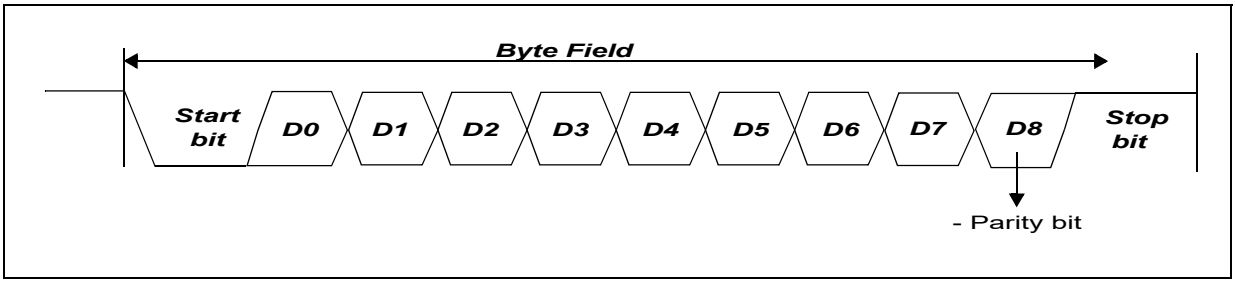
Figure 1672. UART mode 8-bit data frame



63.3.4.2 9-bit frames

The ninth bit should be a parity bit. Even/Odd/0/1 Parity can be selected by the PC[1:0] field in the same register. An even parity will be set if the modulo-2 sum of the seven data bits is one. An odd parity will be cleared in this case. Parity 0 forces a zero logical value. Parity 1 forces a high logical value.

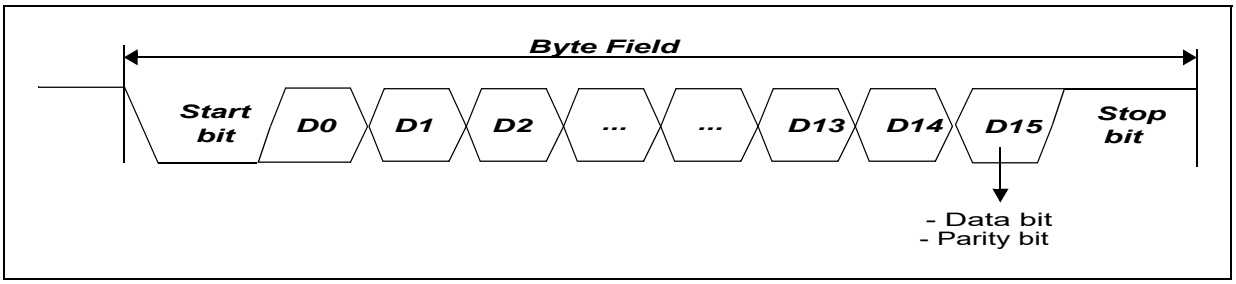
Figure 1673. UART mode 9-bit data frame



63.3.4.3 16-bit data frames

The 16th bit can be a data or parity bit. Even/Odd/0/1 Parity bit can be selected by the PC[1:0] bit in the same register. Parity 0 forces a zero logical value. Parity 1 forces a high logical value.

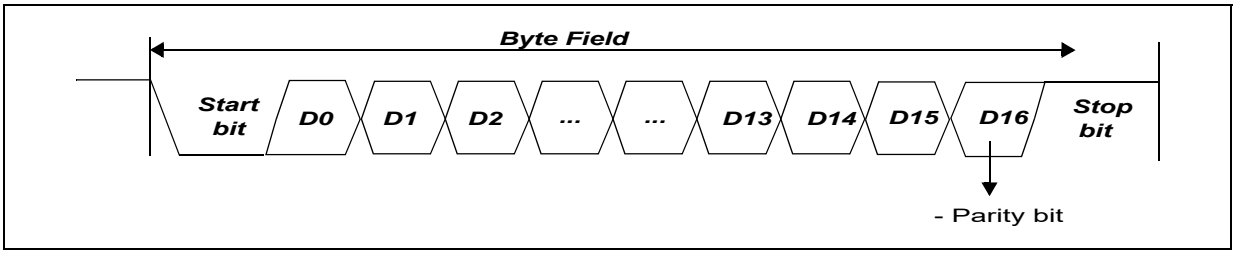
Figure 1674. UART mode 16-bit data frame



63.3.4.4 17-bit frames

The 17th bit is the parity bit. Even/Odd/0/1 Parity bit can be selected by the PC[1:0] bit in the same register. Parity 0 forces a zero logical value. Parity 1 forces a high logical value.

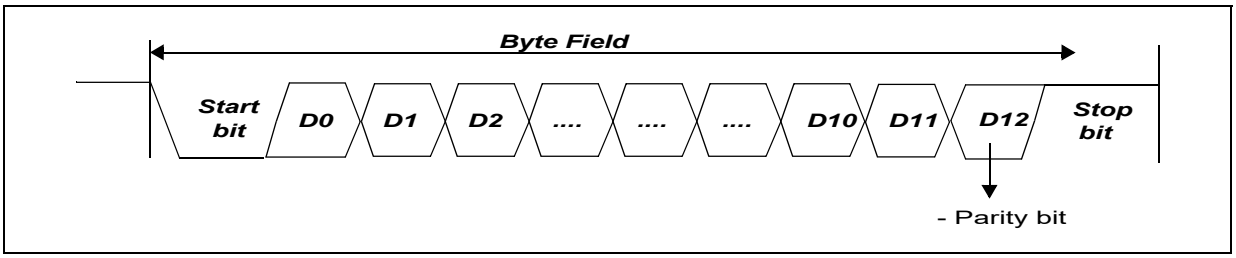
Figure 1675. UART mode 17-bit data frame



63.3.4.5 13-bit frames

Whenever WLS is one, special word length is selected in UART mode. This bit enables 12-bit + parity bit reception only in FIFO mode.

Figure 1676. UART mode 13-bit data frame



63.3.4.6 Buffer in UART mode

The 8-byte buffer is divided into two parts: one for receiver and one for transmitter, as shown in *Figure 1677*.

Figure 1677. Structure of 8-byte buffer

Tx0	BDR0
Tx1	BDR1
Tx2	BDR2
Tx3	BDR3
Rx0	BDR4
Rx1	BDR5
Rx2	BDR6
Rx3	BDR7

For 16-bit frames, the lower eight bits are written in BDR0 and the upper eight bits in BDR1.

The same applies for reception.

63.3.4.7 UART transmitter

In order to start transmission in UART mode, the UART bit should be set and the transmitter enable bit should be set. Transmission starts when the BDR0 (least significant data byte) is programmed and continues until the number of bytes/halfwords transmitted is equal to the value in the TDFL bits in UARTCR.

The transmit buffer is four bytes (when UARTCR[WL1] = 0) or two halfwords (when UARTCR[WL1] = 1), hence a maximum of four bytes (two halfwords) transmission can be triggered. Once the programmed number of bytes (halfwords) has been transmitted, the DTF flag is set in UARTSR. If the Txen bit of UART is reset in the middle of a transmission, then the current transmission is completed and no further transmissions can be invoked.

The buffer can be configured in FIFO mode (mandatory when the DMA Tx is enabled) by setting the control bit UARTCR[TFBM].

Note: If TFF bit is set and a write is performed to the FIFO, the data transmitted may be erroneous.

Invalid writes to BDRL which generates IPS transfer error, also triggers data transmission at the same time.

When the FIFO is full, a read to DATA0 of BDRM reflects the content of DATA3 in BDRL. Proper functionality of the IP is not affected by this.

Table 1647. BDRL access in UART mode

IPS operation(1)	Register	Mode (UARTCR[TFBM])	Word length (UARTCR:WL)	IPS operation result
Write byte0	BDRL	FIFO	Byte	OK
Write byte1-2-3	BDRL	FIFO	Byte	IPS transfer error
Write halfword0-1	BDRL	FIFO	Byte	IPS transfer error
Write word	BDRL	FIFO	Byte	IPS transfer error
Write byte0-1-2-3	BDRL	FIFO	Halfword	IPS transfer error
Write halfword0	BDRL	FIFO	Halfword	OK
Write halfword1	BDRL	FIFO	Halfword	IPS transfer error
Write word	BDRL	FIFO	Halfword	IPS transfer error
Read byte0-1-2-3	BDRL	FIFO	Byte/halfword	IPS transfer error
Read halfword0-1	BDRL	FIFO	Byte/halfword	IPS transfer error
Read word	BDRL	FIFO	Byte/halfword	IPS transfer error
Write byte0-1-2-3	BDRL	Buffer	Byte/halfword	OK
Write halfword0-1	BDRL	Buffer	Byte/halfword	OK
Write word	BDRL	Buffer	Byte/halfword	OK
Read byte0-1-2-3	BDRL	Buffer	Byte/halfword	OK
Read halfword0-1	BDRL	Buffer	Byte/halfword	OK
Read word	BDRL	Buffer	Byte/halfword	OK

1. Halfword0 refers to DATA1 and DATA0 of BDRL register. Halfword1 refers to DATA3 and DATA2 of BDRL register.
Halfword2 refers to DATA5 and DATA4 of BDRM register. Halfword3 refers to DATA7 and DATA6 of BDRM register.

In UART FIFO mode (UARTCR[TFBM] = 1), any read operation causes an IPS transfer error.

63.3.4.8 UART receiver

The reception of a data byte is started as soon as the user exits Initialization mode, sets the Rxen bit, and detects the start bit. There is a dedicated 4-byte (if UARTCR[WL1] = 0) or 2-halfword (if UARTCR[WL1] = 1) data buffer for received data. Once the programmed number (RDFL bits) of bytes have been received, the DRF flag is set in UARTSR and the current reception gets completed. Rxen bit needs to be set only to start the reception.The reception is automatically completed as soon as the programmed number (RDFL bits) of bytes have been received.

The buffer can be configured in FIFO mode (mandatory when the DMA Rx is enabled) by setting the control bit UARTCR[RFBM].

Table 1648. BDRM access in UART mode

IPS operation(1)	Register (2)	Mode (UARTCR[RFBM])	Word length (UARTCR:WL)	IPS operation result
Read byte4	BDRM	FIFO	Byte	OK
Read byte5-6-7	BDRM	FIFO	Byte	IPS transfer error
Read halfword2-3	BDRM	FIFO	Byte	IPS transfer error
Read word	BDRM	FIFO	Byte	IPS transfer error
Read byte4-5-6-7	BDRM	FIFO	Halfword	IPS transfer error
Read halfword2	BDRM	FIFO	Halfword	OK
Read halfword3	BDRM	FIFO	Halfword	IPS transfer error
Read word	BDRM	FIFO	Halfword	IPS transfer error
Write byte4-5-6-7	BDRM	FIFO	Byte/halfword	IPS transfer error
Write halfword2-3	BDRM	FIFO	Byte/halfword	IPS transfer error
Write word	BDRM	FIFO	Byte/halfword	IPS transfer error
Read byte4-5-6-7	BDRM	Buffer	Byte/halfword	OK
Read halfword2-3	BDRM	Buffer	Byte/halfword	OK
Read word	BDRM	Buffer	Byte/halfword	OK
Write byte4-5-6-7	BDRM	Buffer	Byte/halfword	IPS transfer error
Write halfword2-3	BDRM	Buffer	Byte/halfword	IPS transfer error
Write word	BDRM	Buffer	Byte/halfword	IPS transfer error

1. Halfword0 refers to DATA1 and DATA0 of BDRL register. Halfword1 refers to DATA3 and DATA2 of BDRL register.
   Halfword2 refers to DATA5 and DATA4 of BDRM register. Halfword3 refers to DATA7 and DATA6 of BDRM register.
2. Refer to the layout of the registers BDRL and BDRM to identify the mapping between byte x and the data bits of the BDRL and BDRM registers.

Note: If the user does not know in advance how many bytes are to be received, RDFL should not be programmed in advance. The reset value of RDFL is zero. This will ensure that the

reception will happen byte by byte. The state machine will move to the Idle state after each byte reception.

If RDFL is programmed for a certain value but that number bytes are not received, then reception will hang. In that case, software needs to take care of timeout by seeing the flag. Software has to set the sleep bit to move to the Idle state.

If a STOP request arrives in the middle of a reception, it is only acknowledged after all the programmed number of data bytes have been received; it is not served immediately. If the programmed number of data bytes are not received, then software has to take care of timeout; when the state machine moves to Idle state, then only a stop request is served.

If during reception of any byte a parity error occurs, then the corresponding PEx bit in UARTSR is set. No interrupt is generated in this case. If a framing error occurs in any byte (FE bit in UARTSR is set) then an interrupt is generated if the FEIE bit in LINIER is set. As there is only one register bit for a framing error, this interrupt will be helpful in identifying which byte has the framing error.

If the last received frame has not been read from the buffer (in other words, RMB is not reset by the user), then upon reception of the next byte, an overrun error occurs (BOF bit in UARTSR will be set) and one message is lost depending on the RBLM bit of LINCR1. An interrupt is generated if the BOIE bit in LINIER is set.

When WLS bit = 1, BDRM access depends on WL bit setting.

When WLS bit = 1, WL = 0 or 1 should not be used, since this will lead to incorrect reception of data.

In UART FIFO mode, when IP is requested to enter STOP mode, the SW has to first set the INIT bit to send the receiver to IDLE. This ensures that the IP acknowledges STOP request.

63.3.5 DMA interface

63.3.5.1 Main features

The LINFlexD DMA offers a parametric and programmable solution with the following distinctive features:

- LIN Master node, TX mode: single DMA channel
- LIN Master node, RX mode: single DMA channel
- LIN Slave node, TX mode: 1 to N DMA channel where N = max number of ID filters
- LIN Slave node, RX mode: 1 to N DMA channel where N = max number of ID filters
- UART node, TX mode: single DMA channel
- UART node, RX mode: single DMA channel + time-out

This release is backwards-compatible with the existing LINFlexD release if the new features are not enabled.

63.3.5.2 Definitions

Control/status register fields of the LINFlexD macrocell are described in *Table 1649* and *Table 1650*.

Table 1649. LINFlexD control/status fields description

Register	Field	Level	Description
LINCR2	DDRQ	High	Data discard request in reception mode
	DTRQ	High	Data transmission request (slave mode) of the LIN data field stored in BDRL/BDRM
	HTRQ	High	Header transmission request (master mode)
BIDR	DFL[5:0]	—	Data field length: 1 8 bytes: normal frames 1 64 bytes: extended frames
	DIR	—	Direction of the data field: 0 Reception 1 Transmission
	CCS	—	Checksum type: 0 Enhanced 1 Classic
	ID[5:0]	—	Identifier
LINSR	RMB	—	Buffer data ready to be read via CPU/DMA 0 Buffer data is free 1 Buffer data is ready
	DBFF	High	Data buffer full (8 bytes received) and DFL > 7 Used for extended frames.
	DBEF	High	Data buffer empty (8 bytes have been transmitted) and DFL > 7 Used for extended frames.
	DRF	High	Data reception completed
	DTF	High	Data transmission completed
	HRF	High	Header received Used in slave mode in case of Tx filter match.
IFMI	IFMIx	!= zero	Filter match index (slave mode)
UARTCR	FBM	—	FIFO/Buffer mode 0 Buffer mode 1 FIFO mode (mandatory in DMA mode)
UARTSR	RFE	—	Rx FIFO empty 0 Rx FIFO not empty 1 Rx FIFO empty
	TFF	—	Tx FIFO full 0 Tx FIFO not full 1 Tx FIFO full

Table 1649. LINFlexD control/status fields description (continued)

Register	Field	Level	Description

DMATXE[2**TX_CH_NUM](1)		High	DMA Tx channel enable (read/write)
			In UART or LIN master mode only channel 0 can be programmed.
			In LIN slave mode: # DMA TX channel = IFMI[3:0] – 1.
			DMATXE[x] = 0b → TX channel x disabled
			DMATXE[x] = 1b → TX channel x enabled

DMARXE[2**RX_CH_NUM](1)		High	DMA Rx channel enable (read/write)
			In UART or LIN master mode only the channel 0 can be programmed.
			In LIN slave mode: # DMA RX channel = IFMI[3:0] – 1.
			DMARXE[x] = 0b → RX channel x disabled
			DMARXE[x] = 1b → RX channel x enabled

1. ** stands for exponentiation

Table 1650. LINFlexD DMA control fields description

Field	Mode	Value	Level	Description
DMA_TEN	LIN master Tx or UART Tx	DMATXE[0] & dma_enb_req_tx[0]	High	Logical AND between the two DMA enable bits (LINFlexD and eDMA)
DMA_TEN	LIN slave Tx	DMATXE[x] & dma_enb_req_tx[x] x = IFMI – 1	High	
DMA_REN	LIN master Rx or UART Rx	DMARXE[0] & dma_enb_req_rx[0]	High	
DMA_REN	LIN slave Rx	DMARXE[x] & dma_enb_req_rx[x] x = IFMI – 1	High	

Control/status fields of the Tranfer Control Descriptors (TCDs) referred to in this document are described in *Table 1651*.

Table 1651. TCD control fields description

TCD field	Level	Description
CITER[14:0]	—	Current major iteration count
BITER[14:0]	—	Beginning major iteration count
NBYTES[31:0]	—	Inner minor byte transfer count Number of bytes to be transferred in each service request of the channel.
SADDR[31:0]	—	Source address
SOFF[15:0]	—	Source address signed offset Applied to the current source address as each source read is completed.

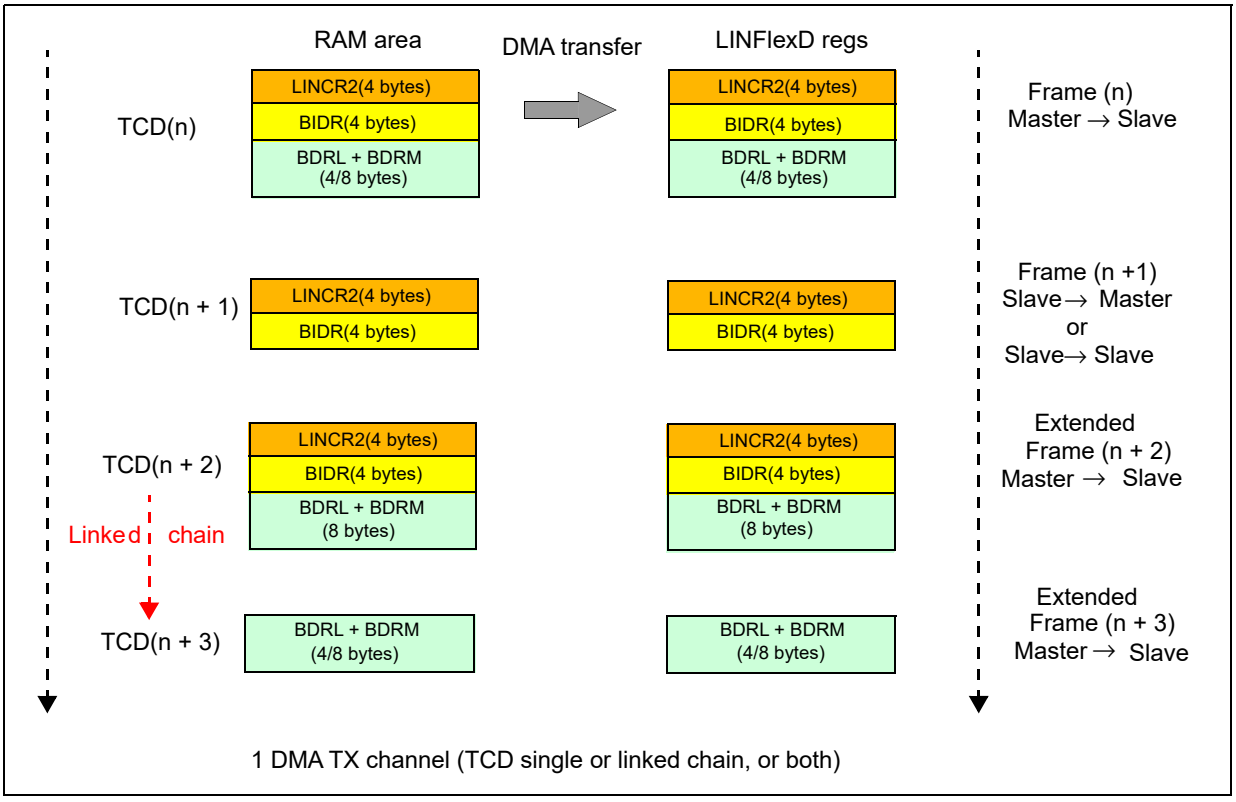
Table 1651. TCD control fields description (continued)

TCD field	Level	Description
SSIZE[2:0]	—	Source data transfer size 000 8-bit 001 16-bit 010 32-bit 011 64-bit
SLAST[31:0]	—	Last source address adjustment
DADDR[31:0]	—	Destination address
DOFF[15:0]	—	Destination address signed offset Applied to the current destination address as each destination write is completed.
DSIZE[2:0]	—	Destination data transfer size 000 8-bit 001 16-bit 010 32-bit 011 64-bit
DLAST_SGA[31:0]	—	Last destination address adjustment or the memory address for the next TCD to be loaded into this channel (scatter/gather)
INT_MAJ	High	Enable an interrupt when major iteration count completes
START	High	The channel is explicitly started via a software initiated service request
DONE	High	Channel done (the DMA has completed the outer major loop)
D_REQ	High	Disable request If this flag is set the DMA hardware automatically clears the corresponding DMAERQ bit when the current major iteration count reaches zero.

63.3.5.3 Master node: TX mode

On a master node, in TX mode the DMA interface requires a single TX channel. Each TCD controls a single frame, except for the extended frames (multiple TCDs). The memory map associated with the TCD chain (RAM area and LINFlexD registers) is given in *Figure 1678*.

Figure 1678. Master node: TX memory map



The TCD chain of the DMA Tx channel on a master node supports:

- Master to Slave: transmission of the entire frame (header + data)
- Slave to Master: transmission of the header; the data reception is controlled by the Rx channel on the master node
- Slave to Slave: transmission of the header

Table 1652 provides the register settings of the LINCR2 and BIDR for each class of LIN frame.

Table 1652. Master node: Tx mode - Register setting

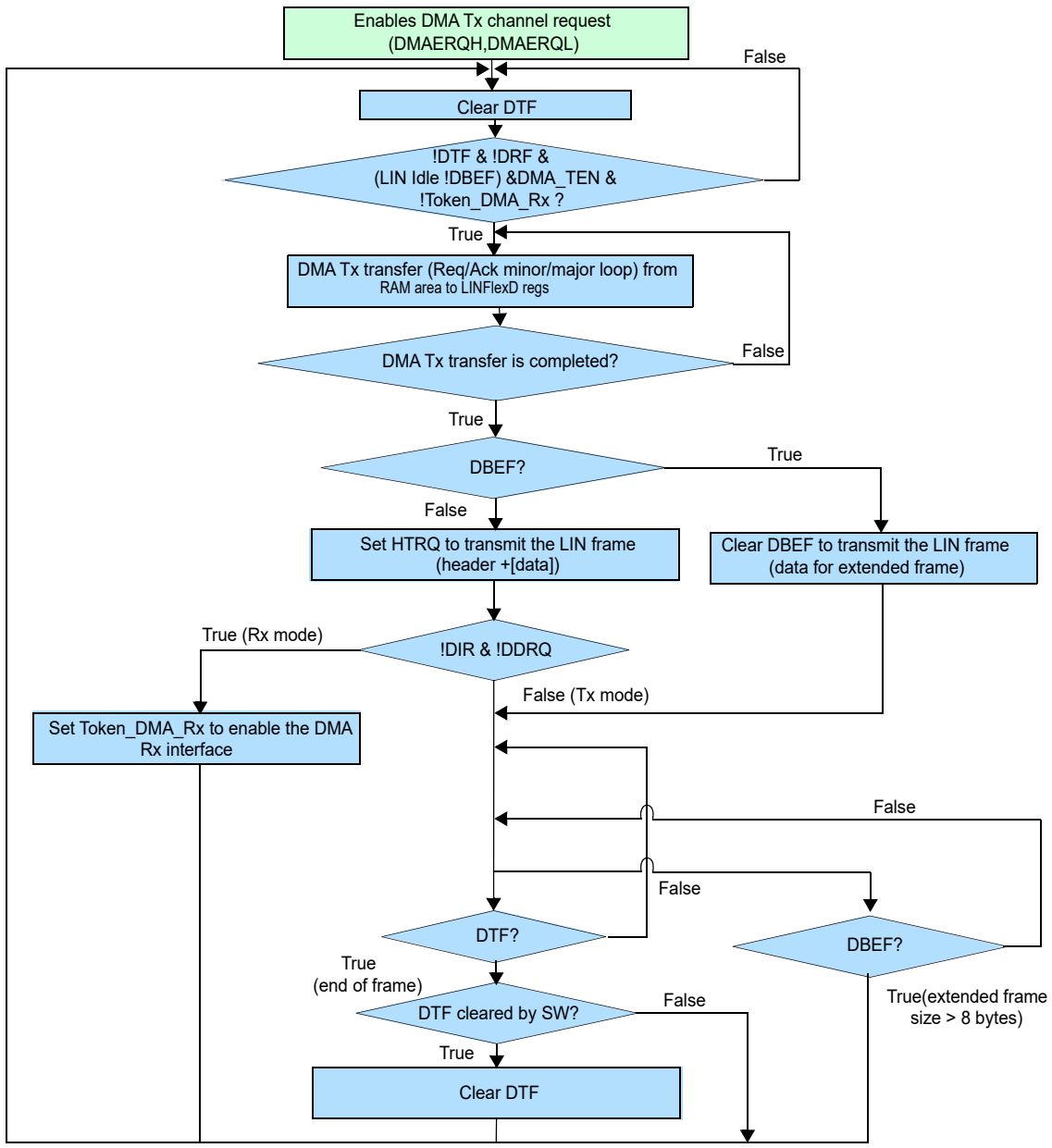
LIN frame	LINCR2	BIDR
Master to Slave	DDRQ = 1 DTRQ = 0 HTRQ = 0	DFL = payload size ID = address CCS = checksum DIR = 1 (TX)
Slave to Master	DDRQ = 0 DTRQ = 0 HTRQ = 0	DFL = payload size ID = address CCS = checksum DIR = 0 (RX)
Slave to Slave	DDRQ = 1 DTRQ = 0 HTRQ = 0	DFL = payload size ID = address CCS = checksum DIR = 0 (RX)

The concept FSM to control the DMA Tx interface is given in *Figure 1679*. DMA TX FSM moves to Idle state immediately at next clock edge if DMATXE[0] = 0. The TCD setting (word transfer) is given in *Table 1653*. All other TCD fields = 0. TCD settings based on halfword or byte transfer are allowed.

Table 1653. TCD setting: Master node -Tx mode

TCD field	Value	Description
CITER[14:0]	1	Single iteration for the major loop
BITER[14:0]	1	Single iteration for the major loop
NBYTES[31:0]	[4 + 4] + 0/4/8 = N	Data buffer is filled with dummy bytes if length is not word-aligned LINCR2 + BIDR + BDRL + BDRM
SADDR[31:0]	—	RAM address
SOFF[15:0]	4	Word increment
SSIZE[2:0]	2	Word transfer
SLAST[31:0]	–N	—
DADDR[31:0]	—	LINCR2 address
DOFF[15:0]	4	Word increment
DSIZE[2:0]	2	Word transfer
DLAST_SGA[31:0]	–N	No scatter/gather processing
INT_MAJ	0/1	Interrupt disabled/enabled
D_REQ	1	Only on the last TCD of the chain
START	0	No software request

Figure 1679. Master node: DMA Tx FSM (concept scheme)

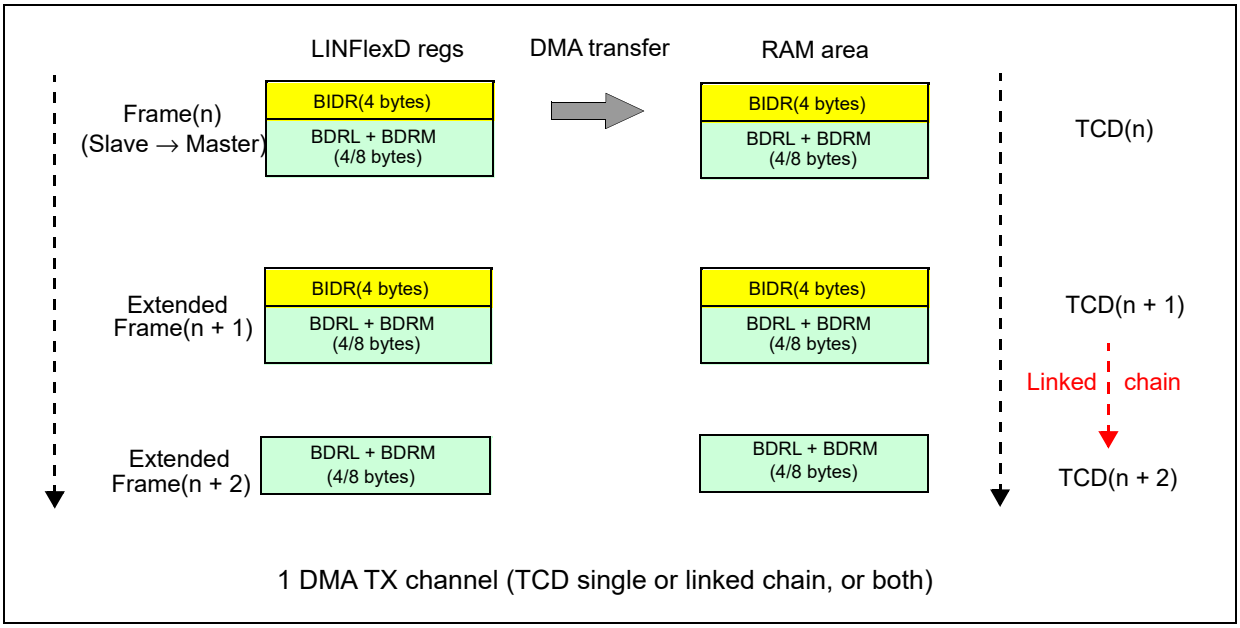


Note: DTF Flag gets auto cleared by Hardware if Software does not clear it within 3 ipg_clk cycles.

63.3.5.4 Master node: RX mode

On a master node in RX mode, the DMA interface requires a single RX channel. Each TCD controls a single frame, except for the extended frames (multiple TCDs). The memory map associated with the TCD chain (RAM area and LINFlexD registers) is given in *Figure 1680*.

Figure 1680. Master node: RX memory map



The TCD chain of the DMA Rx channel on a master node supports Slave to Master: reception of the data field of the header.

The BIDR register is optionally copied into the RAM area. This BIDR field (part of FIFO data) contains the ID of each message, which only allows the CPU to figure out which ID the LINFlexD DMA received if the single DMA channel setup is used.

The concept FSM to control the DMA Rx interface is given in *Figure 1681*. DMA RX FSM moves to Idle state immediately at the next clock edge if DMARXE[0] = 0. The TCD setting (word transfer) is given in *Table 1654*. All other TCD fields = 0. TCD settings based on halfword or byte transfer are allowed.

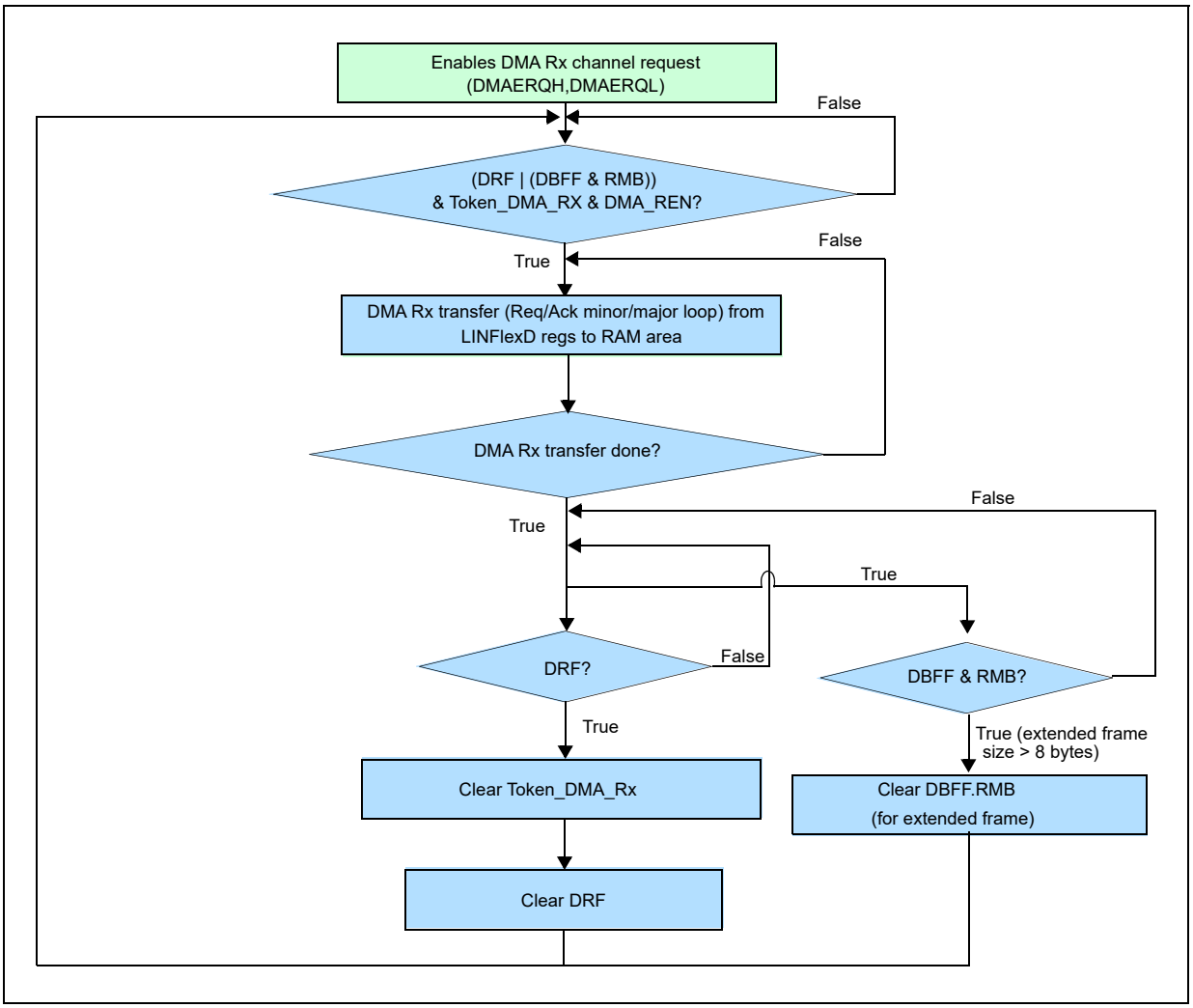
Table 1654. TCD setting: Master node - Rx mode

TCD field	Value	Description
CITER[14:0]	1	Single iteration for the major loop
BITER[14:0]	1	Single iteration for the major loop
NBYTES[31:0]	[4] +4/8 = N	Data buffer is filled with dummy bytes if length is not word-aligned BIDR + BDRL + BDRM
SADDR[31:0]	—	BIDR address
SOFF[15:0]	4	Word increment
SSIZE[2:0]	2	Word transfer
SLAST[31:0]	–N	—
DADDR[31:0]	—	RAM address
DOFF[15:0]	4	Word increment
DSIZE[2:0]	2	Word transfer
DLAST_SGA[31:0]	–N	No scatter/gather processing

Table 1654. TCD setting: Master node - Rx mode (continued)

TCD field	Value	Description
INT_MAJ	0/1	Interrupt disabled/enabled
D_REQ	1	Only on the last TCD of the chain
START	0	No software request

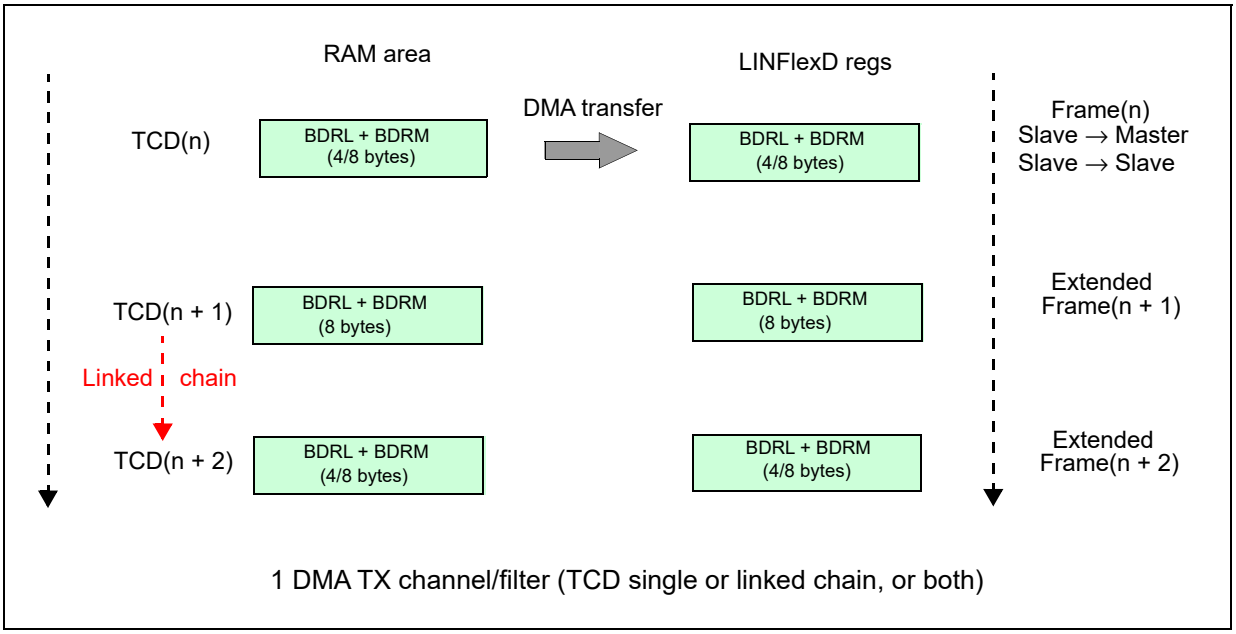
Figure 1681. Master node — DMA Rx FSM (concept scheme)



63.3.5.5 Slave node: TX mode

On a slave node in TX mode, the DMA interface requires a DMA TX channel for each ID filter programmed in TX mode. In case a single DMA TX channel is available, a single ID field filter must be programmed in TX mode. Each TCD controls a single frame, except for the extended frames (multiple TCDs). The memory map associated with the TCD chain (RAM area and LINFlexD registers) is given in *Figure 1682*.

Figure 1682. Slave node — TX memory map



The TCD chain of the DMA Tx channel on a slave node supports:

- Slave to Master: transmission of the data field
- Slave to Slave: transmission of the data field

The register setting of the LINCR2, IFER, IFMR, and IFCR registers is given in *Table 1655*.

Table 1655. Slave node: Tx mode - Register setting

LIN frame	LINCR2	IFER	IFMR	IFCR
Slave to Master or Slave to Slave	DDRQ = 0 DTRQ = 0 HTRQ = 0	To enable an ID filter (Tx mode) for each DMA TX channel	– Identifier list mode Identifier mask mode	DFL = payload size ID = address CCS = checksum DIR = 1 (TX)

The concept FSM to control the DMA Tx interface is given in *Figure 1683*. DMA TX FSM moves to Idle state if DMATXE[x] = 0 where x = IFMI – 1. The TCD setting (word transfer) is given in *Table 1656*. All other TCD fields = 0. TCD settings based on halfword or byte transfer are allowed.

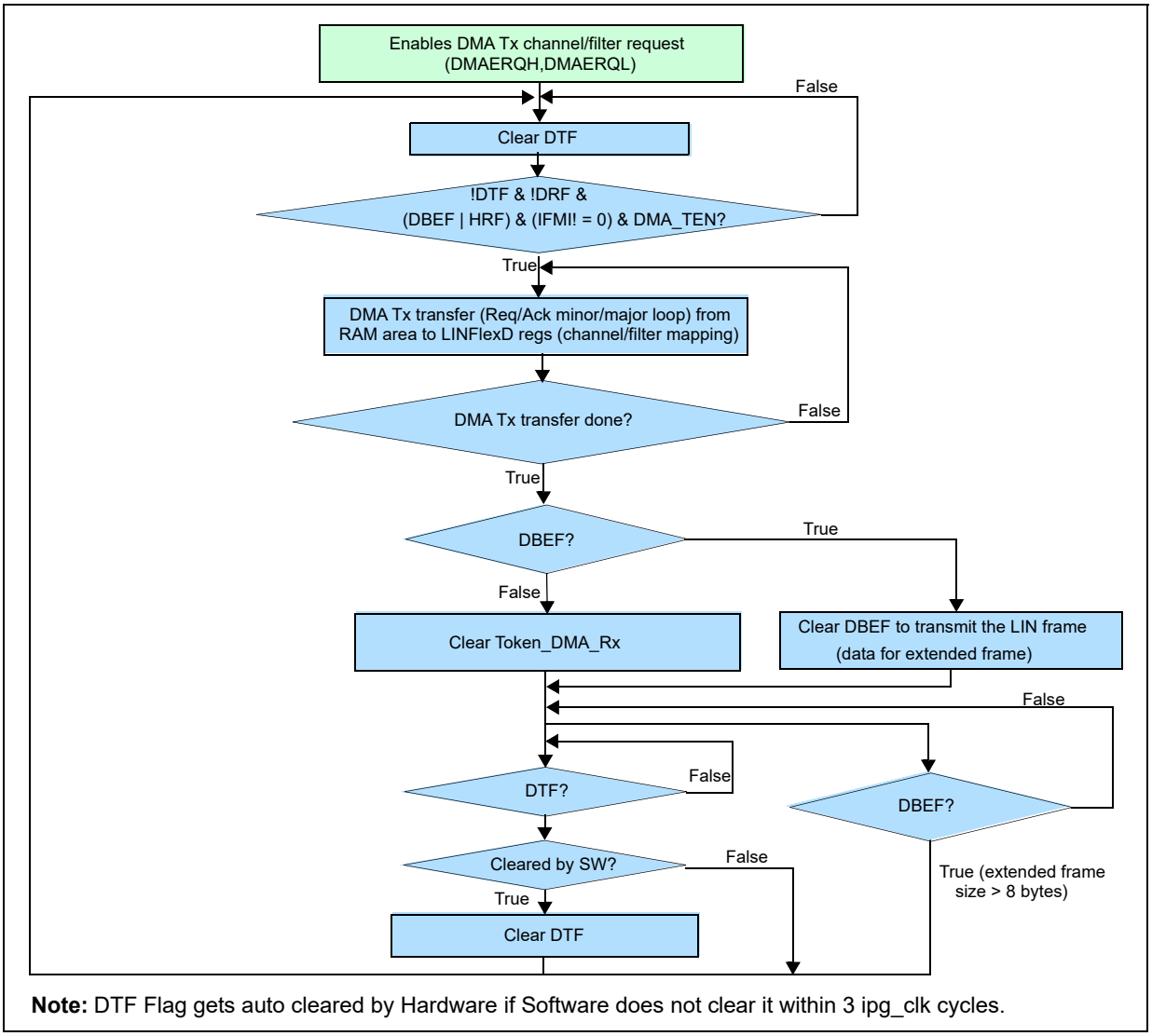
Table 1656. TCD setting: Slave node - Tx mode

TCD field	Value	Description
CITER[14:0]	1	Single iteration for the major loop
BITER[14:0]	1	Single iteration for the major loop
NBYTES[31:0]	4/8 = N	Data buffer is filled with dummy bytes if length is not word-aligned BDRL + BDRM
SADDR[31:0]	—	RAM address
SOFF[15:0]	4	Word increment

Table 1656. TCD setting: Slave node - Tx mode (continued)

TCD field	Value	Description
SSIZE[2:0]	2	Word transfer
SLAST[31:0]	-N	—
DADDR[31:0]	—	BDRL address
DOFF[15:0]	4	Word increment
DSIZE[2:0]	2	Word transfer
DLAST_SGA[31:0]	-N	No scatter/gather processing
INT_MAJ	0/1	Interrupt disabled/enabled
D_REQ	1	Only on the last TCD of the chain
START	0	No software request

Figure 1683. Slave node: DMA Tx FSM (concept scheme)

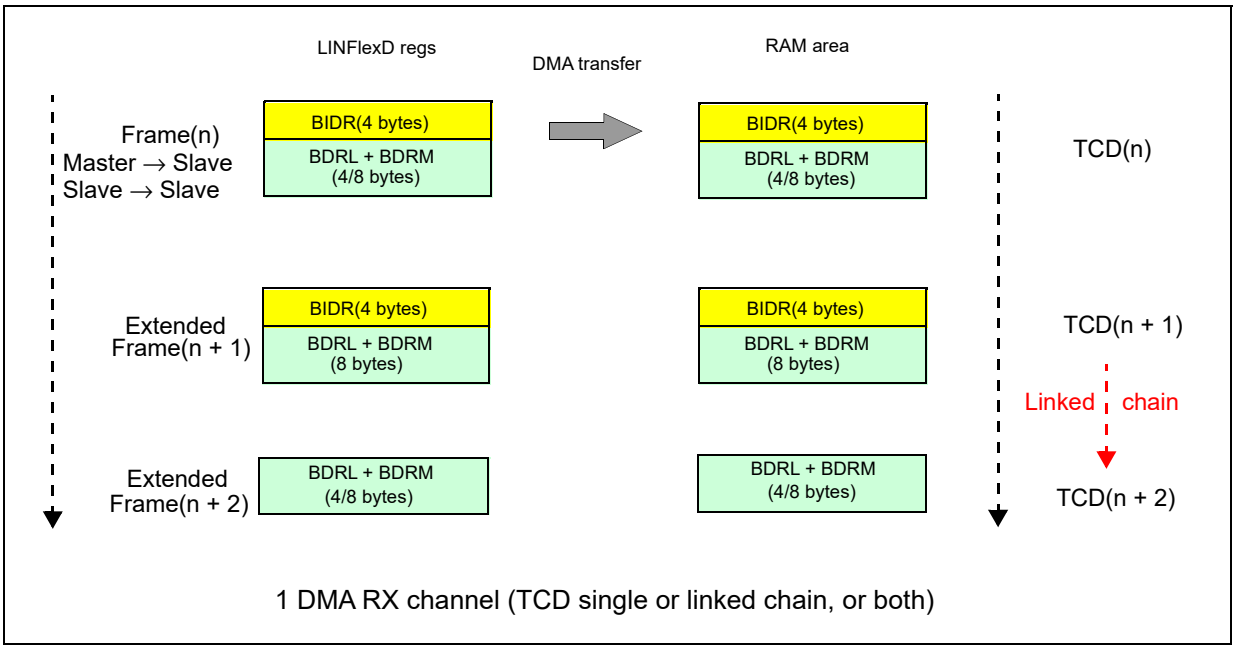


Note: DTF Flag gets auto cleared by Hardware if Software does not clear it within 3 ipg_clk cycles.

63.3.5.6 Slave node: RX mode

On a slave node in RX mode, the DMA interface requires a DMA RX channel for each ID filter programmed in RX mode. In case a single DMA RX channel is available, a single ID field filter must be programmed in RX mode. Each TCD controls a single frame, except for the extended frames (multiple TCDs). The memory map associated with the TCD chain (RAM area and LINFlexD registers) is given in *Figure 1684*.

Figure 1684. Slave node: RX memory map



The TCD chain of the DMA Rx channel on a slave node supports:

- Master to Slave: reception of the data field
- Slave to Slave: reception of the data field

The register setting of the LINCR2, IFER, IFMR, and IFCR registers is given in *Table 1657*.

Table 1657. Slave node: Rx mode - Register setting

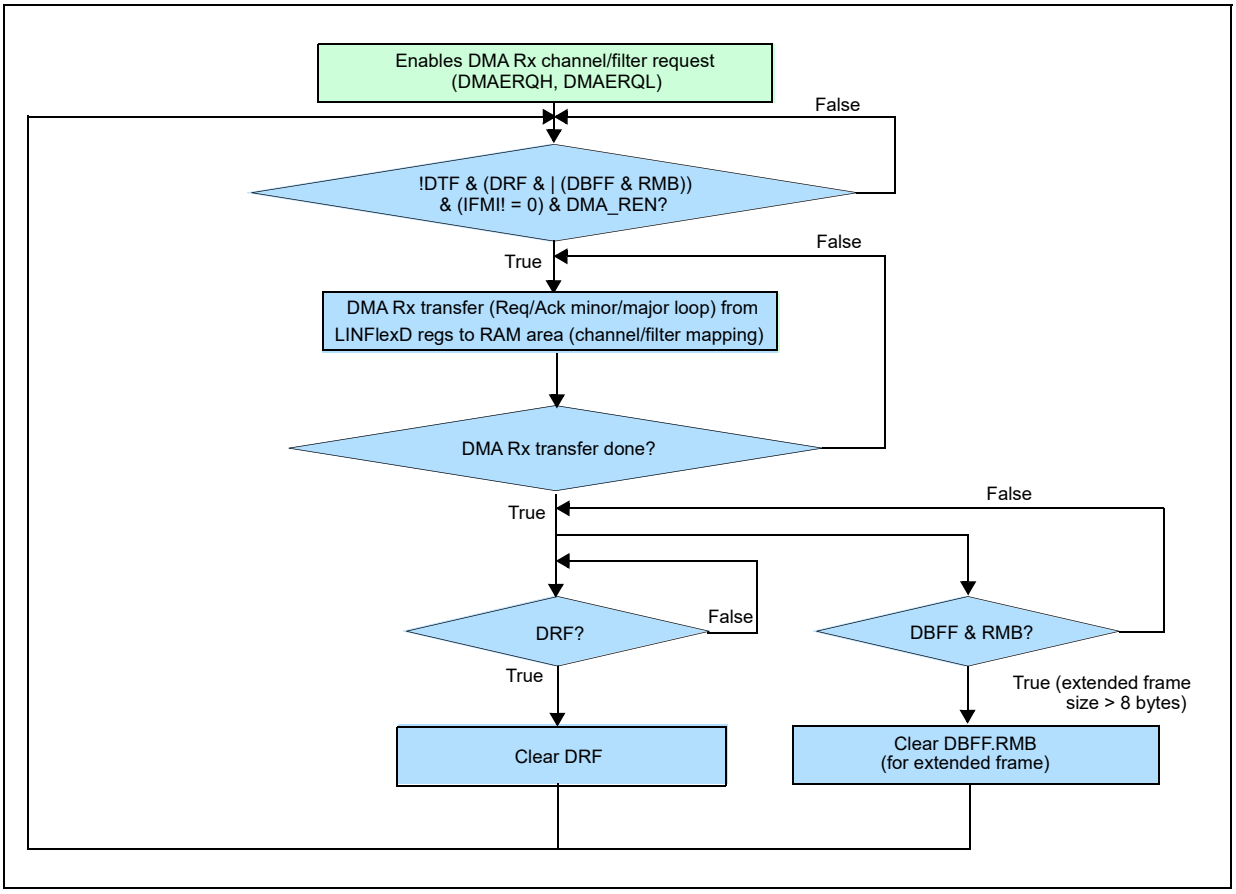
LIN frame	LINCR2	IFER	IFMR	IFCR
Master to Slave or Slave to Slave	DDRQ = 0 DTRQ = 0 HTRQ = 0	To enable an ID filter (Rx mode) for each DMA RX channel	– Identifier list mode Identifier mask mode	DFL = payload size ID = address CCS = checksum DIR = 0 (RX)

The concept FSM to control the DMA Rx interface is given in *Figure 1685*. DMA Rx FSM moves to Idle state if DMARXE[x] = 0 where x = IFMI – 1. The TCD setting (word transfer) is given in *Table 1658*. All other TCD fields = 0. TCD settings based on halfword or byte transfer are allowed.

Table 1658. TCD setting: Slave node - Rx mode

TCD field	Value	Description
CITER[14:0]	1	Single iteration for the major loop
BITER[14:0]	1	Single iteration for the major loop
NBYTES[31:0]	[4] + 4/8 = N	Data buffer is filled with dummy bytes if length is not word-aligned BIDR + BDRL + BDRM
SADDR[31:0]	—	BIDR address
SOFF[15:0]	4	Word increment
SSIZE[2:0]	2	Word transfer
SLAST[31:0]	-N	—
DADDR[31:0]	—	RAM address
DOFF[15:0]	4	Word increment
DSIZE[2:0]	2	Word transfer
DLAST_SGA[31:0]	-N	No scatter/gather processing
INT_MAJ	0/1	Interrupt disabled/enabled
D_REQ	1	Only on the last TCD of the chain
START	0	No software request

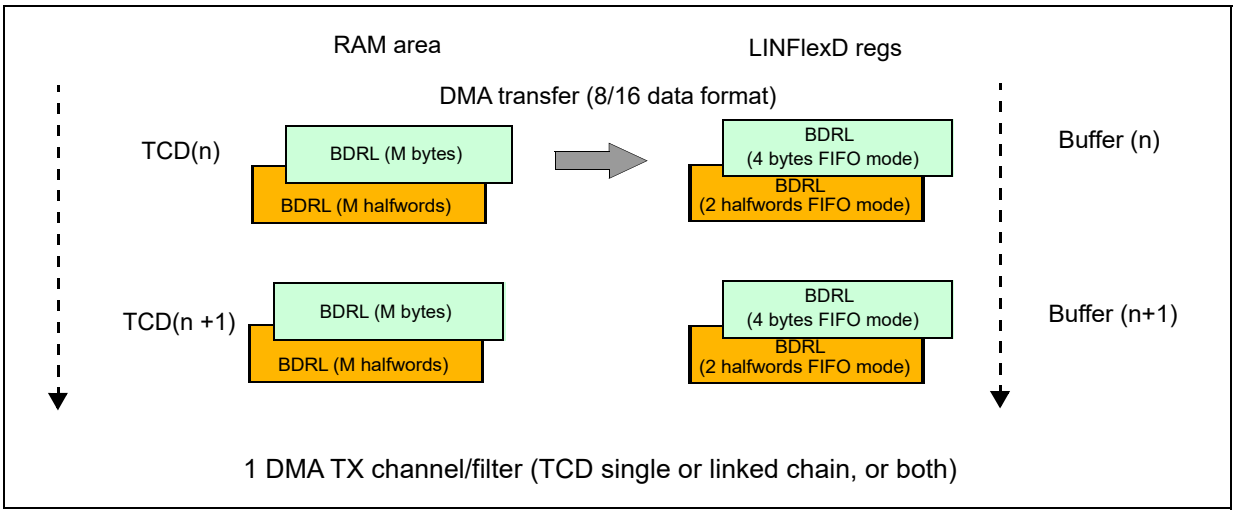
Figure 1685. Slave node — DMA Rx FSM (concept scheme)



63.3.5.7 UART: TX mode

In UART TX mode, the DMA interface requires a DMA TX channel. A single TCD can control the transmission of an entire Tx buffer. The memory map associated with the TCD chain (RAM area and LINFlexD registers) is given in *Figure 1686*.

Figure 1686. UART: TX memory map



The UART TX buffer must be configured in FIFO mode in order to:

- Allow the transfer of large data buffer by a single TCD
- Adsorb the latency, following a DMA request (due to the DMA arbitration), to move data from the RAM to the FIFO
- Use low priority DMA channels

The Tx FIFO size is:

- 4 bytes in 8-bit data format
- 2 halfwords in 16-bit data format

A DMA request is triggered by FIFO-not-full (TX) status signals.

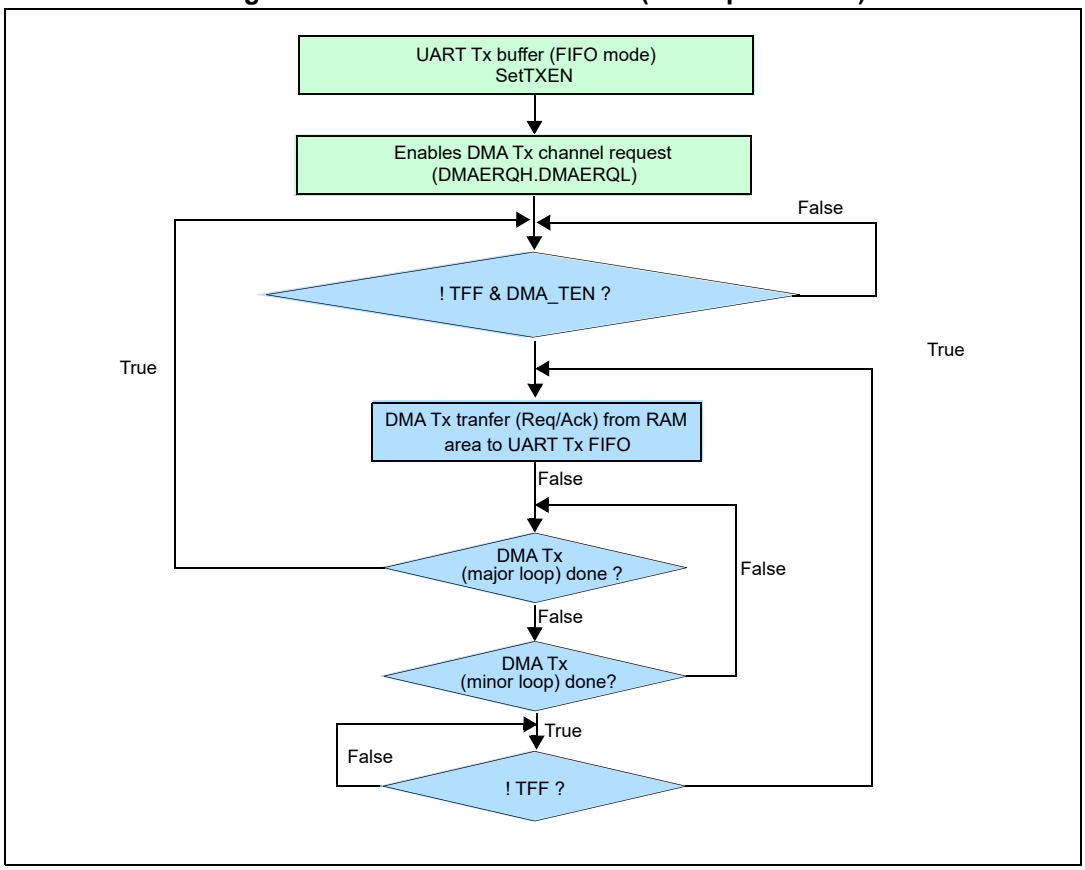
The concept FSM to control the DMA Tx interface is given in *Figure 1687*. DMA Tx FSM will move to Idle state if DMATXE[0] = 0. The TCD setting (typical case) is given in *Table 1659*. All other TCD fields = 0. The minor loop transfers a single byte/halfword as soon as a free entry is available in the Tx FIFO.

Table 1659. TCD setting: UART - Tx mode

TCD field	Value		Description
	8 bits data	16 bits data	
CITER[14:0]	M		Multiple iterations for the major loop
BITER[14:0]	M		Multiple iterations for the major loop
NBYTES[31:0]	1	2	Minor loop transfer = 1 or 2 bytes
SADDR[31:0]	—		RAM address
SOFF[15:0]	1	2	Byte/halfword increment
SSIZE[2:0]	0	1	Byte/halfword transfer
SLAST[31:0]	-M	-M × 2	—
DADDR[31:0](1)	—		BDRL address
DOFF[15:0]	0		No increment (FIFO)
DSIZE[2:0]	0	1	Byte/halfword transfer
DLAST_SGA[31:0]	0		No scatter/gather processing
INT_MAJ	0/1		Interrupt disabled/enabled
D_REQ	1		Only on the last TCD of the chain
START	0		No software request

1. DADDR[31:0] = BDRL + 0x3 for byte transfer — DADDR [31:0] = BDRL + 0x2 for halfword transfer.

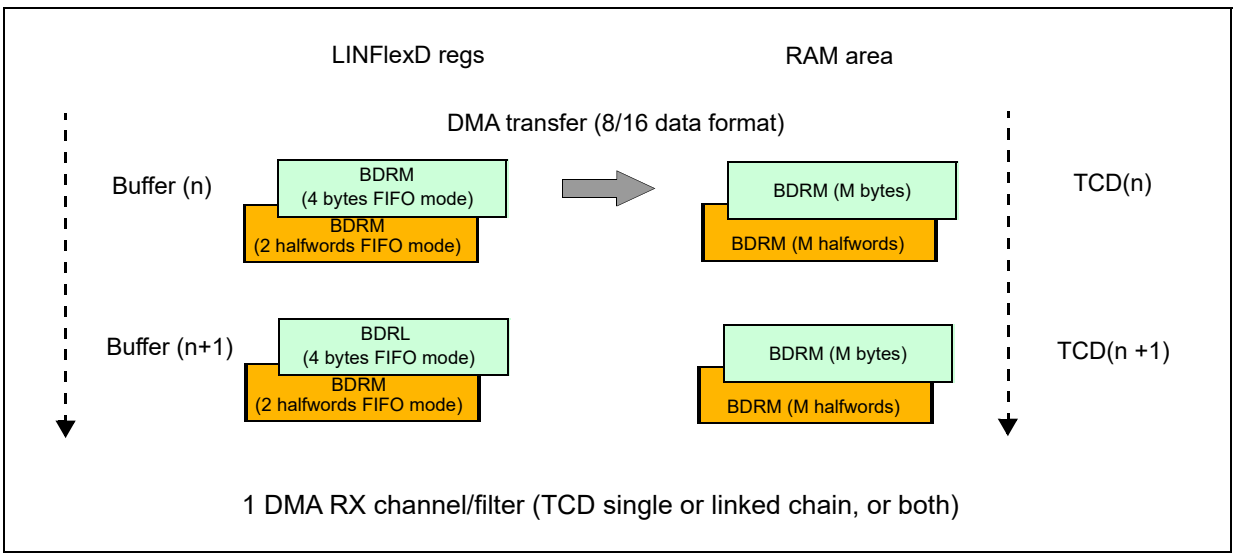
Figure 1687. UART: DMA Tx FSM (concept scheme)



63.3.5.8 UART: RX mode

In UART RX mode, the DMA interface requires a DMA RX channel. A single TCD can control the reception of an entire Rx buffer. The memory map associated with the TCD chain (RAM area and LINFlexD registers) is given in *Table 1660*.

Figure 1688. UART: RX memory map



The UART RX buffer must be configured in FIFO mode in order to:

- Allow the transfer of large data buffer by a single TCD
- Adsorb the latency, following a DMA request (due to the DMA arbitration), to move data from the FIFO to the RAM
- Use low priority DMA channels

The Rx FIFO size is:

- 4 bytes in 8-bit data format
- 2 halfwords in 16-bit data format

This will be sufficient because just one byte allows a reaction time of about 3.8 µs (at 2 Mbit/s) (~450 clock cycles at 120 MHz) before the transmission is affected. A DMA request is triggered by FIFO and not by empty (Rx) status signals.

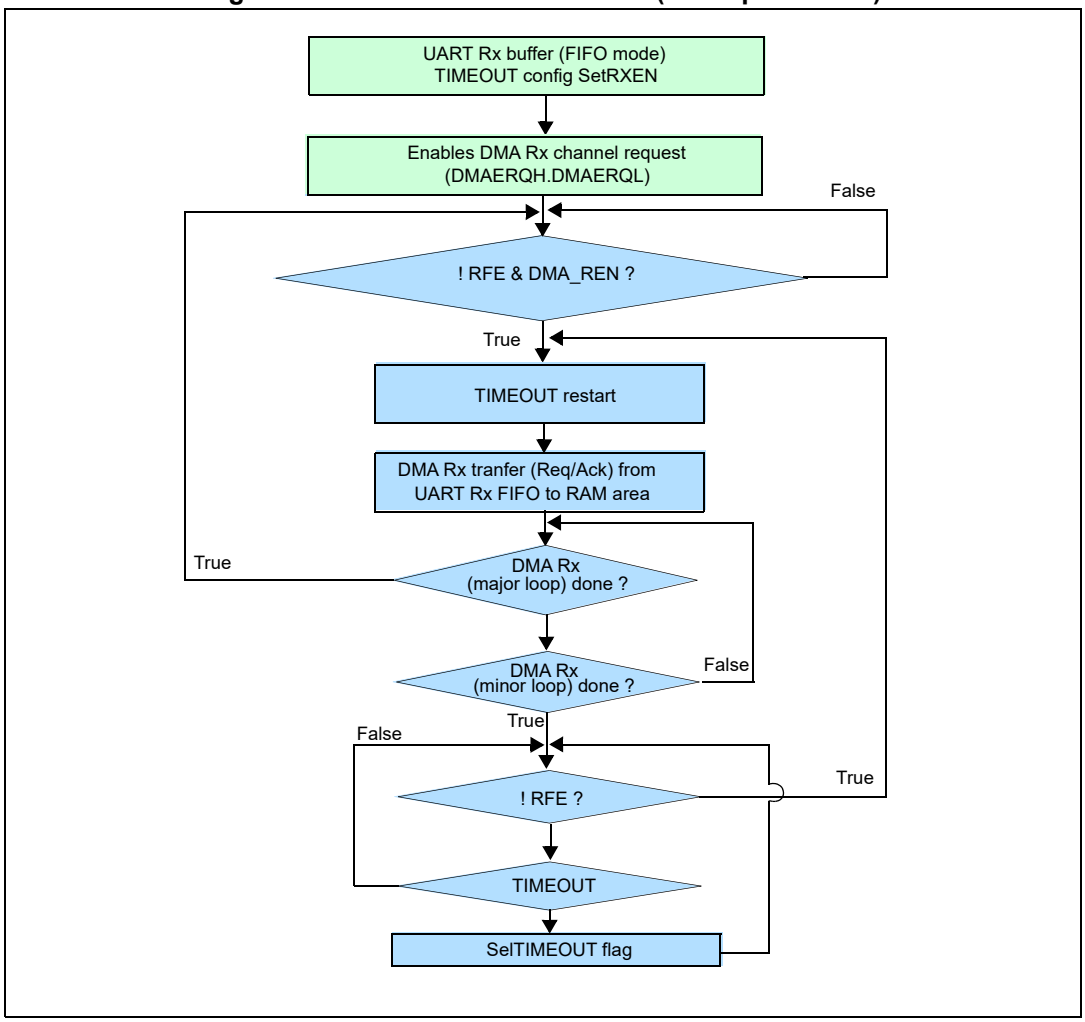
The concept FSM to control the DMA Rx interface is given in *Figure 1689*. DMA Rx FSM will move to Idle state if DMARXE[0] = 0. The TCD setting (typical case) is given in the next table. All other TCD fields equal zero. The minor loop transfers a single byte/halfword as soon an entry is available in the Rx FIFO. A new software reset bit is required that allows the LINFlexD FSMs to be reset in case this timeout state is reached or in any other case. The timeout counter can be re-written by software at any time to extend the timeout period.

Table 1660. TCD setting: UART - Rx mode

TCD field	Value		Description
	8 bits data	16 bits data	
CITER[14:0]	M		Multiple iterations for the major loop
BITER[14:0]	M		Multiple iterations for the major loop
NBYTES[31:0]	1	2	Minor loop transfer = 1 or 2 bytes
SADDR[31:0](1)	—		BDRM address
SOFF[15:0]	0		No increment (FIFO)
SSIZE[2:0]	0	1	Byte/halfword transfer
SLAST[31:0]	0		—
DADDR[31:0]	—		RAM address
DOFF[15:0]	1	2	Byte/halfword increment
DSIZE[2:0]	0	1	Byte/halfword transfer
DLAST_SGA[31:0]	-M	-M × 2	No scatter/gather processing
INT_MAJ	0/1		Interrupt disabled/enabled
D_REQ	1		Only on the last TCD of the chain
START	0		No software request

1. SADDR = BDRM + 0x3 for byte transfer; SADDR[31:0] = BDRM + 0x2 for halfword transfer.

Figure 1689. UART — DMA Rx FSM (concept scheme)



63.3.5.9 DMA interface timing

The LINFlexD DMA interface includes the following signals:

- ipd_req
  - DMA peripheral request, asserted until the DMA transfer has been completed.
- dma_done_lw
  - DMA minor loop done. It indicates that the channel's data transfer of nbytes (the minor loop) has been completed. It is asserted for one cycle after the data phase of the channel's last AMBA AHB write access, or when an error condition is detected.
- dma_ipd_complete
  - DMA major loop finished. It is asserted for one cycle after the data phase of the last write.
- dma_ipd_ack, dma_ipd_done
  - Unused in this implementation.

All signals of the DMA interface must be registered in order to meet timing requirements of the DMA engine.

Figure 1690. DMA interface (major loop — single iteration)

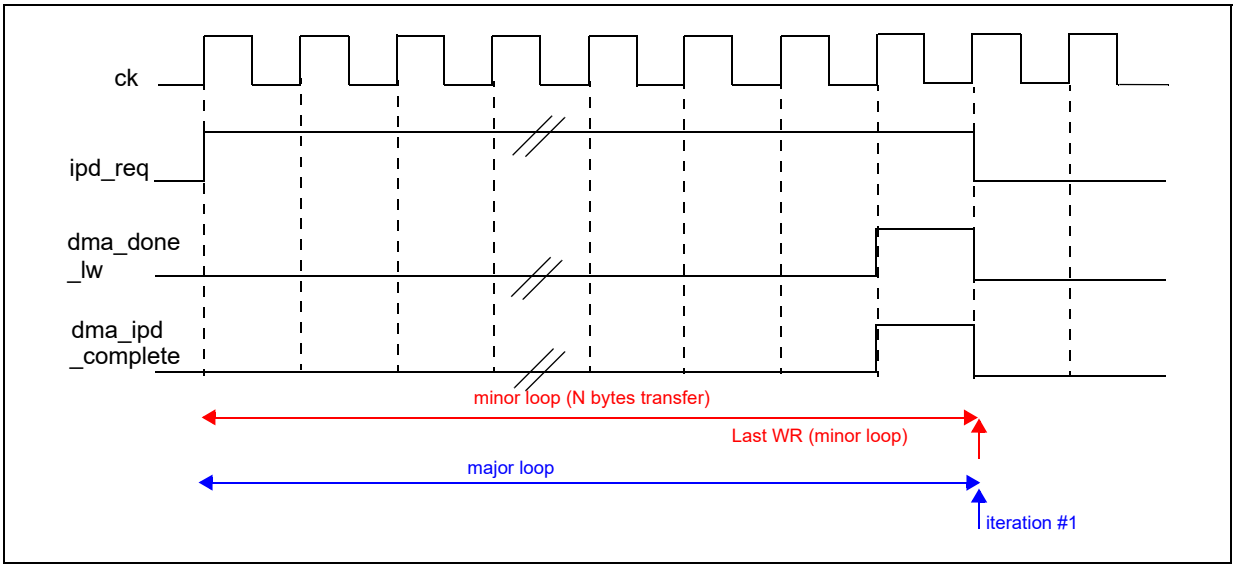
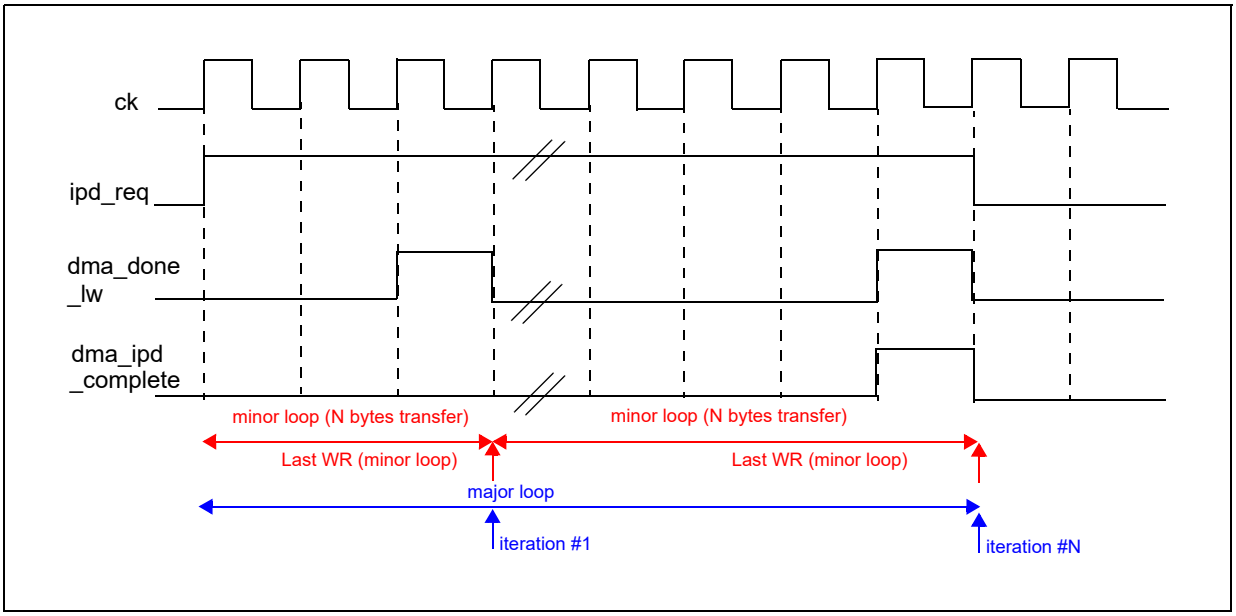


Figure 1691. DMA interface (major loop — N iteration)



63.3.5.10 Use cases and limitations

- In LIN slave mode, the DMA capability can be used only if the ID filtering mode is activated. The number of ID filters enabled must be equal to the number of DMA

channels enabled. The correspondence between channel number and ID filter is based on IFMI (identifier filter match index).

- In LIN master mode both the DMA channels (TX and RX) must be enabled in case the DMA capability is required.
- In UART mode the DMA capability can be used only if the UART Tx/Rx buffers are configured as FIFOs.
- DMA and CPU operating modes are mutually exclusive for the data/frame transfer on a UART or LIN node. Once a DMA transfer is finished the CPU can manage subsequent accesses.
- Error management must always be executed via the CPU enabling the related error interrupt sources. DMA capability does not provide support for error management. Error management means checking status bits, handling IRQs, and potentially canceling DMA transfers.
- The DMA programming model must be coherent with the TCD setting defined in this document
- When IPG_STOP is requested, SW has to first disable DMATXE/DMARXE channel registers, after the current minor loop finishes (indicated by the clearing of ACTIVE bit in DMA TCD register). This ensures that IPG_STOP_ACK is generated and IP enters STOP mode.

63.4 Memory map and register description

Note: Refer to the LINFlexD configuration section of the Device Configuration chapter for registers implementation per LINFlexD instance.

63.4.1 Memory map

Table 1661 shows the LINFlexD memory map.

Note: In Master mode, the registers IFCR0–IFCR15 are not present.

In Master Mode, read access to IFMI register and read/write access to IFER and IFMR registers would result in bus transfer error.

Table 1661. LINFlexD memory map

Address offset (hex)	Register description	Location
0x00	LIN Control register 1 (LINCR1)	*Section 63.4.2.1*
0x04	LIN Interrupt Enable register (LINIER)	*Section 63.4.2.2*
0x08	LIN Status register (LINSR)	*Section 63.4.2.3*
0x0C	LIN Error Status register (LINESR)	*Section 63.4.2.4*
0x10	UART Mode Control register (UARTCR)	*Section 63.4.2.5*
0x14	UART Mode Status register (UARTSR)	*Section 63.4.2.6*
0x18	LIN Timeout Control Status register (LINTCSR)	*Section 63.4.2.7*
0x1C	LIN Output Compare register (LINOCR)	*Section 63.4.2.8*
0x20	LIN Timeout Control register (LINTOCR)	*Section 63.4.2.9*

Table 1661. LINFlexD memory map (continued)

Address offset (hex)	Register description	Location
0x24	LIN Fractional Baud Rate register (LINFBRR)	*Section 63.4.2.10*
0x28	LIN Integer Baud Rate register (LINIBRR)	*Section 63.4.2.11*
0x2C	LIN Checksum Field register (LINCFR)	*Section 63.4.2.12*
0x30	LIN Control register 2 (LINCR2)	*Section 63.4.2.13*
0x34	Buffer Identifier register (BIDR)	*Section 63.4.2.14*
0x38	Buffer Data Register Least Significant (BDRL)	*Section 63.4.2.15*
0x3C	Buffer Data Register Most Significant (BDRM)	*Section 63.4.2.16*
0x40	Identifier Filter Enable register (IFER)	*Section 63.4.2.17*
0x44	Identifier Filter Match index (IFMI)	*Section 63.4.2.18*
0x48	Identifier Filter Mode register (IFMR)	*Section 63.4.2.19*
0x4C	Identifier Filter Control registers (IFCR0)	*Section 63.4.2.20*
0x50	Identifier Filter Control registers (IFCR1)	*Section 63.4.2.21*
0x54	Identifier Filter Control registers (IFCR2)	*Section 63.4.2.20*
0x58	Identifier Filter Control registers (IFCR3)	*Section 63.4.2.21*
0x5C	Identifier Filter Control registers (IFCR4)	*Section 63.4.2.20*
0x60	Identifier Filter Control registers (IFCR5)	*Section 63.4.2.21*
0x64	Identifier Filter Control registers (IFCR6)	*Section 63.4.2.20*
0x68	Identifier Filter Control registers (IFCR7)	*Section 63.4.2.21*
0x6C	Identifier Filter Control registers (IFCR8)	*Section 63.4.2.20*
0x70	Identifier Filter Control registers (IFCR9)	*Section 63.4.2.21*
0x74	Identifier Filter Control registers (IFCR10)	*Section 63.4.2.20*
0x78	Identifier Filter Control registers (IFCR11)	*Section 63.4.2.21*
0x7C	Identifier Filter Control registers (IFCR12)	*Section 63.4.2.20*
0x80	Identifier Filter Control registers (IFCR13)	*Section 63.4.2.21*
0x84	Identifier Filter Control registers (IFCR14)	*Section 63.4.2.20*
0x88	Identifier Filter Control registers (IFCR15)	*Section 63.4.2.21*
0x8C	Global Control register (GCR)	*Section 63.4.2.22*
0x90	UART Preset Timeout register (UARTPTO)	*Section 63.4.2.23*
0x94	UART Current Timeout register (UARTCTO)	*Section 63.4.2.24*
0x98	DMA Tx Enable register (DMATXE)	*Section 63.4.2.25*
0x9C	DMA Rx Enable register (DMARXE)	*Section 63.4.2.26*

63.4.2 Register description

The following points should be considered for the below mentioned LINFlexD registers:

- Reset values are with-slave/without-slave (master only) format.
- If not specified, then the bit can be configured for both master only and master/slave format; in other words, for both values of generic slave = 0 and slave = 1.

63.4.2.1 LIN Control Register 1 (LINCR1)

LINCR1 consists of control bits used to configure features of the LINFlexD.

Address: 0x0000 Access: User read/write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	CCD(1)	CFD(1)	LASE(1)	AUTOWU(1)	MBL(1)				BF(1)	0	LBKM(1)	MME(1)	SSBL(1)	RBLM(1)	SLEEP	INIT
W																
Reset	0	0	0	0	0	0	0	0	1	0	0	0/1(2)	0	0	1	0

1. This field can be read in any modes and written only in *Initialization* mode.
2. When slave = 0, this field reads '1' and cannot be programmed.

Figure 1692. LIN Control Register 1 (LINCR1)

Table 1662. LINCR1 field descriptions

Field	Description
16 CCD	Calculation disable 0 Checksum calculation is done by hardware. When this bit is reset the LINCFR register is read-only. 1 Checksum calculation is disabled. When this bit is set the LINCFR register is read/write. User can program this register to send a software calculated checksum/CRC (provided CFD is reset). This bit can be written during Initialization mode only. It is read-only in Normal mode.
17 CFD	Checksum field disable 0 Checksum field is sent after the required number of data bytes are sent 1 No checksum field is sent in the frame This bit can be configured during Initialization mode only. It is read-only in Normal mode.
18 LASE	LIN Auto Synchronization Enable 0 Auto synchronization disabled 1 Auto synchronization enabled This bit can be programmed in Initialization mode only. It is read-only in Normal mode. (Note: If generic auto_sync = 0, then this bit will always read a 0).

Table 1662. LINCR1 field descriptions (continued)

Field	Description
19 AUTOWU	Auto Wakeup 0 Sleep bit is cleared by software only 1 Sleep bit gets cleared by hardware whenever WUF bit of LINSR is set This bit can be configured during Initialization mode only. This bit is utilized in UART mode also.
20:23 MBL[3:0]	Master Break Length These bits choose the length of the Sync break to be generated by the master. 0x0 10-bit break length 0x1 11-bit break length 0x2 12-bit break length 0x3 13-bit break length 0x4 14-bit break length 0x5 15-bit break length 0x6 16-bit break length 0x7 17-bit break length 0x8 18-bit break length 0x9 19-bit break length 0xA 20-bit break length 0xB 21-bit break length 0xC 22-bit break length 0xD 23-bit break length 0xE 36-bit (cooling) break length 0xF 50-bit break length These bits can be programmed in Initialization mode only. They are read-only in Normal mode.
24 BF	By-pass filter 0 No IRQ if ID does not match any filter 1 A RX IRQ is generated on ID not matching any filter This bit can be programmed during Initialization mode only. Note: If generic slave = 0 or no_of_filters = 0, then this bit will always read a 1, and cannot be programmed.
25	Reserved
26 LBKM	Loop Back mode 0 Loop Back Mode disabled 1 Loop Back mode enabled Refer to *Section 63.3.2.2*. Note that this bit can be programmed only in Initialization mode. This bit is utilized in UART mode also.
27 MME	Master mode enable 0 Slave Mode 1 Master Mode This bit can be programmed in Initialization mode only. It is read-only in Normal mode. If generic *slave* = 0, then this bit will read a ‘1’ always, and cannot be programmed.

Table 1662. LINCR1 field descriptions (continued)

Field	Description
28 SSBL	Slave Mode Sync Break Length 0 11-bit break length 1 10-bit break length This bit can be programmed in Initialization mode only. It is read-only in Normal mode.
29 RBLM	Receiver Buffer Locked mode 0 Receiver Buffer not locked, next incoming message will overwrite the old one 1 Receiver buffer locked against overrun. Once the buffer is full the next incoming message will be discarded if buffer is not released — in other words, RMB is not reset by software. This bit can be programmed in Initialization mode only. It is read-only in Normal mode. This bit is utilized in UART mode also.
30 SLEEP	Sleep Mode Request This bit is set by software to request LINFlexD to enter Sleep mode. This bit is cleared by software or hardware (if AUTOWU bit in LINCR1 and WUF bit in LINSR are set) to exit sleep mode. This bit is utilized in UART mode also.
31 INIT	Initialization Mode Request The software sets this bit to switch the hardware into Initialization mode. On clearing this bit (and if SLEEP bit is also zero) LINFlexD enters normal mode. This bit is utilized in UART mode also.

63.4.2.2 LIN Interrupt enable register (LINIER)

Address: 0x0004 Access: User read/write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	SZIE	OCIE	BEIE	CEIE	HEIE	0	0	FEIE	BOIE	LSIE	WUIE	DBFIE	DBEIE_TOIE	DRIE	DTIE	HRIE
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1693. LIN Interrupt enable register (LINIER)

Table 1663. LINIER field descriptions

Field	Description
16 SZIE	Stuck at Zero Interrupt Enable 0 No interrupt 1 Interrupt enabled An interrupt is generated if this bit is set and the Stuck at Zero Flag (SZF) in LINESR or UARTSR is set.
17 OCIE	Output Compare Interrupt Enable 0 No interrupt 1 Interrupt generated when OCF bit in LINESR or UARTSR is set
18 BEIE	Bit Error Interrupt Enable 0 No interrupt 1 Interrupt generated when BEF bit in LINESR is set
19 CEIE	Checksum Error Interrupt Enable 0 No interrupt 1 Interrupt enabled An interrupt is generated if this bit is set and the Checksum Error Flag (CEF) is set in LINESR.
20 HEIE	Header Error Interrupt Enable 0 No interrupt 1 Interrupt enabled An interrupt is generated when this bit is set and either of the following flags are set SFEF, SDEF, IDPEF in LINESR are set. **Note:** If generic slave = 0, then this bit will always read a 0 and cannot be programmed.
23 FEIE	Frame Error Interrupt Enable 0 No interrupt 1 Interrupt generated if Frame Error Flag (FEF) bit is set in LINESR or UARTSR
24 BOIE	Buffer Overrun Error Interrupt Enable 0 No interrupt 1 Interrupt enabled An interrupt is generated if this bit is set and the Buffer Overrun Flag (BOF) is set in LINESR or UARTSR.
25 LSIE	LIN state Interrupt enable 0 No interrupt 1 Interrupt generated on entering the following states: Sync Del, Sync Field, Identifier field, Checksum Interrupt is generated only when entering the above fields. This interrupt can mainly be used for debugging purposes. The interrupt has no status flag.
26 WUIE	Wakeup interrupt enable 0 No interrupt 1 Interrupt enabled If this bit is set and the WUF in LINSR or UARTSR is set then an interrupt is generated.

Table 1663. LINIER field descriptions (continued)

Field	Description
27 DBFIE	Data Buffer Full Interrupt enable 0 No interrupt 1 Interrupt enabled An interrupt is generated if this bit is set and the DBFF bit in LINSR is also set.
28 DBEIE_TOIE	Data Buffer Empty Interrupt enable/Timeout Interrupt Enable 0 No interrupt 1 Interrupt enabled An interrupt is generated if this bit is set and LINSR[DBEF] status bit is set (in LIN mode) or if this bit is set and UARTSR[TO] status bit is set (in UART mode).
29 DRIE	Data Reception complete Interrupt enable 0 No interrupt 1 Interrupt enabled An interrupt is generated when this bit is set and Data Received flag (DRF) in LINSR or UARTSR is set.
30 DTIE	Data Transmitted Interrupt enable 0 No interrupt 1 Interrupt enabled An interrupt is generated when this bit is set and Data Transmitted flag (DTF) in LINSR or UARTSR is set.
31 HRIE	Header Received Interrupt 0 No interrupt 1 Interrupt enabled An interrupt is generated when this bit is set and the Header Received flag (HRF) in LINSR is set. **Note:** If generic slave = 0, then this bit will always read a 0, and cannot be programmed.

63.4.2.3 LIN Status Register (LINSR)

This register consists of status bits indicating the state of the LINFlexD hardware.

Address: 0x0008 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	AUTOSYNC_COMP	RDC		
W													w1c			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	LINS				0	0	RMB	DRBNE	RXBUSY	RDI	WUF	DBFF	DBEF	DRF	DTF	HRF
W							w1c	w1c			w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0

Figure 1694. LIN Status Register (LINSR)

Table 1664. LINSR field descriptions

Field	Description
12 AUTOSYNC_COMP	Autosynchronization Completed This bit is set after autosynchronization is complete and when the LASE bit in LINCR1 register is enabled. Only after this bit is set, the contents of LINIBRR and LINFBRR registers should be read in autosynchronization mode.
13:15 RDC	Receive Data Byte Count RDC contains the number of entries (bytes) in the Receive data buffer in LIN mode. RDCx is a read-only field available for debug purposes. 000 1 byte 001 2 bytes 010 3 bytes 011 4 bytes 100 5 bytes 101 6 bytes 110 7 bytes 111 8 bytes

Table 1664. LINSR field descriptions (continued)

Field	Description
16:19 LINS	LIN state 0x0 Sleep mode LINFlexD is in Sleep mode to save power consumption. 0x1 Init 0x2 Idle This mode is entered when: SLEEP bit and INIT bit are reset by software, Wakeup pulse has been received on RX pin (AUTOWU set), Previous frame transmission/reception has been completed/aborted. 0x3 Sync Break In slave mode, a falling edge followed by a dominant state has been detected. Receiving sync break. In master mode, sync break transmission ongoing. In slave mode upon any error LIN state could be either Idle or Rec Break, depending on last bit detected on LIN_RX. If last bit detected is dominant then Rec_Break, otherwise Idle. 0x4 Sync Del In Slave mode, valid Sync break has been detected (10 bit or 11 bit). Waiting for a rising edge. In Master mode, Sync break transmission has been completed, sync delimiter transmission is ongoing. 0x5 Sync Field In Slave mode, a valid sync Del has been detected (recessive state for at least one bit time). Receiving sync field. In Master mode, sync field transmission ongoing. 0x6 Identifier Field In Slave mode, a valid sync field has been received. Receiving ID field. In Master mode, identifier transmission is ongoing. 0x7 Header Reception/Transmission In Slave mode, a valid header has been received and Identifier field is available in the BIDR. In Master mode, header transmitted. 0x8 Data Reception/Data Transmission In Receiver mode, reception ongoing. In Transmitter mode, response transmission ongoing. 0x9 Checksum Data transmission/reception completed, checksum transmission/reception ongoing. In UART mode Idle, Init, Sleep, and Data Transmission/Reception states are flagged by the LIN status bits.
22 RMB	Release Message Buffer 0 Buffer data is free. Reset by hardware in when in Initialization mode 1 Buffer data ready to be read by software. This bit should be cleared by software after reading the data received in the buffer.
23 DRBNE	Data Reception Buffer Not Empty Flag This bit is set by hardware as soon as the first byte of response has been received and stored in BDRL (when there is at least one data byte in reception buffer). Software should clear it after reading all the buffers. This bit is also reset by hardware in Initialization mode. This flag could be checked by software in case of a response timeout event.
24 RXBUSY	Receiver Busy flag 0 Receiver is idle 1 Reception ongoing In Slave mode after header reception, if DIR bit is reset and reception starts, then this bit is set. In this case user cannot set the DTRQ bit.

Table 1664. LINSR field descriptions (continued)

Field	Description
25 RDI	LIN Receive signal This bit reflects the current status of the Rx pin. **Note:** After reset is released, RDI reflects the actual value of Rx pin.
26 WUF	Wakeup flag This bit is set by hardware when a falling edge is detected on the Rx pin and when: The slave is in Sleep mode The master is in Sleep mode or Idle mode It can be cleared by software. It gets reset by hardware in Initialization mode.
27 DBFF	Data Buffer full flag This bit is set by hardware when the buffer is full (8 bytes received) and DFL > 7. It should be cleared by software. This bit is also reset by hardware in Initialization mode.
28 DBEF	Data Buffer empty flag This bit is set by hardware when the buffer is empty (8 bytes have been transmitted) and DFL > 7. It should be cleared by software after data buffer is refilled by software. Transmission of next eight bytes will not start unless this bit is reset by software. This bit is also reset by hardware in Initialization mode.
29 DRF	Data Reception Completed flag This bit is set by hardware and indicates that data reception has been completed. This flag should be cleared by software. This bit is also reset by hardware in Initialization mode. **Note:** In case framing error or checksum error occurs then this flag is not set.
30 DTF	Data Transmission Completed flag This bit is set by hardware and indicates that data transmission is completed. This flag should be cleared by software. This bit is also reset by hardware in Initialization mode. **Note:** For LIN mode, in case a bit error occurs (and IOBE is reset) then this flag is not set.
31 HRF	Header Received flag This bit is set when the header reception is completed. It should be cleared by software. This bit is set only when: All filters are inactive and Bypass filter (BF) bit is set No match in any filter and Bypass filter (BF) bit is set TX filter match At end of frame or frame aborted, if HRF is still set, it gets reset by hardware. This bit is also reset by hardware in Initialization mode. **Note:** If generic slave = 0, then this bit will always read a 0, and cannot be programmed.

63.4.2.4 LIN Error Status Register (LINESR)

Address: 0x000C Access: User read/write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	SZF	OCF	BEF	CEF	SFEF	SDEF	IDPEF	FEF	BOF	0	0	0	0	0	0	NF
W	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c							w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1695. LIN error status register (LINESR)

Refer to *Section 63.3.2.5: Errors* for detailed description of all errors.

Table 1665. LINESR field descriptions

Field	Description
16 SZF	Stuck at Zero flag This bit is set when there is a stuck-at-zero timeout error. It should be reset by software.
17 OCF	Output Compare Flag 0 No output compare event occurred 1 The content of the counter has matched the content of OCR1 or OCR2 This bit should be cleared by software. If this bit is set, MODE bit in LINTCSR is cleared, and the IOT bit of LINTCSR is set, then LINFlexD moves to Idle state. If the MODE bit in LINTCSR is clear, then OCF gets reset by hardware in Initialization mode; if the MODE bit is set, then OCF maintains its status irrespective of the LIN state.
18 BEF	Bit Error flag This bit is set by hardware when there is a bit error. It should be cleared by software. Reset by hardware in Initialization mode.
19 CEF	Checksum Error flag This bit is set by hardware if the received checksum does not match the hardware-calculated checksum. It should be cleared by software. This error will never occur if CCD or CFD bit of LINCR1 is set. Reset by hardware in Initialization mode.
20 SFEF	Sync Field Error flag This bit is set by hardware when the received Sync Field is inconsistent. It should be cleared by software. Reset by hardware in Initialization mode. **Note:** If generic slave = 0, then this bit will always read a 0, and cannot be programmed.

Table 1665. LINESR field descriptions (continued)

Field	Description
21 SDEF	Sync Delimiter Error flag This bit is set by hardware when the delimiter is too short (in other words, less than one bit time). It should be cleared by software. Reset by hardware in Initialization mode. **Note:** If generic slave = 0, then this bit will always read a 0, and cannot be programmed.
22 IDPEF	ID Parity Error flag This bit is set by hardware when there is an error in the ID parity. It should be cleared by software. **Note:** Header Error Interrupt is triggered for SFEF or SDEF or IDPEF flag is set and HEIE is set. If generic slave = 0, then this bit will always read a 0, and cannot be programmed. For generic slave = 1,this bit is reset by hardware in Initialization mode.
23 FEF	Framing Error flag This bit is set by hardware when there is a framing error (invalid stop bit). It should be cleared by software. Reset by hardware in initialization mode.
24 BOF	Buffer overrun flag This bit is set by hardware when there is a new byte received and RMB bit is not cleared. It can be cleared by software. Reset by hardware in initialization mode.
31 NF	Noise flag This bit is set by hardware when noise is detected in the received character. It should be cleared by software. Reset by hardware in Initialization mode.

63.4.2.5 UART Mode Control Register (UARTCR)

Address: 0x0010 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	MIS(1)	CSP(1)			OSR(1)				ROSE(1)	NEF(1)(2)			DTU_PCETX(1)	SBUR(1)		WLS(1)
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	TDFL_TFC(2)			RDFL_RFC(3)			RFBM(1)	TFBM(1)	WL1(1)	PC1(1)	RXEN	TXEN	PC0(1)	PCE(1)	WL0(1)	UART(1)
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. This field can be read in any modes and written only in *Initialization* mode.
2. When TFBM = 0, the field is read/write; when TFBM = 1, the field is read-only.
3. When RFBM = 0, the field is read/write; when RFBM = 1, the field is read-only.

Figure 1696. UART Mode Control Register (UARTCR)

Table 1666. UARTCR field descriptions

Field	Description
0 MIS	Monitor Idle State 1 UARTCTO monitors the idle state of the reception line. 0 UARTCTO monitors the number of bits to be received. The default value of this bit is 0.
1:3 CSP	Configurable Sample Point (i) These bits will decide the sample point during reduced over sampling (ROSE=1). CSP can take the following range of values for a certain over sampling rate: – OSR 4: CSP 2, 3 – OSR 5: CSP 2, 3, 4 – OSR 6: CSP 3, 4, 5 – OSR 8: CSP NA For any values of CSP other than the ones mentioned in the above table, data sampled is erroneous.
4:7 OSR	Over Sampling Rate These bits are programmable by the user to configure the number of samples taken for a bit when reduced over sampling is enabled (ROSE=1). For proper functionality, OSR can only take values 4, 5, 6, and 8. For any other values start bit of data is not detected by receiver and thus the reception will never start.
8 ROSE	Reduced Over Sampling Enable 0 Each bit is over sampled sixteen times. 1 OSR bits decide the over sampling rate. The default value of this bit is 0.
9:11 NEF	Number of expected frame These bits are used to configure the number of expected frames in UART reception mode. If the DTU bit is set, then the UART timeout counter will be reset after the configured number of frames have been received.
12 DTU_PCETX	Disable Timeout in UART mode 0 Timeout has to be handled by software 1 Timeout in UART mode is disabled after the configured number of data frames are received This bit can be programmed in initialization mode only when the UART bit is set. There is an internal counter that gives the number of frames received (num_of_frames). When num_of_frames equals NEF, Disable Timeout is set and num_of_frames gets reset to zero, restarting the timer and enabling the timeout again.
13:14 SBUR	Stop bits in UART reception mode When the UART is used for transmission and reception we have to set the same number of stop bits in GCR and SBUR. When the UART is used only as receiver, it is enough to set SBUR bits only. 00 1 stop bit 01 2 stop bits 10 3 stop bits 11 Reserved This bit can be programmed in the initialization mode only, when the UART bit is set.

Table 1666. UARTCR field descriptions (continued)

Field	Description
15 WLS	Special Word Length in UART mode 0 This bit is disabled. 1 This bit enables 12-bit + parity bit in reception (UART mode) for MSC (Micro Second Channel upstream frame) support. This bit can be programmed in initialization mode only when the UART bit is set. Parity check is enabled by default in this mode and PC0/1 bits can be used to select the parity control. This bit is given the highest priority in UART reception mode.
16:18 TDFL_TFC	Transmitter Data Field Length/TX FIFO Counter TDFL_TFC defines the number of bytes to be transmitted in UART buffer mode (TFBM = 0). TDFL_TFC is a read/write field. TDFL_TFC[2] is reserved and not implemented. When the UART data length is configured as a halfword (WL = 10 or WL = 11), only the configurations TDFL_TFC = x01 or TDFL_TFC = x11 are allowed. x00 1-byte x01 2-bytes x10 3-bytes x11 4-bytes TDFL_TFC contains the number of entries (bytes) of the Tx FIFO in UART FIFO mode (TFBM = 1). TDFL_TFCx is a read-only field available for debug purposes. 000 empty 001 1-byte 010 2-bytes 011 3-bytes 100 4-bytes Others: Reserved **Note:** TDFL_TFC can be programmed and are significative only when the UART bit is set.
19:21 RDFL_RFC	Reception Data Field Length/RX FIFO Counter RDFL_RFC defines the number of bytes to be received in UART buffer mode (RFBM = 0). RDFL_RFC is a read/write field. Bit 19 is reserved and not implemented. When the UART data length is configured as a halfword (WL = 10 or WL = 11), only the configurations RDFL_RFC = x01 or RDFL_RFC = x11 are allowed. x00 1-byte x01 2-bytes x10 3-bytes x11 4-bytes RDFL_RFC contains the number of entries (bytes) of the Rx FIFO in UART FIFO mode (RFBM = 1). RDFL_RFCx is a read-only field available for debug purposes. 000 Empty 001 1-byte 010 2-bytes/1.5 byte in case of WLS bit setting 011 3-bytes 100 4-bytes/2×1.5 byte in case of WLS bit setting Others: Reserved RDFL_RFC can be programmed and are significative only when the UART bit is set. **Note:** In buffer mode, RDFL_RFC should be programmed to be greater than or equal to NEF (number of expected frames). In FIFO mode, there is no such constraint.

Table 1666. UARTCR field descriptions (continued)

Field	Description
22 RFBM	RFBM Rx FIFO/Buffer mode 0 Rx Buffer mode enabled 1 Rx FIFO mode enabled (mandatory in DMA Rx mode) This bit can be programmed in Initialization mode, only when the UART bit is set.
23 TFBM	Tx FIFO/Buffer mode 0 Tx Buffer mode enabled 1 Tx FIFO mode enabled (mandatory in DMA Tx mode) This bit can be programmed in initialization mode, only when the UART bit is set.
24 WL1	Word Length in UART mode 00 7 bits data + parity 01 8 bits data when PCE = 0 or 8 bits data + parity when PCE = 1 10 15 bits data + parity 11 16 bits data when PCE = 0 or 16 bits data + parity when PCE = 1 This bit can be programmed in Initialization mode only, when the UART bit is set. **Note:** The two-digit binary values are a combination of the fields “WL1” and “WL0” to describe the functionality of word length in UART mode.
25 PC1	Parity Control 00 Parity sent is Even 01 Parity sent is Odd 10 A logical 0 is always transmitted/checked as parity bit 11 A logical 1 is always transmitted/checked as parity bit This bit can be programmed in Initialization mode, only when UART bit is set. **Note:** The two-digit binary values are a combination of the fields “PC1” and “PC0” to describe the functionality of word length in UART mode.
26 RXEN	Receiver Enable 0 Receiver disabled 1 Receiver enabled This bit can be programmed only when the UART bit is set.
27 TXEN	Transmitter Enable 0 Transmitter disabled 1 Transmitter enabled, transmission starts only when this bit is set and Data byte 0 (DATA0) is programmed This bit can be programmed only when UART bit is set.
28 PC0	Parity Control 00 Parity sent is Even 01 Parity sent is Odd 10 A logical 0 is always transmitted/checked as parity bit 11 A logical 1 is always transmitted/checked as parity bit This bit can be programmed in Initialization mode, only when UART bit is set. **Note:** The two-digit binary values are a combination of the fields “PC1” and “PC0” to describe the functionality of word length in UART mode.
29 PCE	Parity Control Enable 0 Parity transmission, reception, and checking disabled 1 Parity transmission, reception, and checking enabled This bit can be programmed in Initialization mode only, when the UART bit is set.

Table 1666. UARTCR field descriptions (continued)

Field	Description
30 WL0	Word Length in UART mode 00 7 bits data + parity 01 8 bits data when PCE = 0 or 8 bits data + parity when PCE = 1 10 15 bits data + parity 11 16 bits data when PCE = 0 or 16 bits data + parity when PCE = 1 This bit can be programmed in Initialization mode only, when UART bit is set. **Note:** The two-digit binary values are a combination of the fields “WL1” and “WL0” to describe the functionality of word length in UART mode.
31 UART	UART Mode 0 LIN mode 1 UART mode This bit can be programmed in Initialization mode only.

63.4.2.6 UART Mode Status Register (UARTSR)

Address: 0x0014 Access: User read/write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	SZF	OCF	PE				RMB	FEF	BOF	RDI	WUF	RFNE	TO	DRF_RFE(1)	DTF_TFF(2)	NF
W	w1c	w1c	w1c				w1c	w1c	w1c	w1c	w1c		w1c			w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. When RFBM = 0, the field can be read/cleared by software. Writing 1 clears contents; when RFBM = 1, the field is read-only.
2. When TFBM = 0, the field can be read/cleared by software. Writing 1 clears contents; when TFBM = 1, the field is read-only.

Figure 1697. UART Mode Status Register (UARTSR)

Table 1667. UARTSR field descriptions

Field	Description
16 SZF	Stuck at Zero flag This bit is set by hardware when 100 dominant bits are detected. It should be cleared by software. An interrupt will be generated if the SZIE bit in LINIER is set. This bit will reflect the same value as in LINESR, when in Initialization mode and the UART bit is set.
17 OCF	Output Compare Flag 0 No output compare event occurred 1 The content of the counter has matched the content of LINOCR An interrupt will be generated if the OCIE bit in LINIER is set. This bit will reflect the same value as in LINESR, when in Initialization mode and the UART bit is set. For Baud rate above 1 Mbit/s, this flag is not usable.
18:21 PE	Parity Error flag These bits indicate if there is a Parity Error in the corresponding byte. No interrupt is generated if this error occurs. PE[i] (where i = 0,1,2 or 3) can be equal to: 0 No parity error 1 Parity error in the corresponding received byte This bit will reflect the same value as in LINESR, when in initialization mode and UART bit set. **Note:** Parity is checked only after a complete frame is received. When WL bits = 10 or 11, either PE[0] or PE[2] is only set. – When PE[0] is set, it indicates Parity error in either first or second byte received. – When PE[2] is set, it indicates Parity error in either third or fourth byte received.
22 RMB	Release Message Buffer 0 Buffer data is free 1 Buffer data ready to be read by software This bit should be cleared by software. This bit will reflect the same value as in LINESR, when in Initialization mode and the UART bit is set.
23 FEF	Framing Error flag This bit is set by hardware when there is a framing error (invalid stop bit). It should be cleared by software. It will generate an interrupt if the FEIE bit of LINIER is set. This bit will reflect the same value as in LINESR, when in Initialization mode and the UART bit is set.
24 BOF	FIFO/Buffer overrun flag This bit is set by hardware when there is a new byte received and the RMB bit is not cleared in UART buffer mode. In UART FIFO mode this bit is set when there is a new byte and the Rx FIFO is full. In UART FIFO mode, once Rx FIFO is full, the new received message will be discarded irrespective of the value of the RBLM bit. In UART Rx Buffer mode, if RBLM is set then the new message received will be discarded; if RBLM is reset then the new message will overwrite the buffer. It can be cleared by software writing a 1. An interrupt is generated if the BOIE bit of LINIER is set. This bit will reflect the same value as in LINESR, when in Initialization mode and the UART bit is set.

Table 1667. UARTSR field descriptions (continued)

Field	Description
25 RDI	Receiver Data Input signal This bit reflects the current status of the RX pin.
26 WUF	Wakeup flag This bit is set by hardware when a falling edge is detected on the RX pin in sleep mode. It should be cleared by software. An interrupt will be generated if the WUIE bit in LINIER is set. This bit will reflect the same value as in LINESR when in Initialization mode and the UART bit is set.
27 RFNE	Receive FIFO Not Empty RFNE bit is set by hardware in UART FIFO mode (RFBM = 1), when there is at least one data byte present in the receive FIFO. RFNE is a read-only bit for debugging purposes. This flag can be used by software in case of a timeout event.
28 TO	Timeout This bit is set by hardware when a UART timeout occurs — in other words, the value of UARTCTO becomes equal to the preset value of the timeout (UARTPTO register setting). TO should be cleared by software.The SR bit should be used to reset the receiver fsm to Idle state in case of UART TIMEOUT for UART reception, depending on the application, in both buffer and FIFO mode. An interrupt will be generated when LINIER.DBEIE/TOIE bit is set on the Error interrupt line in UART mode.
29 DRF_RFE	Data Reception Completed Flag/Rx FIFO Empty Flag DRF_RFE is set by hardware in UART buffer mode (RFBM = 0) and indicates that the number of bytes programmed in RDFL have been received. DRF_RFE should be cleared by software. An interrupt will be generated if the DRIE bit in LINIER is set. DRF_RFE bit is set only when the configured number of stop bits are received for the last frame (number of frames is configurable by RDFL bits). DRF_RFE is set irrespective of framing error, parity error or overrun error. This bit will reflect the same value as in LINESR when in Initialization mode and the UART bit is set. DRF_RFE is set by hardware in UART FIFO mode (RFBM = 1) when the RX FIFO is empty. DRF_RFE is a read-only bit for debugging purposes. It is internally used by the DMA RX interface.
30 DTF_TFF	Data Transmission Completed Flag/TX FIFO Full Flag DTF_TFF is set by hardware in UART buffer mode (TFBM = 0) and indicates that data transmission is completed. DTF_TFF should be cleared by software. An interrupt will be generated if the DTIE bit in LINIER is set. This bit will reflect the same value as in LINESR when in Initialization mode and the UART bit is set. DTF_TFF is set by hardware in UART FIFO mode (TFBM = 1) when TX FIFO is full. DTF_TFF is a read-only bit for debugging purposes. It is internally used by the DMA TX interface.
31 NF	Noise flag This bit is set by hardware when noise is detected in the received character. It should be cleared by software. This bit will reflect the same value as in LINESR when in Initialization mode and the UART bit is set. During reduced oversampling (ROSE bit = 1), it is enabled only when OSR = 8.

63.4.2.7 LIN Time-Out Control Status Register (LINTCSR)

This register contains control and status bits for the timeout feature.

Address: 0x0018 Access: User read/write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	MODE(1)	IOT(1)	TOCE	CNT							
W																
Reset	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0

1. This field can be read in any modes and written only in *Initialization* mode.

Figure 1698. LIN Time-Out Control Status Register (LINTCSR)

Table 1668. LINTCSR field descriptions

Field	Description
21 MODE	Time-out counter mode 0 LIN mode 1 Output compare mode This bit can be configured only during initialization.
22 IOT	Idle on timeout 0 LIN state machine does not reset to Idle on timeout 1 LIN state machine resets to Idle on timeout event This feature is applicable only when MODE bit in LINTCSR is cleared.
23 TOCE	Time-out counter enable 0 Time-out counter disable. OCF flag is not set on an output compare event. 1 Time-out counter enable. OCF flag is set if an output compare event occurs. TOCE is always configurable by software in Initialization mode. If LIN state is other than INIT and if timer is configured in LIN mode, then hardware takes control of TOCE.
24:31 CNT	Counter Value These bits reflect the value of a free-running counter used for timeout. **Note:** For proper functionality of this counter, LINIBRR should be greater than or equal to 5. This counter increments once in every bit duration, as per the programmed bit rate. The counter starts as soon as the IP comes out of reset and increments based on the slowest baud rate possible (LINIBRR = FFFFFh), until the user programs LINIBRR and LINFBRR with the desired values.

63.4.2.8 LIN Output Compare Register (LINOCR)

This register contains the value to be compared to the count value in LINTOCR register. This register is writable by software only in Output Compare Mode.

Address: 0x001C Access: User read/write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	OC2								OC1							
W																
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Figure 1699. LIN Output Compare Register (LINOCR)

Table 1669. LINOCR field descriptions

Field	Description
16:23 OC2	Output compare value 2 When MODE bit = 0 in LINTCSR register: This field is loaded by HW, with response timeout calculated based on RTO value in LINTOCR register and CNT value in LINTCSR register. When MODE bit = 1 in LINTCSR register: This field can be used as output compare register, upto bit rates of 1 Mbit/s.
24:31 OC1	Output compare value 1 When MODE bit = 0 in LINTCSR register: This field is loaded by HW, with header timeout calculated based on HTO value in LINTOCR register and CNT value in LINTCSR register. When MODE bit = 1 in LINTCSR register: This field can be used as output compare register, upto bit rates of 1 Mbit/s. **Note:** OC1 field can be used as output compare register only in LIN slave mode.

63.4.2.9 LIN Time-Out Control Register (LINTOCR)

Address: 0x0020 Access: User read/write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	RTO				0	HTO						
W																
Reset	0	0	0	0	1	1	1	0	0	0	0(1)/1	1(1)/0	1	1	0	0

1. This reset value is valid only in Master mode (slave = 0).

Figure 1700. LIN Time-Out Control Register (LINTOCR)

Table 1670. LINTOCR field descriptions

Field	Description
20:23 RTO	Response timeout value This is the response timeout duration (in bit time) for 1 byte. The reset value is 0xE = 14, corresponding to $T_{Response_Maximum} = 1.4 * T_{Response_Nominal}$
25:31 HTO	Header timeout value This register contains the header timeout duration (in bit time). This register can be written only for Slave mode. The reset value is 0x2C = 45, corresponding to $T_{Header_Maximum}$ in the LIN specification. If Master mode is enabled then these bits will always reflect 0x1C = 28 (1.4 × 10 bits of sync + 1.4 × 10 bits of ID). For slave, these bits should be programmed without considering 11 bits of break.

63.4.2.10 LIN Fractional Baud Rate Register (LINFBRR)

This register consists of bits that decide the fractional part of the LIN Baud Rate. It can be programmed only in Initialization mode.

Note: When LASE bit is set, this register should be read only after AUTOSYNC_COMP bit in LINSR register is set to obtain the correct value.

This register cannot be used when reduced oversampling is enabled (ROSE bit = 1)

Address: 0x0024 Access: User read/write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	FBR(1)			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. This field can be read in any modes and written only in *Initialization* mode.

Figure 1701. LIN Fractional Baud Rate Register (LINFBRR)

Table 1671. LINFBRR field descriptions

Field	Description
28:31 FBR	Fractional Baud rates 0x0 Fraction (LDIV): 0 0x1 Fraction (LDIV): 1/16 0x2 Fraction (LDIV): 2/16 0x3 Fraction (LDIV): 3/16 0x4 Fraction (LDIV): 4/16 0x5 Fraction (LDIV): 5/16 0x6 Fraction (LDIV): 6/16 0x7 Fraction (LDIV): 7/16 0x8 Fraction (LDIV): 8/16 0x9 Fraction (LDIV): 9/16 0xA Fraction (LDIV): 10/16 0xB Fraction (LDIV): 11/16 0xC Fraction (LDIV): 12/16 0xD Fraction (LDIV): 13/16 0xE Fraction (LDIV): 14/16 0xF Fraction (LDIV): 15/16

63.4.2.11 LIN Integer Baud Rate Register (LINIBRR)

This register consists of control bits that decide the baud rate along with the LINFBRR. It can be programmed only in Initialization mode.

Note: When LASE bit is set, this register should be read only after AUTOSYNC_COMP bit in LINSR register is set to obtain the correct value.

Address: 0x0028 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	IBR(1)			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	IBR(1)															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. This field can be read in any modes and written only in *Initialization* mode.

Figure 1702. LIN Integer Baud Rate Register (LINIBRR)

Table 1672. LINIBRR field descriptions

Field	Description
12:31 IBR	Integer Baud rates These bits along with the fractional baud rate bits decide the LIN baud rate. 0x00000 Mantissa (LDIV): LIN clock disabled 0x00001 Mantissa (LDIV): 1 ... 0xFFFFE Mantissa (LDIV): 1048574 0xFFFFF Mantissa (LDIV): 1048575

63.4.2.12 LIN Checksum Field Register (LINCFR)

*Note: There is a delay between 4 to 6 clock cycles of PBRIDGEx_CLK for the internal checksum's value (which is clocked with LIN_CLK/16 * LDIV) to reflect on LINCFR.*

This register consists of checksum bits.

Address: 0x002C Access: User read/write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	CF							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1703. LIN Checksum Field Register (LINCFR)

Table 1673. LINCFR field descriptions

<table>
<tr><th>Field</th><th>Description</th></tr>
<tr><td>24:31
CF</td><td>Checksum bits
When the LINCR1[CCD] bit is reset these bits are read-only and are calculated by hardware. When the LINCR1[CCD] bit is set, these bits can be written by software.
<table>
<tr><th>LINCR1[CFD]</th><th>LINCR1[CCD]</th><th>LINCHKSUM Read/write</th><th>Checksum sent</th></tr>
<tr><td>1</td><td>1</td><td>read/write</td><td>None</td></tr>
<tr><td>1</td><td>0</td><td>read-only</td><td>None</td></tr>
<tr><td>0</td><td>1</td><td>read/write</td><td>Programmed checksum</td></tr>
<tr><td>0</td><td>0</td><td>read-only</td><td>Calculated checksum</td></tr>
</table>
</td></tr>
</table>

63.4.2.13 LIN Control Register 2 (LINCR2)

This register includes control status bits related to buffer operations.

Address: 0x0030 Access: User read/write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	TBDE(1)	IOBE(1)	IOPE(1)	WURQ(2)	DDRQ(2)	DTRQ(2)	ABRQ(2)	HTRQ(2)	0	0	0	0	0	0	0	0
W																
Reset	0	1	0(3)/1	0	0	0	0	0	0	0	0	0	0	0	0	0

1. This field can be read in any modes and written only in Initialization mode.
2. This field can be read/set by software.
3. When slave = 0, reset value is '0' and the this field cannot be programmed.

Figure 1704. LIN Control Register 2 (LINCR2)

Table 1674. LINCR2 field descriptions

Field	Description
16 TBDE	Two Bit delimiter bit 0 Delimiter length in break field is 1 bit 1 Delimiter length in break field is 2 bits This bit can be set in Initialization mode only.
17 IOBE	Idle on Bit Error 0 Bit Error does not reset LIN state machine 1 Bit Error resets LIN state machine This bit can be set in Initialization mode only.
18 IOPE	Idle on Identifier Parity Error 0 Parity Error does not reset LIN state machine 1 Parity Error resets LIN state machine This bit can be set in Initialization mode only. **Note:** If generic slave = 0, then this bit will always read a 0, and cannot be programmed.
19 WURQ	Wakeup Generate Request Setting this bit will generate a wakeup pulse. It is reset by hardware when the wakeup character has been transmitted. The character sent during wakeup is copied from BDRL (DATA0). Note that this bit cannot be set in Sleep mode — software has to exit Sleep mode before setting this bit. Bit Error is not checked when transmitting the wakeup request.
20 DDRQ	Data Discard request Set by software to stop data reception if the frame does not concern the node. This bit is reset by hardware once LINFlexD ignores the response and moves to Idle state. For LIN slave this bit can be set only when HRF bit is set and Identifier is software-filtered.
21 DTRQ	Data Transmission Request Set by software in slave mode to request the transmission of the LIN Data field stored in the Buffer data register. This bit can be set only when the HRF bit is set (to ensure that data transmission is requested only after a header reception). Cleared by hardware when the request has been completed, or on abort request or error condition. In Master mode, this bit is set by hardware when the DIR bit is set and header transmission is complete.
22 ABRQ	Abort Request Set by software to abort the current transmission. Cleared by hardware when the transmission has been aborted. LINFlexD aborts the transmission at the end of the current bit. This bit can abort a wakeup request also and can be used in UART mode also.
23 HTRQ	Header Transmission Request Set by software to request the transmission of the LIN Header. Cleared by hardware when the request has been completed or on abort request. This bit has no effect in UART mode. **Note:** In master mode, if both HTRQ and ABRQ are set at the same time then ABRQ has no effect. Similarly, in slave mode after header reception, if DTRQ and ABRQ are simultaneously set then ABRQ has no effect.

63.4.2.14 Buffer Identifier Register (BIDR)

This register contains the bits which provide information about the identifier of the transaction and other related information.

Note: All the fields (ID, CSS, DIR, DFL) of the BIDR register must be updated when an ID filter (enabled) in Slave mode (Tx or Rx) matches the ID received.

Address: 0x0034 Access: User read/write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	DFL						DIR	CCS	0	0	ID					
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1705. Buffer Identifier Register (BIDR)

Table 1675. BIDR field descriptions

Field	Description
16:21 DFL	Data Field Length Number of data bytes in the response part of the frame. DFL[5:0] = Number of data bytes – 1 **Note:** DFL[5:3] is provided for extended frames. Normally LIN mode will use DFL[2:0]. Identifier filters are now compatible with DFL[5:0] also.
22 DIR	Direction This bit controls the direction of the data field. 0 LINFlexD receives the data and copies them in the BDRs 1 LINFlexD transmits the data from the BDRs
23 CCS	Classic Checksum This bit controls the type of checksum applied on the current message. 0 Enhanced Checksum covering Identifier and Data fields. This is compatible with LIN specification rev. 2.0 and higher. 1 Classic Checksum covering Data field only. This is compatible with LIN specification 1.3 and lower.
26:31 ID	Identifier Identifier part of the identifier field without the identifier parity. This field can be written only in Master mode (MME = ‘1’).

63.4.2.15 Buffer Data Register Least Significant (BDRL)

This register is a part of an 8-byte data buffer.

Address: 0x0038 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	DATA3								DATA2							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	DATA1								DATA0							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1706. Buffer Data Register Least Significant (BDRL)

Table 1676. BDRL field descriptions

Field	Description
0:7 DATA3	Data Byte 3 Data byte 3 of the data field
8:15 DATA2	Data Byte 2 Data byte 2 of the data field
16:23 DATA1	Data Byte 1 Data byte 1of the data field
24:31 DATA0	Data Byte 0 Data byte 0 of the data field

63.4.2.16 Buffer Data Register Most Significant (BDRM)

This register is a part of an 8-byte data buffer.

Address: 0x003C Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	DATA7								DATA6							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	DATA5								DATA4							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1707. Buffer Data Register Most Significant (BDRM)

Table 1677. BDRM field descriptions

Field	Description
0:7 DATA7	Data Byte 7 Data byte 7 of the data field
8:15 DATA6	Data Byte 6 Data byte 6 of the data field
16:23 DATA5	Data Byte 5 Data byte 5 of the data field
24:31 DATA4	Data Byte 4 Data byte 4 of the data field

63.4.2.17 Identifier Filter Enable Register (IFER)

This register enables/disables a particular filter.

Address: 0x0040 Access: User read/write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	FACT(1)															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. This field can be read in any modes and written only in *Initialization* mode.

Figure 1708. Identifier Filter Enable Register (IFER)

Table 1678. IFER field descriptions

Field	Description
16:31 FACT	Filter active The software sets the bit FACT[x] to activate the filter x in Identifier list mode. 1. In Identifier mask mode, FACT (2*n*+1) have no effect on the corresponding filters as they act as mask for the Identifier 2*n*. 2. The length of this field depends on the value of no_of_filters. 3. The register diagram depicts the maximum no_of_filters, which can be up to 16. **Note:** Refer to device configuration chapter to see the number of filters used in the device.

63.4.2.18 Identifier Filter Match Index (IFMI)

This register contains the index corresponding to the received ID. It can be used to read or write the data directly in RAM.

Address: 0x0044 Access: User read

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	IFMI				
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1709. Identifier Filter Match Index (IFMI)

Table 1679. IFMI field descriptions

Field	Description
27:31 IFMI	Filter match index Upon a filter match with xth filter – IFMI[4:0] = x+1. On no match IFMI is equal to 00h. This register contains the index corresponding to the received identifier. It can be used to directly write or read the data in SRAM (refer to *Section 63.3.2.4* for more details).

63.4.2.19 Identifier Filter Mode Register (IFMR)

This register configures the modes of filters.

Address: 0x0048 Access: User read/write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	IFM(1)							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. This field can be read in any modes and written only in *Initialization* mode.

Figure 1710. Identifier Filter Mode Register (IFMR)

Table 1680. IFMR field descriptions

Field	Description
24:31 IFM	Filter mode 0 Filters 2*n* and 2*n*+1 are in identifier list mode 1 Filters 2*n* and 2*n*+1 are in mask mode, where filter 2*n*+1 is the mask for filter 2*n*

63.4.2.20 Identifier Filter Control Register (IFCR2*n*)

This register is read-only in normal mode and can be programmed only in Initialization mode. This register acts as a filter in both Identifier list and Identifier mask modes.

Address: 0x4C+ *n**0x8 (*n* = 0 to (no_of_filters[2]/2–1)) Access: User read/write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	DFL[1]						DIR[1]	CCS[1]	0	0	ID[1]					
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. This field can be read in any modes and written only in *Initialization* mode.
2. Refer to the Device Configuration chapter to see the number of filters used in this device.

Figure 1711. Identifier Filter Control Register (IFCR2n)

Table 1681. IFCR2*n* field descriptions

Field	Description
16:21 DFL	Data Field Length Number of data bytes in the response part of the frame. DFL[5:0] = Number of data bytes - 1
22 DIR	Direction This bit controls the direction of the data field. 0 LINFlexD receives data and copies to the BDR registers 1 LINFlexD transmits data from the BDR registers
23 CCS	Classic Checksum This bit controls the type of checksum applied on the current message. 0 Enhanced Checksum covering Identifier and Data fields. This is compatible with LIN specification rev. 2.0 and higher. 1 Classic Checksum covering Data field only. This is compatible with LIN specification 1.3 and lower.
26:31 ID	Identifier Identifier part of the identifier field without the identifier parity.

63.4.2.21 Identifier Filter Control Register (IFCR2*n*+1)

This register is read-only in normal mode and can be programmed only in Initialization mode. This register acts as a filter in Identifier list mode but acts as a mask for IFCR2*n* in Identifier mask mode.

Address: 0x50+ *n**0x8 (*n* = 0 to (no_of_filters(2)/2–1)) Access: User read/write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	DFL(1)						DIR(1)	CCS(1)	0	0	ID(1)					
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. This field can be read in any modes and written only in *Initialization* mode.
2. Refer to the Device Configuration chapter to see the number of filters used in this device.

Figure 1712. Identifier Filter Control Register (IFCR2n+1)

Table 1682. IFCR2*n*+1 field descriptions

Field	Description
16:21 DFL	Data Field Length Number of data bytes in the response part of the frame. DFL[5:0] = Number of data bytes – 1
22 DIR	Direction This bit controls the direction of the data field. 0 LINFlexD receives data and copies to the BDR registers 1 LINFlexD transmits data from the BDR registers
23 CCS	Classic Checksum This bit controls the type of checksum applied on the current message. 0 Enhanced Checksum covering Identifier and Data fields. This is compatible with LIN specification rev. 2.0 and higher. 1 Classic Checksum covering Data field only. This is compatible with LIN specification 1.3 and lower.
26:31 ID	Identifier Identifier part of the identifier field without the identifier parity.

63.4.2.22 Global Control Register (GCR)

This register is read-only in Normal mode and can be programmed only in Initialization mode. This is a global control register — in other words, the register configuration will be applied in LIN mode as well as in UART mode.

Address: 0x8C Access: User read/write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	TDFBM(1)	RDFBM(1)	TDLIS(1)	RDLIS(1)	STOP(1)	SR(2)
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. This field can be read in any modes and written only in *Initialization* mode.
2. This field is read 0 by software.

Figure 1713. Global Control Register (GCR)

Table 1683. GCR field descriptions

Field	Description
26 TDFBM	Transmit data first bit MSB This bit controls the first bit of transmit data (payload only) as MSB/LSB in both UART and LIN modes. 0 The first bit of transmitted data is LSB — in other words, the first bit transmitted is mapped on LSB bit (BDR (0), BDR (8), BDR (16), BDR (24)) 1 The first bit of transmitted data is MSB — in other words, the first bit transmitted is mapped on MSB bit (BDR (7), BDR (15), BDR (23), BDR (31))
27 RDFBM	Received data first bit MSB This bit controls the first bit of received data (payload only) as MSB/LSB both in UART and LIN modes. 0 The first bit of received data is LSB – in other words, the first bit received is mapped on LSB bit (BDR (0), BDR (8), BDR (16), BDR (24)) 1 The first bit of received data is MSB – in other words, the first bit received is mapped on MSB bit (BDR (7), BDR (15), BDR (23), BDR (31))
28 TDLIS	Transmit data level inversion selection This bit controls the data inversion of transmitted data (payload only) in both UART and LIN modes. 0 Transmitted data is not inverted 1 Transmitted data is inverted
29 RDLIS	Received data level inversion selection This bit controls the data inversion of received data (payload only) in both UART and LIN modes. 0 Received data is not inverted 1 Received data is inverted

Table 1683. GCR field descriptions (continued)

Field	Description
30 STOP	1/2 stop bit configuration This bit controls the number of stop bit transmitted data in both UART and LIN modes. The stop bit is configured for all the fields (Delimiter, Sync, ID, Checksum, Payload). 0 1 stop bit 1 2 stop bits
31 SR	Soft reset SR executes a soft reset of the LINFlexD controller (FSMs, FIFO pointers, counters, timers, status and error registers) without modifying the configuration registers when a 1 write operation is performed. This bit should be cleared by software to perform further operations (this bit is not cleared by hardware). This bit is always read as 0. The following register fields are reset by SR: LINSR: All fields except RXbusy and AUTOSYNC_COMP LINESR: All fields LINTCSR: Only CNT[0:7] UARTSR: All fields except RFE & TFFt UARTCR: Only TFC & RFC UARTCTO: All fields

63.4.2.23 UART Preset Timeout Register (UARTPTO)

This register contains the preset value of the timeout register in UART mode and is programmed according to the number of bits to be received or to monitor the idle state of the reception line. This register can be written by software any time.

Address: 0x8C + 0x04 Access: User read/write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	PTO											
W																
Reset	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1

Figure 1714. UART Preset Timeout Register (UARTPTO)

Table 1684. UARTPTO field descriptions

Field	Description
20:31 PTO	Preset Timeout PTO defines the preset value of timeout counter. A value of 0, 1 or 2 is forbidden, otherwise the UARTSR[TO] status bit is immediately set. Refer to the UARTCTO register.

63.4.2.24 UART Current Timeout Register (UARTCTO)

This register contains the current timeout value in UART mode, and is used in conjunction with the UARTPTO register (refer to *Section 63.4.2.23*) to monitor the number of bits received by UART or to monitor the idle state of the reception line. UART timeout works in both CPU and DMA modes.

The timeout counter:

- Starts at zero and counts upward
- Is clocked with LIN_CLK / (16 * LDIV) synchronized to PBRIDGEx_CLK (when ROSE = 0)
- Is clocked with LIN_CLK / (OSR * IBRR) synchronized to PBRIDGEx_CLK (when ROSE = 1)
- Is automatically enabled when UARTCR[RXEN] = 1

*Note: Due to this synchronisation, UARTCTO reflects the internal counter's value (which is clocked with LIN_CLK/16 * LDIV when ROSE = 0 or LIN_CLK/OSR * IBRR when ROSE = 1) which was present on it, before 4 to 6 clock cycles of PBRIDGEx_CLK.*

It is reset when:

- Number of frames received is equal to NEF bits
- UARTCTO becomes equal to UARTPTO-1
- Whenever UARTPTO is written
- When DRF is set and DTU bit = 1

Address: 0x8C + 0x08 Access: User read

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	CTO											
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1715. UART Current Timeout Register (UARTCTO)

Table 1685. UARTCTO field descriptions

Field	Description
20:31 CTO	Current Timeout CTO defines the current value of the timeout counter. CTO is a read-only field. CTO is reset every time UARTPTO is re-initialized, or UARTCTO = UARTPTO, or by hard/soft reset. When the CTO value matches the preset value (UARTPTO), the status bit UARTSR[TO] is set.

63.4.2.25 DMA Tx Enable Register (DMATXE)

This register enables the DMA TX interface. This register can be written and read by software any time.

Address: 0x8C + 0x0C Access: User read/write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	DTE															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1716. DMA Tx Enable Register (DMATXE)

Table 1686. DMATXE field descriptions

Field	Description
(32-2^TX_CH_NUM):31 DTE[(2^TX_CH_NUM-1):0]	DMA Tx channel Y enable 0 DMA Tx channel Y disabled 1 DMA Tx channel Y enabled **Note:** The actual size of the register DMATXE depends on the value of the static parameter TX_CH_NUM. When DMATXE = 0x00000000, the DMA TX interface FSM is forced (soft reset) in Idle state. Refer to the device configuration chapter for the value of TX_CH_NUM used in this device.

63.4.2.26 DMA Rx Enable Register (DMARXE)

This register enables the DMA RX interface. This register can be written and read by software any time.

Address: 0x8C + 0x10 Access: User read/write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	DRE															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1717. DMA Rx Enable Register (DMARXE)

Table 1687. DMARXE field descriptions

Field	Description
(32–2^RX_CH_NUM):31 DRE[(2^RX_CH_NUM-1):0]	DMA Rx channel Y enable 0 DMA Rx channel Y disabled 1 DMA Rx channel Y enabled **Note:** The actual size of the register DMARXE depends on the value of the static parameter RX_CH_NUM. When DMARXE = 0x00000000, the DMA RX interface FSM is forced (soft reset) in Idle state. Refer to the device configuration chapter for the value of RX_CH_NUM used in this device.

63.5 Programming considerations

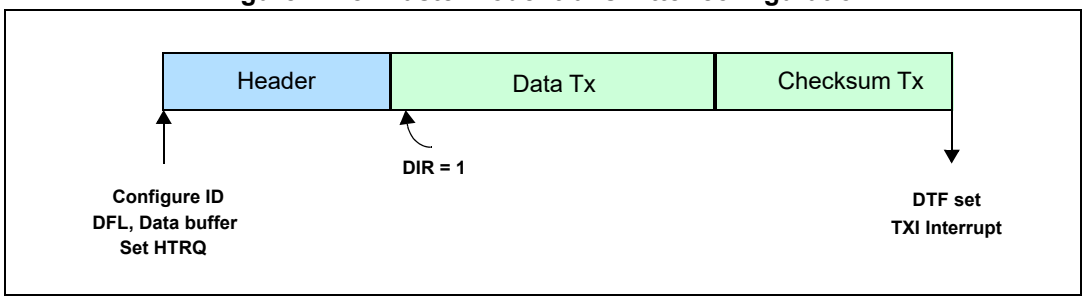
The next subsections describe the various configurations in which the LINFlexD module can be used.

63.5.1 Master node

LINFlexD acts as Master when the MME bit is set.

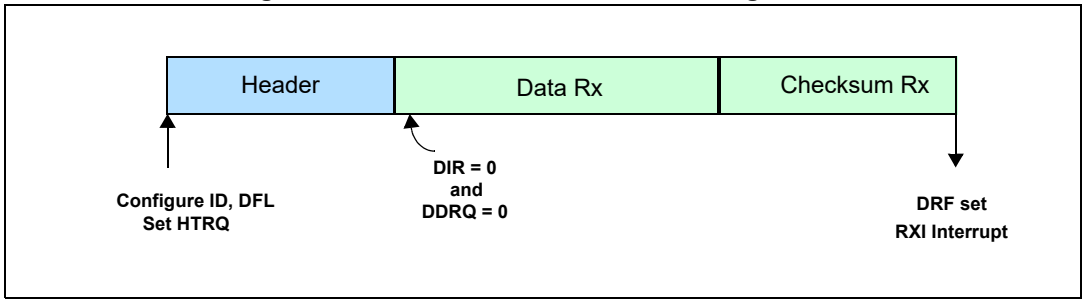
63.5.1.1 Transmitter

Figure 1718. Master node: transmitter configuration



63.5.1.2 Receiver

Figure 1719. Master node: receiver configuration



63.5.1.3 Transmitter, Bit Error

Figure 1720. Master node: transmitter, bit error configuration

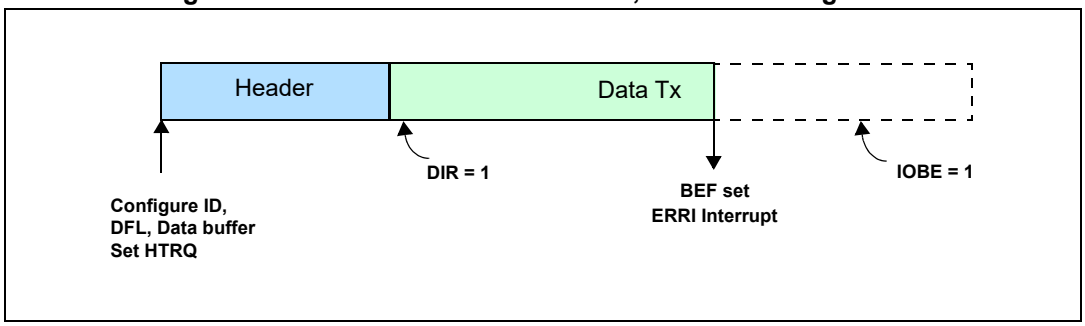
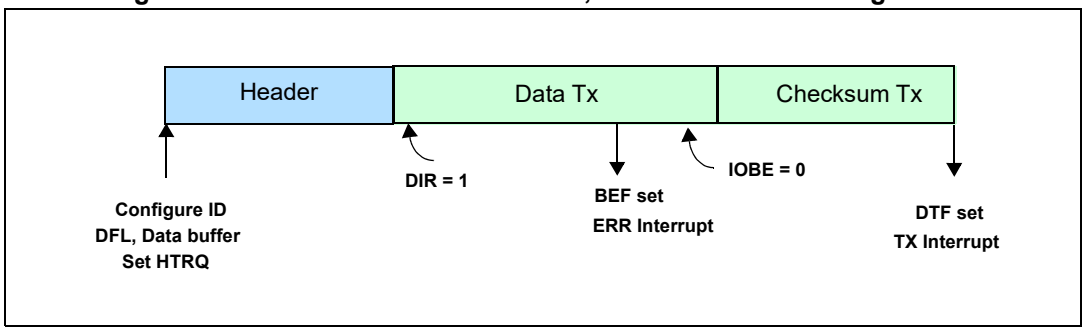
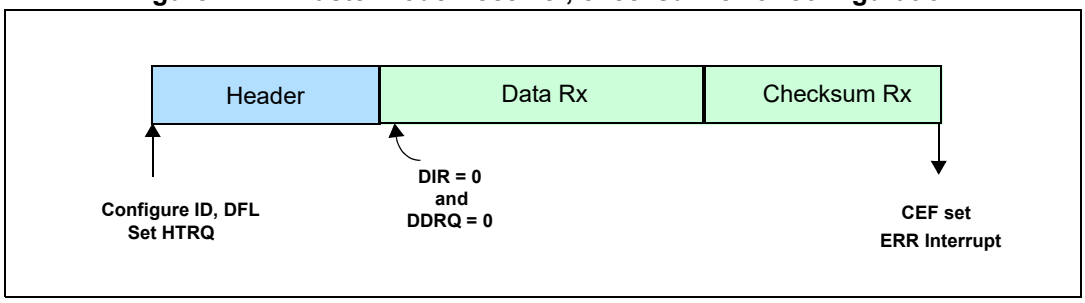


Figure 1721. Master node: transmitter, checksum error configuration



63.5.1.4 Receiver, Checksum Error

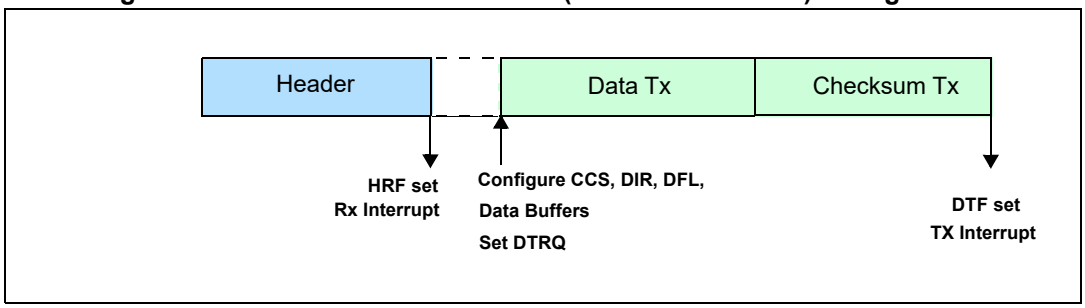
Figure 1722. Master node: receiver, checksum error configuration



63.5.2 Slave node

63.5.2.1 Transmitter (no identifier filters)

Figure 1723. Slave node: transmitter (no identifier filters) configuration



63.5.2.2 Receiver (no identifier filters)

Figure 1724. Slave node: receiver (no identifier filters) configuration

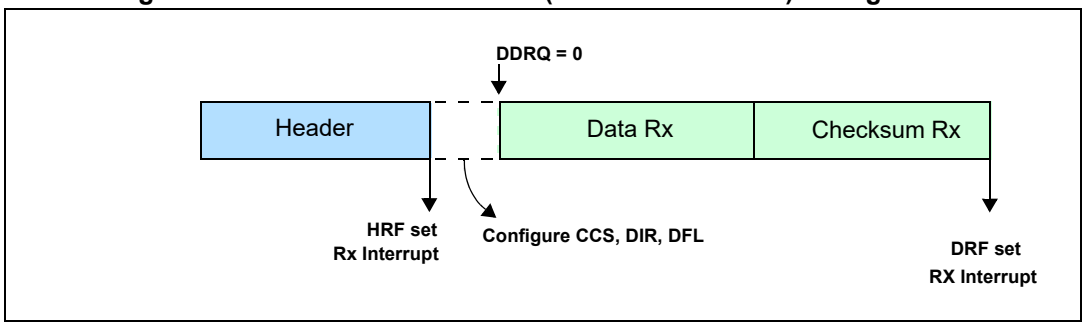
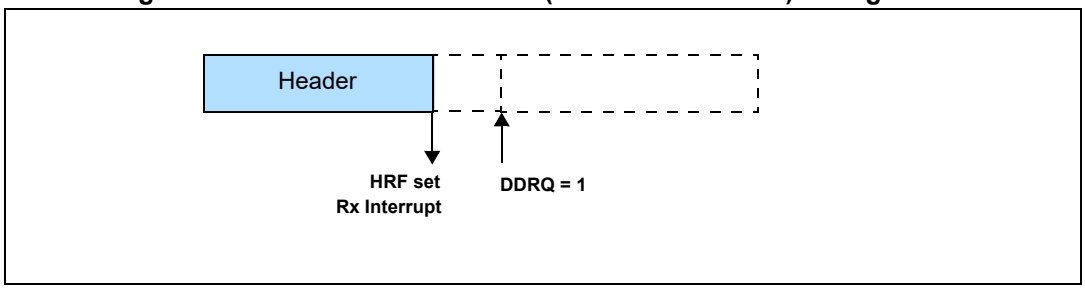
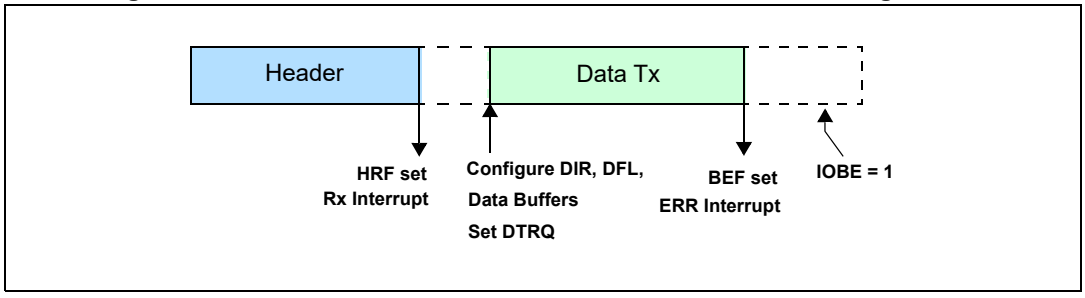


Figure 1725. Slave node: receiver (no identifier filters) configuration



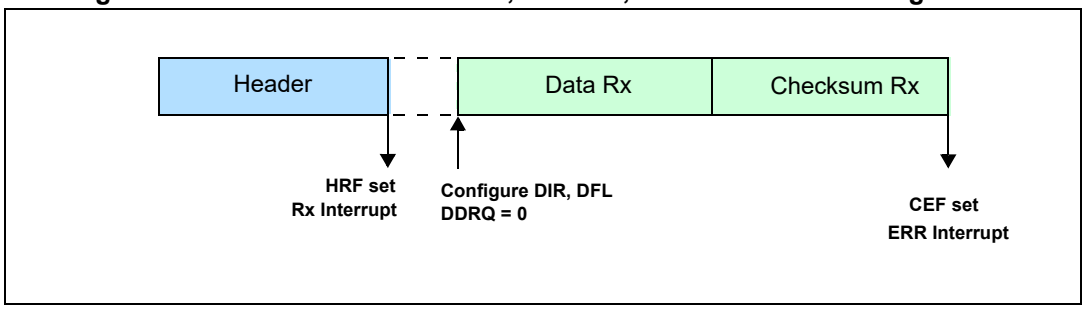
63.5.2.3 No filters, transmitter, bit error

Figure 1726. Slave node: No filters, transmitter, bit error configuration



63.5.2.4 No filters, receiver, checksum error

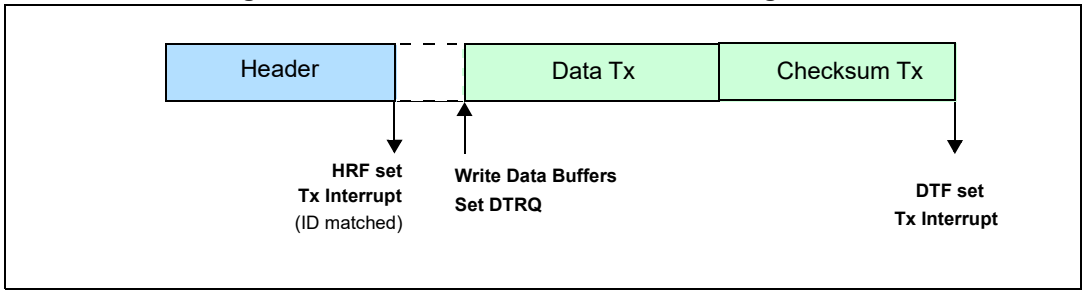
Figure 1727. Slave node: No filters, receiver, checksum error configuration



63.5.2.5 At least one TX filter, BF is reset, ID matches filter

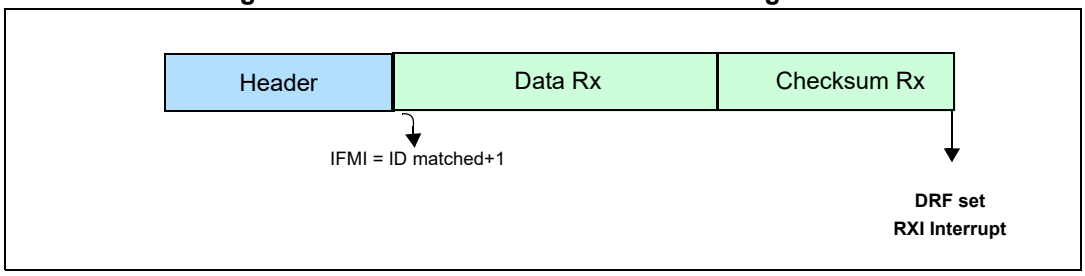
This configuration can be used in case slave never receives data, for example, sensor.

Figure 1728. Slave node: one TX filter configuration



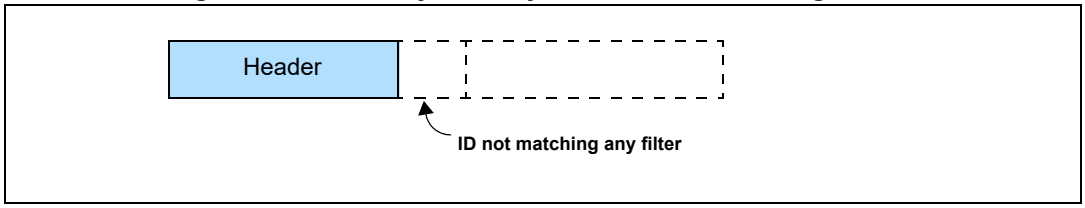
63.5.2.6 At least one RX filter, BF reset, ID matches filter

Figure 1729. Slave node: one RX filter configuration



63.5.2.7 RX only, TX only, RX & TX filters, ID not matching filter, BF reset

Figure 1730. RX only, TX only, RX & TX filters configuration



63.5.2.8 TX filter, BF is set

This configuration is used when:

- All TX IDs are handled by filters.
- There are not enough other filters to handle all reception IDs.

Figure 1731. TX filter, BF is set configuration: ID has matched

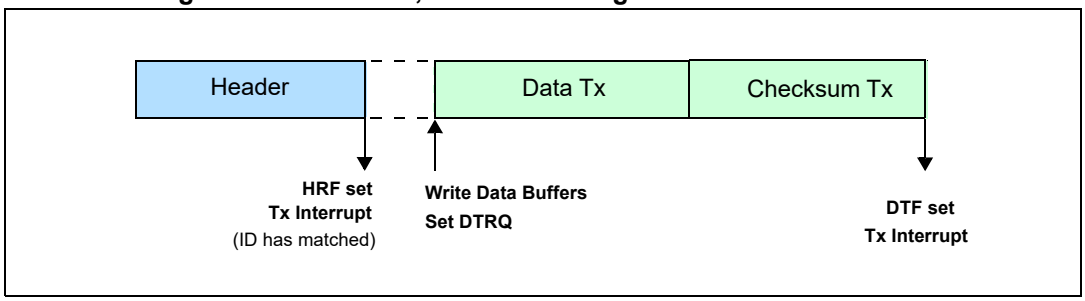
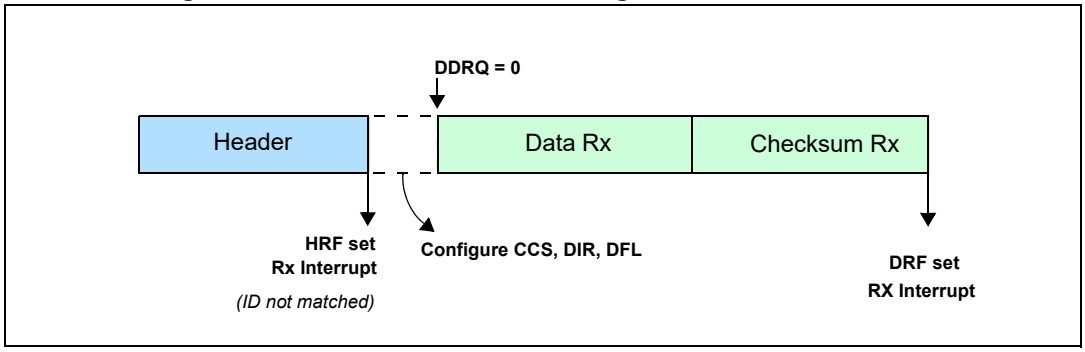


Figure 1732. TX filter, BF is set configuration: ID not matched



63.5.2.9 RX filter, BF is set

Figure 1733. RX filter, BF is set configuration: ID matched

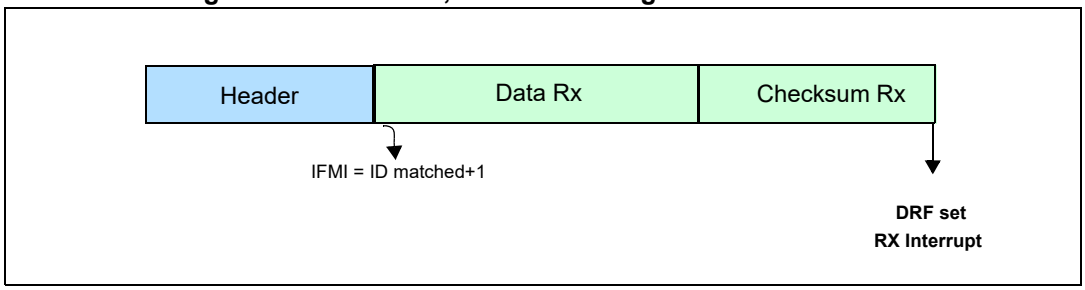


Figure 1734. RX filter, BF is set configuration: ID not matched (ID is Rx)

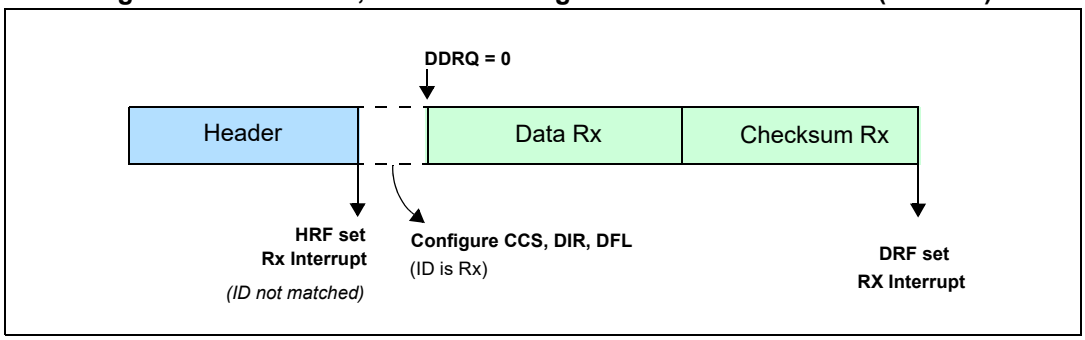
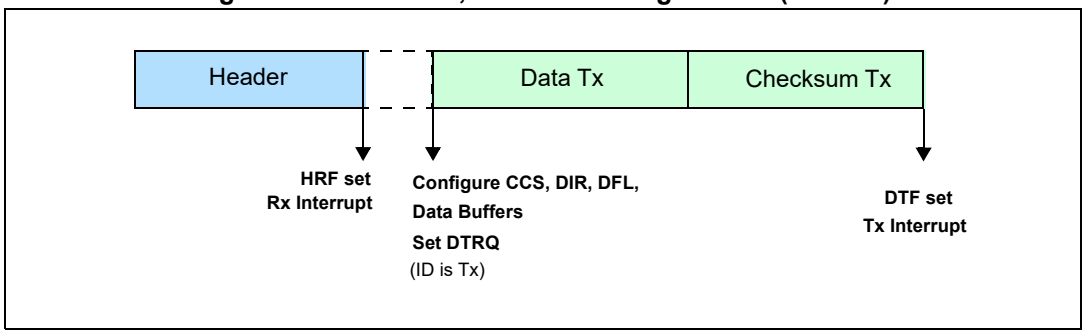


Figure 1735. RX filter, BF is set configuration: (ID is Tx)



63.5.2.10 TX filter, RX filter, BF set

This configuration is used when:

- There are not enough filters.
- The filters are used for most frequently used IDs to reduce CPU usage.

Figure 1736. TX filter, RX filter, BF set configuration: (ID matched)

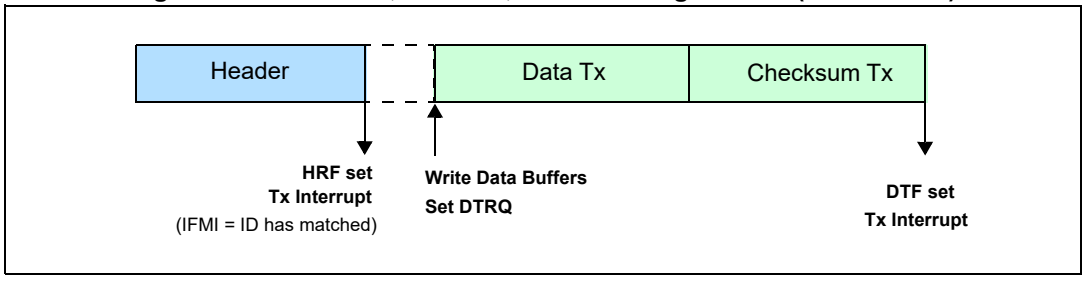


Figure 1737. TX filter, RX filter, BF set configuration: (ID matched + 1)

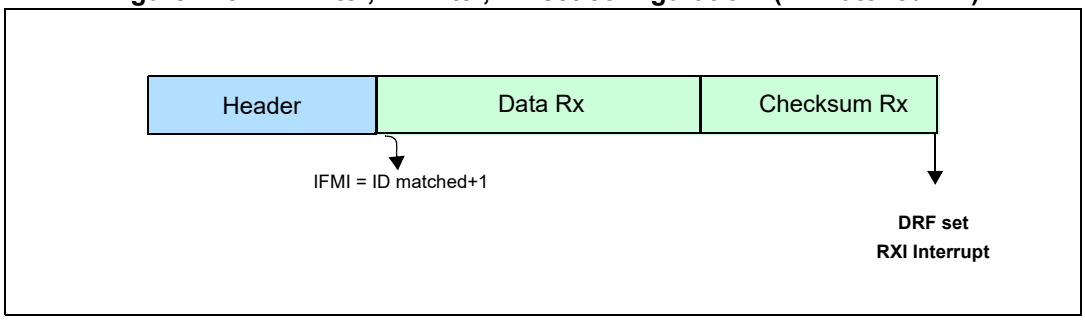
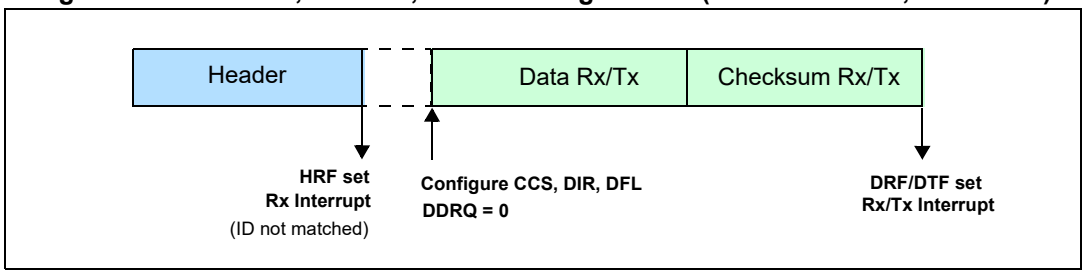


Figure 1738. TX filter, RX filter, BF set configuration: (ID not matched, RX/TX Int.)



63.5.3 Extended frames

Figure 1739. Extended frames (TX Interrupt)

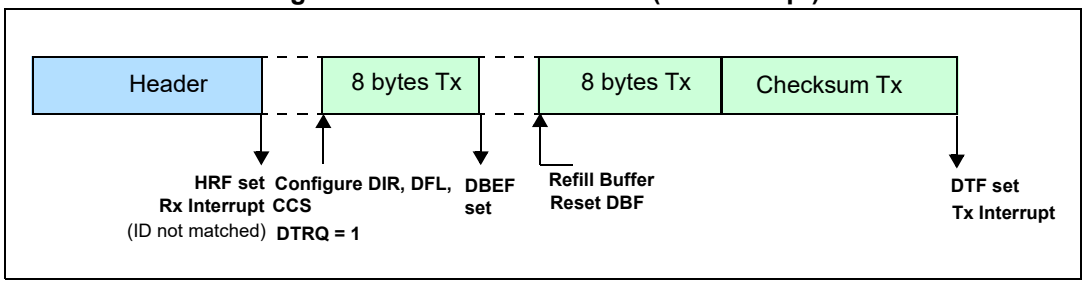
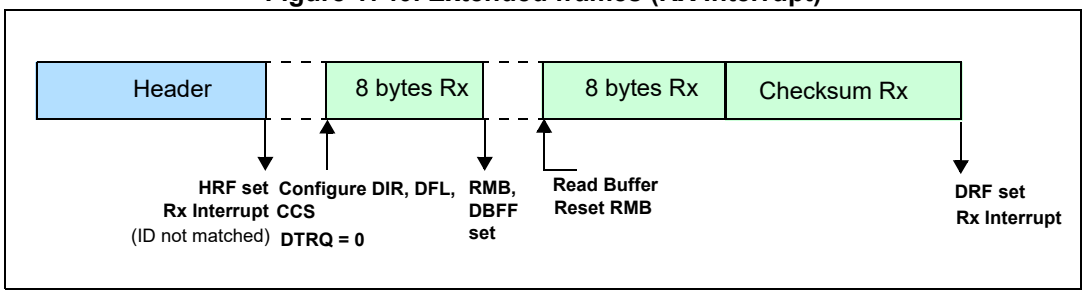


Figure 1740. Extended frames (RX Interrupt)



63.5.4 Timeout

Master Node: Response (during reception only) and frame timeout are checked.

Slave Node: Header, Response (during reception only), and frame timeout are checked.

Figure 1741. Response timeout

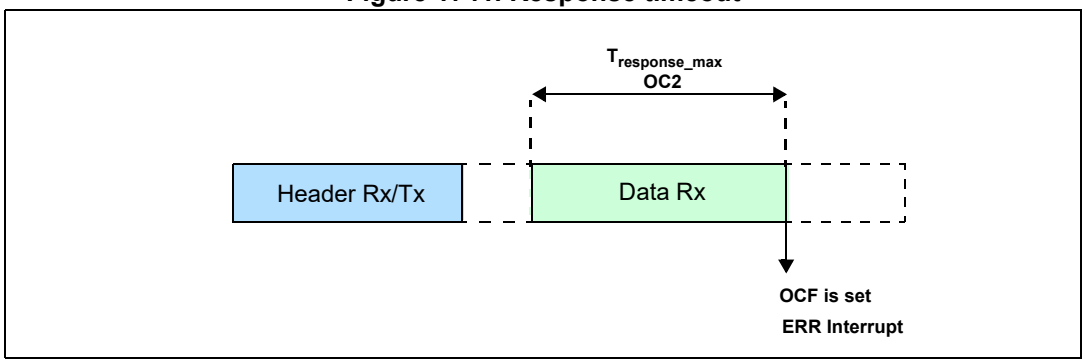
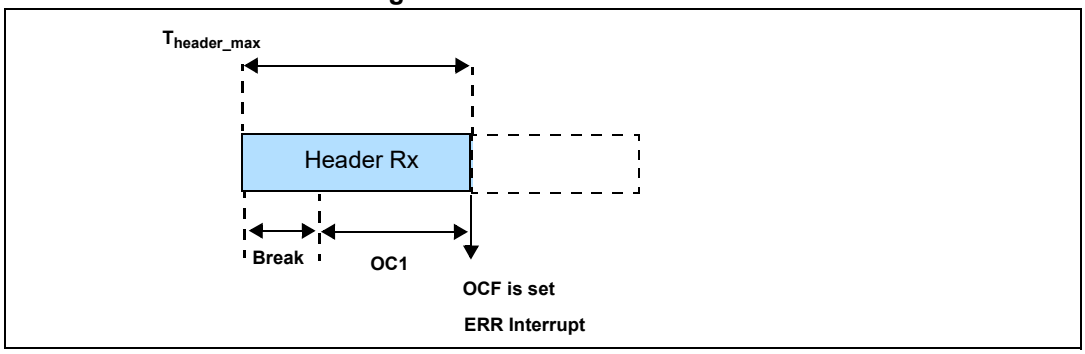
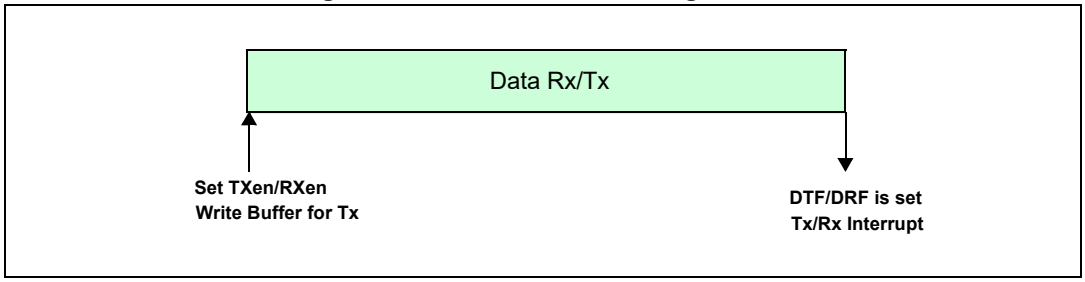


Figure 1742. Header timeout



63.5.5 UART mode

Figure 1743. UART mode configuration



63.5.6 Interrupts

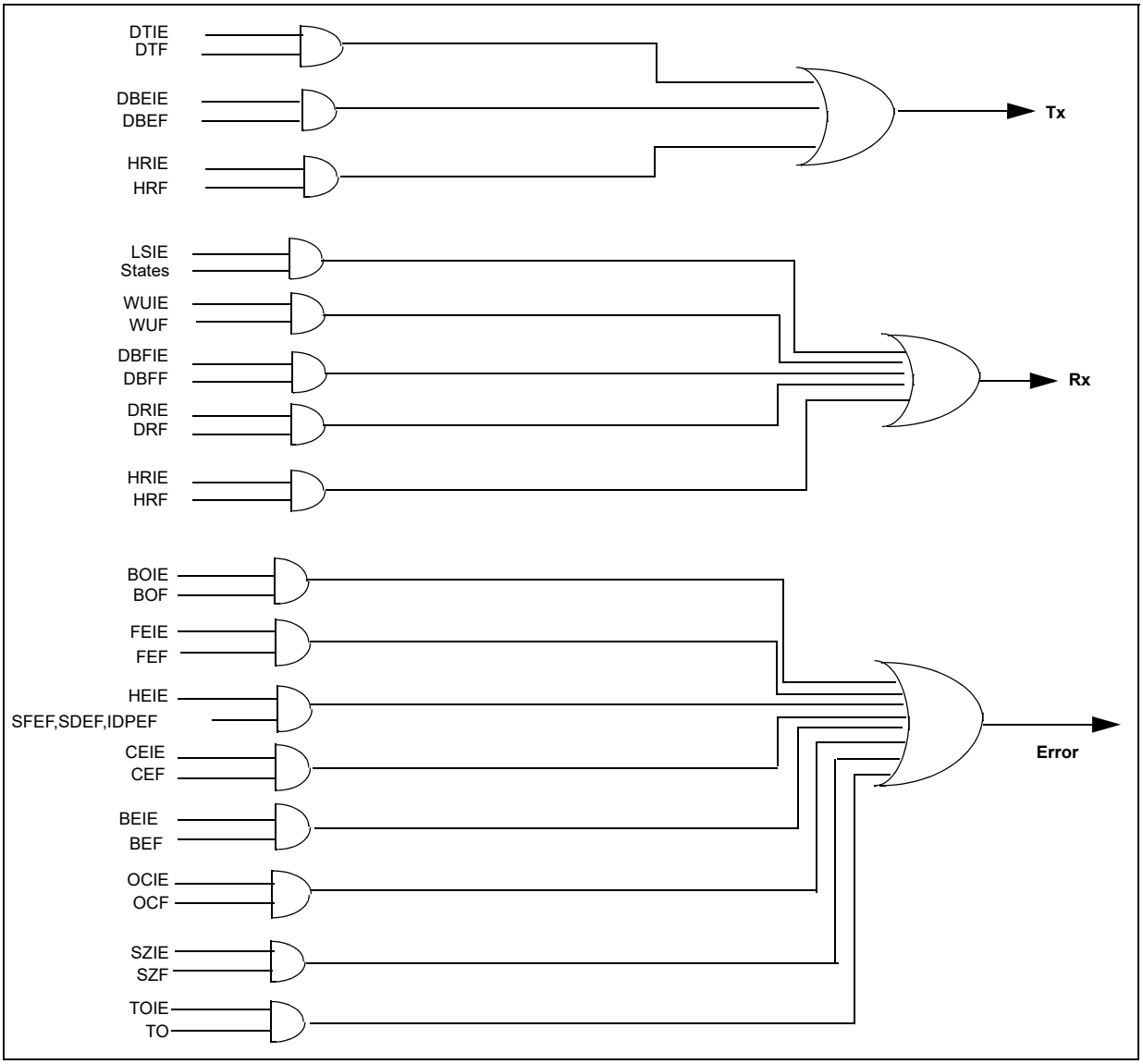
Table 1688. Interrupts

Interrupt event	Event flag	Enable control bit	Interrupt vector
Stuck at zero	SZF	SZIE	Error
Output compare	OCF	OCIE	Error
Bit error	BEF	BEIE	Error
Checksum error	CEF	CEIE	Error
Header error	SFEF or SDEF or IDPEF	HEIE	Error
Frame error	FEF	FEIE	Error
Buffer overrun error	BOF	BOIE	Error
UART timeout error	TO	TOIE	Error
LIN state	Sync Del, Sync Field, Identifier Field, or Checksum Field	LSIE	Rx
Wakeup	WUF	WUIE	Rx
Data buffer full	DBFF	DBFIE	Rx

Table 1688. Interrupts (continued)

Interrupt event	Event flag	Enable control bit	Interrupt vector
Data buffer empty	DBEF	DBEIE	Tx
Data reception complete	DRF	DRIE	Rx
Data transmitted	DTF	DTIE	Tx
Header received	HRF	HRIE	Rx
Header received	HRF	HRIE	Tx (if there is a filter match for Tx identifier)

Figure 1744. Interrupt diagram



64 Reset Generation Module (MC_RGM)

64.1 Introduction

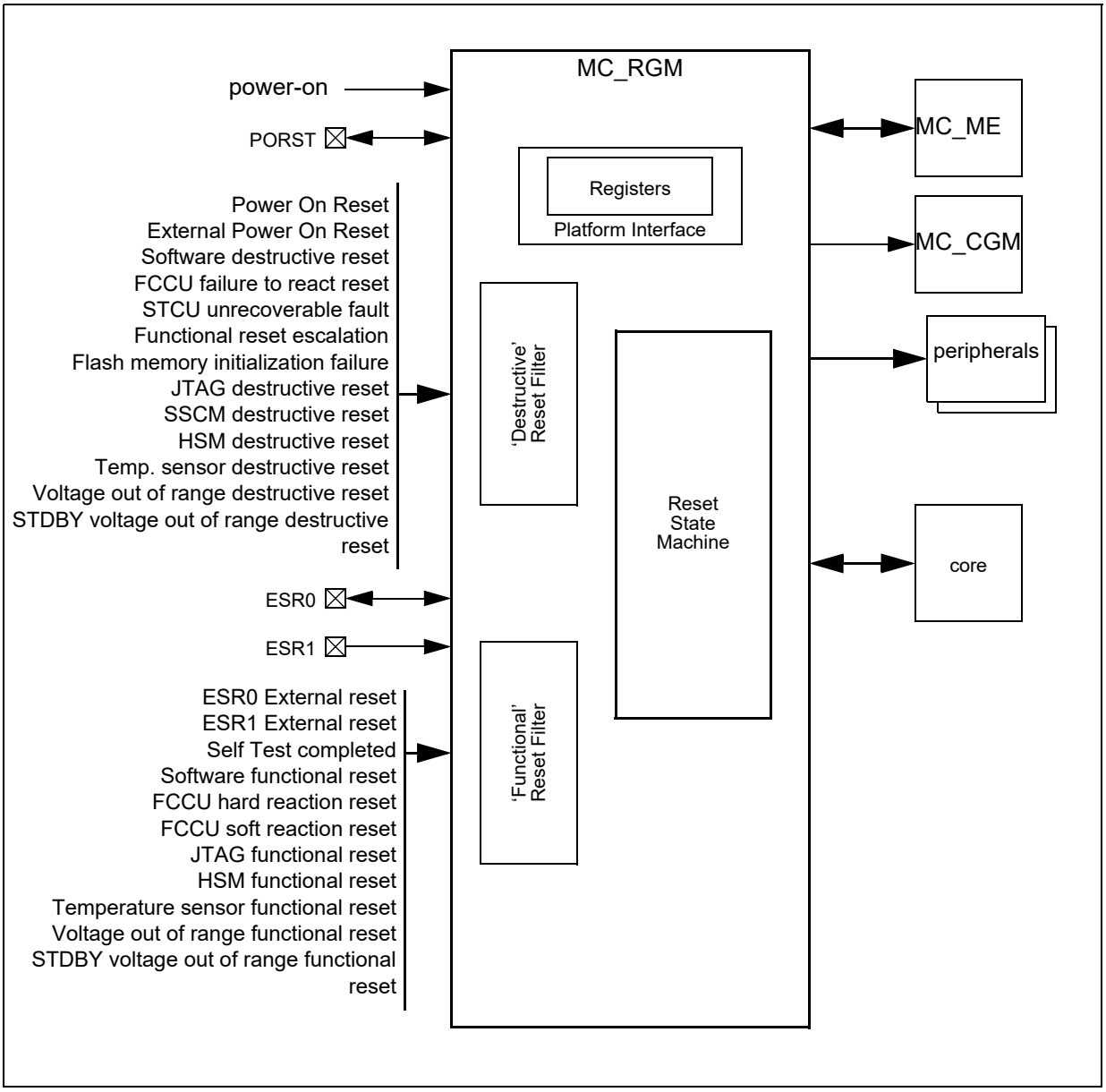
This chapter describes the MC_RGM module.

64.1.1 Overview

The reset generation module (MC_RGM) centralizes the different reset sources and manages the reset sequence of the chip. It provides a register interface and the reset sequencer. Various registers are available to monitor and control the chip reset sequence. The reset sequencer is a state machine which controls the different phases (PHASE0, PHASE1, PHASE2, PHASE3, and IDLE) of the reset sequence and controls the reset signals generated in the system.

Figure 1745 shows the MC_RGM block diagram.

Figure 1745. MC_RGM block diagram



64.1.2 Features

The MC_RGM contains the functionality for the following features:

- 'destructive' resets management
- 'functional' resets management
- signalling of reset events after each reset sequence (reset status flags)
- conversion of reset events to SAFE mode or interrupt request events
- short reset sequence configuration
- bidirectional reset behavior configuration
- boot mode capture on ESR0 deassertion

64.1.3 Reset sources

The different reset sources are organized into two families: ‘destructive’ and ‘functional’.

- A ‘destructive’ reset source is associated with an event related to a critical — usually hardware — error or dysfunction. When a ‘destructive’ reset event occurs, the full reset sequence is applied to the chip starting from PHASE0. This resets the full chip ensuring a safe start-up state for both digital and analog modules.
  ‘Destructive’ resets are:
  - Power On Reset
  - External Power On Reset
  - Software destructive reset
  - FCCU failure to react reset
  - STCU unrecoverable fault
  - Functional reset escalation
  - Flash memory initialization failure
  - JTAG destructive reset
  - SSCM destructive reset
  - HSM destructive reset
  - Temperature sensor destructive reset
  - Voltage out of range destructive reset
  - STDBY voltage out of range destructive reset
- A ‘functional’ reset source is associated with an event related to a less-critical, usually non-hardware, error or dysfunction. When a ‘functional’ reset event occurs, a partial reset sequence is applied to the chip starting from PHASE1. In this case, most digital modules are reset normally, while the state of analog modules or specific digital modules (for example debug modules, flash modules) is preserved .
  ‘Functional’ resets are:
  - ESR0 External reset
  - ESR1 External reset
  - Self Test completed
  - Software functional reset
  - FCCU hard reaction reset
  - FCCU soft reaction reset
  - JTAG functional reset
  - HSM functional reset
  - Temperature sensor functional reset
  - Voltage out of range functional reset
  - STDBY voltage out of range functional reset

When a reset is triggered, the MC_RGM state machine is activated and proceeds through the different phases (PHASE*n* states). Each phase is associated with a particular chip reset being provided to the system. A phase is completed when all corresponding phase completion gates from either the system or internal to the MC_RGM are acknowledged. The chip reset associated with the phase is then released, and the state machine proceeds to the next phase up to entering the IDLE phase. During this entire process, the MC_ME state

machine is held in RESET mode. Only at the end of the reset sequence, when the IDLE phase is reached, does the MC_ME enter the DRUN mode.

Alternatively, it is possible for software to configure some reset source events to be converted from a reset to either a SAFE mode request issued to the MC_ME or to an interrupt issued to the core (see *Section 64.3.1.2: 'Destructive' Event Reset Disable Register (RGM_DERD)* and *Section 64.3.1.3: 'Destructive' Event Alternate Request Register (RGM_DEAR)* for 'destructive' resets and *Section 64.3.1.6: 'Functional' Event Reset Disable Register (RGM_FERD)* and *Section 64.3.1.7: 'Functional' Event Alternate Request Register (RGM_FEAR)* for 'functional' resets).

64.2 External signal description

The MC_RGM interfaces to the bidirectional reset pin ESR0.

64.3 Memory map and register definition

Table 1689. MC_RGM register description

Address	Name	Description	Size	Access			Location
				User	Supervisor	Test	
0x000	RGM_DES	'Destructive' Event Status	word	read	read/write(1)	read/write(1)	64.3.1.1 on page 2832
0x010	RGM_DERD	'Destructive' Event Reset Disable	word	read	read/write(2)	read/write(2)	64.3.1.2 on page 2834
0x020	RGM_DEAR	'Destructive' Event Alternate Request	word	read	read/write	read/write	64.3.1.3 on page 2835
0x030	RGM_DBRE	'Destructive' Bidirectional Reset Enable	word	read	read	read	64.3.1.4 on page 2836
0x300	RGM_FES	'Functional' Event Status	word	read	read/write(1)	read/write(1)	64.3.1.5 on page 2837
0x310	RGM_FERD	'Functional' Event Reset Disable	word	read	read/write(2)	read/write(2)	64.3.1.6 on page 2839
0x320	RGM_FEAR	'Functional' Event Alternate Request	word	read	read/write	read/write	64.3.1.7 on page 2841
0x330	RGM_FBRE	'Functional' Bidirectional Reset Enable	word	read	read/write	read/write	64.3.1.8 on page 2843
0x340	RGM_FESS	'Functional' Event Short Sequence	word	read	read/write	read/write	64.3.1.9 on page 2844
0x604	RGM_FRET	'Functional' Reset Threshold	byte	read	read/write	read/write	64.3.1.10 on page 2846
0x608	RGM_DRET	'Destructive' Reset Escalation Threshold Register	byte	read	read/write	read/write	64.3.1.11 on page 2847

Table 1689. MC_RGM register description (continued)

Address	Name	Description	Size	Access			Location
				User	Supervisor	Test	
0x60C	RGM_EROEC	External Reset Output Extension Control Register	byte	read	read/write	read/write	64.3.1.12 on page 2848
0x610	RGM_PRST0	Peripheral Reset 0	word	read	read/write	read/write	64.3.1.13 on page 2848
0x614	RGM_PRST1	Peripheral Reset 1	word	read	read/write	read/write	64.3.1.14 on page 2850
0x618	RGM_PRST2	Peripheral Reset 2	word	read	read/write	read/write	64.3.1.15 on page 2851
0x61C	RGM_PRST3	Peripheral Reset 3	word	read	read/write	read/write	64.3.1.16 on page 2854
0x620	RGM_PRST4	Peripheral Reset 4	word	read	read/write	read/write	64.3.1.17 on page 2857
0x624	RGM_PRST5	Peripheral Reset 5	word	read	read/write	read/write	64.3.1.18 on page 2860
0x628	RGM_PRST6	Peripheral Reset 6	word	read	read/write	read/write	64.3.1.19 on page 2861
0x62C	RGM_PRST7	Peripheral Reset 7	word	read	read/write	read/write	64.3.1.20 on page 2864
0x630	RGM_PSTAT0	Peripheral Reset Status 0	word	read	read	read	64.3.1.21 on page 2867
0x634	RGM_PSTAT1	Peripheral Reset Status 1	word	read	read	read	64.3.1.22 on page 2868
0x638	RGM_PSTAT2	Peripheral Reset Status 2	word	read	read	read	64.3.1.23 on page 2870
0x63C	RGM_PSTAT3	Peripheral Reset Status 3	word	read	read	read	64.3.1.24 on page 2872
0x640	RGM_PSTAT4	Peripheral Reset Status 4	word	read	read	read	64.3.1.25 on page 2874
0x644	RGM_PSTAT5	Peripheral Reset Status 5	word	read	read	read	64.3.1.26 on page 2875
0x648	RGM_PSTAT6	Peripheral Reset Status 6	word	read	read	read	64.3.1.27 on page 2877
0x64C	RGM_PSTAT7	Peripheral Reset Status 7	word	read	read	read	64.3.1.28 on page 2878

1. individual bits cleared on writing ‘1’
2. write once: ‘0’ = enable, ‘1’ = disable

Note: Any access to unused registers as well as write accesses to read-only registers will:
not change register content
cause a transfer error.

64.3.1 Register descriptions

Unless otherwise noted, all registers may be accessed as 32-bit words, 16-bit half-words, or 8-bit bytes. The bytes are ordered according to big endian. For example, the RGM_DES[8:15] register bits may be accessed as a word at address offset 0x000, as a half-word at address offset 0x002, or as a byte at address offset 0x003.

Some fields may be read-only, and their reset value of ‘1’ or ‘0’ and the corresponding behavior cannot be changed.

64.3.1.1 ‘Destructive’ Event Status Register (RGM_DES)

This register contains the status of the ‘destructive’ reset sources, including those configured to not generate a reset sequence. It can be accessed in read/write on either supervisor mode or test mode. It can be accessed in read only in user mode. Register bits are cleared on writing ‘1’.

This register is reset only on power-on.

Address: 0x000 Access: User read, Supervisor read/write, Test read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	F_VOR_STDBY	F_VOR_DEST	F_TSR_DEST	0	0	0	0	0	0	0
W							w1c	w1c	w1c							
POR	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	F_HSM_DEST	F_SSCM_DEST	0	0	0	F_JTAG_DEST	F_FIF	F_EDR	0	0	F_SUF	F_FFRR	F_SOFT_DEST	0	F_PORST	F_POR
W	w1c	w1c				w1c	w1c	w1c			w1c	w1c	w1c		w1c	w1c
POR	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

Figure 1746. ‘Destructive’ Event Status Register (RGM_DES)

Table 1690. RGM_DES field descriptions

Field	Description
6 F_VOR_STDBY	Flag for voltage out of range ‘from LVDs or HVDs, or both, during standby mode ‘destructive’ reset 0 No voltage out of range ‘destructive’ reset event has occurred since the last clear 1 A voltage out of range ‘destructive’ reset event has occurred **Note:** For LVD100_SB it is not allowed to enable destructive reset, only functional reset is supported.
7 F_VOR_DEST	Flag for voltage out of range ‘destructive’ reset 0 No voltage out of range ‘destructive’ reset event has occurred since the last clear 1 A voltage out of range ‘destructive’ reset event has occurred
8 F_TSR_DEST	Flag for temperature sensor 'destructive' reset 0 No temperature sensor 'destructive' reset event has occurred since the last clear 1 A temperature sensor 'destructive' reset event has occurred
16 F_HSM_DEST	Flag for HSM 'destructive' reset request 0 No HSM 'destructive' reset request event has occurred since the last clear 1 A HSM 'destructive' reset request event has occurred
17 F_SSCM_DEST	Flag for SSCM 'destructive' reset request 0 No SSCM 'destructive' reset request event has occurred since the last clear 1 A SSCM 'destructive' reset request event has occurred
21 F_JTAG_DEST	Flag for JTAG 'destructive' reset request 0 No JTAG 'destructive' reset request event has occurred since the last clear 1 A JTAG 'destructive' reset request event has occurred
22 F_FIF	Flag for flash memory initialization failure 0 No flash memory initialization failure event has occurred since the last clear 1 A flash memory initialization failure event has occurred
23 F_EDR	Flag for ‘functional’ reset escalation 0No ‘functional’ reset escalation event has occurred since the last clear 1 A ‘functional’ reset escalation event has occurred
26 F_SUF	Flag for STCU2 unrecoverable fault 0 No STCU2 unrecoverable fault event during offline BIST has occurred since the last clear 1 A STCU2 unrecoverable fault event during offline BIST has occurred
27 F_FFRR	Flag for FCCU failure to react reset 0 No FCCU failure to react reset event has occurred since the last clear 1 A FCCU failure to react reset event has occurred
28 F_SOFT_DEST	Flag for software ‘destructive’ reset 0 No software ‘destructive’ reset event has occurred since the last clear 1 A software ‘destructive’ reset event has occurred
30 F_PORST	Flag for external power-on reset 0 No external power-on reset has occurred since the last clear 1 A external power-on reset event has occurred **Note:** F_PORST will not be set if an external PORST is asserted during standby mode. Rather MC_RGM.FES.F_JTAG_FUNC will be set
31 F_POR	Flag for Power-On reset 0 No power-on event has occurred since the last clear 1 A power-on event has occurred

64.3.1.2 'Destructive' Event Reset Disable Register (RGM_DERD)

This register provides dedicated bits to disable particular 'destructive' reset sources. When a 'destructive' reset source is disabled, the associated 'destructive' event will trigger either a SAFE mode request or an interrupt request (see *Section 64.3.1.3: 'Destructive' Event Alternate Request Register (RGM_DEAR)*). It can be accessed in read/write in either supervisor mode or test mode. It can be accessed in read only in user mode. Each byte can be written only once after a 'destructive' or power-on reset.

This register is reset only on power-on.

Caution: It is important to clear the RGM_DES register before setting any of the bits in the RGM_DERD register to '1'. Otherwise a redundant SAFE mode request or interrupt request may occur.

Address: 0x010 Access: User read, Supervisor read/write, Test read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	D_VOR_STDBY	D_VOR_DEST	D_TSR_DEST	0	0	0	0	0	0	0
W																
POR	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	D_HSM_DEST	D_SSCM_DEST	0	0	0	D_JTAG_DEST	D_FIF	D_EDR	0	0	D_SUF	D_FFRR	D_SOFT_DEST	0	D_PORST	D_POR
W																
POR	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1747. 'Destructive' Event Reset Disable Register (RGM_DERD)

Table 1691. RGM_DERD field descriptions

Field	Description
6 D_VOR_STDBY	Disable voltage out of range from LVDs or HVDs, or both, during standby mode 'destructive' reset 0 A voltage out of range 'destructive' reset event triggers a reset sequence
7 D_VOR_DEST	Disable voltage out of range 'destructive' reset 0 A voltage out of range 'destructive' reset event triggers a reset sequence
8 D_TSR_DEST	Disable temperature sensor 'destructive' reset 0 A temperature sensor 'destructive' reset event triggers a reset sequence
16 D_HSM_DEST	Disable HSM 'destructive' reset request 0 A HSM 'destructive' reset request event triggers a reset sequence

Table 1691. RGM_DERD field descriptions (continued)

Field	Description
17 D_SSCM_DEST	Disable SSCM 'destructive' reset request 0 A SSCM 'destructive' reset request event triggers a reset sequence
21 D_JTAG_DEST	Disable JTAG 'destructive' reset request 0 A JTAG 'destructive' reset request event triggers a reset sequence
22 D_FIF	Disable flash memory initialization failure 0 A flash memory initialization failure event triggers a reset sequence
23 D_EDR	Disable ‘functional’ reset escalation 0 A ‘functional’ reset escalation event triggers a reset sequence
26 D_SUF	Disable STCU2 unrecoverable fault 0 A STCU2 unrecoverable fault event triggers a reset sequence
27 D_FFRR	Disable FCCU failure to react reset 0 A FCCU failure to react reset event triggers a reset sequence
28 D_SOFT_DEST	Disable software ‘destructive’ reset 0 A software ‘destructive’ reset triggers a reset sequence
30 D_PORST	Disable external power-on reset 0 An external power-on reset event triggers a reset sequence 1 An external power-on reset event generates either a SAFE mode or an interrupt request depending on the values in RGM_DEAR.AR_PORST
31 D_POR	Disable Power-On reset 0 A power-on event triggers a reset sequence

64.3.1.3 ‘Destructive’ Event Alternate Request Register (RGM_DEAR)

This register defines an alternate request to be generated when a reset on a ‘destructive’ event has been disabled. The alternate request can be either a SAFE mode request to MC_ME or an interrupt request to the system. It can be accessed in read/write in either supervisor mode or test mode. It can be accessed in read only in user mode.

This register is reset only on power-on.

Address: 0x020 Access: User read, Supervisor read/write, Test read/write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	AR_PORST	0
W																
POR	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1748. ‘Destructive’ Event Alternate Request Register (RGM_DEAR)

Table 1692. RGM_DEAR field descriptions

Field	Description
30 AR_PORST	Alternate Request for external power-on reset 0 Generate a SAFE mode request on an external power-on reset event if the reset is disabled 1 Generate an interrupt request on an external power-on reset event if the reset is disabled

64.3.1.4 'Destructive' Bidirectional Reset Enable Register (RGM_DBRE)

This register enables the generation of an external reset on 'destructive' reset. It can be accessed in read only in either supervisor mode or test mode. It can be accessed in read only in user mode.

This register is reset only on power-on.

Address: 0x030 Access: User read, Supervisor read, Test read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	BE_VOR_STDBY	BE_VOR_DEST	BE_TSR_DEST	0	0	0	0	0	0	0
W																
POR	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	BE_HSM_DEST	BE_SSCM_DEST	0	0	0	BE_JTAG_DEST	BE_FIF	BE_EDR	0	0	BE_SUF	BE_FFRR	BE_SOFT_DEST	0	BE_PORST	BE_POR
W																
POR	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1749. 'Destructive' Bidirectional Reset Enable Register (RGM_DBRE)

Table 1693. RGM_DBRE field descriptions

Field	Description
6 BE_VOR_STDBY	Bidirectional Reset Enable for voltage out of range from LVDs or HVDs, or both, during standby mode 'destructive' reset 0 ESR0_B is asserted on a voltage out of range 'destructive' reset event
7 BE_VOR_DEST	Bidirectional Reset Enable for voltage out of range 'destructive' reset 0 ESR0_B is asserted on a voltage out of range 'destructive' reset event
8 BE_TSR_DEST	Bidirectional Reset Enable for temperature sensor 'destructive' reset 0 ESR0_B is asserted on a temperature sensor 'destructive' reset event if the reset is enabled

Table 1693. RGM_DBRE field descriptions (continued)

Field	Description
16 BE_HSM_DEST	Bidirectional Reset Enable for HSM 'destructive' reset request 0 ESR0_B is asserted on a HSM 'destructive' reset request event
17 BE_SSCM_DEST	Bidirectional Reset Enable for SSCM 'destructive' reset request 0 ESR0_B is asserted on a SSCM 'destructive' reset request event
21 BE_JTAG_DEST	Bidirectional Reset Enable for JTAG 'destructive' reset request 0 ESR0_B is asserted on a JTAG 'destructive' reset request event 1 ESR0_B is not asserted on a JTAG 'destructive' reset request event
22 BE_FIF	Bidirectional Reset Enable for flash memory initialization failure 0 ESR0_B is asserted on a flash memory initialization failure event
23 BE_EDR	Bidirectional Reset Enable for ‘functional’ reset escalation 0 ESR0_B is asserted on a ‘functional’ reset escalation event
26 BE_SUF	Bidirectional Reset Enable for STCU2 unrecoverable fault 0 ESR0_B is asserted on a STCU2 unrecoverable fault event
27 BE_FFRR	Bidirectional Reset Enable for FCCU failure to react reset 0 ESR0_B is asserted on a FCCU failure to react reset event
28 BE_SOFT_DEST	Bidirectional Reset Enable for software ‘destructive’ reset 0 ESR0_B is asserted on a software ‘destructive’ reset event
30 BE_PORST	Bidirectional Reset Enable for external power-on reset 0 ESR0_B is asserted on an external power-on reset event if the reset is enabled
31 BE_POR	Bidirectional Reset Enable for power-on reset 0 ESR0_B is asserted on a power-on reset event

64.3.1.5 ‘Functional’ Event Status Register (RGM_FES)

This register contains the status of the ‘functional’ reset sources, including those configured to not generate a reset sequence. It can be accessed in read/write on either supervisor mode or test mode. It can be accessed in read only in user mode. Register bits are cleared on writing ‘1’ if the triggering event has already been cleared at the source.

This register is reset only on power-on.

Note: If a ‘functional’ reset source is configured to generate a SAFE mode request or an interrupt request, software needs to clear the event in the source module at least three system clock cycles before it clears the associated RGM_FES status bit in order to avoid multiple SAFE mode requests or interrupts for the same event. In order to avoid having to count cycles, it is good practice for software to check whether the RGM_FES has been properly cleared, and if not, clear it again.

Address: 0x300 Access: User read, Supervisor read/write, Test read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	F_VOR_FUNC_STDBY	F_VOR_FUNC	F_TSR_FUNC	0	0	0	0	0	0	0
W							w1c	w1c	w1c							
POR	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	F_HSM_FUNC	0	0	0	0	F_JTAG_FUNC	0	0	0	F_FCCU_SOFT	F_FCCU_HARD	0	F_SOFT_FUNC	F_ST_DONE	F_ESR1	F_ESR0
W	w1c					w1c				w1c	w1c		w1c	w1c	w1c	w1c
POR	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1750. 'Functional' Event Status Register (RGM_FES)

Table 1694. RGM_FES field descriptions

Field	Description
6 F_VOR_FUNC_STDBY	Flag for voltage out of range from LVDs or HVDs, or both, during standby 'functional' reset 0 No voltage out of range 'functional' reset event has occurred since either the last clear or the last power-on reset assertion 1 A voltage out of range 'functional' reset event has occurred
7 F_VOR_FUNC	Flag for voltage out of range 'functional' reset 0 No voltage out of range 'functional' reset event has occurred since either the last clear or the last power-on reset assertion 1 A voltage out of range 'functional' reset event has occurred
8 F_TSR_FUNC	Flag for temperature sensor 'functional' reset 0 No temperature sensor 'functional' reset event has occurred since either the last clear or the last power-on reset assertion 1 A temperature sensor 'functional' reset event has occurred
16 F_HSM_FUNC	Flag for HSM 'functional' reset request 0 No HSM 'functional' reset request event has occurred since either the last clear or the last power-on reset assertion 1 A HSM 'functional' reset request event has occurred

Table 1694. RGM_FES field descriptions (continued)

Field	Description
21 F_JTAG_FUNC	Flag for JTAG 'functional' reset 0 No JTAG 'functional' reset event has occurred since either the last clear or the last power-on reset assertion 1 A JTAG 'functional' reset event has occurred **Note:** The flag F_JTAG_FUNC indicates a PORST functional reset during the stand-by mode.
25 F_FCCU_SOFT	Flag for FCCU soft reaction 0 No FCCU soft reaction event has occurred since either the last clear or the last power-on reset assertion 1 A FCCU soft reaction event has occurred
26 F_FCCU_HARD	Flag for FCCU hard reaction reset 0 No FCCU hard reaction reset event has occurred since either the last clear or the last power-on reset assertion 1 A FCCU hard reaction reset event has occurred
28 F_SOFT_FUNC	Flag for software 'functional' reset 0 No software 'functional' reset event has occurred since either the last clear or the last power-on reset assertion 1 A software 'functional' reset event has occurred
29 F_ST_DONE	Flag for self test completed 0 No self test completed event has occurred since either the last clear or the last power-on reset assertion 1 A self test completed event has occurred
30 F_ESR1	Flag for ESR1 External Reset 0 No ESR1 external reset event has occurred since either the last clear or the last power-on reset assertion 1 An ESR1 external reset event has occurred
31 F_ESR0	Flag for ESR0 External Reset 0 No ESR0 external reset event has occurred since either the last clear or the last power-on reset assertion 1 An ESR0 external reset event has occurred

64.3.1.6 'Functional' Event Reset Disable Register (RGM_FERD)

This register provides dedicated bits to disable 'functional' reset sources.When a 'functional' reset source is disabled, the associated 'functional' event will trigger either a SAFE mode request or an interrupt request (see *Section 64.3.1.7: 'Functional' Event Alternate Request Register (RGM_FEAR)*). It can be accessed in read/write in either supervisor mode or test mode. It can be accessed in read only in user mode. Each byte can be written only once after power-on reset.

This register is reset only on power-on and any 'destructive' reset.

Caution: It is important to clear the RGM_FES register before setting any of the bits in the RGM_FERD register to '1'. Otherwise a redundant SAFE mode request or interrupt request may occur.

Address: 0x310 Access: User read, Supervisor read/write, Test read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	D_VOR_FUNC_STDBY	D_VOR_FUNC	D_TSR_FUNC	0	0	0	0	0	0	0
W																
'Dest' Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	D_HSM_FUNC	0	0	0	0	D_JTAG_FUNC	0	0	0	D_FCCU_SOFT	D_FCCU_HARD	0	D_SOFT_FUNC	D_ST_DONE	D_ESR1	D_ESR0
W																
'Dest' Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1751. 'Functional' Event Reset Disable Register (RGM_FERD)

Table 1695. RGM_FERD field descriptions

Field	Description
6 D_VOR_FUNC_STDBY	Disable voltage out of range from LVDs or HVDs, or both, during standby 'functional' reset 0 A voltage out of range 'functional' reset event triggers a reset sequence 1 A voltage out of range 'functional' reset event generates either a SAFE mode or an interrupt request depending on the value of RGM_FEAR.AR_VOR_FUNC_STDBY
7 D_VOR_FUNC	Disable voltage out of range 'functional' reset 0 A voltage out of range 'functional' reset event triggers a reset sequence 1 A voltage out of range 'functional' reset event generates either a SAFE mode or an interrupt request depending on the value of RGM_FEAR.AR_VOR_FUNC
8 D_TSR_FUNC	Disable temperature sensor 'functional' reset 0 A temperature sensor 'functional' reset event triggers a reset sequence 1 A temperature sensor 'functional' reset event generates either a SAFE mode or an interrupt request depending on the value of RGM_FEAR.AR_TSR_FUNC
16 D_HSM_FUNC	Disable HSM 'functional' reset request 0 A HSM 'functional' reset request event triggers a reset sequence
21 D_JTAG_FUNC	Disable JTAG 'functional' reset 0 A JTAG 'functional' reset event triggers a reset sequence

Table 1695. RGM_FERD field descriptions (continued)

Field	Description
25 D_FCCU_SOFT	Disable FCCU soft reaction 0 A FCCU soft reaction event triggers a reset sequence 1 A FCCU soft reaction event generates either a SAFE mode or an interrupt request depending on the value of RGM_FEAR.AR_FCCU_SOFT
26 D_FCCU_HARD	Disable FCCU hard reaction reset 0 A FCCU hard reaction reset event triggers a reset sequence
28 D_SOFT_FUNC	Disable software ‘functional’ reset 0 A software ‘functional’ reset event triggers a reset sequence
29 D_ST_DONE	Disable self test completed 0 A self test completed event triggers a reset sequence
30 D_ESR1	Disable ESR1 External Reset 0 An ESR1 external reset event triggers a reset sequence 1 An ESR1 external reset event generates either a SAFE mode or an interrupt request depending on the values in RGM_FEAR.AR_ESR1
31 D_ESR0	Disable ESR0 External Reset 0 An ESR0 external reset event triggers a reset sequence

64.3.1.7 ‘Functional’ Event Alternate Request Register (RGM_FEAR)

This register defines an alternate request to be generated when a reset on a ‘functional’ event has been disabled. The alternate request can be either a SAFE mode request to MC_ME or an interrupt request to the system. It can be accessed in read/write in either supervisor mode or test mode. It can be accessed in read only in user mode.

This register is reset only on power-on and any ‘destructive’ reset.

Address: 0x320 Access: User read, Supervisor read/write, Test read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	AR_VOR_FUNC_STDBY	AR_VOR_FUNC	AR_TSR_FUNC	0	0	0	0	0	0	0
W																
'Dest' Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	AR_ESR1	0
W																
'Dest' Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1752. 'Functional' Event Alternate Request Register (RGM_FEAR)

Table 1696. RGM_FEAR field descriptions

Field	Description
6 AR_VOR_FUNC_STDBY	Alternate Request for LVDs or HVDs, or both, during standby voltage out of range 'functional' reset 0 Generate a SAFE mode request on a voltage out of range 'functional' reset event if the reset is disabled 1 Generate an interrupt request on a voltage out of range 'functional' reset event if the reset is disabled
7 AR_VOR_FUNC	Alternate Request for voltage out of range 'functional' reset 0 Generate a SAFE mode request on a voltage out of range 'functional' reset event if the reset is disabled 1 Generate an interrupt request on a voltage out of range 'functional' reset event if the reset is disabled
8 AR_TSR_FUNC	Alternate Request for temperature sensor 'functional' reset 0 Generate a SAFE mode request on a temperature sensor 'functional' reset event if the reset is disabled 1 Generate an interrupt request on a temperature sensor 'functional' reset event if the reset is disabled
30 AR_ESR1	Alternate Request for ESR1 External Reset 0 Generate a SAFE mode request on a ESR1 external reset event if the reset is disabled 1 Generate an interrupt request on a ESR1 external reset event if the reset is disabled

64.3.1.8 ‘Functional’ Bidirectional Reset Enable Register (RGM_FBRE)

This register enables the generation of an external reset on ‘functional’ reset. It can be accessed in read/write in either supervisor mode or test mode. It can be accessed in read in user mode.

This register is reset only on power-on and any ‘destructive’ reset.

Address: 0x330 Access: User read, Supervisor read/write, Test read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	BE_VOR_FUNC_STDBY	BE_VOR_FUNC	BE_TSR_FUNC	0	0	0	0	0	0	0
W																
‘Dest’ Reset	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	BE_HSM_FUNC	0	0	0	0	BE_JTAG_FUNC	0	0	0	BE_FCCU_SOFT	BE_FCCU_HARD	0	BE_SOFT_FUNC	BE_ST_DONE	BE_ESR1	BE_ESR0
W																
‘Dest’ Reset	1	0	0	0	0	1	0	0	0	1	1	0	1	0	1	0

Figure 1753. ‘Functional’ Bidirectional Reset Enable Register (RGM_FBRE)

Table 1697. RGM_FBRE field descriptions

Field	Description
6 BE_VOR_FUNC_STDBY	Bidirectional Reset Enable for LVDs or HVDs, or both, voltage out of range during standby ‘functional’ reset 0 ESR0_B is asserted on a voltage out of range ‘functional’ reset event if the reset is enabled 1 ESR0_B is not asserted on a voltage out of range ‘functional’ reset event
7 BE_VOR_FUNC	Bidirectional Reset Enable for voltage out of range ‘functional’ reset 0 ESR0_B is asserted on a voltage out of range ‘functional’ reset event if the reset is enabled 1 ESR0_B is not asserted on a voltage out of range ‘functional’ reset event
8 BE_TSR_FUNC	Bidirectional Reset Enable for temperature sensor 'functional' reset 0 ESR0_B is asserted on a temperature sensor 'functional' reset event if the reset is enabled 1 ESR0_B is not asserted on a temperature sensor 'functional' reset event

Table 1697. RGM_FBRE field descriptions (continued)

Field	Description
16 BE_HSM_FUNC	Bidirectional Reset Enable for HSM 'functional' reset request 0 ESR0_B is asserted on a HSM 'functional' reset request event 1 ESR0_B is not asserted on a HSM 'functional' reset request event
21 BE_JTAG_FUNC	Bidirectional Reset Enable for JTAG ‘functional’ reset 0 ESR0_B is asserted on a JTAG ‘functional’ reset event if the reset is enabled 1 ESR0_B is not asserted on a JTAG ‘functional’ reset event
25 BE_FCCU_SOFT	Bidirectional Reset Enable for FCCU soft reaction 0 ESR0_B is asserted on a FCCU soft reaction event 1 ESR0_B is not asserted on a FCCU soft reaction event
26 BE_FCCU_HARD	Bidirectional Reset Enable for a FCCU hard reaction 0 ESR0_B is asserted on a FCCU hard reaction reset event 1 ESR0_B is not asserted on a FCCU hard reaction reset event
28 BE_SOFT_FUNC	Bidirectional Reset Enable for software ‘functional’ reset 0 ESR0_B is asserted on a software ‘functional’ reset event if the reset is enabled 1 ESR0_B is not asserted on a software ‘functional’ reset event
29 BE_ST_DONE	Bidirectional Reset Enable for self test completed 0 ESR0_B is asserted on a self test completed event
30 BE_ESR1	Bidirectional Reset Enable for ESR1 External Reset 0 ESR0_B is asserted on an ESR1 external reset event if the reset is enabled 1 ESR0_B is not asserted on an ESR1 external reset event
31 BE_ESR0	Bidirectional Reset Enable for ESR0 External Reset 0 ESR0_B is asserted on an ESR0 external reset event if the reset is enabled

64.3.1.9 ‘Functional’ Event Short Sequence Register (RGM_FESS)

This register defines which reset sequence will be done when a ‘functional’ reset sequence is triggered. The ‘functional’ reset sequence can either start from PHASE1 or from PHASE3, skipping PHASE1 and PHASE2.

Note: This could be useful for fast reset sequence, for example to skip flash memory reset.

It can be accessed in read/write in either supervisor mode or test mode. It can be accessed in read in user mode.

This register is reset only on power-on and any ‘destructive’ reset.

Address: 0x340 Access: User read, Supervisor read/write, Test read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	SS_VOR_FUNC_STDBY	SS_VOR_FUNC	SS_TSR_FUNC	0	0	0	0	0	0	0
W																
'Dest' Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	SS_HSM_FUNC	0	0	0	0	SS_JTAG_FUNC	0	0	0	SS_FCCU_SOFT	SS_FCCU_HARD	0	SS_SOFT_FUNC	SS_ST_DONE	SS_ESR1	SS_ESR0
W																
'Dest' Reset	1	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0

Figure 1754. 'Functional' Event Short Sequence Register (RGM_FESS)

Table 1698. RGM_FESS field descriptions

Field	Description
6 SS_VOR_FUNC_STDBY	Short Sequence for LVDs or HVDs, or both, voltage out of range during standby 'functional' reset 0 The reset sequence triggered by a voltage out of range 'functional' reset event will start from PHASE1 1 The reset sequence triggered by a voltage out of range 'functional' reset event will start from PHASE3
7 SS_VOR_FUNC	Short Sequence for voltage out of range 'functional' reset 0 The reset sequence triggered by a voltage out of range 'functional' reset event will start from PHASE1 1 The reset sequence triggered by a voltage out of range 'functional' reset event will start from PHASE3
8 SS_TSR_FUNC	Short Sequence for temperature sensor 'functional' reset 0 The reset sequence triggered by a temperature sensor 'functional' reset event will start from PHASE1 1 The reset sequence triggered by a temperature sensor 'functional' reset event will start from PHASE3
16 SS_HSM_FUNC	Short Sequence for HSM 'functional' reset request 1 The reset sequence triggered by a HSM 'functional' reset request event will start from PHASE3

Table 1698. RGM_FESS field descriptions (continued)

Field	Description
21 SS_JTAG_FUNC	Short Sequence for JTAG 'functional' reset 0 The reset sequence triggered by a JTAG 'functional' reset event will start from PHASE1 1 The reset sequence triggered by a JTAG 'functional' reset event will start from PHASE3
25 SS_FCCU_SOFT	Short Sequence for FCCU soft reaction 1 The reset sequence triggered by a FCCU soft reaction event will start from PHASE3
26 SS_FCCU_HARD	Short Sequence for a FCCU hard reaction 0 The reset sequence triggered by a FCCU hard reaction reset event will start from PHASE1
28 SS_SOFT_FUNC	Short Sequence for software 'functional' reset 0 The reset sequence triggered by a software 'functional' reset event will start from PHASE1 1 The reset sequence triggered by a software 'functional' reset event will start from PHASE3
29 SS_ST_DONE	Short Sequence for self test completed 0 The reset sequence triggered by a self test completed event will start from PHASE1
30 SS_ESR1	Short Sequence for ESR1 External Reset 0 The reset sequence triggered by an ESR1 external reset event will start from PHASE1 1 The reset sequence triggered by an ESR1 external reset event if the reset is enabled will start from PHASE3
31 SS_ESR0	Short Sequence for ESR0 External Reset 0 The reset sequence triggered by an ESR0 external reset event will start from PHASE1 1 The reset sequence triggered by an ESR0 external reset event if the reset is enabled will start from PHASE3

64.3.1.10 'Functional' Reset Escalation Threshold Register (RGM_FRET)

This register sets the threshold for 'functional' reset escalation to a 'destructive' reset. It can be accessed in read/write in either supervisor mode or test mode. It can be accessed in read only in user mode.

Writing a non-zero value to the FRET field will enable the 'functional' reset escalation function. Writing any value to this register will reset the 'functional' reset counter. See *Section 64.4.6: 'Functional' reset escalation* for details on the 'functional' reset escalation function.

This register is reset only on power-on and any 'destructive' reset.

Address: 0x604 Access: User read, Supervisor read/write, Test read/write

	0	1	2	3	4	5	6	7
R	0	0	0	0	FRET			
W								
'Dest' Reset	0	0	0	0	0	1	0	0

Figure 1755. 'Functional' Reset Escalation Threshold Register (RGM_FRET)

Table 1699. RGM_FRET field descriptions

Field	Description
4:7 FRET	'Functional' Reset Escalation Threshold 0000 'Functional' reset escalation function disabled 0001 Number of 'functional' reset events before escalation to destructive reset ... 1111 Number of 'functional' reset events before escalation to destructive reset

64.3.1.11 'Destructive' Reset Escalation Threshold Register (RGM_DRET)

This register sets the threshold for 'destructive' reset escalation keeping the chip in the reset state until the next power-on reset triggers a new reset sequence. It can be accessed in read/write in either supervisor mode or test mode. It can be accessed in read only in user mode.

Writing a non-zero value to the DRET field will enable the 'destructive' reset software escalation function. Writing any value to this register will reset the 'destructive' reset counter. See *Section 64.4.7: 'Destructive' reset escalation*, for details on the 'destructive' reset escalation function.

This register is reset only on power-on.

Address: 0x608 Access: User read, Supervisor read/write, Test read/write

	0	1	2	3	4	5	6	7
R	0	0	0	0	DRET			
W								
POR	0	0	0	0	1	0	0	0

Figure 1756. 'Destructive' Reset Escalation Threshold Register (RGM_DRET)

Table 1700. RGM_DRET field descriptions

Field	Description
4:7 DRET	'Destructive' Reset Escalation Threshold 0000 'Destructive' reset escalation function disabled 0001 Number of 'destructive' reset events before chip enters permanent reset state ... 1111 Number of 'destructive' reset events before chip enters permanent reset state

64.3.1.12 External Reset Output Extension Control Register (RGM_EROEC)

This register controls the assertion of the ESR0_B pin after the reset sequence has completed. It can be accessed in read/write in either supervisor mode or test mode. It can be accessed in read only in user mode. The register bit is cleared on write ‘1’.

Clearing the EROEC bit will cause the MC_RGM to stop asserting the ESR0_B pin within two 16 MHz internal RC oscillator clock cycles. The ESR0_B pin cannot be asserted by writing to this register.

This register also provides the current value sampled on ESR0 via the ERIS bit. This is useful for software to poll if, for example, ESR0_B is used in the system to synchronize the application software with an external event.

This register is reset only on power-on.

Address: 0x60C Access: User read, Supervisor read/write, Test read/write

	0	1	2	3	4	5	6	7
R	0	0	0	0	0	0	ERIS	EROEC
W								w1c
POR	0	0	0	0	0	0	0	1

Figure 1757. External Reset Output Extension Control Register (RGM_EROEC)

Table 1701. RGM_EROEC field descriptions

Field	Description
6 ERIS	External Reset Input Status 0 A ‘0’ is being observed on ESR0 1 A ‘1’ is being observed on ESR0
7 EROEC	External Reset Output Assertion Control 0 ESR0_B is not being asserted by the MC_RGM 1 ESR0_B is being asserted by the MC_RGM

64.3.1.13 Peripheral Reset Register 0 (RGM_PRST0)

This register provides individual resets for various peripherals. It can be accessed in read/write in either supervisor mode or test mode. It can be accessed in read only in user mode. See *Table 1702* for details.

Address: 0x610 Access: User read, Supervisor read/write, Test read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	PIT_0_RST	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	SIUL_RST	0	0	0	SIPI_0_RST	0	LFAST_0_RST	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1758. Peripheral Reset Register 0 (RGM_PRST0)

Table 1702. RGM_PRST0 register field descriptions

Field	Description
1 PIT_0_RST	PIT_0 reset Writing a ‘1’ to this bit resets the PIT_0 peripheral. Writing a ‘0’ to this bit releases the reset to the PIT_0 peripheral if the bit’s current value is ‘1’, otherwise it has no effect. 0 No forced reset of peripheral 1 Forced reset of peripheral
16 SIUL_RST	SIUL reset Writing a ‘1’ to this bit resets the SIUL peripheral. Writing a ‘0’ to this bit releases the reset to the SIUL peripheral if the bit’s current value is ‘1’, otherwise it has no effect. 0 No forced reset of peripheral 1 Forced reset of peripheral
20 SIPI_0_RST	SIPI_0 reset Writing a ‘1’ to this bit resets the SIPI_0 peripheral. Writing a ‘0’ to this bit releases the reset to the SIPI_0 peripheral if the bit’s current value is ‘1’, otherwise it has no effect. 0 No forced reset of peripheral 1 Forced reset of peripheral
22 LFAST_0_RST	LFAST_0 reset Writing a ‘1’ to this bit resets the LFAST_0 peripheral. Writing a ‘0’ to this bit releases the reset to the LFAST_0 peripheral if the bit’s current value is ‘1’, otherwise it has no effect. 0 No forced reset of peripheral 1 Forced reset of peripheral

64.3.1.14 Peripheral Reset Register 1 (RGM_PRST1)

This register provides individual resets for various peripherals. It can be accessed in read/write in either supervisor mode or test mode. It can be accessed in read only in user mode. See *Table 1703* for details.

Address: 0x614 Access: User read, Supervisor read/write, Test read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	CCCU_RST	0	SD_ADC_0_RST	SD_ADC_2_RST	SD_ADC_4_RST	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	CRC_0_RST	0	DMAMUX_4_2_0_RST	0	0	PSI5S_0_RST	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1759. Peripheral Reset Register 1 (RGM_PRST1)

Table 1703. RGM_PRST1 register field descriptions

Field	Description
1 CCCU_RST	CCCU reset Writing a ‘1’ to this bit resets the CCCU peripheral. Writing a ‘0’ to this bit releases the reset to the CCCU peripheral if the bit’s current value is ‘1’, otherwise it has no effect. 0 No forced reset of peripheral 1 Forced reset of peripheral
3 SD_ADC_0_RST	SD_ADC_0 reset Writing a ‘1’ to this bit resets the SD_ADC_0 peripheral. Writing a ‘0’ to this bit releases the reset to the SD_ADC_0 peripheral if the bit’s current value is ‘1’, otherwise it has no effect. 0 No forced reset of peripheral 1 Forced reset of peripheral
4 SD_ADC_2_RST	SD_ADC_2 reset Writing a ‘1’ to this bit resets the SD_ADC_2 peripheral. Writing a ‘0’ to this bit releases the reset to the SD_ADC_2 peripheral if the bit’s current value is ‘1’, otherwise it has no effect. 0 No forced reset of peripheral 1 Forced reset of peripheral

Table 1703. RGM_PRST1 register field descriptions (continued)

Field	Description
5 SD_ADC_4_RST	SD_ADC_4 reset Writing a ‘1’ to this bit resets the SD_ADC_4 peripheral. Writing a ‘0’ to this bit releases the reset to the SD_ADC_4 peripheral if the bit’s current value is ‘1’, otherwise it has no effect. 0 No forced reset of peripheral 1 Forced reset of peripheral
25 CRC_0_RST	CRC_0 reset Writing a ‘1’ to this bit resets the CRC_0 peripheral. Writing a ‘0’ to this bit releases the reset to the CRC_0 peripheral if the bit’s current value is ‘1’, otherwise it has no effect. 0 No forced reset of peripheral 1 Forced reset of peripheral
27 DMAMUX_4_2_0_RST	DMAMUX_4_2_0 reset Writing a ‘1’ to this bit resets the DMAMUX_4_2_0 peripheral. Writing a ‘0’ to this bit releases the reset to the DMAMUX_4_2_0 peripheral if the bit’s current value is ‘1’, otherwise it has no effect. 0 No forced reset of peripheral 1 Forced reset of peripheral
30 PSI5S_0_RST	PSI5S_0 reset Writing a ‘1’ to this bit resets the PSI5S_0 peripheral. Writing a ‘0’ to this bit releases the reset to the PSI5S_0 peripheral if the bit’s current value is ‘1’, otherwise it has no effect. 0 No forced reset of peripheral 1 Forced reset of peripheral

64.3.1.15 Peripheral Reset Register 2 (RGM_PRST2)

This register provides individual resets for various peripherals. It can be accessed in read/write in either supervisor mode or test mode. It can be accessed in read only in user mode. See *Table 1704* for details.

Address: 0x618 Access: User read, Supervisor read/write, Test read/write

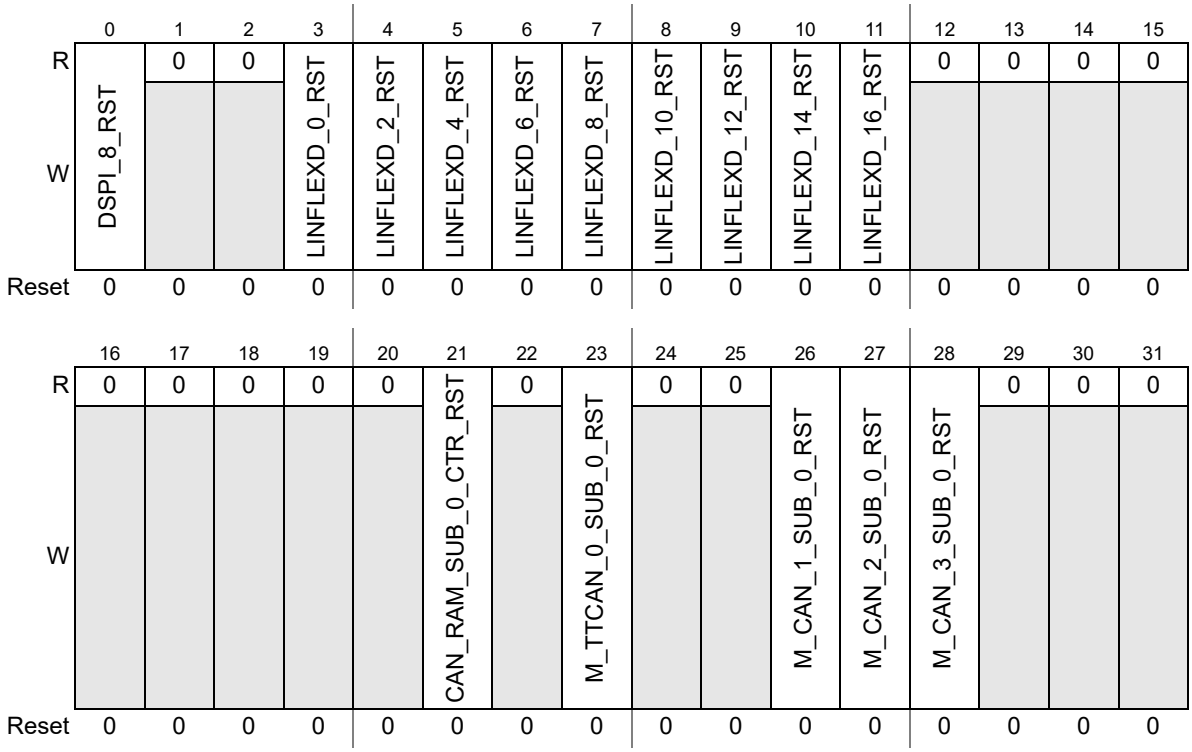


Figure 1760. Peripheral Reset Register 2 (RGM_PRST2)

Table 1704. RGM_PRST2 register field descriptions

Field	Description
0 DSPI_8_RST	DSPI_8 reset Writing a ‘1’ to this bit resets the DSPI_8 peripheral. Writing a ‘0’ to this bit releases the reset to the DSPI_8 peripheral if the bit’s current value is ‘1’, otherwise it has no effect. 0 No forced reset of peripheral 1 Forced reset of peripheral
3 LINFLEXD_0_RST	LINFLEXD_0 reset Writing a ‘1’ to this bit resets the LINFLEXD_0 peripheral. Writing a ‘0’ to this bit releases the reset to the LINFLEXD_0 peripheral if the bit’s current value is ‘1’, otherwise it has no effect. 0 No forced reset of peripheral 1 Forced reset of peripheral
4 LINFLEXD_2_RST	LINFLEXD_2 reset Writing a ‘1’ to this bit resets the LINFLEXD_2 peripheral. Writing a ‘0’ to this bit releases the reset to the LINFLEXD_2 peripheral if the bit’s current value is ‘1’, otherwise it has no effect. 0 No forced reset of peripheral 1 Forced reset of peripheral

Table 1704. RGM_PRST2 register field descriptions (continued)

Field	Description
5 LINFLEXD_4_RST	LINFLEXD_4 reset Writing a ‘1’ to this bit resets the LINFLEXD_4 peripheral. Writing a ‘0’ to this bit releases the reset to the LINFLEXD_4 peripheral if the bit’s current value is ‘1’, otherwise it has no effect. 0 No forced reset of peripheral 1 Forced reset of peripheral
6 LINFLEXD_6_RST	LINFLEXD_6 reset Writing a ‘1’ to this bit resets the LINFLEXD_6 peripheral. Writing a ‘0’ to this bit releases the reset to the LINFLEXD_6 peripheral if the bit’s current value is ‘1’, otherwise it has no effect. 0 No forced reset of peripheral 1 Forced reset of peripheral
7 LINFLEXD_8_RST	LINFLEXD_8 reset Writing a ‘1’ to this bit resets the LINFLEXD_8 peripheral. Writing a ‘0’ to this bit releases the reset to the LINFLEXD_8 peripheral if the bit’s current value is ‘1’, otherwise it has no effect. 0 No forced reset of peripheral 1 Forced reset of peripheral
8 LINFLEXD_10_RST	LINFLEXD_10 reset Writing a ‘1’ to this bit resets the LINFLEXD_10 peripheral. Writing a ‘0’ to this bit releases the reset to the LINFLEXD_10 peripheral if the bit’s current value is ‘1’, otherwise it has no effect. 0 No forced reset of peripheral 1 Forced reset of peripheral
9 LINFLEXD_12_RST	LINFLEXD_12 reset Writing a ‘1’ to this bit resets the LINFLEXD_12 peripheral. Writing a ‘0’ to this bit releases the reset to the LINFLEXD_12 peripheral if the bit’s current value is ‘1’, otherwise it has no effect. 0 No forced reset of peripheral 1 Forced reset of peripheral
10 LINFLEXD_14_RST	LINFLEXD_14 reset Writing a ‘1’ to this bit resets the LINFLEXD_14 peripheral. Writing a ‘0’ to this bit releases the reset to the LINFLEXD_14 peripheral if the bit’s current value is ‘1’, otherwise it has no effect. 0 No forced reset of peripheral 1 Forced reset of peripheral
11 LINFLEXD_16_RST	LINFLEXD_16 reset Writing a ‘1’ to this bit resets the LINFLEXD_16 peripheral. Writing a ‘0’ to this bit releases the reset to the LINFLEXD_16 peripheral if the bit’s current value is ‘1’, otherwise it has no effect. 0 No forced reset of peripheral 1 Forced reset of peripheral

Table 1704. RGM_PRST2 register field descriptions (continued)

Field	Description
21 CAN_RAM_SUB_0_CTR_RST	CAN_RAM_SUB_0_CTR reset It is not possible to access CAN RAM content if corresponding CAN RAM controller is kept under reset by CAN_RAM_SUB_0_CTR_RST bit Writing a '1' to this bit resets the CAN_RAM_SUB_0_CTR peripheral. Writing a '0' to this bit releases the reset to the CAN_RAM_SUB_0_CTR peripheral if the bit's current value is '1', otherwise it has no effect. 0 No forced reset of peripheral 1 Forced reset of peripheral
23 M_TTCAN_0_SUB_0_RST	M_TTCAN_0_SUB_0 reset Writing a '1' to this bit resets the M_TTCAN_0_SUB_0 peripheral. Writing a '0' to this bit releases the reset to the M_TTCAN_0_SUB_0 peripheral if the bit's current value is '1', otherwise it has no effect. 0 No forced reset of peripheral 1 Forced reset of peripheral
26 M_CAN_1_SUB_0_RST	M_CAN_1_SUB_0 reset Writing a '1' to this bit resets the M_CAN_1_SUB_0 peripheral. Writing a '0' to this bit releases the reset to the M_CAN_1_SUB_0 peripheral if the bit's current value is '1', otherwise it has no effect. 0 No forced reset of peripheral 1 Forced reset of peripheral
27 M_CAN_2_SUB_0_RST	M_CAN_2_SUB_0 reset Writing a '1' to this bit resets the M_CAN_2_SUB_0 peripheral. Writing a '0' to this bit releases the reset to the M_CAN_2_SUB_0 peripheral if the bit's current value is '1', otherwise it has no effect. 0 No forced reset of peripheral 1 Forced reset of peripheral
28 M_CAN_3_SUB_0_RST	M_CAN_3_SUB_0 reset Writing a '1' to this bit resets the M_CAN_3_SUB_0 peripheral. Writing a '0' to this bit releases the reset to the M_CAN_3_SUB_0 peripheral if the bit's current value is '1', otherwise it has no effect. 0 No forced reset of peripheral 1 Forced reset of peripheral

64.3.1.16 Peripheral Reset Register 3 (RGM_PRST3)

This register provides individual resets for various peripherals. It can be accessed in read/write in either supervisor mode or test mode. It can be accessed in read only in user mode. See *Table 1705* for details.

Address: 0x61C Access: User read, Supervisor read/write, Test read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	SAR_ADC_12BIT_0_RST	SAR_ADC_12BIT_2_RST	0	0	0	0	0	0	SAR_ADC_10BIT_0_RST	0	0	0	0	SAR_ADC_10BIT_STDBY_RST	0	SAR_ADC_12BIT_B_RST
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	PSI5_0_RST	0	0	0	FLEXRAY_0_RST	0	0	SENT_0_RST	0	0	IIC_0_RST	0	DSPI_0_RST	DSPI_2_RST	DSPI_4_RST	DSPI_6_RST
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1761. Peripheral Reset Register 3 (RGM_PRST3)

Table 1705. RGM_PRST3 register field descriptions

Field	Description
0 SAR_ADC_12BIT_0_RST	SAR_ADC_12BIT_0 reset Writing a ‘1’ to this bit resets the SAR_ADC_12BIT_0 peripheral. Writing a ‘0’ to this bit releases the reset to the SAR_ADC_12BIT_0 peripheral if the bit’s current value is ‘1’, otherwise it has no effect. **Note:** This peripheral is coupled with RGM_PRST4.SAR_ADC_12BIT_0_SEQ_RST therefore they have to be identically set 0 No forced reset of peripheral 1 Forced reset of peripheral
1 SAR_ADC_12BIT_2_RST	SAR_ADC_12BIT_2 reset Writing a ‘1’ to this bit resets the SAR_ADC_12BIT_2 peripheral. Writing a ‘0’ to this bit releases the reset to the SAR_ADC_12BIT_2 peripheral if the bit’s current value is ‘1’, otherwise it has no effect. **Note:** This peripheral is coupled with RGM_PRST4.SAR_ADC_12BIT_2_SEQ_RST therefore they have to be identically set 0 No forced reset of peripheral 1 Forced reset of peripheral

Table 1705. RGM_PRST3 register field descriptions (continued)

Field	Description
8 SAR_ADC_10BIT_0_RST	SAR_ADC_10BIT_0 reset Writing a '1' to this bit resets the SAR_ADC_10BIT_0 peripheral. Writing a '0' to this bit releases the reset to the SAR_ADC_10BIT_0 peripheral if the bit's current value is '1', otherwise it has no effect. **Note:** This peripheral is coupled with RGM_PRST4.SAR_ADC_10BIT_0_SEQ_RST therefore they have to be identically set 0 No forced reset of peripheral 1 Forced reset of peripheral
13 SAR_ADC_10BIT_STDBY_RST	SAR_ADC_10BIT_STDBY reset Writing a '1' to this bit resets the SAR_ADC_10BIT_STDBY peripheral. Writing a '0' to this bit releases the reset to the SAR_ADC_10BIT_STDBY peripheral if the bit's current value is '1', otherwise it has no effect. **Note:** This peripheral is coupled with RGM_PRST4.SAR_ADC_10BIT_STDBY_SEQ_RST therefore they have to be identically set 0 No forced reset of peripheral 1 Forced reset of peripheral
15 SAR_ADC_12BIT_B_RST	SAR_ADC_12BIT_B reset Writing a '1' to this bit resets the SAR_ADC_12BIT_B peripheral. Writing a '0' to this bit releases the reset to the SAR_ADC_12BIT_B peripheral if the bit's current value is '1', otherwise it has no effect. **Note:** This peripheral is coupled with RGM_PRST4.SAR_ADC_12BIT_B_SEQ_RST therefore they have to be identically set 0 No forced reset of peripheral 1 Forced reset of peripheral
16 PSI5_0_RST	PSI5_0 reset Writing a '1' to this bit resets the PSI5_0 peripheral. Writing a '0' to this bit releases the reset to the PSI5_0 peripheral if the bit's current value is '1', otherwise it has no effect. 0 No forced reset of peripheral 1 Forced reset of peripheral
20 FLEXRAY_0_RST	FLEXRAY_0 reset Writing a '1' to this bit resets the FLEXRAY_0 peripheral. Writing a '0' to this bit releases the reset to the FLEXRAY_0 peripheral if the bit's current value is '1', otherwise it has no effect. 0 No forced reset of peripheral 1 Forced reset of peripheral
23 SENT_0_RST	SENT_0 reset Writing a '1' to this bit resets the SENT_0 peripheral. Writing a '0' to this bit releases the reset to the SENT_0 peripheral if the bit's current value is '1', otherwise it has no effect. 0 No forced reset of peripheral 1 Forced reset of peripheral

Table 1705. RGM_PRST3 register field descriptions (continued)

Field	Description
26 IIC_0_RST	IIC_0 reset Writing a ‘1’ to this bit resets the IIC_0 peripheral. Writing a ‘0’ to this bit releases the reset to the IIC_0 peripheral if the bit’s current value is ‘1’, otherwise it has no effect. 0 No forced reset of peripheral 1 Forced reset of peripheral
28 DSPI_0_RST	DSPI_0 reset Writing a ‘1’ to this bit resets the DSPI_0 peripheral. Writing a ‘0’ to this bit releases the reset to the DSPI_0 peripheral if the bit’s current value is ‘1’, otherwise it has no effect. 0 No forced reset of peripheral 1 Forced reset of peripheral
29 DSPI_2_RST	DSPI_2 reset Writing a ‘1’ to this bit resets the DSPI_2 peripheral. Writing a ‘0’ to this bit releases the reset to the DSPI_2 peripheral if the bit’s current value is ‘1’, otherwise it has no effect. 0 No forced reset of peripheral 1 Forced reset of peripheral
30 DSPI_4_RST	DSPI_4 reset Writing a ‘1’ to this bit resets the DSPI_4 peripheral. Writing a ‘0’ to this bit releases the reset to the DSPI_4 peripheral if the bit’s current value is ‘1’, otherwise it has no effect. 0 No forced reset of peripheral 1 Forced reset of peripheral
31 DSPI_6_RST	DSPI_6 reset Writing a ‘1’ to this bit resets the DSPI_6 peripheral. Writing a ‘0’ to this bit releases the reset to the DSPI_6 peripheral if the bit’s current value is ‘1’, otherwise it has no effect. 0 No forced reset of peripheral 1 Forced reset of peripheral

64.3.1.17 Peripheral Reset Register 4 (RGM_PRST4)

This register provides individual resets for various peripherals. It can be accessed in read/write in either supervisor mode or test mode. It can be accessed in read only in user mode. See *Table 1706* for details.

Address: 0x620 Access: User read, Supervisor read/write, Test read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	PIT_1_RST	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R					0	0	0	0			0	0	0			
W	SAR_ADC_12BIT_0_SEQ_RST	SAR_ADC_12BIT_2_SEQ_RST	SAR_ADC_12BIT_1_SEQ_RST	SAR_ADC_12BIT_3_SEQ_RST					SAR_ADC_10BIT_0_SEQ_RST	SAR_ADC_10BIT_1_SEQ_RST				SAR_ADC_10BIT_STDBY_SEQ_RST	SAR_ADC_12BIT_B_SEQ_RST	GTMINT_RST
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1762. Peripheral Reset Register 4 (RGM_PRST4)

Table 1706. RGM_PRST4 register field descriptions

Field	Description
1 PIT_1_RST	PIT_1 reset Writing a ‘1’ to this bit resets the PIT_1 peripheral. Writing a ‘0’ to this bit releases the reset to the PIT_1 peripheral if the bit’s current value is ‘1’, otherwise it has no effect. 0 No forced reset of peripheral 1 Forced reset of peripheral
16 SAR_ADC_12BIT_0_SEQ_RST	SAR_ADC_12BIT_0_SEQ reset Writing a ‘1’ to this bit resets the SAR_ADC_12BIT_0_SEQ peripheral. Writing a ‘0’ to this bit releases the reset to the SAR_ADC_12BIT_0_SEQ peripheral if the bit’s current value is ‘1’, otherwise it has no effect. 0 No forced reset of peripheral 1 Forced reset of peripheral
17 SAR_ADC_12BIT_2_SEQ_RST	SAR_ADC_12BIT_2_SEQ reset Writing a ‘1’ to this bit resets the SAR_ADC_12BIT_2_SEQ peripheral. Writing a ‘0’ to this bit releases the reset to the SAR_ADC_12BIT_2_SEQ peripheral if the bit’s current value is ‘1’, otherwise it has no effect. 0 No forced reset of peripheral 1 Forced reset of peripheral

Table 1706. RGM_PRST4 register field descriptions (continued)

Field	Description
18 SAR_ADC_12BIT_1_SEQ_RST	SAR_ADC_12BIT_1_SEQ reset Writing a ‘1’ to this bit resets the SAR_ADC_12BIT_1_SEQ peripheral. Writing a ‘0’ to this bit releases the reset to the SAR_ADC_12BIT_1_SEQ peripheral if the bit’s current value is ‘1’, otherwise it has no effect. 0 No forced reset of peripheral 1 Forced reset of peripheral
19 SAR_ADC_12BIT_3_SEQ_RST	SAR_ADC_12BIT_3_SEQ reset Writing a ‘1’ to this bit resets the SAR_ADC_12BIT_3_SEQ peripheral. Writing a ‘0’ to this bit releases the reset to the SAR_ADC_12BIT_3_SEQ peripheral if the bit’s current value is ‘1’, otherwise it has no effect. 0 No forced reset of peripheral 1 Forced reset of peripheral
24 SAR_ADC_10BIT_0_SEQ_RST	SAR_ADC_10BIT_0_SEQ reset Writing a ‘1’ to this bit resets the SAR_ADC_10BIT_0_SEQ peripheral. Writing a ‘0’ to this bit releases the reset to the SAR_ADC_10BIT_0_SEQ peripheral if the bit’s current value is ‘1’, otherwise it has no effect. 0 No forced reset of peripheral 1 Forced reset of peripheral
25 SAR_ADC_10BIT_1_SEQ_RST	SAR_ADC_10BIT_1_SEQ reset Writing a ‘1’ to this bit resets the SAR_ADC_10BIT_1_SEQ peripheral. Writing a ‘0’ to this bit releases the reset to the SAR_ADC_10BIT_1_SEQ peripheral if the bit’s current value is ‘1’, otherwise it has no effect. 0 No forced reset of peripheral 1 Forced reset of peripheral
29 SAR_ADC_10BIT_STDBY_SEQ_RST	SAR_ADC_10BIT_STDBY_SEQ reset Writing a ‘1’ to this bit resets the SAR_ADC_10BIT_STDBY_SEQ peripheral. Writing a ‘0’ to this bit releases the reset to the SAR_ADC_10BIT_STDBY_SEQ peripheral if the bit’s current value is ‘1’, otherwise it has no effect. 0 No forced reset of peripheral 1 Forced reset of peripheral
30 SAR_ADC_12BIT_B_SEQ_RST	SAR_ADC_12BIT_B_SEQ reset Writing a ‘1’ to this bit resets the SAR_ADC_12BIT_B_SEQ peripheral. Writing a ‘0’ to this bit releases the reset to the SAR_ADC_12BIT_B_SEQ peripheral if the bit’s current value is ‘1’, otherwise it has no effect. 0 No forced reset of peripheral 1 Forced reset of peripheral
31 GTMINT_RST	GTMINT reset Writing a ‘1’ to this bit resets the GTMINT peripheral. Writing a ‘0’ to this bit releases the reset to the GTMINT peripheral if the bit’s current value is ‘1’, otherwise it has no effect. 0 No forced reset of peripheral 1 Forced reset of peripheral

64.3.1.18 Peripheral Reset Register 5 (RGM_PRST5)

This register provides individual resets for various peripherals. It can be accessed in read/write in either supervisor mode or test mode. It can be accessed in read only in user mode. See *Table 1707* for details.

Address: 0x624 Access: User read, Supervisor read/write, Test read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	SD_ADC_1_RST	SD_ADC_3_RST	SD_ADC_5_RST	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	FCCU_RST	0	0	CRC_1_RST	0	DMAMUX_5_3_1_RST	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1763. Peripheral Reset Register 5 (RGM_PRST5)

Table 1707. RGM_PRST5 register field descriptions

Field	Description
3 SD_ADC_1_RST	SD_ADC_1 reset Writing a ‘1’ to this bit resets the SD_ADC_1 peripheral. Writing a ‘0’ to this bit releases the reset to the SD_ADC_1 peripheral if the bit’s current value is ‘1’, otherwise it has no effect. 0 No forced reset of peripheral 1 Forced reset of peripheral
4 SD_ADC_3_RST	SD_ADC_3 reset Writing a ‘1’ to this bit resets the SD_ADC_3 peripheral. Writing a ‘0’ to this bit releases the reset to the SD_ADC_3 peripheral if the bit’s current value is ‘1’, otherwise it has no effect. 0 No forced reset of peripheral 1 Forced reset of peripheral
5 SD_ADC_5_RST	SD_ADC_5 reset Writing a ‘1’ to this bit resets the SD_ADC_5 peripheral. Writing a ‘0’ to this bit releases the reset to the SD_ADC_5 peripheral if the bit’s current value is ‘1’, otherwise it has no effect. 0 No forced reset of peripheral 1 Forced reset of peripheral

Table 1707. RGM_PRST5 register field descriptions (continued)

Field	Description
22 FCCU_RST	FCCU reset Writing a ‘1’ to this bit resets the FCCU peripheral. Writing a ‘0’ to this bit releases the reset to the FCCU peripheral if the bit’s current value is ‘1’, otherwise it has no effect. 0 No forced reset of peripheral 1 Forced reset of peripheral
25 CRC_1_RST	CRC_1 reset Writing a ‘1’ to this bit resets the CRC_1 peripheral. Writing a ‘0’ to this bit releases the reset to the CRC_1 peripheral if the bit’s current value is ‘1’, otherwise it has no effect. 0 No forced reset of peripheral 1 Forced reset of peripheral
27 DMAMUX_5_3_1_RST	DMAMUX_5_3_1 reset Writing a ‘1’ to this bit resets the DMAMUX_5_3_1 peripheral. Writing a ‘0’ to this bit releases the reset to the DMAMUX_5_3_1 peripheral if the bit’s current value is ‘1’, otherwise it has no effect. 0 No forced reset of peripheral 1 Forced reset of peripheral

64.3.1.19 Peripheral Reset Register 6 (RGM_PRST6)

This register provides individual resets for various peripherals. It can be accessed in read/write in either supervisor mode or test mode. It can be accessed in read only in user mode. See *Table 1708* for details.

Address: 0x628 Access: User read, Supervisor read/write, Test read/write

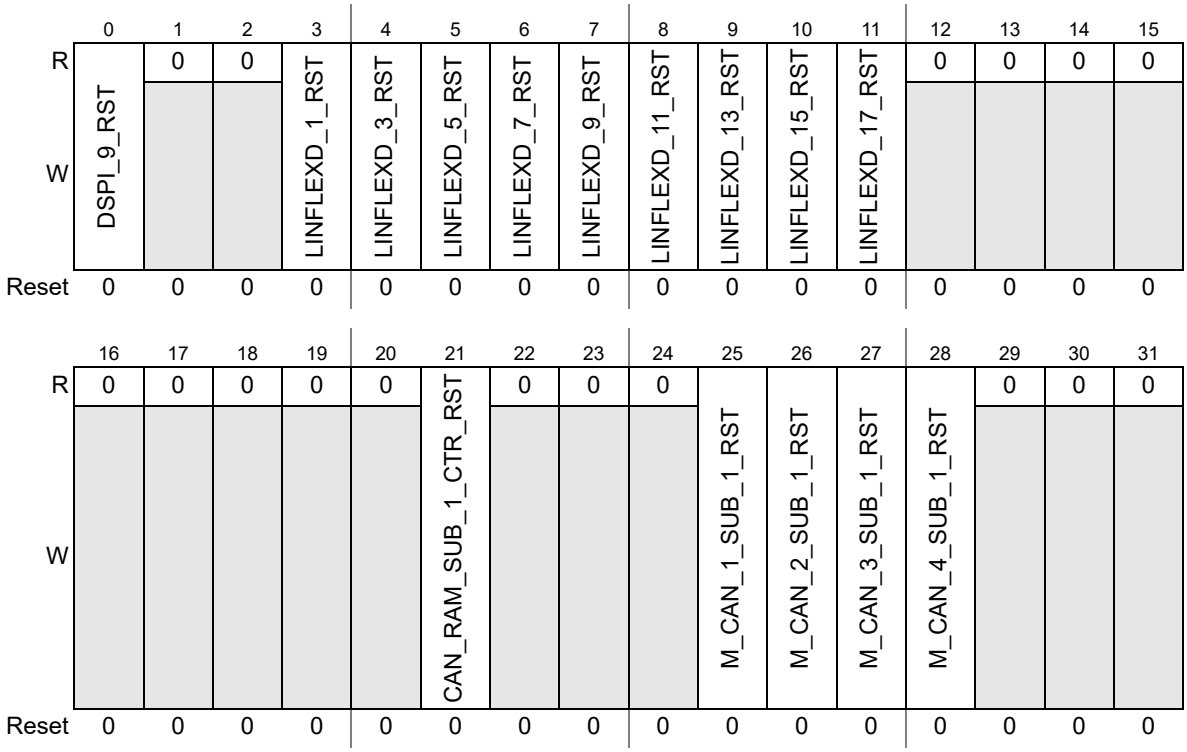


Figure 1764. Peripheral Reset Register 6 (RGM_PRST6)

Table 1708. RGM_PRST6 register field descriptions

Field	Description
0 DSPI_9_RST	DSPI_9 reset Writing a ‘1’ to this bit resets the DSPI_9 peripheral. Writing a ‘0’ to this bit releases the reset to the DSPI_9 peripheral if the bit’s current value is ‘1’, otherwise it has no effect. 0 No forced reset of peripheral 1 Forced reset of peripheral
3 LINFLEXD_1_RST	LINFLEXD_1 reset Writing a ‘1’ to this bit resets the LINFLEXD_1 peripheral. Writing a ‘0’ to this bit releases the reset to the LINFLEXD_1 peripheral if the bit’s current value is ‘1’, otherwise it has no effect. 0 No forced reset of peripheral 1 Forced reset of peripheral
4 LINFLEXD_3_RST	LINFLEXD_3 reset Writing a ‘1’ to this bit resets the LINFLEXD_3 peripheral. Writing a ‘0’ to this bit releases the reset to the LINFLEXD_3 peripheral if the bit’s current value is ‘1’, otherwise it has no effect. 0 No forced reset of peripheral 1 Forced reset of peripheral

Table 1708. RGM_PRST6 register field descriptions (continued)

Field	Description
5 LINFLEXD_5_RST	LINFLEXD_5 reset Writing a ‘1’ to this bit resets the LINFLEXD_5 peripheral. Writing a ‘0’ to this bit releases the reset to the LINFLEXD_5 peripheral if the bit’s current value is ‘1’, otherwise it has no effect. 0 No forced reset of peripheral 1 Forced reset of peripheral
6 LINFLEXD_7_RST	LINFLEXD_7 reset Writing a ‘1’ to this bit resets the LINFLEXD_7 peripheral. Writing a ‘0’ to this bit releases the reset to the LINFLEXD_7 peripheral if the bit’s current value is ‘1’, otherwise it has no effect. 0 No forced reset of peripheral 1 Forced reset of peripheral
7 LINFLEXD_9_RST	LINFLEXD_9 reset Writing a ‘1’ to this bit resets the LINFLEXD_9 peripheral. Writing a ‘0’ to this bit releases the reset to the LINFLEXD_9 peripheral if the bit’s current value is ‘1’, otherwise it has no effect. 0 No forced reset of peripheral 1 Forced reset of peripheral
8 LINFLEXD_11_RST	LINFLEXD_11 reset Writing a ‘1’ to this bit resets the LINFLEXD_11 peripheral. Writing a ‘0’ to this bit releases the reset to the LINFLEXD_11 peripheral if the bit’s current value is ‘1’, otherwise it has no effect. 0 No forced reset of peripheral 1 Forced reset of peripheral
9 LINFLEXD_13_RST	LINFLEXD_13 reset Writing a ‘1’ to this bit resets the LINFLEXD_13 peripheral. Writing a ‘0’ to this bit releases the reset to the LINFLEXD_13 peripheral if the bit’s current value is ‘1’, otherwise it has no effect. 0 No forced reset of peripheral 1 Forced reset of peripheral
10 LINFLEXD_15_RST	LINFLEXD_15 reset Writing a ‘1’ to this bit resets the LINFLEXD_15 peripheral. Writing a ‘0’ to this bit releases the reset to the LINFLEXD_15 peripheral if the bit’s current value is ‘1’, otherwise it has no effect. 0 No forced reset of peripheral 1 Forced reset of peripheral
11 LINFLEXD_17_RST	LINFLEXD_17 reset Writing a ‘1’ to this bit resets the LINFLEXD_17 peripheral. Writing a ‘0’ to this bit releases the reset to the LINFLEXD_17 peripheral if the bit’s current value is ‘1’, otherwise it has no effect. 0 No forced reset of peripheral 1 Forced reset of peripheral

Table 1708. RGM_PRST6 register field descriptions (continued)

Field	Description
21 CAN_RAM_SUB_1_CTR_RST	CAN_RAM_SUB_1_CTR reset It is not possible to access CAN RAM content if corresponding CAN RAM controller is kept under reset by CAN_RAM_SUB_1_CTR_RST bit Writing a '1' to this bit resets the CAN_RAM_SUB_1_CTR peripheral. Writing a '0' to this bit releases the reset to the CAN_RAM_SUB_1_CTR peripheral if the bit's current value is '1', otherwise it has no effect. 0 No forced reset of peripheral 1 Forced reset of peripheral
25 M_CAN_1_SUB_1_RST	M_CAN_1_SUB_1 reset Writing a '1' to this bit resets the M_CAN_1_SUB_1 peripheral. Writing a '0' to this bit releases the reset to the M_CAN_1_SUB_1 peripheral if the bit's current value is '1', otherwise it has no effect. 0 No forced reset of peripheral 1 Forced reset of peripheral
26 M_CAN_2_SUB_1_RST	M_CAN_2_SUB_1 reset Writing a '1' to this bit resets the M_CAN_2_SUB_1 peripheral. Writing a '0' to this bit releases the reset to the M_CAN_1_SUB_0 peripheral if the bit's current value is '1', otherwise it has no effect. 0 No forced reset of peripheral 1 Forced reset of peripheral
27 M_CAN_3_SUB_1_RST	M_CAN_3_SUB_1 reset Writing a '1' to this bit resets the M_CAN_3_SUB_1 peripheral. Writing a '0' to this bit releases the reset to the M_CAN_3_SUB_1 peripheral if the bit's current value is '1', otherwise it has no effect. 0 No forced reset of peripheral 1 Forced reset of peripheral
28 M_CAN_4_SUB_1_RST	M_CAN_4_SUB_1 reset Writing a '1' to this bit resets the M_CAN_4_SUB_1 peripheral. Writing a '0' to this bit releases the reset to the M_CAN_4_SUB_1 peripheral if the bit's current value is '1', otherwise it has no effect. 0 No forced reset of peripheral 1 Forced reset of peripheral

64.3.1.20 Peripheral Reset Register 7 (RGM_PRST7)

This register provides individual resets for various peripherals. It can be accessed in read/write in either supervisor mode or test mode. It can be accessed in read only in user mode. See *Table 1709* for details.

Address: 0x62C Access: User read, Supervisor read/write, Test read/write

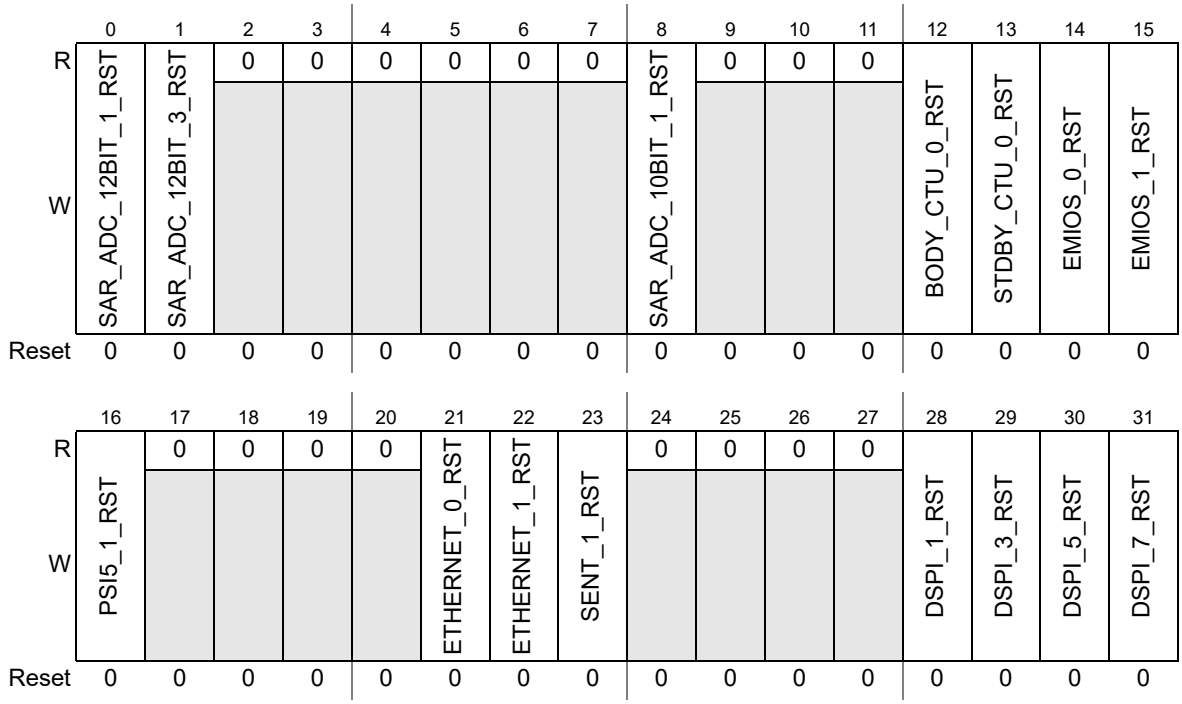


Figure 1765. Peripheral Reset Register 7 (RGM_PRST7)

Table 1709. RGM_PRST7 register field descriptions

Field	Description
0 SAR_ADC_12BIT_1_RST	SAR_ADC_12BIT_1 reset Writing a ‘1’ to this bit resets the SAR_ADC_12BIT_1 peripheral. Writing a ‘0’ to this bit releases the reset to the SAR_ADC_12BIT_1 peripheral if the bit’s current value is ‘1’, otherwise it has no effect. **Note:** This peripheral is coupled with RGM_PRST4.SAR_ADC_12BIT_1_SEQ_RST therefore they have to be identically set 0 No forced reset of peripheral 1 Forced reset of peripheral
1 SAR_ADC_12BIT_3_RST	SAR_ADC_12BIT_3 reset Writing a ‘1’ to this bit resets the SAR_ADC_12BIT_3 peripheral. Writing a ‘0’ to this bit releases the reset to the SAR_ADC_12BIT_3 peripheral if the bit’s current value is ‘1’, otherwise it has no effect. **Note:** This peripheral is coupled with RGM_PRST4.SAR_ADC_12BIT_3_SEQ_RST therefore they have to be identically set 0 No forced reset of peripheral 1 Forced reset of peripheral

Table 1709. RGM_PRST7 register field descriptions (continued)

Field	Description
8 SAR_ADC_10BIT_1_RST	SAR_ADC_10BIT_1 reset Writing a ‘1’ to this bit resets the SAR_ADC_10BIT_1 peripheral. Writing a ‘0’ to this bit releases the reset to the SAR_ADC_10BIT_1 peripheral if the bit’s current value is ‘1’, otherwise it has no effect. **Note:** This peripheral is coupled with RGM_PRST4.SAR_ADC_10BIT_1_SEQ_RST therefore they have to be identically set 0 No forced reset of peripheral 1 Forced reset of peripheral
12 BODY_CTU_0_RST	BODY_CTU_0 reset Writing a ‘1’ to this bit resets the BODY_CTU_0 peripheral. Writing a ‘0’ to this bit releases the reset to the BODY_CTU_0 peripheral if the bit’s current value is ‘1’, otherwise it has no effect. 0 No forced reset of peripheral 1 Forced reset of peripheral
13 STDBY_CTU_0_RST	STDBY_CTU_0 reset Writing a ‘1’ to this bit resets the STDBY_CTU_0 peripheral. Writing a ‘0’ to this bit releases the reset to the STDBY_CTU_0 peripheral if the bit’s current value is ‘1’, otherwise it has no effect. 0 No forced reset of peripheral 1 Forced reset of peripheral
14 EMIOS_0_RST	EMIOS_0 reset Writing a ‘1’ to this bit resets the EMIOS_0 peripheral. Writing a ‘0’ to this bit releases the reset to the EMIOS_0 peripheral if the bit’s current value is ‘1’, otherwise it has no effect. 0 No forced reset of peripheral 1 Forced reset of peripheral
15 EMIOS_1_RST	EMIOS_1 reset Writing a ‘1’ to this bit resets the EMIOS_1 peripheral. Writing a ‘0’ to this bit releases the reset to the EMIOS_1 peripheral if the bit’s current value is ‘1’, otherwise it has no effect. 0 No forced reset of peripheral 1 Forced reset of peripheral
16 PSI5_1_RST	PSI5_1 reset Writing a ‘1’ to this bit resets the PSI5_1 peripheral. Writing a ‘0’ to this bit releases the reset to the PSI5_1 peripheral if the bit’s current value is ‘1’, otherwise it has no effect. 0 No forced reset of peripheral 1 Forced reset of peripheral
21 ETHERNET_0_RST	ETHERNET_0 reset Writing a ‘1’ to this bit resets the ETHERNET_0 peripheral. Writing a ‘0’ to this bit releases the reset to the ETHERNET_0 peripheral if the bit’s current value is ‘1’, otherwise it has no effect. 0 No forced reset of peripheral 1 Forced reset of peripheral

Table 1709. RGM_PRST7 register field descriptions (continued)

Field	Description
22 ETHERNET_1_RST	ETHERNET_1 reset Writing a ‘1’ to this bit resets the ETHERNET_1 peripheral. Writing a ‘0’ to this bit releases the reset to the ETHERNET_1 peripheral if the bit’s current value is ‘1’, otherwise it has no effect. 0 No forced reset of peripheral 1 Forced reset of peripheral
23 SENT_1_RST	SENT_1 reset Writing a ‘1’ to this bit resets the SENT_1 peripheral. Writing a ‘0’ to this bit releases the reset to the SENT_1 peripheral if the bit’s current value is ‘1’, otherwise it has no effect. 0 No forced reset of peripheral 1 Forced reset of peripheral
28 DSPI_1_RST	DSPI_1 reset Writing a ‘1’ to this bit resets the DSPI_1 peripheral. Writing a ‘0’ to this bit releases the reset to the DSPI_1 peripheral if the bit’s current value is ‘1’, otherwise it has no effect. 0 No forced reset of peripheral 1 Forced reset of peripheral
29 DSPI_3_RST	DSPI_3 reset Writing a ‘1’ to this bit resets the DSPI_3 peripheral. Writing a ‘0’ to this bit releases the reset to the DSPI_3 peripheral if the bit’s current value is ‘1’, otherwise it has no effect. 0 No forced reset of peripheral 1 Forced reset of peripheral
30 DSPI_5_RST	DSPI_5 reset Writing a ‘1’ to this bit resets the DSPI_5 peripheral. Writing a ‘0’ to this bit releases the reset to the DSPI_5 peripheral if the bit’s current value is ‘1’, otherwise it has no effect. 0 No forced reset of peripheral 1 Forced reset of peripheral
31 DSPI_7_RST	DSPI_7 reset Writing a ‘1’ to this bit resets the DSPI_7 peripheral. Writing a ‘0’ to this bit releases the reset to the DSPI_7 peripheral if the bit’s current value is ‘1’, otherwise it has no effect. 0 No forced reset of peripheral 1 Forced reset of peripheral

Caution: Use the RGM_PRST*n* registers with care. See *Section 64.4.8: Individual peripheral resets* for the proper sequence to prevent unexpected chip behavior.

64.3.1.21 Peripheral Reset Status Register 0 (RGM_PSTAT0)

This register provides the current reset status for various peripherals. It can be accessed in read only in all modes. See *Table 1710* for details.

Address: 0x630 Access: User read/write, Supervisor read, Test read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	PIT_0_STAT	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	SIUL_STAT	0	0	0	SIPI_0_STAT	0	LFAST_0_STAT	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1766. Peripheral Reset Status Register 0 (RGM_PSTAT0)

Table 1710. RGM_PSTAT0 register field descriptions

Field	Description
1 PIT_0_STAT	PIT_0 reset status 0 peripheral is not in reset 1 peripheral is in reset
16 SIUL_STAT	SIUL reset status 0 peripheral is not in reset 1 peripheral is in reset
20 SIPI_0_STAT	SIPI_0 reset status 0 peripheral is not in reset 1 peripheral is in reset
22 LFAST_0_STAT	LFAST_0 reset status 0 peripheral is not in reset 1 peripheral is in reset

64.3.1.22 Peripheral Reset Status Register 1 (RGM_PSTAT1)

This register provides the current reset status for various peripherals. It can be accessed in read only in all modes. See *Table 1711* for details.

Address: 0x634 Access: User read/write, Supervisor read, Test read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	CCCU_STAT	0	SD_ADC_0_STAT	SD_ADC_2_STAT	SD_ADC_4_STAT	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	CRC_0_STAT	0	DMAMUX_4_2_0_STAT	0	0	PSI5S_0_STAT	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1767. Peripheral Reset Status Register 1 (RGM_PSTAT1)

Table 1711. RGM_PSTAT1 register field descriptions

Field	Description
1 CCCU_STAT	CCCU reset status 0 peripheral is not in reset 1 peripheral is in reset
3 SD_ADC_0_STAT	SD_ADC_0 reset status 0 peripheral is not in reset 1 peripheral is in reset
4 SD_ADC_2_STAT	SD_ADC_2 reset status 0 peripheral is not in reset 1 peripheral is in reset
5 SD_ADC_4_STAT	SD_ADC_4 reset status 0 peripheral is not in reset 1 peripheral is in reset
25 CRC_0_STAT	CRC_0 reset status 0 peripheral is not in reset 1 peripheral is in reset

Table 1711. RGM_PSTAT1 register field descriptions (continued)

Field	Description
27 DMAMUX_4_2_0_STAT	DMAMUX_4_2_0 reset status 0 peripheral is not in reset 1 peripheral is in reset
30 PSI5S_0_STAT	PSI5S_0 reset status 0 peripheral is not in reset 1 peripheral is in reset

64.3.1.23 Peripheral Reset Status Register 2 (RGM_PSTAT2)

This register provides the current reset status for various peripherals. It can be accessed in read only in all modes. See *Table 1712* for details.

Address: 0x638 Access: User read/write, Supervisor read, Test read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	DSPI_8_STAT	0	0	LINFLEXD_0_STAT	LINFLEXD_2_STAT	LINFLEXD_4_STAT	LINFLEXD_6_STAT	LINFLEXD_8_STAT	LINFLEXD_10_STAT	LINFLEXD_12_STAT	LINFLEXD_14_STAT	LINFLEXD_16_STAT	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	CAN_RAM_SUB_0_CTR_STAT	0	M_TTCAN_0_SUB_0_STAT	0	0	M_CAN_1_SUB_0_STAT	M_CAN_2_SUB_0_STAT	M_CAN_3_SUB_0_STAT	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1768. Peripheral Reset Status Register 2 (RGM_PSTAT2)

Table 1712. RGM_PSTAT2 register field descriptions

Field	Description
0 DSPI_8_STAT	DSPI_8 reset status 0 peripheral is not in reset 1 peripheral is in reset
3 LINFLEXD_0_STAT	LINFLEXD_0 reset status 0 peripheral is not in reset 1 peripheral is in reset
4 LINFLEXD_2_STAT	LINFLEXD_2 reset status 0 peripheral is not in reset 1 peripheral is in reset
5 LINFLEXD_4_STAT	LINFLEXD_4 reset status 0 peripheral is not in reset 1 peripheral is in reset
6 LINFLEXD_6_STAT	LINFLEXD_6 reset status 0 peripheral is not in reset 1 peripheral is in reset
7 LINFLEXD_8_STAT	LINFLEXD_8 reset status 0 peripheral is not in reset 1 peripheral is in reset
8 LINFLEXD_10_STAT	LINFLEXD_10 reset status 0 peripheral is not in reset 1 peripheral is in reset
9 LINFLEXD_12_STAT	LINFLEXD_12 reset status 0 peripheral is not in reset 1 peripheral is in reset
10 LINFLEXD_14_STAT	LINFLEXD_14 reset status 0 peripheral is not in reset 1 peripheral is in reset
11 LINFLEXD_16_STAT	LINFLEXD_16 reset status 0 peripheral is not in reset 1 peripheral is in reset
21 CAN_RAM_SUB_0_CTR_STAT	CAN_RAM_SUB_0_CTR reset status 0 peripheral is not in reset 1 peripheral is in reset
23 M_TTCAN_0_SUB_0_STAT	M_TTCAN_0_SUB_0 reset status 0 peripheral is not in reset 1 peripheral is in reset
26 M_CAN_1_SUB_0_STAT	M_CAN_1_SUB_0 reset status 0 peripheral is not in reset 1 peripheral is in reset

Table 1712. RGM_PSTAT2 register field descriptions (continued)

Field	Description
27 M_CAN_2_SUB_0_STAT	M_CAN_2_SUB_0 reset status 0 peripheral is not in reset 1 peripheral is in reset
28 M_CAN_3_SUB_0_STAT	M_CAN_3_SUB_0 reset status 0 peripheral is not in reset 1 peripheral is in reset

64.3.1.24 Peripheral Reset Status Register 3 (RGM_PSTAT3)

This register provides the current reset status for various peripherals. It can be accessed in read only in all modes. See *Table 1713* for details.

Address: 0x63C Access: User read/write, Supervisor read, Test read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	SAR_ADC_12BIT_0_STAT	SAR_ADC_12BIT_2_STAT	0	0	0	0	0	0	SAR_ADC_10BIT_0_STAT	0	0	0	0	SAR_ADC_10BIT_STDBY_STAT	0	SAR_ADC_12BIT_B_STAT
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	PSI5_0_STAT	0	0	0	FLEXRAY_0_STAT	0	0	SENT_0_STAT	0	0	IIC_0_STAT	0	DSPI_0_STAT	DSPI_2_STAT	DSPI_4_STAT	DSPI_6_STAT
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1769. Peripheral Reset Status Register 3 (RGM_PSTAT3)

Table 1713. RGM_PSTAT3 register field descriptions

Field	Description
0 SAR_ADC_12BIT_0_STAT	SAR_ADC_12BIT_0 reset status 0 peripheral is not in reset 1 peripheral is in reset
1 SAR_ADC_12BIT_2_STAT	SAR_ADC_12BIT_2 reset status 0 peripheral is not in reset 1 peripheral is in reset
8 SAR_ADC_10BIT_0_STAT	SAR_ADC_10BIT_0 reset status 0 peripheral is not in reset 1 peripheral is in reset
13 SAR_ADC_10BIT_STDBY_STAT	SAR_ADC_10BIT_STDBY reset status 0 peripheral is not in reset 1 peripheral is in reset
15 SAR_ADC_12BIT_B_STAT	SAR_ADC_12BIT_B reset status 0 peripheral is not in reset 1 peripheral is in reset
16 PSI5_0_STAT	PSI5_0 reset status 0 peripheral is not in reset 1 peripheral is in reset
20 FLEXRAY_0_STAT	FLEXRAY_0 reset status 0 peripheral is not in reset 1 peripheral is in reset
23 SENT_0_STAT	SENT_0 reset status 0 peripheral is not in reset 1 peripheral is in reset
26 IIC_0_STAT	IIC_0 reset status 0 peripheral is not in reset 1 peripheral is in reset
28 DSPI_0_STAT	DSPI_0 reset status 0 peripheral is not in reset 1 peripheral is in reset
29 DSPI_2_STAT	DSPI_2 reset status 0 peripheral is not in reset 1 peripheral is in reset
30 DSPI_4_STAT	DSPI_4 reset status 0 peripheral is not in reset 1 peripheral is in reset
31 DSPI_6_STAT	DSPI_6 reset status 0 peripheral is not in reset 1 peripheral is in reset

64.3.1.25 Peripheral Reset Status Register 4 (RGM_PSTAT4)

This register provides the current reset status for various peripherals. It can be accessed in read only in all modes. See *Table 1714* for details.

Address: 0x640 Access: User read/write, Supervisor read, Test read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	PIT_1_STAT	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	SAR_ADC_12BIT_0_SEQ_STAT	SAR_ADC_12BIT_2_SEQ_STAT	SAR_ADC_12BIT_1_SEQ_STAT	SAR_ADC_12BIT_3_SEQ_STAT	0	0	0	0	SAR_ADC_10BIT_0_SEQ_STAT	SAR_ADC_10BIT_1_SEQ_STAT	0	0	0	SAR_ADC_10BIT_STDBY_SEQ_STAT	SAR_ADC_12BIT_B_SEQ_STAT	GTMINT_STAT
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1770. Peripheral Reset Status Register 4 (RGM_PSTAT4)

Table 1714. RGM_PSTAT4 register field descriptions

Field	Description
1 PIT_1_STAT	PIT_1 reset status 0 peripheral is not in reset 1 peripheral is in reset
16 SAR_ADC_12BIT_0_SEQ_STAT	SAR_ADC_12BIT_0_SEQ reset status 0 peripheral is not in reset 1 peripheral is in reset
17 SAR_ADC_12BIT_2_SEQ_STAT	SAR_ADC_12BIT_2_SEQ reset status 0 peripheral is not in reset 1 peripheral is in reset
18 SAR_ADC_12BIT_1_SEQ_STAT	SAR_ADC_12BIT_1_SEQ reset status 0 peripheral is not in reset 1 peripheral is in reset

Table 1714. RGM_PSTAT4 register field descriptions (continued)

Field	Description
19 SAR_ADC_12BIT_3_SEQ_STAT	SAR_ADC_12BIT_3_SEQ reset status 0 peripheral is not in reset 1 peripheral is in reset
24 SAR_ADC_10BIT_0_SEQ_STAT	SAR_ADC_10BIT_0_SEQ reset status 0 peripheral is not in reset 1 peripheral is in reset
25 SAR_ADC_10BIT_1_SEQ_STAT	SAR_ADC_10BIT_1_SEQ reset status 0 peripheral is not in reset 1 peripheral is in reset
29 SAR_ADC_10BIT_STDBY_SEQ_STAT	SAR_ADC_10BIT_STDBY_SEQ reset status 0 peripheral is not in reset 1 peripheral is in reset
30 SAR_ADC_12BIT_B_SEQ_STAT	SAR_ADC_12BIT_B_SEQ reset status 0 peripheral is not in reset 1 peripheral is in reset
31 GTMINT_STAT	GTMINT reset status 0 peripheral is not in reset 1 peripheral is in reset

64.3.1.26 Peripheral Reset Status Register 5 (RGM_PSTAT5)

This register provides the current reset status for various peripherals. It can be accessed in read only in all modes. See *Table 1715* for details.

Address: 0x644 Access: User read/write, Supervisor read, Test read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	SD_ADC_1_STAT	SD_ADC_3_STAT	SD_ADC_5_STAT	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	FCCU_STAT	0	0	CRC_1_STAT	0	DMAMUX_5_3_1_STAT	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1771. Peripheral Reset Status Register 5 (RGM_PSTAT5)

Table 1715. RGM_PSTAT5 register field descriptions

Field	Description
3 SD_ADC_1_STAT	SD_ADC_1 reset status 0 peripheral is not in reset 1 peripheral is in reset
4 SD_ADC_3_STAT	SD_ADC_3 reset status 0 peripheral is not in reset 1 peripheral is in reset
5 SD_ADC_5_STAT	SD_ADC_5 reset status 0 peripheral is not in reset 1 peripheral is in reset
22 FCCU_STAT	FCCU reset status 0 peripheral is not in reset 1 peripheral is in reset
25 CRC_1_STAT	CRC_1 reset status 0 peripheral is not in reset 1 peripheral is in reset
27 DMAMUX_5_3_1_STAT	DMAMUX_5_3_1 reset status 0 peripheral is not in reset 1 peripheral is in reset

64.3.1.27 Peripheral Reset Status Register 6 (RGM_PSTAT6)

This register provides the current reset status for various peripherals. It can be accessed in read only in all modes. See *Table 1716* for details.

Address: 0x648 Access: User read/write, Supervisor read, Test read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	DSPI_9_STAT	0	0	LINFLEXD_1_STAT	LINFLEXD_3_STAT	LINFLEXD_5_STAT	LINFLEXD_7_STAT	LINFLEXD_9_STAT	LINFLEXD_11_STAT	LINFLEXD_13_STAT	LINFLEXD_15_STAT	LINFLEXD_17_STAT	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	CAN_RAM_SUB_1_CTR_STAT	0	0	0	M_CAN_1_SUB_1_STAT	M_CAN_2_SUB_1_STAT	M_CAN_3_SUB_1_STAT	M_CAN_4_SUB_1_STAT	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1772. Peripheral Reset Status Register 6 (RGM_PSTAT6)

Table 1716. RGM_PSTAT6 register field descriptions

Field	Description
0 DSPI_9_STAT	DSPI_9 reset status 0 peripheral is not in reset 1 peripheral is in reset
3 LINFLEXD_1_STAT	LINFLEXD_1 reset status 0 peripheral is not in reset 1 peripheral is in reset
4 LINFLEXD_3_STAT	LINFLEXD_3 reset status 0 peripheral is not in reset 1 peripheral is in reset
5 LINFLEXD_5_STAT	LINFLEXD_5 reset status 0 peripheral is not in reset 1 peripheral is in reset

Table 1716. RGM_PSTAT6 register field descriptions (continued)

Field	Description
6 LINFLEXD_7_STAT	LINFLEXD_7 reset status 0 peripheral is not in reset 1 peripheral is in reset
7 LINFLEXD_9_STAT	LINFLEXD_9 reset status 0 peripheral is not in reset 1 peripheral is in reset
8 LINFLEXD_11_STAT	LINFLEXD_11 reset status 0 peripheral is not in reset 1 peripheral is in reset
9 LINFLEXD_13_STAT	LINFLEXD_13 reset status 0 peripheral is not in reset 1 peripheral is in reset
10 LINFLEXD_15_STAT	LINFLEXD_15 reset status 0 peripheral is not in reset 1 peripheral is in reset
11 LINFLEXD_17_STAT	LINFLEXD_17 reset status 0 peripheral is not in reset 1 peripheral is in reset
21 CAN_RAM_SUB_1_CTR_STAT	CAN_RAM_SUB_1_CTR reset status 0 peripheral is not in reset 1 peripheral is in reset
25 M_CAN_1_SUB_1_STAT	M_CAN_1_SUB_1 reset status 0 peripheral is not in reset 1 peripheral is in reset
26 M_CAN_2_SUB_1_STAT	M_CAN_2_SUB_1 reset status 0 peripheral is not in reset 1 peripheral is in reset
27 M_CAN_3_SUB_1_STAT	M_CAN_3_SUB_1 reset status 0 peripheral is not in reset 1 peripheral is in reset
28 M_CAN_4_SUB_1_STAT	M_CAN_4_SUB_1 reset status 0 peripheral is not in reset 1 peripheral is in reset

64.3.1.28 Peripheral Reset Status Register 7 (RGM_PSTAT7)

This register provides the current reset status for various peripherals. It can be accessed in read only in all modes. See *Table 1717* for details.

Address: 0x64C Access: User read/write, Supervisor read, Test read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	SAR_ADC_12BIT_1_STAT	SAR_ADC_12BIT_3_STAT	0	0	0	0	0	0	SAR_ADC_10BIT_1_STAT	0	0	0	BODY_CTU_0_STAT	STDBY_CTU_0_STAT	EMIOS_0_STAT	EMIOS_1_STAT
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	PSI5_1_STAT	0	0	0	0	ETHERNET_0_STAT	ETHERNET_1_STAT	SENT_1_STAT	0	0	0	0	DSPI_1_STAT	DSPI_3_STAT	DSPI_5_STAT	DSPI_7_STAT
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1773. Peripheral Reset Status Register 7 (RGM_PSTAT7)

Table 1717. RGM_PSTAT7 register field descriptions

Field	Description
0 SAR_ADC_12BIT_1_STAT	SAR_ADC_12BIT_1 reset status 0 peripheral is not in reset 1 peripheral is in reset
1 SAR_ADC_12BIT_3_STAT	SAR_ADC_12BIT_3 reset status 0 peripheral is not in reset 1 peripheral is in reset
8 SAR_ADC_10BIT_1_STAT	SAR_ADC_10BIT_1 reset status 0 peripheral is not in reset 1 peripheral is in reset
12 BODY_CTU_0_STAT	BODY_CTU_0 reset status 0 peripheral is not in reset 1 peripheral is in reset
13 STDBY_CTU_0_STAT	STDBY_CTU_0 reset status 0 peripheral is not in reset 1 peripheral is in reset
14 EMIOS_0_STAT	EMIOS_0 reset status 0 peripheral is not in reset 1 peripheral is in reset

Table 1717. RGM_PSTAT7 register field descriptions (continued)

Field	Description
15 EMIOS_1_STAT	EMIOS_1 reset status 0 peripheral is not in reset 1 peripheral is in reset
16 PSI5_1_STAT	PSI5_1 reset status 0 peripheral is not in reset 1 peripheral is in reset
21 ETHERNET_0_STAT	ETHERNET_0 reset status 0 peripheral is not in reset 1 peripheral is in reset
22 ETHERNET_1_STAT	ETHERNET_1 reset status 0 peripheral is not in reset 1 peripheral is in reset
23 SENT_1_STAT	SENT_1 reset status 0 peripheral is not in reset 1 peripheral is in reset
28 DSPI_1_STAT	DSPI_1 reset status 0 peripheral is not in reset 1 peripheral is in reset
29 DSPI_3_STAT	DSPI_3 reset status 0 peripheral is not in reset 1 peripheral is in reset
30 DSPI_5_STAT	DSPI_5 reset status 0 peripheral is not in reset 1 peripheral is in reset
31 DSPI_7_STAT	DSPI_7 reset status 0 peripheral is not in reset 1 peripheral is in reset

64.4 Functional description

64.4.1 Reset state machine

The main role of the MC_RGM is the generation of the reset sequence which ensures that the correct parts of the chip are reset based on the reset source event. This is summarized in *Table 1718*.

Table 1718. MC_RGM reset implications

Source	What Gets Reset	External Reset Assertion(1)
power-on reset	all executes a start-up self test if enabled	yes
'destructive' resets	all except some clock/reset management executes a start-up self test if enabled	yes
long external (ESR0) reset	all except some clock/reset management and debug executes a start-up self test if enabled	programmable(2)
long external (ESR0) and long 'functional' resets	all except some clock/reset management, STCU, FCCU, and debug	programmable(2)
shortened 'functional' resets(3)	flip-flops except some clock/reset management , STCU, FCCU, and debug	programmable(2)

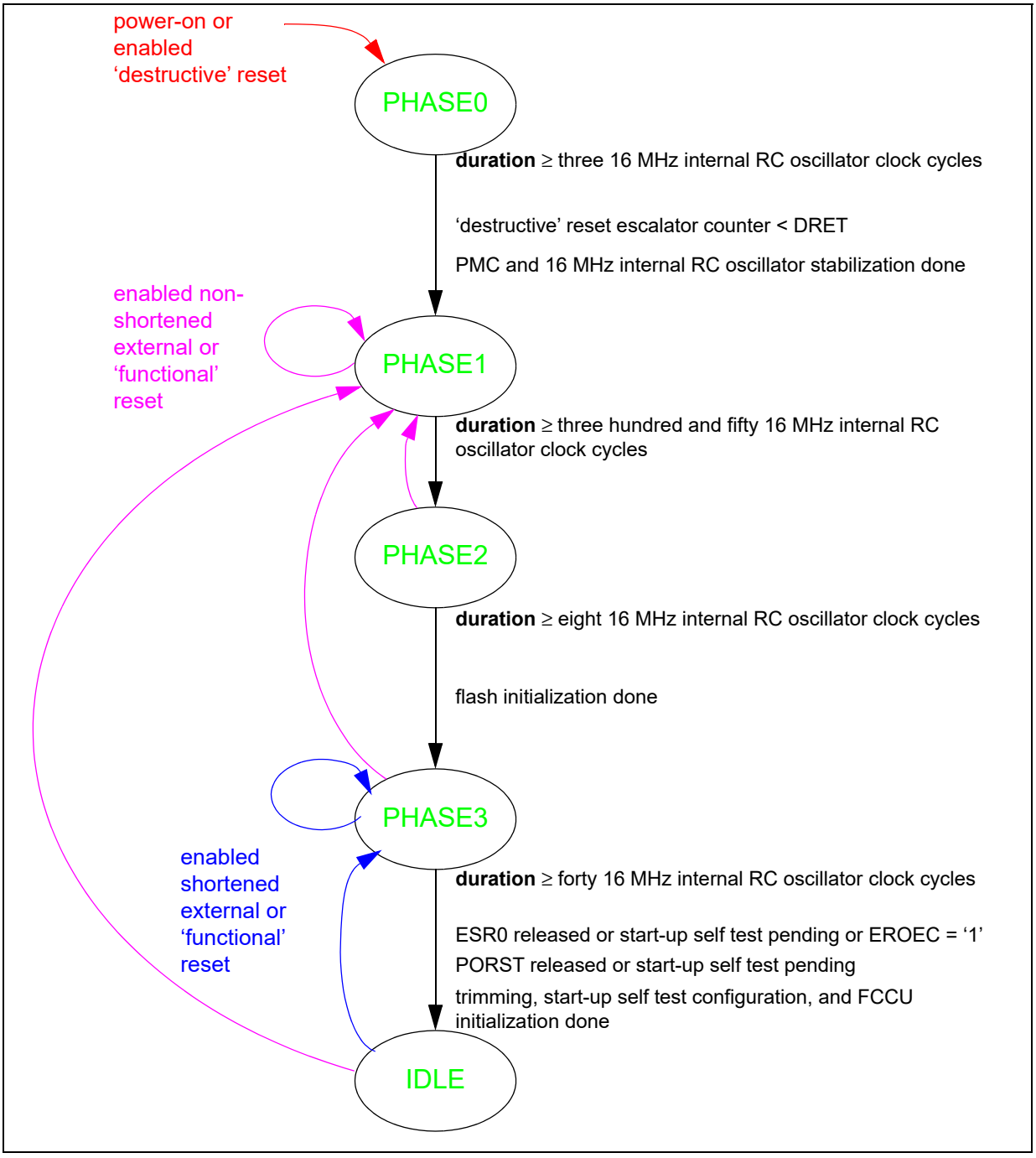
1. 'external reset assertion' means that the ESR0_B pin is asserted by the MC_RGM from the start of the reset sequence until the end of reset PHASE3
2. the assertion of the external reset is controlled via the RGM_FBRE register
3. the short sequence is enabled via the RGM_FESS register

Note: JTAG logic has its own independent reset control and is not controlled by the MC_RGM in any way.

The reset sequence is comprised of five phases managed by a state machine, which ensures that all phases are correctly processed through waiting for a minimum duration and until all processes that need to occur during that phase have been completed before proceeding to the next phase.

The state machine used to produce the reset sequence is shown in *Figure 1774*.

Figure 1774. MC_RGM state machine



64.4.1.1 PHASE0 Phase

This phase is entered immediately from any phase on a power-on or enabled ‘destructive’ reset event. The reset state machine exits PHASE0 and enters PHASE1 on verification of the following:

- all enabled ‘destructive’ resets have been processed
- all processes that need to be done in PHASE0 are completed
  - PMC and 16 MHz internal RC oscillator stabilization
- a minimum of three 16 MHz internal RC oscillator clock cycles have elapsed since power-up completion and the last enabled ‘destructive’ reset event
- the ‘destructive’ reset escalator counter has not reached the value in the RGM_DRET[DRET] field

64.4.1.2 PHASE1 phase

This phase is entered either on exit from PHASE0 or immediately from PHASE2, PHASE3, or IDLE on a non-masked external or ‘functional’ reset event if it has not been configured to trigger a ‘short’ sequence. The reset state machine exits PHASE1 and enters PHASE2 on verification of the following:

- all enabled, non-shortened ‘functional’ resets have been processed
- a minimum of three hundred and fifty 16 MHz internal RC oscillator clock cycles have elapsed since the last enabled external or non-shortened ‘functional’ reset event

64.4.1.3 PHASE2 phase

This phase is entered on exit from PHASE1. The reset state machine exits PHASE2 and enters PHASE3 on verification of the following:

- all processes that need to be done in PHASE2 are completed
  - flash initialization
- a minimum of eight 16 MHz internal RC oscillator clock cycles have elapsed since entering PHASE2

64.4.1.4 PHASE3 phase

This phase is entered either on exit from PHASE2 or immediately from IDLE on an enabled, shortened ‘functional’ reset event. The reset state machine exits PHASE3 and enters IDLE on verification of the following:

- all processes that need to be done in PHASE3 are completed
  - trimming, start-up self test configuration, and FCCU initialization
- a minimum of forty 16 MHz internal RC oscillator clock cycles have elapsed since the last enabled, shortened ‘functional’ reset event
- ESR0 is not asserted, except if a start-up self test is to be executed on exit of the current reset sequence or RGM_EROEC[EROEC] = 1
- PORST is not asserted, except if a start-up self test is to be executed on exit of the current reset sequence

64.4.1.5 IDLE phase

This is the final phase and is entered on exit from PHASE3. When this phase is reached, the MC_RGM releases control of the system to the platform and waits for new reset events that can trigger a reset sequence.

64.4.2 'Destructive' resets

A 'destructive' reset indicates that an event has occurred after which critical register or memory content can no longer be guaranteed. Exceptions to the latter are the software 'destructive' reset and the external power-on reset which do allow the memory content to be preserved if the chip is configured to not execute a self-test on power-on, 'destructive', and external resets.

The status flag associated with a given 'destructive' reset event (RGM_DES.F_<destructive reset> bit) is set when the 'destructive' reset is asserted and the power-on reset is not asserted. It is possible for multiple status bits to be set simultaneously, and it is software's responsibility to determine which reset source is the most critical for the application.

A given 'destructive' reset can be optionally disabled by writing bit RGM_DERD.D_<destructive reset>.

Note: The RGM_DERD register can be written only once between two 'destructive' reset events.

The chip's low-voltage detector threshold ensures that, when 'destructive' voltage out of range (LVD or HVD, or both) is enabled, the supply is sufficient to have the 'destructive' event correctly propagated through the digital logic. Therefore, if a given 'destructive' reset is enabled, the MC_RGM ensures that the associated reset event will be correctly triggered to the full system. However, if the given 'destructive' reset is disabled and the voltage goes below the digital functional threshold, functionality can no longer be ensured, and the reset may or may not be asserted.

An enabled 'destructive' reset will trigger a reset sequence starting from the beginning of PHASE0.

64.4.2.1 External power-on reset—PORST

The bidirectional external power-on reset pin PORST is a special case 'destructive' reset.

The detection of a falling edge on PORST will start the reset sequence.

The chip asserts PORST if the reset sequence was triggered by one of the following:

- a power-on reset
- a 'destructive' reset event

In this case, PORST is asserted until the end of PHASE0.

The PORST input is disabled immediately as of the PORST output being asserted in order to prevent a falling edge from being detected while the pin is being driven from the chip. The input is then re-enabled 4 us after the PORST output stops being driven by the chip in order to allow the pull-up on the pin to take effect.

In addition, the reset sequence is not exited into application until PORST is no longer driven low from off-chip. In the case the chip is configured to execute a self-test on power-on, 'destructive', and external resets, this is the exit of PHASE3 after self test execution has completed.

64.4.3 External reset

The MC_RGM manages the external reset coming from ESR0. The detection of a falling edge on ESR0 will start the reset sequence.

The status flag associated with the external reset falling edge event (RGM_FES.F_ESR0 bit) is set when the external reset is asserted and the power-on reset is not asserted.

Note: The RGM_FERD register can be written only once between two 'destructive' reset events.

An enabled external reset will normally trigger a reset sequence starting from the beginning of PHASE1. Nevertheless, the RGM_FESS register enables the further configuring of the reset sequence triggered by the external reset. When RGM_FESS.SS_ESR0 is set, the external reset will trigger a reset sequence starting directly from the beginning of PHASE3, skipping PHASE1 and PHASE2. This can be useful especially when an external reset should not reset the flash memory.

The MC_RGM may also assert the external reset if the reset sequence was triggered by one of the following:

- a power-on reset
- a 'destructive' reset event
- an external reset event if configured via the RGM_FBRE register to assert the external reset
- a 'functional' reset event configured via the RGM_FBRE register to assert the external reset

In this case, ESR0_B is asserted until all conditions for exiting PHASE3 have been met with the exception of the ESR0 assertion check.

If the OPT_RGM_ESR0_HW bit of the "UTEST Miscellaneous" DCF client is set to '0', the MC_RGM continues to assert ESR0 until the RGM_EROEC[EROEC] bit has been cleared by software.

In addition, the MC_RGM asserts ESR0 while the start-up self test is being executed.

The ESR0 input is disabled immediately as of the ESR0_B output being asserted in order to prevent a falling edge from being detected while the pin is being driven from the chip. The input is then re-enabled 4 µs after the ESR0 output stops being driven by the chip in order to allow the pull-up on the pin to take effect.

If ESR0_B is stopped being asserted by the MC_RGM and the 4 µs input mask period expires during the reset IDLE phase (for example when the EROEC bit is cleared by the software), and now a low value is detected on ESR0, an external reset falling edge event will not occur, and no new reset sequence will be triggered. For a new reset sequence to trigger, ESR0 must first be de-asserted.

The correct timing between PORST and ESR0 is mandatory to avoid slightly random behavior on RGM_FES(F_ESR0) flag, adding capacitances in their circuits is highly discouraged. Considering two cases: device kept in reset at power-up and external switch reset.

In case of power-up, device can still be kept in reset by keeping ESR0 or PORST low externally. Once the device releases the control of ESR0_B, it waits for 4 us for enabling the ESR0 input, as said. If ESR0 still doesn't rise-up after these 4 us, RGM_FES(F_ESR0) is triggered although no reset sequence is started.

The latter is triggered at falling edge of ESR0 input only. This implies that if we have a slow ramp on ESR0, a flag can be seen during reset release. Moreover if ESR0 is kept asserted externally after reset phases, F_ESR0 flag would be seen. In case of external switch reset, where both PORST and ESR0 are asserted together, the ESR0_B release from device happens after PORST is released.

So, the 4 us window will start after PORST release. When PORST and ESR0 are asserted contemporary, destructive reset is the superset signal and ESR0 is masked at the stage of reset entry. In case ESR0 has a slower ramp than PORST, RGM_FES(F_ESR0) might not get set because ESR0 can take longer to ramp down. But in case ESR0 has a very slow ramp expected to PORST, it may not rise in 4 us after PORST release and so RGM_FES(F_ESR0) might get set at the time of reset release.

64.4.4 'Functional' resets

A 'functional' reset indicates that an event has occurred after which it can be guaranteed that critical register and memory content is still intact.

The status flag associated with a given 'functional' reset event (RGM_FES.F_<functional reset> bit) is set when the 'functional' reset is asserted and the power-on reset is not asserted. It is possible for multiple status bits to be set simultaneously, and it is software's responsibility to determine which reset source is the most critical for the application.

A given 'functional' reset can be optionally disabled by software writing bit RGM_FERD.D_<functional reset>.

Note: The RGM_FERD register can be written only once between two power-on reset events.

An enabled 'functional' reset will normally trigger a reset sequence starting from the beginning of PHASE1. Nevertheless, the RGM_FESS register enables the further configuring of the reset sequence triggered by a 'functional' reset. When RGM_FESS.SS_<functional reset> is set, the associated 'functional' reset will trigger a reset sequence starting directly from the beginning of PHASE3, skipping PHASE1 and PHASE2. This can be useful especially in case a 'functional' reset should not reset the flash memory module.

See *Chapter 69: Mode Entry Module (MC_ME)* for details on the STANDBY0 and DRUN modes.

64.4.5 Alternate event generation

The MC_RGM provides alternative events to be generated on reset source assertion. When a reset source is asserted, the MC_RGM normally enters the reset sequence. Alternatively, it is possible for some reset source events to be converted from a reset to either a SAFE mode request issued to the MC_ME or to an interrupt request issued to the core.

Alternate event selection for a given reset source is made via the RGM_DERD/FERD and RGM_DEAR/FEAR registers as shown in *Table 1719*.

Table 1719. MC_RGM alternate event selection

RGM_DERD/FERD bit value	RGM_DEAR/FEAR bit value	Generated event
0	X	reset

Table 1719. MC_RGM alternate event selection (continued)

RGM_DERD/FERD bit value	RGM_DEAR/FEAR bit value	Generated event
1	0	SAFE mode request
1	1	interrupt request

The alternate event is cleared by deasserting the source of the request (at the reset source that caused the alternate request) and also clearing the appropriate RGM_DES/FES status bit.

Note: SAFE mode requests are generated regardless of whether the system clock is running. Interrupt requests are generated with the system clock, and therefore, if the system clock was disabled at the time of the event, the interrupt request is not asserted until the system clock is re-enabled.

Note: If a masked 'destructive' reset event which is configured to generate a SAFE mode/interrupt request occurs during PHASE0, it is ignored, and the MC_RGM will not send any safe mode/interrupt request to the MC_ME.

Note: If a masked 'functional' reset event which is configured to generate a SAFE mode/interrupt request occurs during PHASE1, it is ignored, and the MC_RGM will not send any safe mode/interrupt request to the MC_ME.

64.4.6 'Functional' reset escalation

'Functional' reset escalation can be used to generate a 'destructive' reset if a number of 'functional' or external resets has occurred between software writes to the RGM_FRET register. This function is enabled by writing a non-zero value to the FRET field of this register.

Once 'functional' reset escalation has been enabled, the MC_RGM increases a counter on each 'functional' or external reset which causes a reset sequence to be initiated (entrance into PHASE1 or PHASE3 from the IDLE phase). This counter is cleared on a write of any value to the RGM_FRET register and on any power-on or 'destructive' reset. If the counter reaches the value in the RGM_FRET register's FRET field, the MC_RGM starts a reset sequence at PHASE0 and asserts the F_EDR bit in the RGM_DES register.

64.4.7 'Destructive' reset escalation

'Destructive' reset escalation can be used to keep the chip in the reset state until a new power-on triggers a reset sequence if a number of 'destructive' resets has occurred between software writes to the RGM_DRET register. This function is enabled by writing a non-zero value to the DRET field of this register.

Once 'destructive' reset escalation has been enabled, the MC_RGM increases a counter on each 'destructive' reset[29] which causes a reset sequence to be initiated (entrance into PHASE0 from the IDLE phase) or an ongoing reset sequence to restart (entrance into PHASE0 from PHASE1, PHASE2, or PHASE3). This counter is cleared on a write of any value to the RGM_DRET register and on any power-on reset. If the counter reaches the

29. Only exception is the external power-on reset pin PORST that does not affect the increase of counter.

value in the RGM_DRET register's DRET field, the MC_RGM enters reset PHASE0 and stays there until the next power-on reset occurs.

64.4.8 Individual peripheral resets

A peripheral may be reset individually without resetting the rest of the chip by setting the appropriate bit in the RGM_PRST*n* registers. However, in order to prevent unexpected behavior by the chip, the following sequence must be observed by software.

1. Disable the peripheral that is to be reset via the MC_ME (see the MC_ME chapter for details).
2. Wait for the mode change to complete.
3. Set the bit corresponding to the peripheral in the RGM_PRST*n* registers.
4. Wait until the bit corresponding to the peripheral in the RGM_PSTAT*n* registers is set.
5. Clear the bit corresponding to the peripheral in the RGM_PRST*n* registers.
6. Wait until the bit corresponding to the peripheral in the RGM_PSTAT*n* registers is cleared.

The peripheral can be kept in reset indefinitely by not executing steps *5.* and *6.*. After the completion of step *6.*, the peripheral can be enabled again via the MC_ME.

65 Boot Assist Flash (BAF)

65.1 Introduction

The BAF code is programmed by ST. The two main tasks of BAF are to provide a serial bootloader feature and to search for the application entry point.
For details regarding how the BAF code is executed, refer to *Chapter 8: Reset and Boot*.

The BAF is executed via the IRCOSC as clock source.

The MCU is booted through a collaboration of several blocks, hardware and firmware. The first boot phases are performed by a state machine inside the System Status and Configuration Module (SSCM). Once completed, the SSCM sends a reset vector to the HW boot core of the device pointing into the Boot Assist Flash (BAF).

The BAF code then checks the life cycle of the device. When the Life Cycle is set to FAIL_ANALYSIS, the BAF code shall check the UOPS[BAF_FA] bit inside SSCM as described in *Chapter 66: System Status and Configuration Module (SSCM)*. This bit mirrors the bit 31 of the UTEST_Miscellaneous_2 DCF. It tells whether the BAF shall stay in an endless loop or not when the Life Cycle is set to FAIL_ANALYSIS. In case the DCF is not programmed, the bit shall be 0 (refer to *Figure 1775*). Otherwise, it searches for a boot header and boots the application code in internal flash memory.

If no boot header is found in internal flash memory, it downloads application code serially using the LINFlexD module or the M_CAN module. The package pins used by LINFlexDs are the same pins used by the M_CAN modules. Hence, the external PHY can be either LIN or CAN (refer to *Section 65.3.9* for configuration details).

65.2 Memory map

For the BAF location in flash, refer to the Embedded memories chapter. The BAF block base address is 0x0040_4000. It is one time programmable (OTP) and is programmed during factory test. The address space and memory used by the BAF code are shown in *Table 1720*.

Note: The BAF Flash block is assigned as OTP and it is protected from programming by a bit in the PASS LOCK0 registers; but the BAF Flash block can further be protected from programming by an external hardware pin. If the Hardware Interlock feature is enabled, while this pin is logic '0' the BAF Flash block can not be programmed.

Table 1720. BAF memory organization

BAF Component	Address
BAF image header	0x0040_4000
BAF code entry point	0x0040_4100
BAF data area and stack	0x5280_0000–0x5280_0603

65.2.1 BAF image header

BAF image contains a 256-byte header starting from address 0x0040_4000. The BAF header is explained in *Table 1721*.

Table 1721. BAF image header

Address offset	Size (bits)	Field description
0x0000	32	SSCM boot record tag
0x0004	32	BAF start address
0x0008	32	BAF version number
0x000C–0x000F	Reserved	
0x0010	32	Part Clock Jitter Activation Constant
0x0014	32	Clock Jitter Activation Constant
0x0018–0x00FF	Reserved	

65.2.1.1 BAF image version

The BAF version is a 32-bit field in the image header starting at address 0x0040_4000. The 32-bit field is explained in *Table 1722*.

Table 1722. BAF image version

Field	Description
0:7	Major Number: This field contains the major version number for the BAF image
8:15	Minor Number: This field contains the minor version number for the BAF image
16:31	Reserved

65.2.1.2 Clock Jitter Activation constant

The Clock Jitter Activation constant is a 32-bit flag, but the smallest element in flash that can be programmed is 64 bits, so the first 32 bits are reserved to allow programming of this constant. This 32-bit flag can be programmed by the customer and is used to activate the Clock Jitter feature that adds clock edge jitter to the baud rate clocks of several communications modules.As part of the BAF startup procedure, BAF copies this constant from flash to the PASS Clock Jitter Enable Register.

Table 1723. BAF Clock Jitter Activation Constant

Field	Description
0:31	Clock Jitter Activation Constant. The initial state of production disable flag is erased - 0xFFFF_FFFF. This equates to NO clock jitter.

65.3 Functional description

A functional description of the BAF modes is contained in the following sections.

65.3.1 Boot modes

Depending on the state and the content of its flash memory, the device can enter one of these different boot modes.

- Application code execution

  If the internal flash contains application code with a valid boot header, the application is executed.

- Serial boot

  If no valid boot header is found and the current life cycle phase of the device is less than Infield, an attempt is made to download the application by a serial protocol using the LINFlexD modules or M_CAN modules. The downloaded code is then executed by the boot CPU.

- Static mode

  The device enters power safe mode and waits for an external reset.

65.3.2 Device initialization

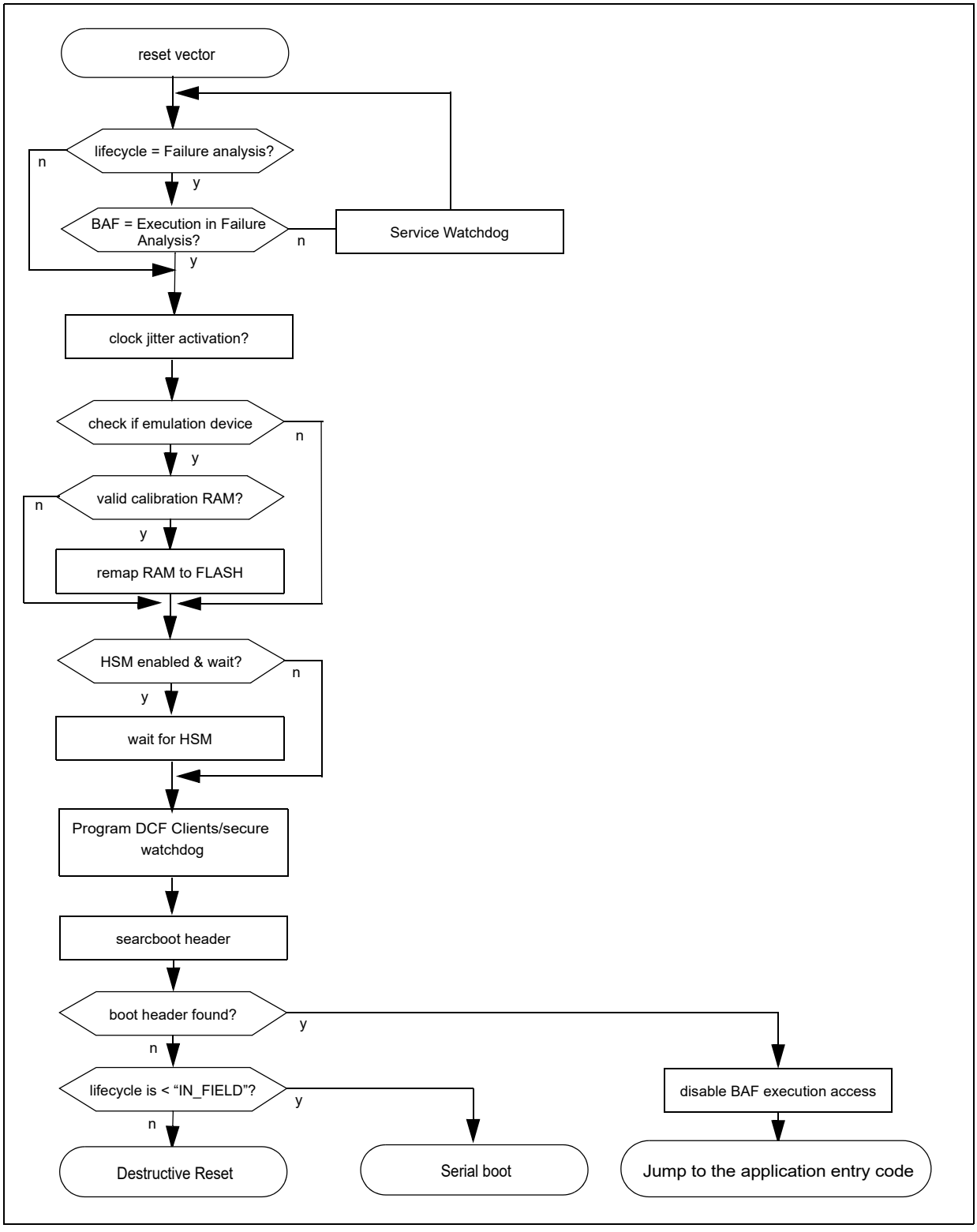
The BAF performs a minimal initialization of core registers and clocks required to enter the above described boot modes, then attempts to interact with the Hardware Security Module (HSM) in the following way:

- HSM: If the HSM is enabled and the “wait for HSM” flag is present, then the boot core waits for the HSM to become ready.
- BAF sets up the exception vector table to catch machine check exceptions, which may occur if there are any uncorrectable ECC errors. In the event of an uncorrectable ECC error, the BAF issues a destructive reset from the machine check exception handler if the exception is due to the reading of instructions from the BAF region. If an exception occurs during reading of the Boot header, the BAF ignores the ECC error and jumps to next boot header. If an ECC error occurs during reading of D-MEM, the BAF will still issue a destructive reset. BAF programs the Mode Entry module’s MC_ME_MCTL[TARGET_MODE] to issue a destructive reset.
- BAF uses DMEM memory area of the boot CPU as in *Table 1720: BAF memory organization* for its stack, data and any code that executes from volatile memory. During the BAF start-up procedure, it initializes this section of memory so that there is no ECC error when BAF accesses this memory area.
- The memory area of the BAF is protected from execution for functional safety considerations. This is achieved by disabling BAF execution in PFLASH control register3 (PFLASHC_PFCR3[BAF_DIS]). Finally a mode change (from DRUN to DRUN again) is requested from Mode Entry module to activate the new core configuration. At this point the BAF has finished its operation.

65.3.3 Flow of control

Figure 1775 shows the overall flow of control in the BAF. The individual steps are explained in the following sections.

Figure 1775. Flow of control



65.3.4 Optionally wait for HSM

If the HSM is enabled (SSCM_UOPS[HSE] = 1), the boot core will need to wait for the HSM to become ready (SSCM_UOPS[HSB = 1]). The core then polls the HSM until a ready flag is set. Both SSCM registers can be set by a DCF record. The boot core polls the HSM to HOST FLAGS register (HSM_HSM2HTF) until its least significant bit is set.[end] The boot core also services the watchdog while polling the HSM_HSM2HTF register. This is just in case the HSM is not ready in 16 ms, servicing the watchdog prevents the system going to reset.

65.3.5 Initialization of BAF DCF clients

65.3.5.1 'Soft' DCF clients

DCF clients are 32-bit hardware registers that are written by the SSCM during reset with data that is stored in DCF record format in UTEST region of flash memory. These DCF records are used to configure module registers, and select device operating modes.

The BAF extends the DCF record concept by implementing soft DCF clients. The 64-bit DCF records include:

- 32 bits of data
- 15-bit chip select field (selects the module which implements the DCF clients)
- 15-bit address field (selects a specific DCF client within the selected module)
- Parity bit (not used with UTEST DCF records)
- Stop bit (to indicate the end of DCF record has been reached)

As part of its execution flow, BAF searches through the list of DCF records in UTEST flash to determine if any of the records are to be processed by BAF.

The search starts at one of two addresses: either the start address of DCF records in UTEST (DCF Start Record), or the address specified in UTEST (Soft DCF Record Start Address). If either of the two least significant bits of the 32-bit address stored at the Soft DCF Record Start Address is set, the default start address at the DCF Start Record is used. If the two least significant bits are clear, this 32-bit value is assumed to be the start address of the search.

The default setting of the two least significant bits is 11b (erased value of flash memory is 0xFFFF_FFFF). A user may intentionally program a dummy address with the least two significant bits set in order to disable the search for an alternate address (as the UTEST block is OTP). Valid search start address must start on a 4-byte boundary.

The DCF record search finishes when a DCF record is read with a STOP bit set. The STOP bit may be set because that memory location is not programmed yet (so the list can be added to) or it can be intentionally programmed STOP record to indicate that no more records are added to the list. BAF does not execute the DCF record with STOP bit set. BAF parses the STOP bit before parsing the rest of the records.

BAF uses Chip Select 14 (CS14) for BAF DCF clients. For records with CS14 asserted, the BAF uses the address field (address[14:0]) in DCF records to determine how to process the 32-bits of data. The soft DCF clients programmed by BAF is listed in *Table 1724*.

The DCF clients which are used for initialization of security watchdog are explained in *Section 65.3.5.2: Security watchdog configuration*.

A group of four Soft DCF clients is used to provide a mechanism to write configuration information to address locations. The first DCF client receive 32-bit address information. The next three DCF Clients receive the data that is to be written to the address contained in the first client. The three data clients allow 32-, 16- and 8-bit data to be written to the address in the first client. These four clients can be accessed repeatedly to write whole string of data to memory.

Another soft DCF client as stated in *Table 1724* provides a simple callback feature. The soft DCF record data contains the start address of a subroutine that the BAF calls before it starts searching for the host boot header and exits to customer application code. The callback routine can reside within the BAF or UTEST blocks only. The callback DCF client can be accessed repeatedly in order to execute a callback sequence.

The serial boot DCF client provides a mechanism using which application can inform BAF to perform serial boot or boot from internal flash. Details can be viewed in *Section 65.3.5.6: SER_BOOT_CBACK – BAF serial boot DCF client*.

Table 1724. BAF DCF client list

Address (hex)	Name	Label	Description
0x0000	Security Watchdog Control Register	SEC_SWT_CR	Data to write to SWT Control Register
0x0001	Security watchdog Time out register	SEC_SWT_TO	Security watchdog counter value
0x0002	Security watchdog service address	SEC_IAC8	Instruction Address compare register value - Code present at this address services the watchdog.
0x0003	Security watchdog CPU select	SEC_SWT_CPU	Determines which CPU is to use the security watchdog
0x0004–0x01FF	Reserved		
0x0200	Address	ADDR	The address to which DATAX is written
0x0201	32-bit Data	DATA32	The 32-bit data that is written to ADDR
0x0202	16-bit Data	DATA16	The 16-bit data that is written to ADDR
0x0203	8-bit Data	DATA8	The 8-bit data that is written to ADDR
0x0204–0x020F	Reserved		
0x0210	Callback Address	CALLBACK	The address of a callback function
0x0211	Serial boot Callback Address	SER_BOOT_CBACK	The address of serial boot callback function
0x0212–0x0213	Reserved		Reserved for future use.
0x0214	Reserved		Reserved for future use.
0x0215–0x7FFF	Reserved		Reserved for future use.

Note: BAF boots with a watchdog timeout window set to 16ms. Parsing more and more DCF Records may increase boot time. But it should be taken care that number of DCF records should not be very high as it can cause watchdog window timeout.

65.3.5.2 Security watchdog configuration

BAF initializes the security watchdog before it jumps to the HOST CPU application. BAF implements four DCF clients for security watchdog initialization. BAF parses the DCF records and looks for configuration data to be used to program the security watchdog registers. BAF first collects all information required to program the security watchdog from DCF records and then program the security watchdog registers. BAF configures only SWT2 as security watchdog for the CPU2 core. Configuration of the Security watchdog for other cores has to be performed by the application running on that core as BAF cannot program other core's IAC8 register. Security watchdog detailed configuration steps are explained in the section below.

65.3.5.2.1 Security Watchdog Control Register (SEC_SWT_CR)

BAF parses the DCF records. If BAF finds a record with CS14 set and address field as 0x0000 while parsing, BAF programs the DCF client "Security watchdog control register" with the 32-bit value present in the DATA section of the DCF records. Once written, any subsequent writes to this DCF Client are ignored by BAF.

65.3.5.2.2 Security Watchdog TImeout (SEC_SWT_TO)

The BAF parses the DCF records. While parsing, if the BAF finds a DCF record with CS14 set and address field as 0001h, the BAF programs the security watchdog timeout DCF Clients. This is the security watchdog timeout register. The BAF checks the value obtained from DCF records. If the value yields a timeout greater than 30 seconds, then the BAF changes this value to 30 seconds. Once written, any subsequent writes to this DCF client are ignored.

65.3.5.2.3 Security Watchdog Service Address (SEC_IAC8)

BAF configures the security watchdog in 'fixed address execution service mode'. In this mode, the watchdog is serviced by executing code at the address loaded into the designated IAC8 (Instruction Address Compare) register. If BAF finds a DCF record with CS14 set and address field as 0x0002 while parsing the DCF records, BAF programs the DCF clients for SEC_IAC8. This DCF client can only be written once. Subsequent writes are ignored.

Note: To program the core IAC8 special purpose register, the ownership of this register should be with software. If hardware has its ownership software will not be able to write to it using mtspr instruction.

65.3.5.2.4 Security Watchdog CPU select (SEC_SWT_CPU)

This DCF client indicates to BAF which CPU will use the security watchdog. BAF parses the DCF record and if it finds a record with CS14 set and address field set as 0003h, BAF programs this DCF client. This means that BAF uses the two least significant bits out of the 32-bit data in the DCF record to find the CPU number which will use the security watchdog. *Figure 1776* and *Table 1725* indicate the data layout for this client in the DCF record. This client is written once during reset. Subsequent writes are ignored.

Writing this client causes the BAF to configure the security watchdog based on data written to the three previous DCF clients. If any of the three previous DCF clients are not written, data is configured erroneously.

During normal BAF operation, SWT2 is configured to perform a conventional task watchdog function (cause reset if not regularly serviced). If the security watchdog is assigned to CPU2 (IOP), first SWT3 is configured to perform the watchdog function task. The timeout value, configuration, and current time is read from SWT2 and copied to SWT3. Then SWT2 is unlocked with current sequence and enabled as security watchdog.

DCF Client Address 0x0003

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

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

Figure 1776. SEC_SWT_CPU bit configuration

Table 1725. SEC_SWT_CPU field description

Field	Description
30:31 SC	These two bits select the CPU which will use the security watchdog. 00 Reserved 01 Reserved 10 CPU2 (IOP) 11 Reserved

Note: BAF programs FCCU for enabling SWT2 and SWT3 as fault source and makes them hardware recoverable. This is done to ensure that fault latched in FCCU get cleared on reset and on next SWT timeout FCCU can again trigger reset. BAF does not clear the faults latched in FCCU as they can be of interest for application.

65.3.5.3 BAF DCF client - ADDRESS

BAF parses the DCF records. If BAF finds a DCF record with CS14 set and Address field as 0x0200 while parsing, BAF programs this DCF client. This client is written with a 32-bit address from the Data section of DCF record. This 32-bit address is used in conjunction with any one of the DATA32, DATA16 or DATA8 DCF Data clients. BAF does not immediately use the ADDRESS information, but it is stored and used later in conjunction with a DATAX DCF client (DATA32, DATA16 or DATA8). Then BAF writes the data to the address in memory specified by this ADDRESS DCF client.

The user can create a series of data writes to memory by programming a series of DCF records consisting of pairs of records. The first DCF record of the pair writes to ADDR to provide the address. The second DCF record of the pair writes to the DATA client (DATAX) to provide the data. Subsequent pairs of DCF records can be programmed to write to more memory locations. The number of pairs of DCF records that can be programmed is only limited by the amount UTEST space set aside for DCF records.

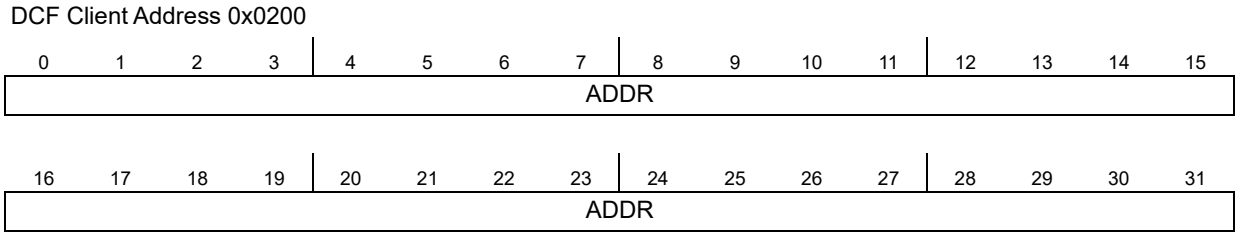


Figure 1777. BAF DCF ADDR client configuration

Table 1726. BAF DCF ADDR client description

Field	Description
0:31 ADDR	32-Bit Address DCF client This ADDRESS value is used in conjunction with a DATAX value. The BAF writes DATAX value to ADDRESS value in the memory space of the MCU.

65.3.5.4 BAF DCF client - DATAX

Caution: While using DATAX DCF records, it should be taken in account that BAF does not take care of ECC scheme and can generate ECC, if write is performed in memory locations which are protected by ECC and are not pre-initialized as required.

65.3.5.4.1 DATA32 – 32-bit data DCF client

BAF parses the DCF records. If BAF finds a record with CS14 set and address field set as 0x0201 while parsing, BAF programs this 32-bit DCF client. BAF takes the memory address provided by the ADDR client, masks the least significant 2 bits to ‘0’ (Only 32-bit aligned writes are supported) and writes this 32-bit data to the address.

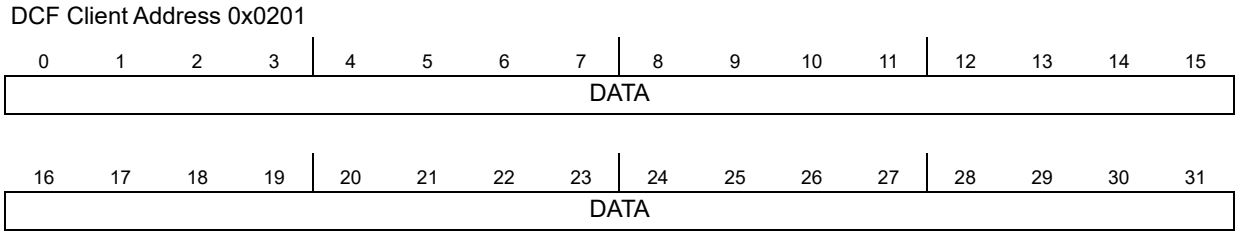


Figure 1778. DATA32 DCF client configuration

Table 1727. DATA32 DCF client configuration

Field	Description
0:31 DATA	32-Bit DCF data

65.3.5.4.2 DATA16 – 16-bit data DCF client

BAF parses the DCF records. If BAF finds a record with CS14 set and address field set as 0x0202 while parsing, BAF programs this 16-bit DCF client. BAF takes the memory address provided by the ADDR client, masks the least significant bits to ‘0’ (Only 16-bit aligned

writes are supported) and writes least significant 16-bit data to the address. Data in the 16 most significant bits are ignored.

DCF Client Address 0x0202

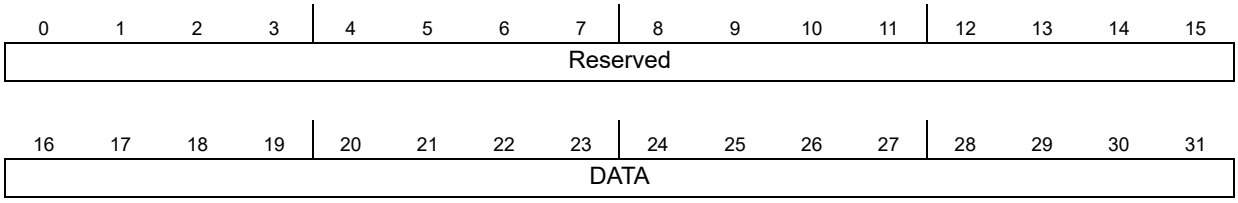


Figure 1779. DATA16 DCF Client Configuration

Table 1728. DATA16 DCF client configuration

Field	Description
16:31 DATA	16-Bit DCF data

65.3.5.4.3 DATA8 – 8-bit DCF client

BAF parses the DCF records. If BAF finds a record with CS14 set and address field set as 0x0203 while parsing, BAF programs this 8-bit DCF client. BAF takes the memory address provided by the ADDR client, and writes least significant 8-bit data to the address. Data in the 24 most significant bits are ignored.

DCF Client Address 0x0203

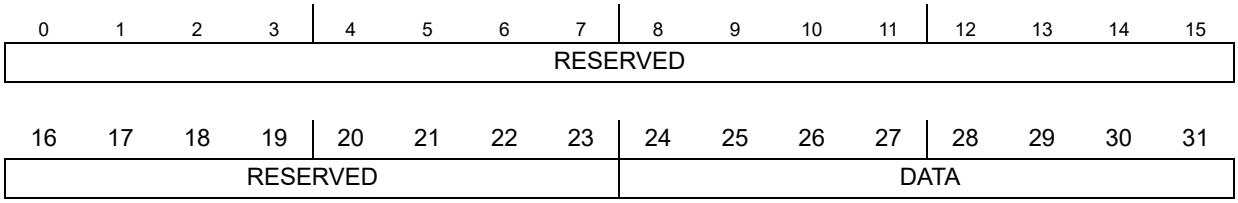


Figure 1780. DATA8 DCF Client Configuration

Table 1729. DATA8 DCF client configuration

Field	Description
24:31 DATA	8-Bit DCF data

65.3.5.5 CALLBACK – BAF callback DCF client

BAF parses the DCF records. If BAF finds a record with CS14 set and address field as 0x0210 while parsing, BAF programs this callback DCF client. This client is written with a 32-bit start address of a function which is invoked by BAF. The BAF code checks the start address is within the UTEST or within the BAF flash memory blocks, and then passes program flow to this address. If the address is found to be outside the UTEST and BAF flash memory blocks, no function call is executed. The callback is aborted and BAF moves on to the next DCF records. The function that is called is expected to comply with Embedded Application Binary Interface (EABI) for PowerPC found in Power Architecture web site and

to terminate in an orderly manner. An incorrectly written function causes unpredictable operations in BAF. No parameters are passed to this function and no return value is expected.

Caution: Excessively long callback functions could cause the watchdog timer to time-out and cause a reset.

BAF executes the callback function as the DCF records are parsed. Therefore, several callback functions maybe executed by writing several DCF records that write to the callback DCF clients. The callback functions are executed in the same order as the DCF records.

65.3.5.6 SER_BOOT_CBACK – BAF serial boot DCF client

BAF parses the DCF records. If BAF finds a record with CS14 set and address field as 0x0211 while parsing, BAF programs this serial boot DCF client. This DCF client is written with a 32 bit start address of function which is invoked by BAF. The BAF code checks the start address is within UTEST or within the BAF flash memory block, and then passes program flow to this address. If the address is found to be outside the UTEST and BAF flash memory block, no function call is executed. The DCF client execution is aborted and BAF moves on to the next DCF record. The function that is called is expected to comply with Embedded Application Binary Interface (EABI) for PowerPC found in over Architecture web site and to terminate in orderly manner. An incorrectly written function causes unpredictable operations in BAF. No parameters are passed to this function.

Warning: No parameters are passed to the function invoked by the BAF.Excessively long callback functions could cause the watchdog timer to time-out and cause a reset.

The serial boot callback function returns a char value. The returned value informs BAF if serial boot or normal internal flash boot should be executed (refer to *Table 1730* for details regarding return value).

Table 1730. Serial boot callback function return value definition

Return values	Description
0x00	Boot from internal flash using standard search for boot record (refer to ‘Search boot header and boot options’ in *Chapter 7: Device configuration*).
0x01	Serial boot using both M_CAN and LINFlexD (in UART mode) (refer to ‘Search boot header and boot options’ in *Chapter 7: Device configuration*). **Note:** Serial boot is conditioned by life cycle as described in *Figure 1775*

65.3.6 Production disable activation protocol and implementation

Module failure analysis is a standard requirement. But some OEMs have an additional requirement that upon entering failure analysis mode, the MCU operation is modified in such a way that the MCU can never be used in production ECU. The chosen operational modification for production disable is to apply clock jitters to can modules so that it prevents active communications with another can device on a can network. For detail regarding production disable refer to the PASS chapter.

The BAF maintains a 64-bit flag in BAF code flash region as explained in *Section 65.2.1.2: Clock Jitter Activation constant*. On every POR BAF copies the clock jitter activation constant from this flag to clock jitter enable register in PASS module to enable jitters on can baud clock. Default value of this flag is shown in *Section 65.2.1.2: Clock Jitter Activation constant*. Hence, jitter on the MCAN baud clock is disabled if the default value is used. To enable jitter on can clock the value of this flag should be modified to 0xFFFF_FFFE.

The BAF flash memory is write protected and locked. To enable modification of this flag and activate production disable feature, BAF implements production disable activation protocol as explained in *Table 1732*. BAF implements this protocol in a power.ORG EABI compliant callback routine. This callback routine can be invoked by programming a callback based soft DCF client as explained in *Section 65.3.5.5: CALLBACK – BAF callback DCF client*. The callback function address is shown in *Table 1731*.

Table 1731. Production Disable Callback routine entry point

Description	Address
Production disable protocol callback function address	0x0040_40E0

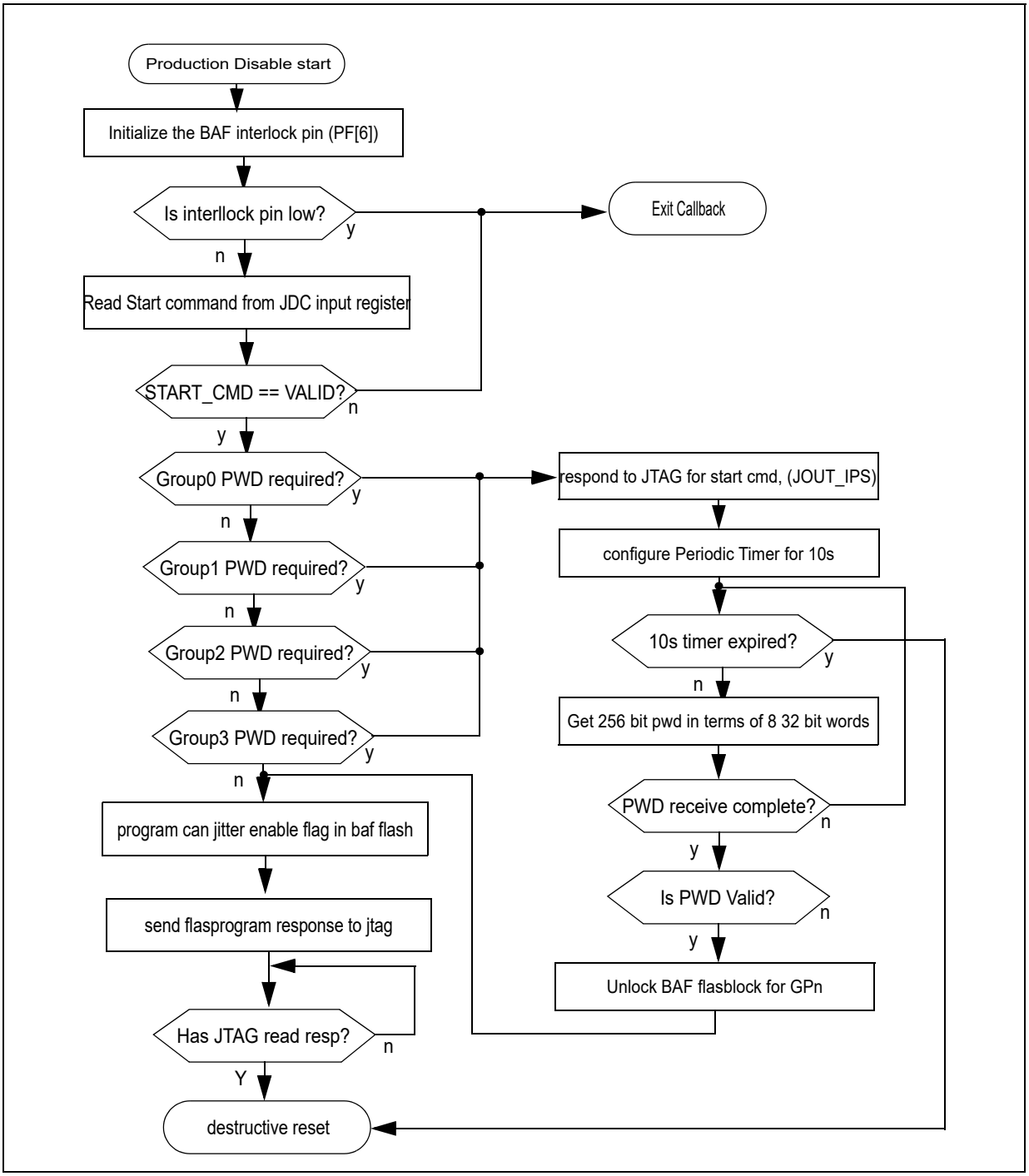
Table 1732. Production Disable protocol description

Step	Description	Tool Action/Response	MCU Action/Response
1	After power-on tool holds the MCU in reset and writes the activation code to JDC via JTAG interface. The tool pulls hardware interlock pin high, then releases reset.	Assert hardware interlock pin Write 0xF5A0_AA55 to JDC	Starts production disable activation
2	The MCU responds by writing to the JDC an acknowledgment code and which of the 4 PASS passwords are required to unlock the BAF flash block		Write 0xFA50_000*n* to JDC if password for pass group n is required to unlock the BAF flash block. (n is either 0,1,2 or 3)
3	The tool reads the acknowledge from JDC module and writes the relevant password to the JDC module. MCU performs password comparison and writes error code to JDC module password provided by tool was incorrect.	Read JDC Write 256 bit password for pass group n to JDC in eight 32 bit words. (n is 0,1,2 or 3)	Write 0xFA50_F00*n* if password does NOT match. Once the status has been read by the tool then cause reset. (n is either 0,1,2 or 3)

Table 1732. Production Disable protocol description (continued)

Step	Description	Tool Action/Response	MCU Action/Response
4	If more than one password is required to unlock the BAF flash block step 2 and step 3 are repeated until all relevant password has been supplied.		If password matches step 2 and step 3 are repeated for the next pass group until all pass group that lock BAF flash has been processed.
5	The MCU then attempts to program the can jitter activation constant within the BAF flash block. The can jitter activation constant flag is a write once flag. Once written it cannot be re-programed as BAF flash block is OTP. MCU write a code to JDC to signal success or failure of flash programming operation and issue reset.		Program can jitter activation constant in BAF flash block. If successful write 0xFA51_0000 to JDC. If unsuccessful write 0xFA51_F000 to JDC Issue reset.

Figure 1781. Production Disable callback function flow chart



65.3.7 TDM Diary Operation

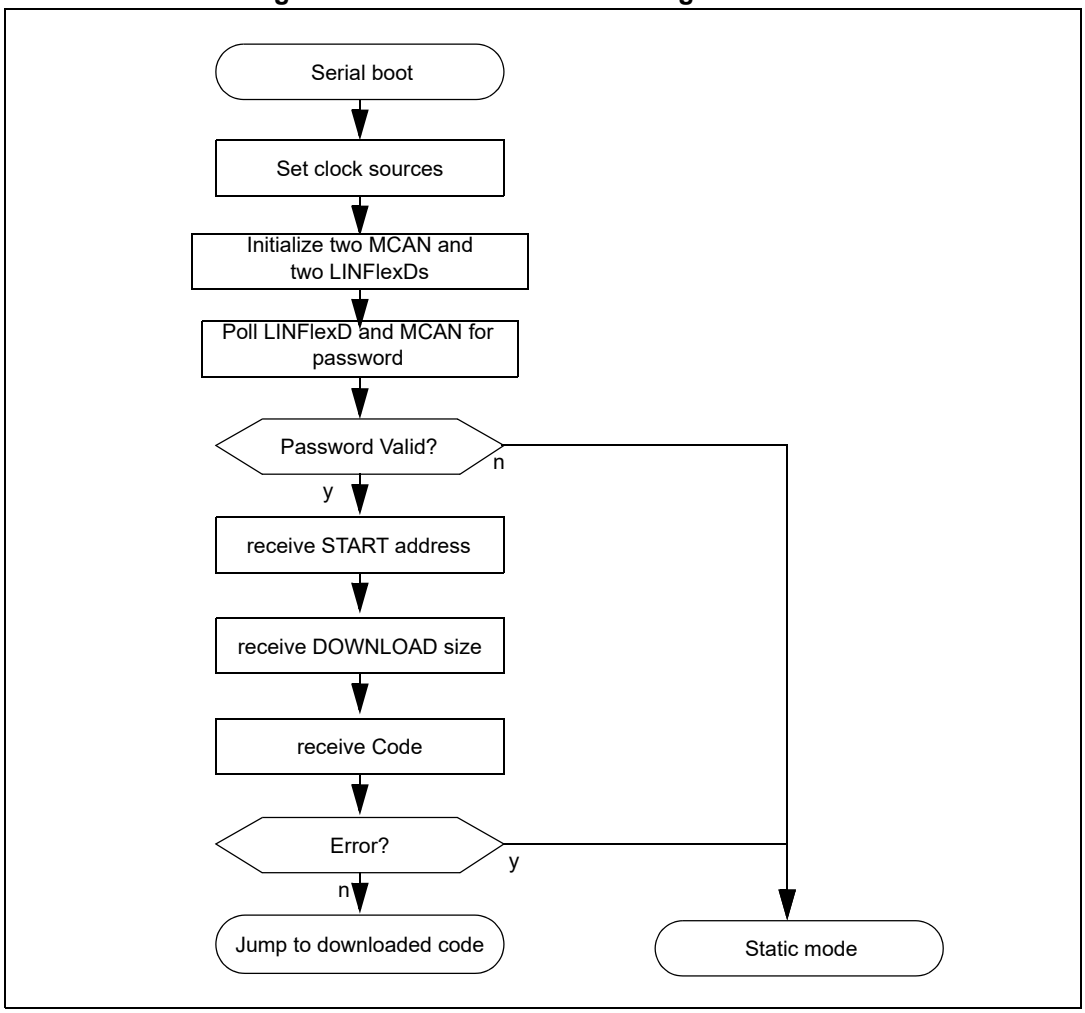
On every reset, BAF copies the first four bytes of last record of every Tamper Detect Region to corresponding Software Tamper Override Key Region Registers in TDM. BAF reads the base address of TDM Diary from DBA register in TDM. For more details on TDM refer to the corresponding section. This is done as a part of TDM Diary Override enhancement.

65.3.8 Optionally perform a serial boot

If no boot header is found in one of the locations mentioned above, then BAF determines the Life Cycle status of the device. If the Life Cycle is lower than In Field, it attempts a serial boot. Otherwise, the boot has failed and the BAF issues a destructive reset.

If the customer selects serial boot using serial boot using serial boot DCF client, then BAF determines the Life Cycle status of the device. If the Life Cycle is lower than Failure Analysis, it attempts a serial boot.

Figure 1782. Flow of control during serial boot



The clock source in the MC_CGM is configured (with IRCOSC for LINFlexD and with XOSC for CAN) and the PADs are set up for two LINFlexDs and two M_CANs signals, then the code is downloaded using LINFlexD or M_CAN in Serial Boot mode. In the absence of errors causing the device to be placed in Static mode, the device state is restored to resemble the out-of-reset state and the downloaded code is executed.

Serial boot allows the download of application to internal SRAM via LINFlexD or M_CAN interface. After downloading the application image to SRAM, the BAF jumps to the start address provided as part of download protocol mentioned below and executes the downloaded application.

At start of serial boot sequence the Rx-channels of all two serial interfaces shall be enabled and the BAF code shall take care for polling all two Rx-channels for detecting the valid serial boot message. After a successful detection of the active Rx-channel the corresponding Tx-channel shall be enabled and the inactive Rx-channel shall be disabled.

At the start of the process both LINFlexDs and M_CANs are configured in passive listening mode, connected to the same pair of pads. As soon as either interface receives a valid message, the receiving interface is configured in full Tx and Rx mode. The non receiving interface is switched back to reset state and does not participate further in serial boot.

If a CAN message of 8 bytes with ID 0x11 is received on the M_CAN controller first, the BAF program

- Disables LINFlexD.
- Configures LINFlexD RXD input mux back to reset state.
- Transitions to CAN serial download protocol routine.

If a CAN message not containing 8 bytes or NOT with ID 0x11 is received by M_CAN controller or a M_CAN controller generates a CAN error, it is probable that the tool is sending a LINFlexD frame. Under such circumstances BAF program

- Ignores the CAN message, if a message arrived.
- Clear any pending CAN error.
- Continue monitoring for valid CAN messages.

If any byte from LINFlexD is received first, the BAF program:

- Checks if the first byte is 0xFE of the password. If not continue polling. The data may be a valid CAN message.
- If it is first byte of the password (0xFE) then disable the M_CAN controller and restore to its reset state.
- Configures the M_CAN pads to their reset state.
- Transition to LINFlexD serial boot (UART mode).

65.3.8.1 Serial boot mode protocol

From a high-level perspective, the download protocol over M_CAN and LIN interfaces follows the steps below:

1. Host transmits the 64-bit password to MCU.
2. Host transmits start address, size of downloaded code in bytes, NO_ECHO bit and VLE bit to MCU.

   NO_ECHO bit signifies if actual download data to be echoed or not. If NO_ECHO bit is set, then data will not be echoed otherwise if NO_ECHO bit is clear, then data will be echoed.

Note: Since this device supports only VLE code and no Book E code, the VLE bit is used only for backward compatibility.

3. Host transmits a sequence of data bytes that are to be executed.
4. The boot CPU executes code from start address provided in step 2.

Each step must be completed before the next step starts.

The exact protocol between LINFlexD and M_CAN are slightly different.

65.3.8.2 LINFlexD download protocol

The communication is performed in half-duplex mode, any transmission from the host is followed by the MCU transmission.

1. The host sends data to the MCU and starts waiting.
2. The MCU echoes the received 8-byte of password, 4-byte of load address and 4-byte of VLE-bit, NO_ECHO bit and 30-bit of download size to the host. The MCU echoes actual download data depending upon NO_ECHO bit.
3. The host verifies whether the echo is correct.
   – If the data is correct, the host can send the next byte of data.
   – If data is not correct, it is assumed the MCU has not correctly received the data and the only practical option is for the Host to reset the MCU and start again.

A more detailed description of these steps follows.

All multi-byte data structures are sent with MSB first.

The tool sends data in the following order:

1. 8 bytes that represent the password.
2. 4 bytes that represent the start address where the code is to be downloaded.
3. 4 bytes that represent the number of bytes to be downloaded, NO_ECHO bit and the VLE bit.
4. *x* bytes of code that is downloaded to SRAM and executed (*x* must match the value specified in step *3.*).

65.3.8.3 M_CAN download protocol

The M_CAN download protocol uses very similar structure to the LINFlexD protocol, but it makes use of the message ID's provided by M_CAN.

The tool sends data packed into standard CAN message in the following manner:

1. A message with 0x11 as the ID and an 8-byte length to send the password. The MCU transmits the same message, but with an ID of 0x1.
2. A message with 0x12 as the ID and 8-byte length to send the start address, size of the code and VLE mode bit.The MCU transmits back the same data but with an ID of 0x2.
3. A message with 0x13 as the ID is used to send data that is to be downloaded to SRAM. The MCU transmit the same data with an ID of 0x3.

65.3.8.4 Download 64-bit password and password check

The first 64 bits received represent the password. The MCU only supports public passwords. The received password is always compared with the public password 0xFEED_FACE_CAFE_BEEF.

- If the correct password is received, BAF continues its task.
- If an incorrect password is supplied, BAF puts the device into static mode (low power safe mode).
- If 64 bits of password data is not received after approximately 30s, BAF causes a destructive reset of the MCU.

65.3.8.5 Download start address, VLE bit, NO_ECHO bit and code size

After the password, the next 8 bytes received by the MCU contains a 32-bit start address, the VLE mode bit, NO_ECHO bit and 30-bit code length as shown in *Figure 1783* and *Figure 1784*.

The VLE bit (Variable Length Encoding) is used to indicate for which instruction set the downloaded code is compiled. The UART serial boot protocol supports download of VLE Code (VLE = 1) and Book E Code (VLE = 0).

The MCU only supports VLE code. If the BAF detects that the host is sending Book E code, the BAF puts the device into static mode.

The NO_ECHO bit is used to indicate if download data should be echoed by the MCU. If the NO_ECHO bit is set (for example, NO_ECHO = 1), then only 8 bytes of password, 4 bytes of download address, and 4 bytes of download size are echoed, but download data is not echoed. If the NO_ECHO bit is clear, then every byte sent to the MCU must be echoed.

The start address defines where the received data is stored and where the MCU branches after the download is finished. The three LSB bits of the start address are ignored by BAF, such that the load address is 64-bit double-word aligned.

The length determines how many data bytes has to be loaded.

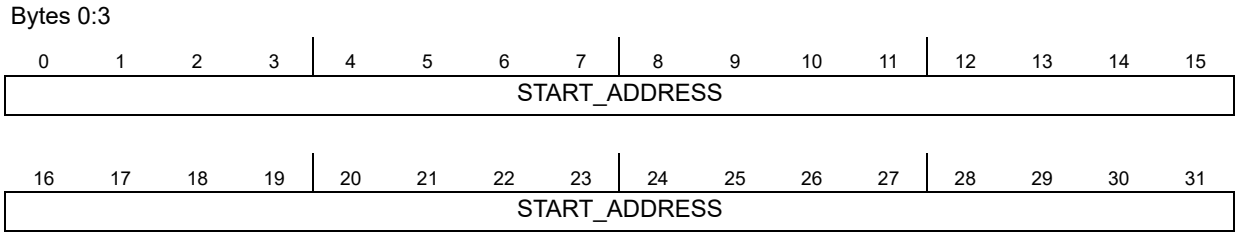


Figure 1783. Start address

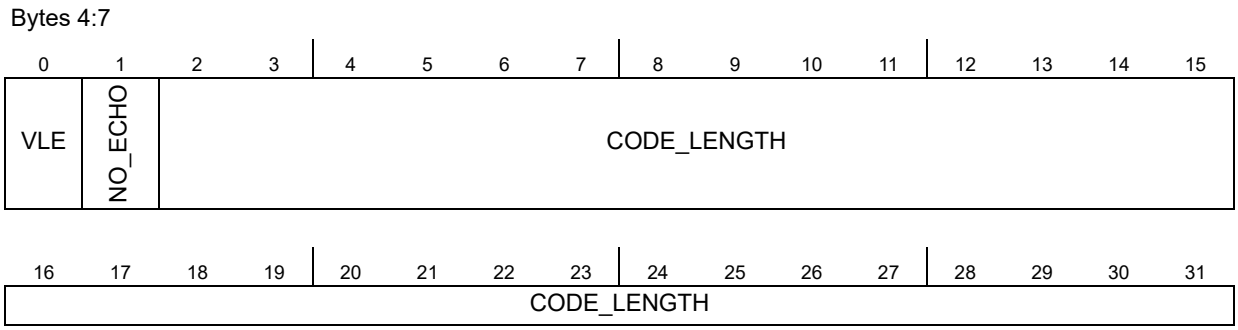


Figure 1784. VLE bit, NO_ECHO bit and code size

65.3.8.6 Download data

Each byte of data received is downloaded into device SRAM starting from the address specified in the previous protocol step. It is not verified whether the provided address is a valid address in SRAM or is writable.

The address increments until a number of bytes of data received matches the code length specified in the previous protocol step.

Since the SRAM is protected by 64-bit wide Error Correction Code (ECC), BAF always writes to SRAM grouped in 64-bit double-words. If the last byte does not fall into a 64-Bit bounder, the BAF fills it with zero's and renders the data aligned on a 64-bit bounder. After the last data is written to SRAM, the BAF writes a dummy double-word (0x0000_0000_0000_0000) to the next address location. This is written to avoid possible ECC error during core prefetch.

65.3.8.7 Execute code

The BAF program waits for the last echo message transmission to be completed. Then it restores the initial MCU configuration and jumps to the loaded code at Start Address which was received in step 2 of the protocol. At this point the BAF has finished its tasks and the MCU is controlled by new code executing from SRAM.

65.3.9 Serial boot configuration

65.3.9.1 LINFlexD configuration

Boot using UART protocol is implemented by LINFlexD module.

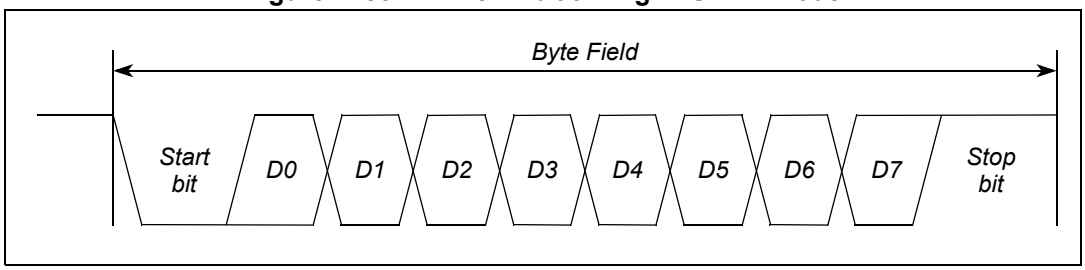
Note: Refer to the pin configuration table in Chapter 7: Device configuration for pins used by the LINFlexD module.

When Serial Boot mode is started the LINFlexD_Rx is configured as an input and multiplexed to the LINFlexD module. The LINFlexD_Tx remains in a passive input state until the first byte of protocol is received. It is then configured as an output and multiplexed to LINFlexD module. The application may not be using the Serial Boot Mode function, and may be using the pin used by LINFlexD_Tx as an input, which may have some external hardware driving the pin. Inadvertently driving this pin as an output would cause contention and possible damage.

The LINFlexD module operates in half-duplex mode. The interface is either transmitting a byte or receiving a byte, but never both at the same time. It is expected that LINFlexD pins are connected to a physical layer bus driver/receiver (PHY). Many PHYs enforce half-duplex mode by using a single data channel for both transmitting and receiving data . The side effect of this is that all data transmitted on Tx pin is also received on Rx pin. To prevent the LINFlexD module receiving its own transmission, the Rx pin is blocked during Tx pin transmission.

The LINFlexD controller is configured to operate with an 8-bit data frame without parity bit and 1 stop bit.

Figure 1785. LINFlexD bit timing in UART mode

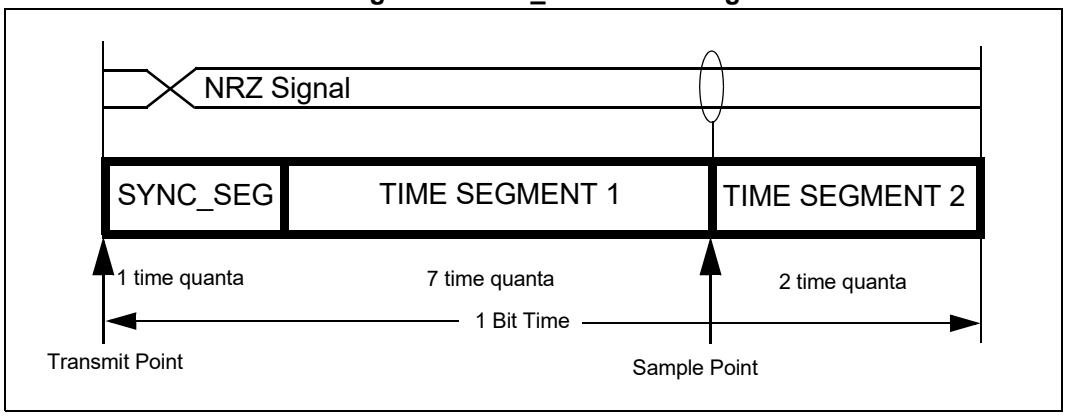


65.3.9.2 M_CAN configuration

The M_CAN controller is configured to operate at a baud rate equal to crystal frequency divided by 40, using the standard 11-bit identifier format detailed in CAN 2.0A specification (refer to *Table 1733* for baud rates).

The Bit timing is configured as shown in *Figure 1786*

Figure 1786. M_CAN Bit Timing



65.3.9.3 Baud rates

The LINFlexD baud rate is set at 250kbaud. *Table 1733* below shows some possible baud rates.

The M_CAN baud rate is set by the frequency of the external crystal/40.

Table 1733. M_CAN baud rates

Frequency	M_CAN baud rate
8 MHz	200 Kbaud
12 MHz	300 Kbaud
16 MHz	400 Kbaud
20 MHz	500 Kbaud
40 MHz	1000 Kbaud

65.3.9.4 LINFlexD and M_CAN pin configuration

Refer to *Chapter 7: Device configuration* for pin details.

65.3.9.5 Protocol summary

Table 1734 summarizes the LINFlexD protocol and BAF action during this boot mode.

Table 1734. LINFlexD serial boot mode download protocol

Protocol steps	Host message sent	BAF response message	Action
1	64-bit password (MSB first)	64-bit password (MSB first)	Password checked for validity and compared with stored password.
2	32-bit store address	32-bit store address	Load address is stored for future use.
3	VLE bit + NO_ECHO bit + 30 bits of download data size in bytes (MSB first)	VLE bit + NO_ECHO bit + 30 bits of download data size in bytes (MSB first)	Size of download is stored for future use. Verify VLE bit.
4	8 bits of raw binary data	If NO_ECHO bit is set, data is not echoed, otherwise data is echoed.	8 x 8 bits of data are packed into 64-bit double-words. These double-words are stored in SRAM starting from the Load address. The Load address increments until the number of data received and stored matches the size as specified in the previous steps.
5	None	None	Branch to downloaded code

Table 1735 shows the M_CAN serial boot protocol.

Table 1735. M_CAN serial boot mode download protocol

Protocol steps	Host message sent	BAF response message	Action
1	CAN ID 0x011 + 64-bit password	CAN ID 0x001 + 64-bit password	Password checked against fixed password (0xFEED_FACE_CAFE_BEEF). Watchdog timer is refreshed if the password check is successful.
2	CAN ID 0x012 + 32-bit store address + 32-bit (1 VLE bit, 1 NO_ECHO bit and 30 of download data size in bytes)	CAN ID 0x002 + 32-bit store address + 32-bit (1 VLE bit, 1 NO_ECHO bit and 30 of download data size in bytes)	Load address and size of download are stored for future use.
3	CAN ID 0x013 + 8 to 64-bits of raw binary data	VLE bit + 31 bit number of bytes (MSB first)	Each byte of data received is store in MCU memory, starting at the address specified in the previous step and incrementing until the amount of data received and stored, matched the size as specified in the previous step.
4	None	None	The BAF returns IO pins and CAN module to their reset state, then branches to the first address the data was stored to (As specified in step 2)
5	None	None	Branch to downloaded code

65.4 Resources accessed by BAF code execution

The BAF code execution uses several IPs to achieve the main functions.

- Magic carpet (clocks)

Several clock configurations are made depending on the boot mode of the BAF. All settings are restored before execution of the application code (in flash memory or received via serial download).

- LINFlexD and M_CAN

The LINFlexD module is only used if normal boot header search fails and a serial download boot is attempted, or from the SER_BOOT_CBACK callback function. All settings are restored before the application code in system RAM is executed.

- PFLASH

PFLASH module is used to disable BAF execution. Remapping of the external overlay memory is configured automatically under certain conditions. The two PFLASH are used.

- Interrupts

No interrupts/exceptions are used or enabled, except the Machine Check exception, to handle ECC errors.

- Security watchdog

When SWT2 is chosen as security watchdog, SWT3 is used for task monitoring purpose.

- FCCU

FCCU is used to configure Fault Reaction of SWT2 and SWT3.

- TDM

TDM for implementing TDM Diary enhancements.

- SSCM

Used to read some register bits (mirrors of DCF)

- PASS

Used for clock jitter enabling

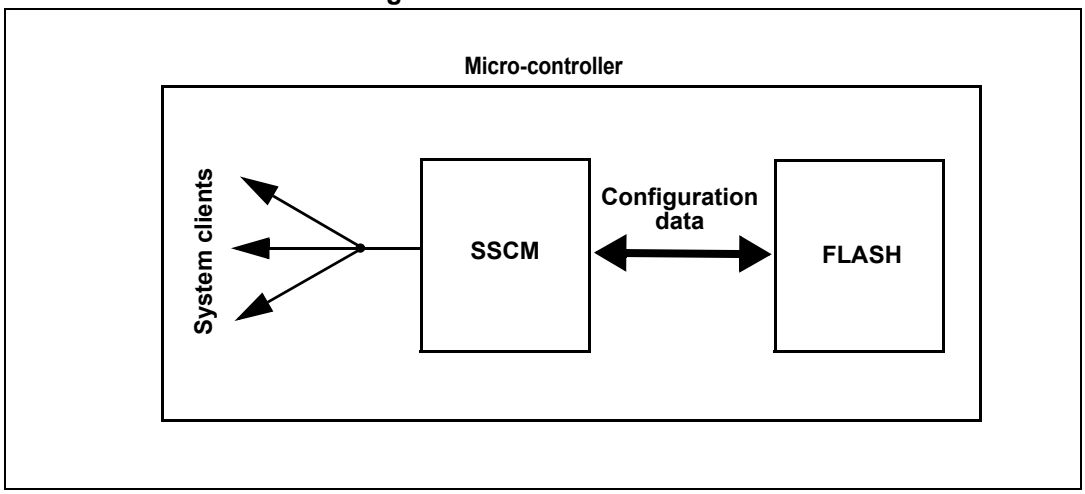
66 System Status and Configuration Module (SSCM)

66.1 Introduction

66.1.1 Overview

The System Status and Configuration Module (SSCM) is aimed at reading the system configuration from the Flash and distributing to various clients inside the micro-controller, pictured in *Figure 1787*.

Figure 1787. SSCM overview



66.1.2 Features

The SSCM includes these distinct features:

- System Configuration and Status
  - Device Mode and System Status
  - Determine primary boot vector
  - Determine HSM boot vector
  - Decodes the Device Life Cycle

66.1.3 Modes of operation

The SSCM operates identically in all system modes.

66.2 Memory map and register definition

This section provides a detailed description of all memory-mapped registers in the SSCM.

Table 1736 shows the memory map for the SSCM. Note that all addresses are offsets; the absolute address may be calculated by adding the specified offset to the base address of the SSCM.

Note: All registers are accessible via 8-bit, 16-bit or 32-bit accesses. However, 16-bit accesses must be aligned to 16-bit boundaries, and 32-bit accesses must be aligned to 32-bit

boundaries. As an example, the STATUS register is accessible by a 16-bit READ/WRITE to address Base + 0x0002, but performing a 16-bit access to Base + 0x0003 is illegal.

Table 1736. SSCM memory map

Address offset	Register	Access	Reset value	Section
0x0000	SSCM System Status (STATUS)	R/W	0xyyy0(1)	*Section 66.2.1.1*
0x0002	SSCM System Memory and ID Register (MEMCONFIG)	R	0xyyyy(1)	*Section 66.2.1.2*
0x0004–0x0005	Reserved			
0x0006	SSCM Error Configuration Register (ERROR)	R/W	0x0000	*Section 66.2.1.3*
0x0008–0x0009	Reserved(2)			
0x000A–0x001C	Reserved			
0x0020	SSCM HSM and User Option Status Register (UOPS)	R	0xyyyy_yyyy(1)	*Section 66.2.1.4*
0x0024–0x0027	Reserved			
0x0028	Processor Start Address Register (PSA)	R	0xyyyy_yyyy(1)	*Section 66.2.1.5*
0x002C–0x002F	Reserved			
0x0030	SSCM HSM Start Address Register (HSA)	R	0xyyyy_yyyy(1)	*Section 66.2.1.6*
0x0034	Life Cycle Status Register (LCSTAT)	R	0x0000_0y0y(1)	*Section 66.2.1.7*
0x0038–0x004F	Reserved			

1. Refer to register description for reset value details.
2. No register abort on this location.

66.2.1 Register descriptions

The following registers are available in the SSCM. Those bits that are shaded out are reserved for future use. To optimize future compatibility, these bits should be masked out during any read/write operations to avoid conflict with future revisions.

66.2.1.1 System Status Register (STATUS)

The System Status register reflects the current state of the system.

Offset: 0x0000 Access: Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	CER	0	NXEN1	NXEN	1	0	0	BMODE			VLE	0	0	0	0
W		w1c														
Reset	0	0	0	-(2)	-(2)	1	0	0	-(2)			-(1)	0	0	0	0

1. Reset value depends on the associated option bits in flash memory.
2. Reset value depends on the device status after leaving reset.

Figure 1788. System Status Register (STATUS)

Table 1737. STATUS field descriptions

Field	Description
1 CER	Configuration Error This field indicates that the SSCM has detected a configuration error while loading DCF clients (with PARITY Enabled). This error may have been detected during the reset sequence. 0 No configuration problem detected by the SSCM 1 Device configuration is not correct If a non-permanent (transient) error is detected, this bit can be cleared by writing a 1. If the error is permanent, the bit will reappear immediately.
3 NXEN1	Processor 1 Nexus enable status 0 Processor 1 Nexus disabled. 1 Processor 1 Nexus enabled
4 NXEN	Processor 0 Nexus enable status. 0 Processor 0 Nexus disabled. 1 Processor 0 Nexus enabled
8:10 BMODE	Device Boot Mode 000 Reserved 001 Reserved 010 Reserved 011 User Code Boot (BAF not executed) 100 Reserved 101 BAF Code Boot (BAF executed) 110 Reserved 111 Reserved This field is only updated during reset. If the device goes into standby mode and wakes up from it again, the bits retain their original value.
11 VLE	Variable Length Instruction Mode When booting in User Code Boot mode, this field indicates that the code stored in the flash memory is using the VLE instruction set. The value of this field is determined by the RCHW field of the flash memory boot sector. 0 User Code Boot Location contains standard PPC code 1 User Code Boot Location contains VLE code

66.2.1.2 System Memory and ID Register (MEMCONFIG)

The System Memory and ID register is a read-only register which reflects the memory configuration of the system. It also contains the JTAG ID.

Offset: 0x0002 Access: Read Only

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	JPIN										1	MREV				0
W																
Reset	0/1(1)	0/1(1)	0/1(1)	0/1(1)	0/1(1)	0/1(1)	0/1(1)	0/1(1)	0/1(1)	0/1(1)	1	0/1(1)	0/1(1)	0/1(1)	0/1(1)	0

1. Reset value reflects the JTAG ID of the device.

Figure 1789. System memory and ID register (MEMCONFIG)

Table 1738. MEMCONFIG field descriptions

Field	Description
0:9 JPIN	JTAG Part ID Number
11:14 MREV	Minor Mask Revision

66.2.1.3 Error Configuration Register (ERROR)

The Error Configuration register is a read-write register that controls the error handling of the system.

Offset: 0x0006 Access: Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	PAE	RAE
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1790. Error Configuration Register (ERROR)

Table 1739. ERROR field descriptions

Field	Description
14 PAE	Peripheral Bus Abort Enable This bit enables bus aborts on: – any access to a peripheral slot that is not used on the device. – illegal accesses to off-platform peripherals. On platform peripherals cannot be controlled by SSCM peripheral abort. 0 Illegal accesses to non-existing peripherals do not produce a Prefetch or Data Abort exception 1 Illegal accesses to non-existing peripherals produce a Prefetch or Data Abort exception
15 RAE	Register Bus Abort Enable This bit enables bus aborts on illegal accesses to off-platform peripherals. Illegal accesses are defined as reads or writes to reserved addresses within the address space for a particular peripheral. 0 Illegal accesses to peripherals do not produce a Prefetch or Data Abort exception 1 Illegal accesses to peripherals produce a Prefetch or Data Abort exception **Note:** Transfers to peripheral bus resources may be aborted even before they reach the peripheral bus (in example, at the PBRIDGE level). In this case, the PAE and RAE register bits have no effect on the abort.

66.2.1.4 HSM and User Option Status Register (UOPS)

Offset: 0x0020 Access: Read Only

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	BAF_FA	FL_STDBY	FL_RUN	0	0	0	0	0	0	0	0	0	HSB			HSE
W																
Reset	0/1(1)	0	0	0	0	0	0	0	0	0	0	0	0/1(1)	0/1(1)	0/1(1)	0/1(1)

1. Values are determined by DCF record and life cycle - default is 0.

Figure 1791. HSM and User Option Status Register (UOPS)

Table 1740. UOPS field descriptions

Field	Description
16 BAF_FA	BAF execution mode when bypass mode DCF is zero and lifecycle is FA 0 BAF enters an infinite loop 1 BAF executes as normal **Note:** This bit is mirrored from bit 31 of the UTEST_Miscellaneous_2 DCF client.
17 FL_STDBY	Code flash on after standby mode, despite MC_ME programming 0 Code Flash can be switched off when exiting from STANDBY mode, if programmed in MC_ME.MC_DRUN.CFLAON 1 Code Flash cannot be switched off when exiting from STANDBY mode, even if programmed in MC_ME.MC_DRUN.CFLAON **Note:** This bit is mirrored from bit 30 of UTEST_Miscellaneous_2 DCF client. When this bit is set, writing on MC_ME.MC_DRUN.CFLAON has no effect.
18 FL_RUN	Code flash on during RUNx mode, despite MC_ME programming 0 Code Flash can be switched off in RUNx mode, if programmed in MC_ME.MC_DRUx.CFLAON 1 Code Flash can be switched off in RUNx mode, even if programmed in MC_ME.MC_RUNx.CFLAON **Note:** This bit is mirrored from bit 29 of UTEST_Miscellaneous_2 DCF client. When this bit is set, writing on MC_ME.MC_RUNx.CFLAON has no effect.
28:30 HSB	HSM Boot Configuration 001 When the device is in BAF Code Boot mode, BAF waits for HSM ready flag before allowing application code to start. In User Code Boot mode the SSCM waits for HSM ready flag before allowing application code to start. Others: When the device is in BAF Code Boot mode, BAF will not wait for HSM ready flag before allowing application code to start. In User Code Boot mode the SSCM will not wait for HSM ready flag before allowing application code to start. This field can be set via DCF record (refer to the SPC58xEx/SPC58xGx Microcontroller Security Reference Manual for the HSM details).
31 HSE	HSM Enabled This bit is set to 1 if HSM is enabled via DCF record.

66.2.1.5 Processor Start Address Register (PSA)

Offset 0x0028 Access: Read Only

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	SADR(1)															
W																
Reset	0/1	0/1	0/1	0/1	0/1	0/1	0/1	0/1	0/1	0/1	0/1	0/1	0/1	0/1	0/1	0/1

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	SADR(1)															
W																
Reset	0/1	0/1	0/1	0/1	0/1	0/1	0/1	0/1	0/1	0/1	0/1	0/1	0/1	0/1	0/1	0/1

1. Reset value is the start address.

Figure 1792. Processor Start Address Register (PSA)

Table 1741. PSA field descriptions

Field	Description
0:31 SADR	Start Address The boot processor will start executing application code from this address. In User Code Boot mode this indicates the location determined by the RCHW search. In BAF Code Boot mode this will show the start address of the BAF code. In the case of a serial download, this register will point to the BAM start location.

66.2.1.6 HSM Start Address Register (HSA)

Offset 0x0030 Access: Read Only

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	HADR															
W																
Reset	0/1	0/1	0/1	0/1	0/1	0/1	0/1	0/1	0/1	0/1	0/1	0/1	0/1	0/1	0/1	0/1

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	HADR															
W																
Reset	0/1	0/1	0/1	0/1	0/1	0/1	0/1	0/1	0/1	0/1	0/1	0/1	0/1	0/1	0/1	0/1

Figure 1793. HSM Start Address Register (HSA)

Table 1742. HSA field descriptions

Field	Description
0:31 HADR	HSM Start Address HSM will start executing application code from this address. The value of this register is determined by the HSM boot header search.

66.2.1.7 Life Cycle Status Register (LCSTAT)

Offset 0x0034 Access: Read Only

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	LC		
W																
Reset	0	0	0	0	0	0	0/1(1)	0/1(1)	0	0	0	0	0	0/1(2)	0/1(2)	0/1(2)

1. Reset value reflects the Test Life Cycle of the device.
2. Reset value is the Life Cycle of the device.

Figure 1794. Life Cycle Status Register (LCSTAT)

Table 1743. LCSTAT field descriptions

Field	Description
29:31 LC	Life Cycle As determined by the SSCM by reading the LC slots. The encoding is: 000 FAIL_ANALYSIS 001 Reserved 010 PROD_OEM 011 CUST_DELIV 100 Reserved 101 Reserved 110 ST_PROD 111 IN_FIELD

66.3 Functional description

The primary purpose of the SSCM is to provide information about the current state and configuration of the system that may be useful for configuring application software and for debug of the system.

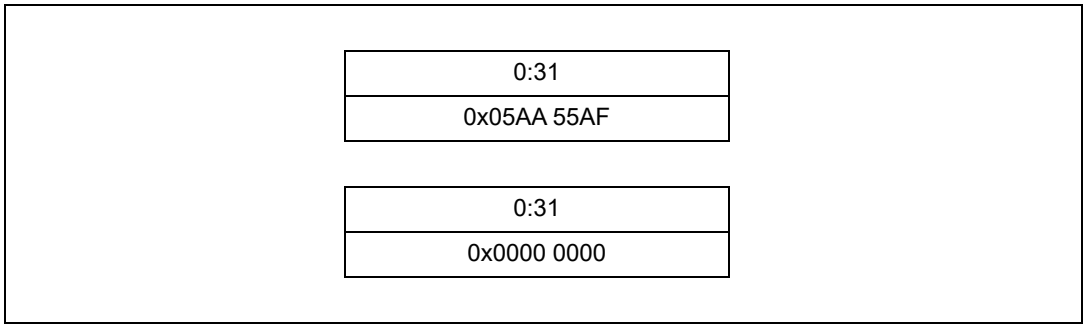
66.3.1 DCF mechanism

The DCF mechanism has been developed to handle settings of device parameters via OTP flash memory.

Starting at location UTEST_OFFSET the user can store a series of DCF records - the device will process these records during the system reset sequence before the CPU leaves reset. The first record needs to be a start record followed by an application-specific number of data records as required, and finally a stop record.

Each record is 64 bits in length. *Figure 1795* shows the DCF start record. The start record must be placed at UTEST_OFFSET to indicate to the device that the following data records should be processed.

Figure 1795. DCF start record



0:31
0x05AA 55AF

0:31
0x0000 0000

The data records are structured similar to a CPU write instruction - 15 bits of client select, 15 bits of address, 32 bits of payload data plus a STOP bit - refer to *Figure 1796*. In a data record the STOP bit must always be programmed to 0. Some clients support a parity bit.

Figure 1796. Format of a DCF data record

0:31
WDATA[31:0]

0:14	15:29	30	31
CS[14:0]	ADDR[16:2]	PRTY	STOP

The stop record indicates that processing should stop. The general format of the stop record is shown in *Figure 1797*. Only the STOP bit needs to be 1 in order to form a stop record – All other bits are ignored. An unprogrammed location in UTEST flash will be interpreted as a stop record.

Figure 1797. DCF stop record

0:31
Reserved

0:30	31
Reserved	1

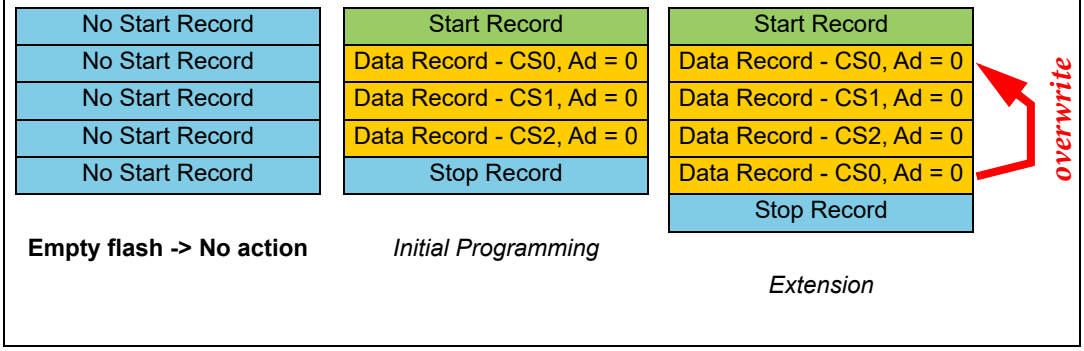
If *n* data records are to be stored in UTEST, the data structure must be as shown in *Table 1744*.

Table 1744. Series of DCF records in UTEST

ADDR offset	DATA			
0x00	0x05AA_55AF			
0x04	0x0000_0000			STOP=0
0x08	WDATA[31:0]			
0x0C	CS[14:0]	ADDR[16:2]	PRTY	STOP=0
0x10	WDATA[31:0]			
0x14	CS[14:0]	ADDR[16:2]	PRTY	STOP=0
...	...			
8n - 1 + 0x0	Reserved			
8n - 1 + 0x4	Reserved			1
8n + 0x0				
8n + 0x4				

There must never be an unprogrammed record in the data structure, as that would be interpreted as a stop record, so subsequent records would be ignored. This allows programming the records in several sessions, each time appending new records at the end of the list, as shown in *Figure 1798*.

Figure 1798. Appending DCF records



In this case, a record may overwrite a client's value which was already set by a previous record. However whether such an operation is allowed depends on the rules given for the client - some clients are write-once, only allow setting bits but not clearing, and so on.

The parity bit (PRTY) is required for some DCF records. The parity scheme used is even parity, so the number of 1s in the WDATA and PRTY fields needs to be even. (For example, if the WDATA field has the value 0x0000_0001 then the PRTY field needs to be set to '1', so that the total number of 1s is even).

66.3.2 ECC error monitoring

During flash memory scanning (such as Life Cycle or DCF records), the SSCM will monitor the flash memory status signals for ECC and other unexpected access errors. If such an error occurs the SSCM will request a destructive reset.

66.3.3 Boot mode functionality

The device supports the following boot modes for the main boot core:

- User Code Boot mode (SSCM): the device will boot from the first bootable section of the flash memory main array (BOOT_ID = 0x5A).
- BAF Code Boot: the device will boot from UTest NVM Flash Block, (interface with the HSM if available) and perform device setup (BOOT_ID = 0xA5).
- Standby-Boot : after waking up from Standby Mode, the processor will execute from the Standby RAM. The flash memory will not be accessible in this mode.

If booting is not possible with the selected configuration (for example, no Boot ID is found in the selected boot location for SSCM and BAF, that is 0x5A and 0xA5 respectively) then the device will enter serial boot mode.

If the HSM is enabled via DCF record, the HSM will boot from the location found via the HSM boot header search. Refer to *Section 66.3.6: HSM boot header search* for details.

66.3.4 BAF configuration

In BAF Code Boot mode the boot core will start to execute the BAF code at the location provided in the “Embedded Flash Memory” chapter (TestFlash block memory map). The BMODE field in the SSCM status register will be set to the value 0b101. Details of the BAF operation are described in the BAF chapter.

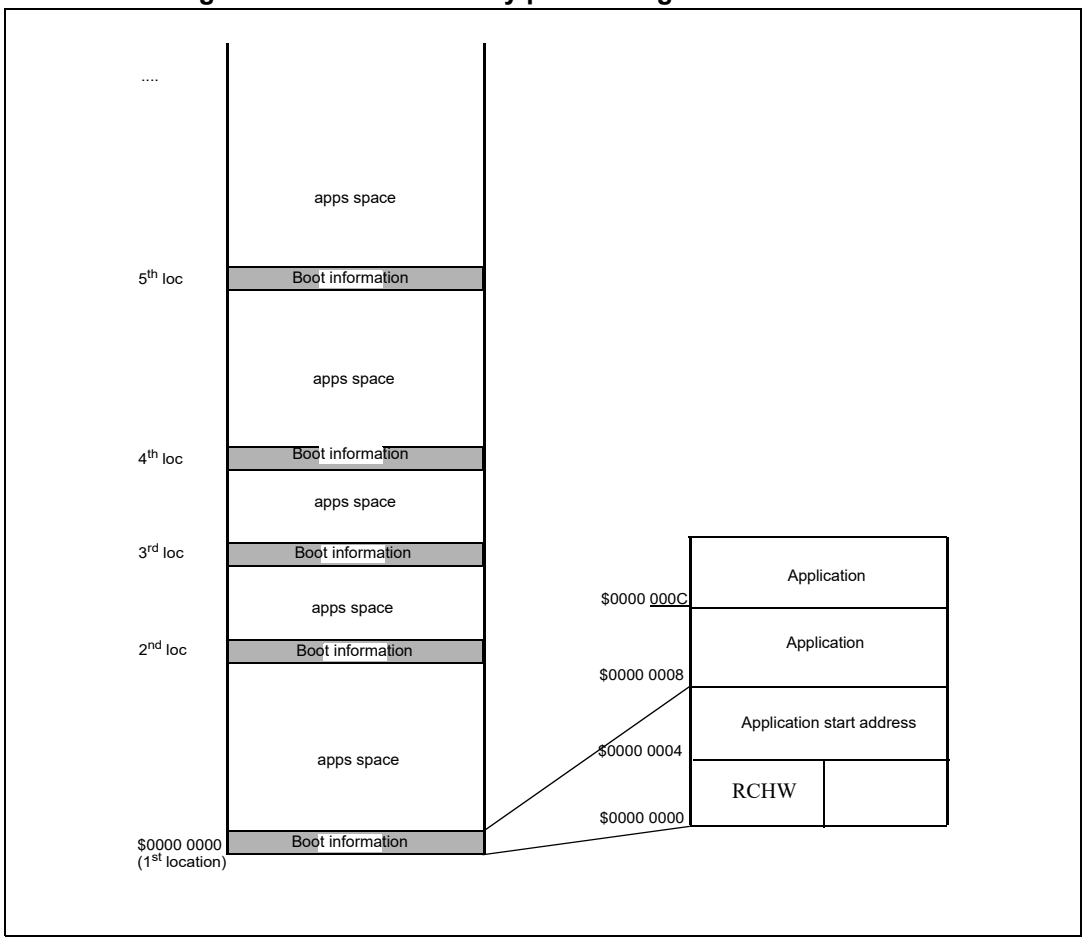
66.3.5 User code boot sector search

66.3.5.1 Potential boot sectors

As shown in *Figure 1799* in User Code Boot mode, several locations are searched for a valid boot ID. The lowest sector which starts with a valid boot ID will be used to boot the device.

The flash memory locations that are searched for boot information are shown in the “Reset and Boot” chapter (section “HW boot record search (SSCM search)”).

Figure 1799. Flash memory partitioning and RCHW search



66.3.5.2 Reset Configuration Half-Word

For each boot sector in the flash memory, SSCM checks the content of the Reset Configuration Half-Word (RCHW) at offset 0x00. If the RCHW field BOOT_ID holds the value 0x5A then the sector is considered bootable. In addition, there is a flag which indicates that the code is VLE code. (If a device only supports VLE code for future compatibility the flag must always be set to '1'.) All other bits are reserved.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	VLE	BOOT_ID							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

= Reserved

Figure 1800. Reset Configuration Half-Word

Once the device detects that it needs to boot from flash memory, and finds a valid BOOT_ID, it will boot from the application start address at offset 0x04 within the boot sector.

66.3.5.3 Standby Boot mode

In Standby Boot mode, the MCU can be configured to boot from internal flash memory (User Code Boot mode) or from Standby RAM. If the MCU boots from Standby RAM the reset sequence can be abridged - in this case the flash memory will not be available (refer to RGM block guide for mode selection).

The device security status will be stored in the SSCM Standby area, so debugging will be possible if the device is unsecured.

To enter Standby Boot mode (SBM) first the RGM needs to be configured appropriately, then low power standby mode must be entered and subsequently a wakeup interrupt must be received.

The intent of SBM is to allow quick wakeup from low power standby mode. Upon wakeup application code could perform a quick check whether the device is needed, perform a short task, then return to standby again and conserve power.

To further aid this, the device allows to boot without activating the flash memory - which significantly reduces the length of the wakeup sequence (refer to the MC_RGM chapter).

In Standby Boot mode the device will boot from Standby RAM offset 0x00. The application needs to load code to this location before going into low power standby mode. In some cases the entire code will be located here, but it is also possible to merely place a jump command at offset 0x00 and continue execution from another location (for example, the flash memory if it is activated or from the external bus).

If the device boots without enabling the flash memory, it is not possible to determine the security status again. Hence the security status is stored in the SSCM standby area, and the device will therefore have the same access restrictions as were in place before entering low power standby mode. For example, if the device was secure before entering low power standby mode, then debug will not be possible in Standby Boot mode.

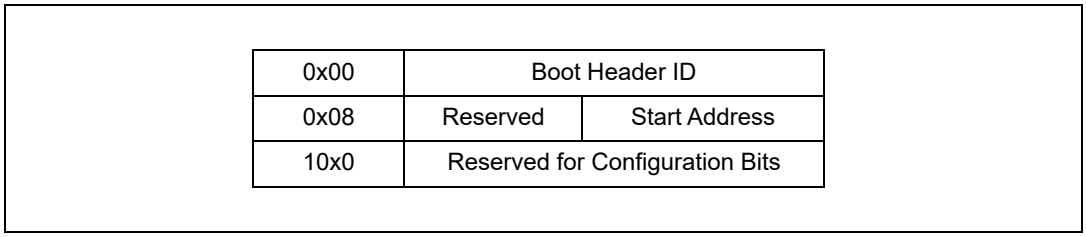
Sometimes it is desirable to activate the flash memory even though it was initially bypassed in Standby Boot mode. To do this the application needs to select a run mode which enables the flash memory - the device will transition into that run mode once the flash memory reset sequence is complete.

66.3.6 HSM boot header search

The SSCM searches the code flash memory of the HSM module for a valid boot header (refer to the SSCM section of the “Device Configuration” chapter for address(es)) and provides the boot vector to the CPU of the HSM module.

The HSM Boot Header has the format as described in *Figure 1801*.

Figure 1801. Boot header structure



A valid Boot Header ID has the value 0xFFFF_0000_FFFF_0000.

The first valid Boot Header found is the one which is passed to the HSM CPU. In order to switch from one Boot Header to a Boot Header at a higher address, the Boot Header at the lower address must be invalidated. This can be achieved by over-programming the Boot Header ID location with 0x0000_0000_0000_0000. Since this value results in the same ECC encoding it will still be possible to read the location without causing an ECC error.

66.3.7 Life Cycle

The SSCM will determine the Life Cycle of the device by reading the Life Cycle slots from UTEST flash memory area. The read operation is done during the reset phase with normal timings and it is protected by both operating monitors and ECC check. In addition a set of sanity checks executed over the LC read data guarantee the integrity of the final LC value.

At the end of the reset phase, the LC can have one of the following values:

- ST Production
- Customer Delivery
- OEM Production
- In Field
- FA

The LC is written into 5 slots, 256 bits each, at fixed locations in UTEST flash memory block. Each LC slot is read in one single atomic operation and it is organized into two fields:

- The valid field
- The invalid field

Depending on the possible combination of the data programmed into these fields, each LC slot can have one of the possible 4 status:

Table 1745. Life Cycle slots

Use case	Offset	LC slot	Valid field (128 bits)		Invalid field (128 bits)		LC slot status
1	0x00-0x10	1	Erased	Erased	Erased	Erased	Erased
2	0x00-0x10	1	Marked	Erased	Marked	Erased	Inactive
3	0x00-0x10	1	Marked	Erased	Erased	Erased	Active
4	0x00-0x10	1	Any other value				Illegal

“Marked” in this case means that the value has been programmed with the bit pattern 0x55AA_50AF_55AA_50AF, “Erased” is detected by the bit pattern 0xFFFF_FFFF_FFFF_FFFF.

Table 1746 shows how the slots are arranged in memory.

Table 1746. Life Cycle Slots in Memory

Offset	LC Slot word
0x00 - 0x07	Valid[63:0]
0x08 - 0x0F	Don't care

The Life Cycle is determined as shown in *Table 1747*. The priority of the entries is from top to bottom, so the first row which applies determines the resulting Life Cycle.

Table 1747. Resulting Life Cycle

LC Slot 0 ST_PROD(1)	LC Slot 1 CUST_DEL(1)	LC Slot 2 OEM_PROD(1)	LC Slot 3 IN_FIELD(1)	LC Slot 4 FA(1)	Resulting LifeCycle
Active	Erased	Erased	Erased	Erased	ST Production
Inactive	**Active**	Erased	Erased	Erased	Customer Delivery
Inactive	*Inactive*	**Active**	Erased	Erased	OEM Production
Inactive	*Inactive*	*Inactive*	**Active**	Erased	In Field
Inactive	*Inactive*	*Inactive*	*Inactive*	**Active**	FA
Any other combination				Erased	In Field
Any other combination					System Reset

1. Erased = all slot bits at '1'

66.4 Initialization and application information

66.4.1 Reset

The reset state of each individual bit is shown within the Register Description section (refer to *Section 66.2.1: Register descriptions*).

66.5 Additional safety measures

66.5.1 Spurious reset protection

The SSCM implements protection against spurious module resets (for example, resets which only effect the SSCM but not other modules) by gating the state machines with the expected status of the MC_RGM. In case of a spurious reset the SSCM will not interfere with the flash memory bus or overwrite configuration registers. Status internal to the SSCM will be lost, however.

67 Power management controller digital interface (PMC_Dig)

67.1 Introduction

The Control PMU manages the PMU at reset, standby entry, standby exit and during run. It checks the LVD or HVD accuracy.

The PMC analog block interfaces the digital part of the device by the PMC_Dig block.

This chapter describes the digital block of the PMU.

The PMC_Dig contains all of the registers and digital logic to generate all of the enable signals, two stage trim control bits, power on reset (POR) generation, standby entry/exit sequence and test mode logic for the analog block. The PMU digital logic also contains the controls for PMU analog outputs to be sensed with the ADC module.

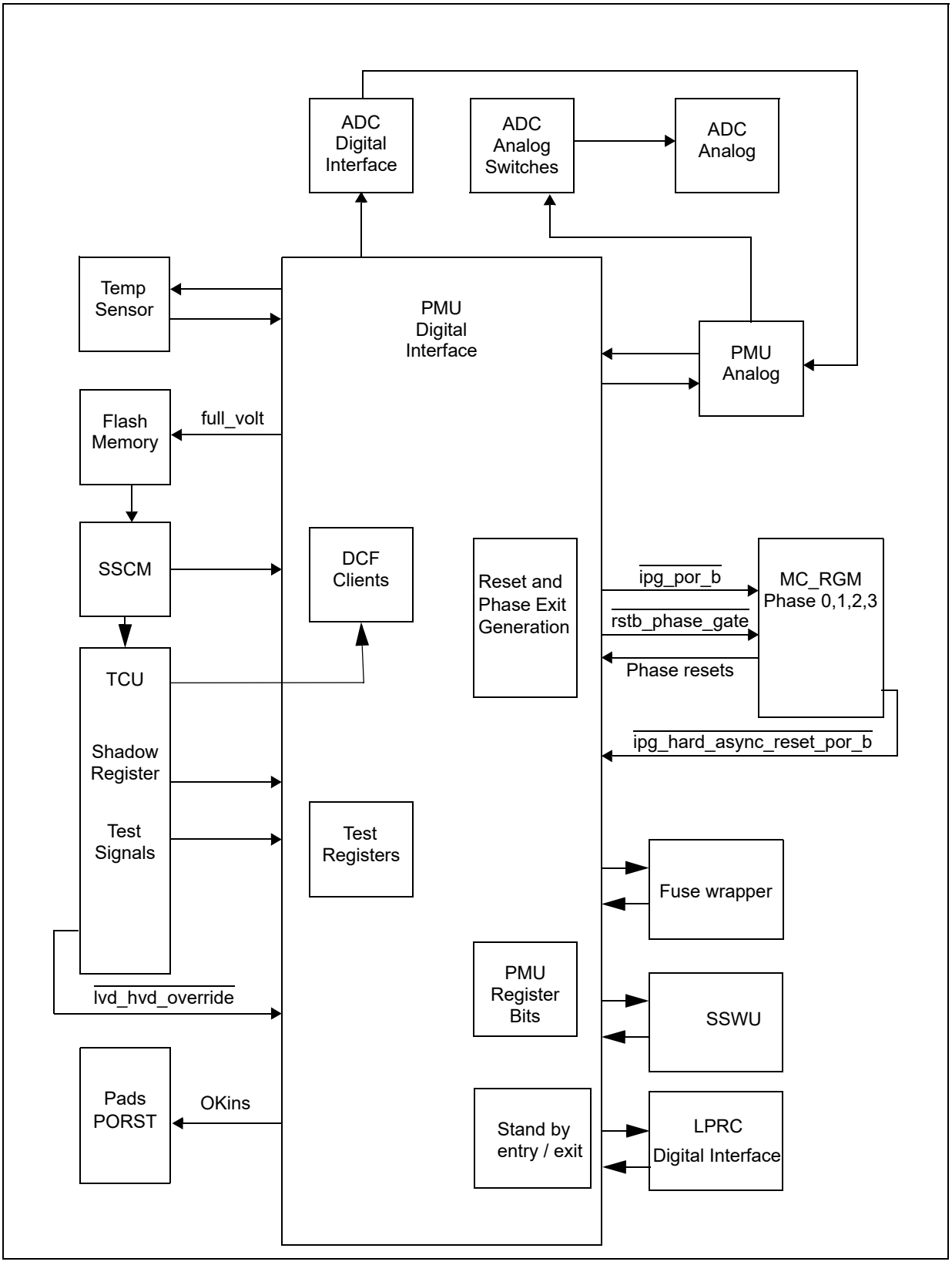
In addition, the PMU digital logic contains the enable signals, trim bits and other registers for the temperature sensor for SOC and a dedicated temperature sensor for RC regulator.

There are also several custom interfaces to various other blocks: the Reset Generation Module (MC_RGM) block receives the LVD values during POR to use for phase transition conditions and reset generation of fuse_wrapper.

There are two custom interfaces, one to Flash memory via the SSCM (and DCF client blocks) that is used to load the trim values during initial POR. The other to the fuse block via fuse_wrapper that is responsible for first stage trimming, that is to say, bandgap trimming of both regulator and monitor.

The PMU digital block generates a transfer error event to the system if the selected address is out of range of PMU digital memory map. A transfer error is not generated if an unused address within the PMU digital memory space is accessed. *Figure 1802* shows the top level block diagram of the digital PMU and its interfaces to the device modules.

Figure 1802. Digital PMU block diagram



67.2 Main features

- PMU management at a main reset
- Resets signal generation

67.2.1 Standard features

- IPS bus interface

67.3 IPS bus interface

The IPS bus interface is a slave bus used for configuration purposes via CPU. The following bus operations (contiguous byte enables) are supported:

- word (32 bits) data write/read operations to any registers
- low and high half-words (16 bits, data[31:16] or data[15:0]) data write/read operations to any registers
- byte(8 bits, data[31:24] or data[23:16] or data[15:8] or data[7:0]) data write/read operations to any registers
- any other operation (free byte enables or other operations) must be avoided

The PMC_Dig block generates a transfer error in the following cases:

- The PMU digital block generates a transfer error event to the system if the selected address is out of range of PMU digital memory map. A transfer error is not generated if an unused address within the PMU digital memory space is accessed
- any write/read operation different from byte/hword/word (free byte enables or other operations) on each register

The registers of the PMC_Dig block are accessible (read/write) in each access mode: user, supervisor or test.

About the Reset used among the registers:

- The POR reset is used on EPR_LVx, EPR_HVx, REE_LVx, REE_HVx, RES_LVx and RES_HVx registers
- The destructive reset is used on REE_TD, RES_TD, FEE_LVx, FEE_HVx and FEE_TD registers
- The functional reset is used on IE_HVx, IE_LVx, IE_G, CTL_TD registers

67.4 Memory map/register definition

Table 1748 shows the PMU memory map. The PMU includes many registers for configuring and monitoring LVD monitors and some other control registers.

Table 1748. Power Management Controller Memory Map

Offset	Register name	Location
0x0000	Event Pending Register LV0 (EPR_LV0)	*Section 67.5.1*
0x0004	Reset Event Enable LV0 (REE_LV0)	*Section 67.5.2*

Table 1748. Power Management Controller Memory Map (continued)

Offset	Register name	Location
0x0008	Reset Event Select LV0 (RES_LV0)	*Section 67.5.3*
0x000C	Interrupt Enable LV0 (IE_LV0)	*Section 67.5.4*
0x0010	FCCU Event Enable LV0 (FEE_LV0)	*Section 67.5.5*
0x0014–0x001F	***Reserved***	
0x0020	Event Pending Register LV1 (EPR_LV1)	*Section 67.5.6*
0x0024	Reset Event Enable LV1 (REE_LV1)	*Section 67.5.7*
0x0028	Reset Event Select LV1 (RES_LV1)	*Section 67.5.8*
0x002C	Interrupt Enable LV1 (IE_LV1)	*Section 67.5.9*
0x0030	FCCU Event Enable LV1 (FEE_LV1)	*Section 67.5.10*
0x0034–0x003F	***Reserved***	
0x0040	Event Pending Register HV0 (EPR_HV0)	*Section 67.5.11*
0x0044	Reset Event Enable HV0 (REE_HV0)	*Section 67.5.12*
0x0048	Reset Event Selection HV0 (RES_HV0)	*Section 67.5.13*
0x004C	Interrupt Enable HV0 (IE_HV0)	*Section 67.5.14*
0x0050	FCCU Event Enable HV0 (FEE_HV0)	*Section 67.5.15*
0x0054–0x005F	***Reserved***	
0x0060	Event Pending Register HV1 (EPR_HV1)	*Section 67.5.16*
0x0064	Reset Event Enable HV1 (REE_HV1)	*Section 67.5.17*
0x0068	Reset Event Selection HV1 (RES_HV1)	*Section 67.5.18*
0x006C	Interrupt Enable HV1 (IE_HV1)	*Section 67.5.19*
0x0070	FCCU Event Enable HV1 (FEE_HV1)	*Section 67.5.20*
0x0074–0x007F	***Reserved***	
0x0080	Supply Gauge Status Register (GR_S)	*Section 67.5.21*
0x0084	Pending Gauge Status Register (GR_P)	*Section 67.5.22*
0x0088	Global Interrupt Enable Register (IE_G)	*Section 67.5.23*
0x008C–0x0103	***Reserved***	
0x0104	Voltage selection of IO (VSIO)	*Section 67.5.24*
0x0108–0x023F	***Reserved***	
0x0240	PMOS/NMOS Regulator Status Register (PNREG_S)	*Section 67.5.25*
0x0244	HPREG SMPS Sel Status Register (HPREG_SMPS_SEL_S)	*Section 67.5.26*
0x0248	Miscellaneous Ctrl Register (MISC_CTRL_REG)	*Section 67.5.27*
0x024C	SSWU Ctrl Register (SSWU_CTRL_REG)	*Section 67.5.28*
0x0250 - 0x02FF	***Reserved***	

Table 1748. Power Management Controller Memory Map (continued)

Offset	Register name	Location
0x0300	Temperature Event Pending register (EPR_TD)	*Section 67.5.29*
0x0304	Temperature Reset Event Enable register (REE_TD)	*Section 67.5.30*
0x0308	Temperature Reset Event Selection register (RES_TD)	*Section 67.5.31*
0x030C	Temperature sensor configuration register (CTL_TD)	*Section 67.5.32*
0x0310–0x0317	***Reserved***	
0x0318	Temperature FCCU Event Enable register (FEE_TD)	*Section 67.5.33*
0x031C–0x03C7	***Reserved***	
0x03C8	BIST Mask Status register (BIST_MASK_STATUS)	*Section 67.5.34*
0x03CC	User BIST Flags register phase1(BIST_FLAGS_PHASE1)	*Section 67.5.35*
0x03D0	User BIST Flags register phase2(BIST_FLAGS_PHASE2)	*Section 67.5.36*
0x03D4	User BIST Control register (BIST_CTRL)	*Section 67.5.37*
0x03D8	User BIST Time1 and Time0 register (BIST_TIME10)	*Section 67.5.38*
0x03DC	User BIST Time3 and Time2 register (BIST_TIME32)	*Section 67.5.39*
0x03E0	User BIST Time6 and Time5 register (BIST_TIME65)	*Section 67.5.40*
0x03E4	User BIST VD Under Test Monitor Register (BIST_DEBUG)	*Section 67.5.41*

Table 1749 shows how the 32-bit registers for LV0, LV1, HV0, and HV1 are organized.

Table 1749. 32-bit registers LV0, LV1, HV0, HV1 arrangement table

Register class	BYTE 3[1] (bits [0:7])	BYTE 2[1] (bits [8:15])	BYTE 1[1] (bits [16:23])	BYTE 0[1] (bits [24-31])
LV0	VD3 = LVD100_FL_C_SB	VD2 = Reserved	VD1 = MVD082T_C	VD0 = Reserved
LV1	VD7 = Reserved	VD6 = HVD134_C	VD5 = Not implemented	VD4 = Not implemented
HV0	VD11 = LVD290_AD_AS LVD290_FL_C_IF_IJ	VD10 = Reserved	VD9 = Not implemented	VD8 = Reserved
HV1	VD15 = Reserved	VD14 = LVD400_AD_AS LVD400_IF_IJ_IM	VD13 = HVD400_C_IF_IJ	VD12 = Not implemented

1. Refer to *Section 67.5: Registers description*.

Table 1750 shows the correspondence between VD number and VD name, where HV means a 3.3 V/5.0 V supply and LV means a 1.2 V supply.

Table 1750. VD number versus VD name table

VD Number	VD Name	VD Meaning
VD0	POR031_C	LV detector asserts when PMU supply < 0.31 V
VD1	MVD082T_C	LV detector asserts when PMU supply < 0.82 V
VD2	MVD094_C	LV detector asserts when PMU supply < 0.94 V
VD2	MVD094_FA/FB	LV detector asserts when Flash supply < 0.94 V
VD3	LVD100_C	LV detector asserts when PMU supply < 1.0 V
VD3	LVD100_FL	LV detector asserts when Flash supply < 1.0 V
VD3	LVD100_SB	LV detector asserts when standby supply < 1.0 V
VD6	HVD134_C	LV detector asserts when PMU supply > 1.34 V
VD7	UVD140_C	LV detector asserts when PMU supply > 1.40 V
VD7	UVD140_FL	LV detector asserts when Flash supply > 1.40 V
VD8	POR200_C	HV detector asserts when PMU supply < 2.00 V
VD10	MVD270_C	HV detector asserts when PMU supply < 2.70 V
VD10	MVD270_FL	HV detector asserts when Flash supply < 2.70 V
VD10	MVD270_SB	HV detector asserts when standby supply < 2.70 V
VD11	LVD290_AD	HV detector asserts when sigma delta ADC supply < 2.90 V
VD11	LVD290_AS	HV detector asserts when SAR ADC supply < 2.90 V
VD11	LVD290_C	HV detector asserts when PMU supply < 2.90 V
VD11	LVD290_FL	HV detector asserts when Flash supply < 2.90 V
VD11	LVD290_IF	HV detector asserts when IO FLEXRAY supply < 2.90 V
VD11	LVD290_IJ	HV detector asserts when IO JTAG supply < 2.90 V
VD13	HVD400_C	HV detector asserts when PMU supply > 4.00 V
VD13	HVD400_IF	HV detector asserts when IO FLEXRAY supply > 4.00 V
VD13	HVD400_IJ	HV detector asserts when IO JTAG supply > 4.00 V
VD14	LVD400_AD	HV detector asserts when sigma delta ADC supply < 4.00 V
VD14	LVD400_AS	HV detector asserts when SAR ADC supply < 4.00 V
VD14	LVD400_IF	HV detector asserts when IO FLEXRAY supply < 4.00 V
VD14	LVD400_IJ	HV detector asserts when IO JTAG supply < 4.00 V
VD14	LVD400_IM	HV detector asserts when IO MAINS supply < 4.00 V
VD15	UVD600_C	HV detector asserts when PMU supply > 6.00 V
VD15	UVD600_FL	HV detector asserts when Flash supply > 6.00 V

Table 1750. VD number versus VD name table (continued)

VD Number	VD Name	VD Meaning
VD15	UVD600_IF	HV detector asserts when IO FLEXRAY supply > 6.00 V
VD15	UVD600_IJ	HV detector asserts when IO JTAG supply > 6.00 V

67.5 Registers description

67.5.1 Event Pending Register (EPR_LV0)

This Event Pending Register indicates the present and past state of the voltage detect signals in LOW VOLTAGE 0 range. If the voltage detect event has ever passed the trigger event since the last clearing event, the flag bit is set. In order to clear the flag, a one must be written to the flag bit that needs to be cleared.

Offset: 0x0000 Access: User read/write, Supervisor read/write, Test read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	LVD3_SB	0	LVD3_FL	0	LVD3_C	0	0	0	0	0	0	0	0
W				w1c		w1c		w1c								
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	MVD1_B	0	0	0	0	0	0	0	0
W								w1c								
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1803. Event Pending Register (EPR_LV0)

Table 1751. EPR_LV0 field descriptions

Field	Description
3 LVD3_SB	LVD3_SB flag This bit is the low-voltage status flag associated with LVD100_SB VD, the voltage level detect for the low voltage of 1.00 V supply always on IO ring. It is asserted when the supply falls below its corresponding Voltage Detector 3 threshold, and clears when the supply rises above its corresponding LVD3 threshold and a ‘1’ is written to this bit location. 0 Currently no occurrence 1 VD occurrence detected on the low voltage 1.00 V point supply
5 LVD3_FL	LVD3_FL flag. This bit is the low-voltage status flag associated with LVD100_FL VD, the voltage level detect for the low voltage Flash 1.00 V supply. It is asserted when the supply falls below its corresponding Voltage Detector 3 threshold, and clears when the supply rises above its corresponding LVD3 threshold and a ‘1’ is written to this bit location. 0 Currently no occurrence 1 VD occurrence detected on the low voltage 1.00 V point supply
7 LVD3_C	LVD3_C flag This bit is the low-voltage status flag associated with LVD100_C VD, the voltage level detect for the low voltage CORE 1.00 V supply. It is asserted when the supply falls below its corresponding Voltage Detector 3 threshold, and clears when the supply rises above its corresponding LVD3 threshold and a ‘1’ is written to this bit location. 0 Currently no occurrence 1 VD occurrence detected on the low voltage 1.00 V point supply
23 MVD1_B	MVD1_B flag This bit is the low-voltage status flag associated with MVD082B VD, the voltage level detect for the low voltage 0.82 V of buddy device supply. It is asserted when the supply falls below its corresponding Voltage Detector 1 threshold, and clears when the supply rises above its corresponding MVD threshold and a ‘1’ is written to this bit location. 0 Currently no occurrence 1 VD occurrence detected on the low voltage 0.82 V point supply

67.5.2 Reset Event Enable Register (REE_LV0)

This Reset Event Enable Register controls whether the voltage detect signal event causes a reset. If the desired flag bit is enabled, the voltage detect event causes a reset to be generated when the selected voltage passes the trigger event. These bits are loaded from the Flash during the boot sequence. If the Flash loaded bits are programmed to be enabled (set), these bits cannot be disabled (cleared).

Offset: 0x0004 Access: User read/write, Supervisor read/write, Test read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	REE3_SB	0	REE3_FL	0	REE3_C	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

Figure 1804. Reset Event Enable Register (REE_LV0)

Table 1752. REE_LV0 field descriptions

Field	Description
3 REE3_SB	REE3_SB reset enable This bit defines whether LVD100_SB VD, the voltage level detect for the low voltage of 1.00 V always on IO ring supply, generates a system reset. The RES3_SB bit determines whether the reset is functional or destructive. 0 Disabled. LVD assertion on the supply of the cold point does not cause system reset. 1 Enabled. LVD assertion on the supply of the cold point causes system reset.
5 REE3_FL	REE3_FL reset enable This bit defines whether LVD100_FL VD, the voltage level detect for the low voltage of 1.00 V on Flash supply, generates a system reset. The RES3_FL bit determines whether the reset is functional or destructive. 0 Disabled. LVD assertion on the supply of the cold point does not cause system reset. 1 Enabled. LVD assertion on the supply of the cold point causes system reset.
7 REE3_C	REE3_C reset enable This bit defines whether LVD100_C VD, the voltage level detect for the low voltage Core 1.00 V supply, generates a system reset. The RES3_C bit determines whether the reset is functional or destructive. 0 Disabled. LVD assertion on the supply of the cold point does not cause system reset. 1 Enabled. LVD assertion on the supply of the cold point causes system reset.

67.5.3 Reset Event Select Register (RES_LV0)

This Reset Event Select Register controls, in LOW VOLTAGE 0 range, whether the voltage detect signal event causes a destructive or functional reset. If the desired flag bit is enabled, the voltage detect event causes either a destructive or functional reset to be generated when the selected voltage passes the trigger event. These bits are loaded from the Flash during the boot sequence.

Offset: 0x0008 Access: User read/write, Supervisor read/write, Test read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	RES3_SB	0	RES3_FL	0	RES3_C	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

Figure 1805. Reset Event Select Register (RES_LV0)

Table 1753. RES_LV0 field descriptions

Field	Description
3 RES3_SB	RES3_SB reset event select This bit defines whether LVD100_SB VD, the voltage level detect for the low voltage of 1.00 V always on IO ring supply of the cold point, generates a destructive reset or a functional reset. The REE3_SB bit determines whether the reset event is enabled. 0 Destructive Reset Generated. LVD assertion on the supply of the cold point causes a destructive system reset. 1 Functional Reset Generated. LVD assertion on the supply of the cold point causes a functional system reset. For LVD100_SB it is not allowed to enable destructive reset, only functional reset is supported.
5 RES3_FL	RES3_FL reset event select This bit defines whether LVD100_FL VD, the voltage level detect for the low voltage Flash 1.00 V supply of the cold point, generates a destructive reset or a functional reset. The REE3_FL bit determines whether the reset event is enabled. 0 Destructive Reset Generated. LVD assertion on the supply of the cold point causes a destructive system reset. 1 Functional Reset Generated. LVD assertion on the supply of the cold point causes a functional system reset.
7 RES3_C	RES3_C reset event select This bit defines whether LVD100_C VD, the voltage level detect for the low voltage Core 1.00 V supply of the cold point, generates a destructive reset or a functional reset. The REE3_C bit determines whether the reset event is enabled. 0 Destructive Reset Generated. LVD assertion on the supply of the cold point causes a destructive system reset. 1 Functional Reset Generated. LVD assertion on the supply of the cold point causes a functional system reset.

67.5.4 Interrupt Enable (IE_LV0)

This configuration register contains a set of LV0 interrupt Enable bits that correspond to each of the LV0 Event Pending Registers (EPR_LV0). These bits indicate whether an interrupt is enabled waiting for a potential voltage event. A ‘0’ indicates that no interrupt is enabled, and a ‘1’ indicates that an interrupt is enabled. These bits are replicated as a read-

only monitor in the register (IE_G) at address 0x0088. The IE_LV0 bits can be read or written at any time.

Offset: 0x000C Access: User read/write, Supervisor read/write, Test read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	VD3IE_SB	0	VD3IE_FL	0	VD3IE_C	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

Figure 1806. Interrupt Enable Register (IE_LV0)

Table 1754. IE_LV0 field descriptions

Field	Description
3 VD3IE_SB	VD3IE_SB Interrupt Enable This bit determines whether an interrupt is enabled or disabled for the LVD100_SB VD, the voltage level detect for the low voltage of 1.00 V always on IO ring supply event. 0 Interrupt disabled 1 Interrupt enabled Interrupt occurrence in case of LVD happens only in normal mode (interrupt being not powered in standby mode).
5 VD3IE_FL	VD3IE_FL Interrupt Enable This bit determines whether an interrupt is enabled or disabled for the LVD100_FL VD, the voltage level detect for the low voltage Flash 1.00 V supply event. 0 Interrupt disabled 1 Interrupt enabled
7 VD3IE_C	VD3IE_C Interrupt Enable This bit determines whether an interrupt is enabled or disabled for the LVD100_C VD, the voltage level detect for the low voltage Core 1.00 V supply event. 0 Interrupt disabled 1 Interrupt enabled

67.5.5 FCCU Event Enable Register (FEE_LV0)

This FCCU Event Enable Register LV0 controls whether the voltage detect signal events cause an event signal to be sent to the FCCU (Fault Collection and Control Unit). If the desired flag bit is enabled, the voltage detect event causes an FCCU event to be generated when the selected voltage passes the trigger event.

Offset: 0x0010 Access: User read/write, Supervisor read/write, Test read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	FEE3_SB	0	FEE3_FL	0	FEE3_C	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

Figure 1807. FCCU Event Enable Register (FEE_LV0)

Table 1755. FEE_LV0 field descriptions

Field	Description
3 FEE3_SB	FEE3_SB FCCU event enable This bit defines whether a VD assertion of LVD100_SB VD, the voltage level detect for the low voltage of 1.00 V always on IO ring supply, generates an FCCU event. 0 Disabled. A VD assertion of LV0 on the supply of the always on IO ring does not cause an FCCU event. 1 Enabled. A VD assertion of LV0 on the supply of the always on IO ring causes an FCCU event. FCCU event in case of LVD happens only in normal mode (FCCU being not powered in standby mode).
5 FEE3_FL	FEE3_FL FCCU event enable This bit defines whether a VD assertion of LVD100_FL VD, the voltage level detect for the low voltage Flash 1.00 V supply, generates an FCCU event. 0 Disabled. A VD assertion of LV0 on the supply of the Flash does not cause an FCCU event. 1 Enabled. A VD assertion of LV0 on the supply of the Flash causes an FCCU event.
7 FEE3_C	FEE3_C FCCU event enable This bit defines whether a VD assertion of LVD100_C VD, the voltage level detect for the low voltage Core 1.00 V supply, generates an FCCU event. 0 Disabled. A VD assertion of LV0 on the supply of the Core does not cause an FCCU event. 1 Enabled. A VD assertion of LV0 on the supply of the Core causes an FCCU event.

67.5.6 Event Pending Register (EPR_LV1)

This Event Pending Register indicates the present and past state of the voltage detect signals in LOW VOLTAGE 1 range. If the voltage detect event has ever passed the trigger event since the last clearing event, the flag bit is set. In order to clear the flag, a ‘1’ must be written to the flag bit that needs to be cleared.

Offset: 0x0020 Access: User read/write, Supervisor read/write, Test read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	HVD6_C
W																w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

Figure 1808. Event Pending Register (EPR_LV1)

Table 1756. EPR_LV1 field descriptions

Field	Description
15 HVD6_C	HVD6_C flag This bit is the low-voltage status flag associated with HVD134_C VD, the voltage level detect for the low voltage Core 1.34 V supply. It is asserted when the supply falls below its corresponding Voltage Detector 6 threshold, and clears when the supply rises above its corresponding HVD6 threshold and a '1' is written to this bit location. 0 Currently no occurrence 1 VD occurrence detected on the low voltage 1.34 V point supply

67.5.7 Reset Event Enable Register (REE_LV1)

This Reset Event Enable Register controls, in LOW VOLTAGE 1 range, whether the voltage detect signal event causes a reset. If the desired flag bit is enabled, the voltage detect event causes a reset to be generated when the selected voltage passes the trigger event. These bits are loaded from the Flash during the boot sequence. If the Flash loaded bits are programmed to be enabled (set), these bits cannot be disabled (cleared).

Offset: 0x0024 Access: User read/write, Supervisor read/write, Test read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	REE6_C
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

Figure 1809. Reset Event Enable Register (REE_LV1)

Table 1757. REE_LV1 field descriptions

Field	Description
15 REE6_C	REE6_C reset enable This bit defines whether a HVD134_C VD, the voltage level detect for the low voltage Core 1.34 V supply of the cold point, generates a system reset. RES_LV1[RES6_C] determines whether the reset is functional or destructive. 0 Disabled. HVD assertion on the supply of the cold point does not cause system reset. 1 Enabled. HVD assertion on the supply of the cold point causes system reset.

67.5.8 Reset Event Select Register (RES_LV1)

This Reset Event Select Register controls, in LOW VOLTAGE 1 range, whether the voltage detect signal event causes a destructive or functional reset. If the desired flag bit is enabled, the voltage detect event causes either a destructive or functional reset to be generated when the selected voltage passes the trigger event. These bits are loaded from the Flash during the boot sequence.

Offset: 0x0028 Access: User read/write, Supervisor read/write, Test read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	RES6_C
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

Figure 1810. Reset Event Select Register (RES_LV1)

Table 1758. RES_LV1 field descriptions

Field	Description
15 RES6_C	RES6_C event select This bit defines whether a HVD134_C VD, the voltage level detect for the low voltage Core 1.34 V supply of the cold point, generates a destructive reset or a functional reset. REE_LV1[REE6_C] bit determines whether the reset event is enabled. 0 Destructive Reset Generated. LVD assertion on the supply of the cold point causes a destructive system reset. 1 Functional Reset Generated. LVD assertion on the supply of the cold point causes a functional system reset.

67.5.9 Interrupt Enable (IE_LV1)

This configuration register contains a set of LV1 interrupt Enable bits that correspond to each of the LV1 Event Pending Registers (EPR_LV1). These bits indicate whether an interrupt is enabled waiting for a potential voltage event. A ‘0’ indicates that no interrupt is enabled, and a ‘1’ indicates that an interrupt is enabled. These bits are replicated as a read-

only monitor in the register (IE_G) at address 0x0088. The IE_LV1 bits can be read or written at any time.

Offset: 0x002C Access: User read/write, Supervisor read/write, Test read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	HVD6IE_C
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

Figure 1811. Interrupt Enable Register (IE_LV1)

Table 1759. IE_LV1 field descriptions

Field	Description
15 HVD6IE_C	HVD6IE_C Interrupt Enable This bit determines whether an interrupt is enabled or disabled for the HVD134_C VD, the voltage level detect for the low voltage Core 1.34 V supply the voltage event occurs. 0 Interrupt disabled 1 Interrupt enabled

67.5.10 FCCU Event Enable Register (FEE_LV1)

This FCCU Event Enable Register LV1 controls whether the voltage detect signal events cause an event signal to be sent to the FCCU (Fault Collection and Control Unit). If the desired flag bit is enabled, the voltage detect event causes an FCCU event to be generated when the selected voltage passes the trigger event.

Offset: 0x0030 Access: User read/write, Supervisor read/write, Test read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	FEE6_C
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

Figure 1812. FCCU Event Enable Register (FEE_LV1)

Table 1760. FEE_LV1 field descriptions

Field	Description
15 FEE6_C	FEE6_C FCCU event enable This bit defines whether a HVD134_C VD assertion of Core 1.34 V supply generates an FCCU event. 0 Disabled. A VD assertion of LV1 on the supply of the Core does not cause an FCCU event. 1 Enabled. A VD assertion of LV1 on the supply of the Core causes an FCCU event.

67.5.11 Event Pending Register (EPR_HV0)

This Event Pending Register indicates the present and past state of the voltage detect signals in HIGH VOLTAGE 0 range. If the voltage detect event has ever passed the trigger event since the last clearing event, the flag bit is set. In order to clear the flag, a '1' must be written to the flag bit that needs to be cleared.

Offset: 0x0040 Access: User read/write, Supervisor read/write, Test read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	LVD11_AD	LVD11_AS	LVD11_IF	LVD11_IJ	0	LVD11_FL	LVD11_C	0	0	0	0	0	0	0	0
W		w1c	w1c	w1c	w1c		w1c	w1c								
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

Figure 1813. Event Pending Register (EPR_HV0)

Table 1761. EPR_HV0 field descriptions

Field	Description
1 LVD11_AD	LVD11_AD flag This bit is the high-voltage status flag associated with LVD290_AD, the voltage level detect for the low voltage Sigma Delta ADC 2.90 V supply. It is asserted when the supply falls below its corresponding Voltage Detector 11 threshold, and clears when the supply rises above its corresponding LVD11 threshold and a ‘1’ is written to this bit location. 0 Currently no occurrence 1 VD occurrence detected on the low voltage 2.90 V point supply
2 LVD11_AS	LVD11_AS flag This bit is the high-voltage status flag associated with LVD290_AS, the voltage level detect for the low voltage SAR ADC 2.90 V supply. It is asserted when the supply falls below its corresponding Voltage Detector 11 threshold, and clears when the supply rises above its corresponding LVD11 threshold and a ‘1’ is written to this bit location. 0 Currently no occurrence 1 VD occurrence detected on the low voltage 2.90 V point supply
3 LVD11_IF	LVD11_IF flag This bit is the high-voltage status flag associated with LVD290_IF, the voltage level detect for the low voltage IO Flexray 2.90 V supply. It is asserted when the supply falls below its corresponding Voltage Detector 11 threshold, and clears when the supply rises above its corresponding LVD11 threshold and a ‘1’ is written to this bit location. 0 Currently no occurrence 1 VD occurrence detected on the low voltage 2.90 V point supply
4 LVD11_IJ	LVD11_IJ flag This bit is the high-voltage status flag associated with LVD290_IJ, the voltage level detect for the low voltage IO JTAG 2.90 V supply. It is asserted when the supply falls below its corresponding Voltage Detector 11 threshold, and clears when the supply rises above its corresponding LVD11 threshold and a ‘1’ is written to this bit location. 0 Currently no occurrence 1 VD occurrence detected on the low voltage 2.90 V point supply
6 LVD11_FL	LVD11_FL flag This bit is the high-voltage status flag associated with LVD290_FL VD, the voltage level detect for the high voltage assertion on the Flash 2.90 V supply. It is asserted when the supply falls below its corresponding Voltage Detector 11 threshold, and clears when the supply rises above its corresponding LVD11 threshold and a ‘1’ is written to this bit location. 0 Currently no occurrence 1 VD occurrence detected on the low voltage 2.90 V point supply
7 LVD11_C	LVD11_C flag This bit is the high-voltage status flag associated with LVD290_C VD, the voltage level detect for the high voltage Core 2.90 V supply. It is asserted when the supply falls below its corresponding Voltage Detector 11 threshold, and clears when the supply rises above its corresponding LVD11 threshold and a ‘1’ is written to this bit location. 0 Currently no occurrence 1 VD occurrence detected on the low voltage 2.90 V point supply

67.5.12 Reset Event Enable Register (REE_HV0)

This Reset Event Enable Register controls whether the voltage detect signal event causes a reset. If the desired flag bit is enabled, the voltage detect event causes a reset to be generated when the selected voltage passes the trigger event. These bits are loaded from the Flash during the boot sequence. If the Flash loaded bits are programmed to be enabled (set), these bits cannot be disabled (cleared).

Offset: 0x0044 Access: User read/write, Supervisor read/write, Test read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	REE11_AD	REE11_AS	REE11_IF	REE11_IJ	0	REE11_FL	REE11_C	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

Figure 1814. Reset Event Enable Register (REE_HV0)

Table 1762. REE_HV0 field descriptions

Field	Description
1 REE11_AD	LVD11_AD reset enable This bit defines whether LVD290_AD VD, the voltage level detect for the high voltage assertion on the Sigma Delta ADC 2.90 V supply generates a system reset. The RES11_AD bit determines whether a system reset is enabled or not. 0 Disabled. VD assertion on the supply of the cold point does not cause system reset. 1 Enabled. VD assertion on the supply of the cold point causes system reset.
2 REE11_AS	LVD11_AS reset enable This bit defines whether LVD290_AS VD, the voltage level detect for the high voltage assertion on the SAR ADC 2.90 V supply generates a system reset. The RES11_AS bit determines whether a system reset is enabled or not. 0 Disabled. VD assertion on the supply of the cold point does not cause system reset. 1 Enabled. VD assertion on the supply of the cold point causes system reset.
3 REE11_IF	LVD11_IF reset enable This bit defines whether LVD290_IF VD, the voltage level detect for the high voltage assertion on the IO Flexray 2.90 V supply generates a system reset. The RES11_IF bit determines whether a system reset is enabled or not. 0 Disabled. VD assertion on the supply of the cold point does not cause system reset. 1 Enabled. VD assertion on the supply of the cold point causes system reset.

Table 1762. REE_HV0 field descriptions (continued)

Field	Description
4 REE11_IJ	LVD11_IJ reset enable This bit defines whether LVD290_IJ VD, the voltage level detect for the high voltage assertion on the IO JTAG 2.90 V supply generates a system reset. The RES11_IJ bit determines whether a system reset is enabled or not. 0 Disabled. VD assertion on the supply of the cold point does not cause system reset. 1 Enabled. VD assertion on the supply of the cold point causes system reset.
6 REE11_FL	LVD11_FL reset enable This bit defines whether LVD290_FL VD, the voltage level detect for the high voltage assertion on the Flash 2.90 V supply of the cold point generates a system reset. The RES11_FL bit determines whether a system reset is enabled or not. 0 Disabled. LVD assertion on the supply of the cold point does not cause system reset. 1 Enabled. LVD assertion on the supply of the cold point causes system reset.
7 REE11_C	LVD11_C reset enable This bit defines whether LVD290_C VD, the voltage level detect for the high voltage Core 2.90 V supply, of the cold point generates a system reset. The RES11_C bit determines whether a system reset is enabled or not. 0 Disabled. HVD assertion on the supply of the cold point does not cause system reset. 1 Enabled. HVD assertion on the supply of the cold point causes system reset.

67.5.13 Reset Event Select Register (RES_HV0)

This Reset Event Select Register controls, in HIGH VOLTAGE 0 range, whether the voltage detect signal event causes a destructive or functional reset. If the desired flag bit is enabled, the voltage detect event causes either a destructive or functional reset to be generated when the selected voltage passes the trigger event. These bits are loaded from the Flash during the boot sequence.

Offset: 0x0048 Access: User read/write, Supervisor read/write, Test read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	RES11_AD	RES11_AS	RES11_IF	RES11_IJ	0	RES11_FL	RES11_C	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

Figure 1815. Reset Event Select Register (RES_HV0)

Table 1763. RES_HV0 field descriptions

Field	Description
1 RES11_AD	RES11_AD reset event select This bit defines whether an LVD assertion on the Sigma Delta ADC 2.90 V supply of the cold point generates a destructive reset or a functional reset. The REE11_ad bit determines whether the reset event is enabled. 0 Destructive Reset Generated. LVD assertion on the supply of the cold point causes a destructive system reset. 1 Functional Reset Generated. LVD assertion on the supply of the cold point causes a functional system reset.
2 RES11_AS	RES11_AS reset event select This bit defines whether an LVD assertion on the SAR ADC 2.90 V supply of the cold point generates a destructive reset or a functional reset. The REE11_AS bit determines whether the reset event is enabled. 0 Destructive Reset Generated. LVD assertion on the supply of the cold point causes a destructive system reset. 1 Functional Reset Generated. LVD assertion on the supply of the cold point causes a functional system reset.
3 RES11_IF	RES11_IF reset event select This bit defines whether an LVD assertion on the IO Flexray 2.90 V supply of the cold point generates a destructive reset or a functional reset. The REE11_IF bit determines whether the reset event is enabled. 0 Destructive Reset Generated. LVD assertion on the supply of the cold point causes a destructive system reset. 1 Functional Reset Generated. LVD assertion on the supply of the cold point causes a functional system reset.
4 RES11_IJ	RES11_IJ reset event select This bit defines whether an LVD assertion on the IO JTAG 2.90 V supply of the cold point generates a destructive reset or a functional reset. The REE11_IJ bit determines whether the reset event is enabled. 0 Destructive Reset Generated. LVD assertion on the supply of the cold point causes a destructive system reset. 1 Functional Reset Generated. LVD assertion on the supply of the cold point causes a functional system reset.
6 RES11_FL	RES11_FL reset event select This bit defines whether an LVD assertion on the Flash 2.90 V supply of the cold point generates a destructive reset or a functional reset. The REE11_FL bit determines whether the reset event is enabled. 0 Destructive Reset Generated. LVD assertion on the supply of the cold point causes a destructive system reset. 1 Functional Reset Generated. LVD assertion on the supply of the cold point causes a functional system reset.
7 RES11_C	RES11_C reset event select This bit defines whether an LVD assertion on the Core 2.90 V supply of the cold point generates a destructive reset or a functional reset. The REE11_C bit determines whether the reset event is enabled. 0 Destructive Reset Generated. LVD assertion on the supply of the cold point causes a destructive system reset. 1 Functional Reset Generated. LVD assertion on the supply of the cold point causes a functional system reset.

67.5.14 Interrupt Enable (IE_HV0)

This configuration register contains a set of HV0 interrupt Enable bits that correspond to each of the HV0 Event Pending Registers (EPR_HV0). These bits indicate whether an interrupt enabled waiting for a potential voltage event. A ‘0’ indicates that no interrupt is enabled, and a ‘1’ indicates that an interrupt is enabled. These bits are replicated as a read-only monitor in the register (IE_G) at address 0x0088. The IE_HV0 bits can be read or written at any time.

Offset: 0x004C Access: User read/write, Supervisor read/write, Test read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	LVD11IE_AD	LVD11IE_AS	LVD11IE_IF	LVD11IE_IJ	0	LVD11IE_FL	LVD11IE_C	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

Figure 1816. Interrupt Enable Register (IE_HV0)

Table 1764. IE_HV0 field descriptions

Field	Description
1 LVD11IE_AD	LVD11IE_AD Interrupt Enable This bit determines whether an interrupt is enabled or disabled for the LVD290_AD VD, the voltage level detect for the high voltage assertion on the Sigma Delta ADC 2.90 V supply event occurs. 0 Interrupt disabled 1 Interrupt enabled
2 LVD11IE_AS	LVD11IE_AS Interrupt Enable This bit determines whether an interrupt is enabled or disabled for the LVD290_AS VD, the voltage level detect for the high voltage assertion on the SAR ADC 2.90 V supply event occurs. 0Interrupt disabled 1 Interrupt enabled
3 LVD11IE_IF	LVD11IE_IF Interrupt Enable This bit determines whether an interrupt is enabled or disabled for the LVD290_IF VD, the voltage level detect for the high voltage assertion on the IO Flexray 2.90 V supply event occurs. 0 Interrupt disabled 1 Interrupt enabled

Table 1764. IE_HV0 field descriptions (continued)

Field	Description
4 LVD11IE_IJ	LVD11IE_IJ Interrupt Enable This bit determines whether an interrupt is enabled or disabled for the LVD290_IJ VD, the voltage level detect for the high voltage assertion on the IO JTAG 2.90 V supply event occurs. 0 Interrupt disabled 1 Interrupt enabled
6 LVD11IE_FL	LVD11IE_FL Interrupt Enable This bit determines whether an interrupt is enabled or disabled for the LVD290_FL VD, the voltage level detect for the high voltage assertion on the Flash 2.90 V supply event occurs. 0 Interrupt disabled 1 Interrupt enabled
7 LVD11IE_C	LVD11IE_C Interrupt Enable This bit determines whether an interrupt is enabled or disabled for the IVD290_C VD, the voltage level detect for the high voltage Core 2.90 V supply event occurs. 0 Interrupt disabled 1 Interrupt enabled

67.5.15 FCCU Event Enable Register (FEE_HV0)

This FCCU Event Enable Register HV0 controls whether the voltage detect signal events cause an event signal to be sent to the FCCU (Fault Collection and Control Unit). If the desired flag bit is enabled, the voltage detect event causes an FCCU event to be generated when the selected voltage passes the trigger event.

Offset: 0x0050 Access: User read/write, Supervisor read/write, Test read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	FEE11_AD	FEE11_AS	FEE11_IF	FEE11_IJ	0	FEE11_FL	FEE11_C	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

Figure 1817. FCCU Event Enable Register (FEE_HV0)

Table 1765. FEE_HV0 field descriptions

Field	Description
1 FEE11_AD	FEE11_AD FCCU event enable This bit defines whether a LVD290_AD VD, the voltage level detect for the high voltage assertion on the Sigma Delta ADC 2.90 V supply assertion generates an FCCU event. 0 Disabled. A VD assertion of HV0 on the supply of the ADC does not cause an FCCU event. 1 Enabled. A VD assertion of HV0 on the supply of the ADC causes an FCCU event.
2 FEE11_AS	FEE11_AS FCCU event enable This bit defines whether a LVD290_AS VD, the voltage level detect for the high voltage assertion on the SAR ADC 2.90 V supply assertion generates an FCCU event. 0 Disabled. A VD assertion of HV0 on the supply of the ADC does not cause an FCCU event. 1 Enabled. A VD assertion of HV0 on the supply of the ADC causes an FCCU event.
3 FEE11_IF	FEE11_IF FCCU event enable This bit defines whether a LVD290_IF VD, the voltage level detect for the high voltage assertion on the IO Flexray 2.90 V supply assertion generates an FCCU event. 0 Disabled. A VD assertion of HV0 on the supply of the IO Flexray does not cause an FCCU event. 1 Enabled. A VD assertion of HV0 on the supply of the IO Flexray causes an FCCU event.
4 FEE11_IJ	FEE11_IJ FCCU event enable This bit defines whether a LVD290_IJ VD, the voltage level detect for the high voltage assertion on the IO JTAG 2.90 V supply assertion generates an FCCU event. 0 Disabled. A VD assertion of HV0 on the supply of the IO JTAG does not cause an FCCU event. 1 Enabled. A VD assertion of HV0 on the supply of the IO JTAG causes an FCCU event.
6 FEE11_FL	FEE11_FL FCCU event enable This bit defines whether a LVD290_FL VD, the voltage level detect for the high voltage assertion on the Flash 2.90 V supply assertion generates an FCCU event. 0 Disabled. A VD assertion of HV0 on the supply of the Flash does not cause an FCCU event. 1 Enabled. A VD assertion of HV0 on the supply of the Flash causes an FCCU event.
7 FEE11_C	FEE11_C FCCU event enable This bit defines whether a LVD290_C VD, the voltage level detect for the high voltage assertion on the Core 2.90 V supply assertion generates an FCCU event. 0 Disabled. A VD assertion of HV0 on the supply of the Core does not cause an FCCU event. 1 Enabled. A VD assertion of HV0 on the supply of the Core causes an FCCU event.

67.5.16 Event Pending Register (EPR_HV1)

This Event Pending Register indicates the present and past state of the voltage detect signals in HIGH VOLTAGE 1 range. If the voltage detect event has ever passed the trigger event since the last clearing event, the flag bit is set. In order to clear the flag, a one must be written to the flag bit that needs to be cleared.

Offset: 0x0060 Access: User read/write, Supervisor read/write, Test read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	LVD14_AD	LVD14_AS	0	LVD14_IF	LVD14_IJ	LVD14_IM	0
W										w1c	w1c		w1c	w1c	w1c	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	HVD13_IF	HVD13_IJ	HVD13_C	0	0	0	0	0	0	0	0
W						w1c	w1c	w1c								
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1818. Event Pending Register (EPR_HV1)

Table 1766. EPR_HV1 field descriptions

Field	Description
9 LVD14_AD	LVD14_AD flag This bit is the High voltage status flag associated with LVD400_AD VD, the VD for the high voltage Sigma Delta ADC 4.00 V supply. It is asserted when the supply falls below its corresponding Voltage Detector 14 threshold, and clears when the supply rises under its corresponding LVD14_AD threshold and a ‘1’ is written to this bit location. 0 Currently no occurrence. 1 VD occurrence detected on the high voltage 4.00 V point supply.
10 LVD14_AS	LVD14_AS flag This bit is the High voltage status flag associated with LVD400_AS VD, the VD for the high voltage SAR ADC 4.00 V supply. It is asserted when the supply falls below its corresponding Voltage Detector 14 threshold, and clears when the supply rises under its corresponding LVD14_AS threshold and a ‘1’ is written to this bit location. 0 Currently no occurrence. 1 VD occurrence detected on the high voltage 4.00 V point supply.

Table 1766. EPR_HV1 field descriptions (continued)

Field	Description
12 LVD14_IF	LVD14_IF flag This bit is the High voltage status flag associated with LVD400_IF VD, the VD for the high voltage IO Flexray 4.00 V supply. It is asserted when the supply falls below its corresponding Voltage Detector 14 threshold, and clears when the supply rises under its corresponding LVD14_IF threshold and a '1' is written to this bit location. 0 Currently no occurrence. 1 VD occurrence detected on the high voltage 4.00 V point supply.
13 LVD14_IJ	LVD14_IJ flag This bit is the High voltage status flag associated with LVD400_IJ VD, the VD for the high voltage IO JTAG 4.00 V supply. It is asserted when the supply falls below its corresponding Voltage Detector 14 threshold, and clears when the supply rises under its corresponding LVD14_IJ threshold and a '1' is written to this bit location. 0 Currently no occurrence. 1 VD occurrence detected on the high voltage 4.00 V point supply.
14 LVD14_IM	LVD14_IM flag This bit is the High voltage status flag associated with LVD400_IM VD, the VD for the high voltage IO main 4.00 V supply. It is asserted when the supply falls below its corresponding Voltage Detector 14 threshold, and clears when the supply rises under its corresponding LVD14_IM threshold and a '1' is written to this bit location. 0 Currently no occurrence. 1 VD occurrence detected on the high voltage 4.00 V point supply.
21 HVD13_IF	HVD13_IF flag This bit is the High Voltage status flag associated with HVD400_IF VD, the VD for the high voltage IO Flexray 4.00 V supply. It is asserted when the supply is above its corresponding Voltage Detector 13 threshold, and clears when the supply falls below its corresponding HVD13_IF threshold and a '1' is written to this bit location. 0 Currently no occurrence. 1 VD occurrence detected on the high voltage 4.00 V point supply.
22 HVD13_IJ	HVD13_IJ flag This bit is the High Voltage status flag associated with HVD400_IJ VD, the VD for the high voltage IO JTAG 4.00 V supply. It is asserted when the supply is above its corresponding Voltage Detector 13 threshold, and clears when the supply falls below its corresponding HVD13_IJ threshold and a '1' is written to this bit location. 0 Currently no occurrence. 1 VD occurrence detected on the high voltage 4.00 V point supply.
23 HVD13_C	HVD13_C flag This bit is the High Voltage status flag associated with HVD400_C VD, the VD for the high voltage Core 4.00 V supply. It is asserted when the supply is above its corresponding Voltage Detector 13 threshold, and clears when the supply below its corresponding HVD13_C threshold and a '1' is written to this bit location. 0 Currently no occurrence. 1 VD occurrence detected on the high voltage 4.00 V point supply.

67.5.17 Reset Event Enable Register (REE_HV1)

This Reset Event Enable Register controls, in HIGH VOLTAGE 1 range, whether the voltage detect signal event causes a reset. If the desired flag bit is enabled, the voltage detect event

causes a reset to be generated when the selected voltage passes the trigger event. These bits are loaded from the Flash during the boot sequence. If the Flash loaded bits are programmed to be enabled (set), these bits cannot be disabled (cleared).

Offset: 0x0064 Access: User read/write, Supervisor read/write, Test read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	REE14_AD	REE14_AS	0	REE14_IF	REE14_IJ	REE14_IM	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	REE13_IF	REE13_IJ	REE13_C	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1819. Reset Event Enable Register (REE_HV1)

Table 1767. REE_HV1 field descriptions

Field	Description
9 REE14_AD	REE14_AD reset enable This bit defines whether an LVD assertion on the high voltage supply of the cold point generates a system reset. The RES14_AD bit determines whether the reset is functional or destructive. 0 Disabled. LVD assertion on the supply of the cold point does not cause system reset. 1 Enabled. LVD assertion on the supply of the cold point causes system reset.
10 REE14_AS	REE14_AS reset enable This bit defines whether an LVD assertion on the high voltage supply of the cold point generates a system reset. The RES14_AS bit determines whether the reset is functional or destructive. 0 Disabled. LVD assertion on the supply of the cold point does not cause system reset. 1 Enabled. LVD assertion on the supply of the cold point causes system reset.
12 REE14_IF	REE14_IF reset enable This bit defines whether an LVD assertion on the high voltage supply of the cold point generates a system reset. The RES14_IF bit determines whether the reset is functional or destructive. 0 Disabled. LVD assertion on the supply of the cold point does not cause system reset. 1 Enabled. LVD assertion on the supply of the cold point causes system reset.

Table 1767. REE_HV1 field descriptions (continued)

Field	Description
13 REE14_IJ	REE14_IJ reset enable This bit defines whether an LVD assertion on the high voltage supply of the cold point generates a system reset. The RES14_IJ bit determines whether the reset is functional or destructive. 0 Disabled. LVD assertion on the supply of the cold point does not cause system reset. 1 Enabled. LVD assertion on the supply of the cold point causes system reset.
14 REE14_IM	REE14_IM reset enable This bit defines whether an LVD assertion on the high voltage supply of the cold point generates a system reset. The RES14_IM bit determines whether the reset is functional or destructive. 0 Disabled. LVD assertion on the supply of the cold point does not cause system reset. 1 Enabled. LVD assertion on the supply of the cold point causes system reset.
21 REE13_IF	REE13_IF reset enable This bit defines whether HVD400_IF VD, the VD for the high voltage IO Flexray 4.00 V supply generates a system reset. The RES13_IF bit determines whether the reset is functional or destructive. 0 Disabled. HVD assertion on the supply of the cold point does not cause system reset. 1 Enabled. HVD assertion on the supply of the cold point causes system reset.
22 REE13_IJ	REE13_IJ reset enable This bit defines whether HVD400_IJ VD, the VD for the high voltage IO JTAG 4.00 V supply generates a system reset. The RES13_IJ bit determines whether the reset is functional or destructive. 0 Disabled. HVD assertion on the supply of the cold point does not cause system reset. 1 Enabled. HVD assertion on the supply of the cold point causes system reset.
23 REE13_C	REE13_C reset enable This bit defines whether HVD400_C VD, the VD for the high voltage Core 4.00 V supply generates a system reset. The RES13_C bit determines whether the reset is functional or destructive. 0 Disabled. HVD assertion on the supply of the cold point does not cause system reset. 1 Enabled. HVD assertion on the supply of the cold point causes system reset.

67.5.18 Reset Event Select Register (RES_HV1)

This Reset Event Select Register controls, in HIGH VOLTAGE 1 range, whether the voltage detect signal event causes a destructive or functional reset. If the desired flag bit is enabled, the voltage detect event causes either a destructive or functional reset to be generated when the selected voltage passes the trigger event. These bits are loaded from the Flash during the boot sequence.

Offset: 0x0068 Access: User read/write, Supervisor read/write, Test read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	RES14_AD	RES14_AS	0	RES14_IF	RES14_IJ	RES14_IM	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	RES13_IF	RES13_IJ	RES13_C	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1820. Reset Event Select Register (RES_HV1)

Table 1768. RES_HV1 field descriptions

Field	Description
9 RES14_AD	RES14_AD reset event select This bit defines whether a LVD400_AD VD, the VD for the high voltage Sigma Delta ADC 4.00 V supply generates a destructive reset or a functional reset by LVD14_AD bit. 0 Destructive Reset Generated. LVD assertion on the supply of the cold point causes a destructive system reset. 1 Functional Reset Generated. LVD assertion on the supply of the cold point causes a functional system reset.
10 RES14_AS	RES14_AS reset event select This bit defines whether a LVD400_AS VD, the VD for the high voltage SAR ADC 4.00 V supply generates a destructive reset or a functional reset by LVD14_AS bit. 0 Destructive Reset Generated. LVD assertion on the supply of the cold point causes a destructive system reset. 1 Functional Reset Generated. LVD assertion on the supply of the cold point causes a functional system reset.
12 RES14_IF	RES14_IF reset event select This bit defines whether a LVD400_IF VD, the VD for the high voltage IO Flexray 4.00 V supply generates a destructive reset or a functional reset by LVD14_IF bit. 0 Destructive Reset Generated. LVD assertion on the supply of the cold point causes a destructive system reset. 1 Functional Reset Generated. LVD assertion on the supply of the cold point causes a functional system reset.
13 RES14_IJ	RES14_IJ reset event select This bit defines whether a LVD400_IJ VD, the VD for the high voltage IO JTAG 4.00 V supply generates a destructive reset or a functional reset by LVD14_IJ bit. 0 Destructive Reset Generated. LVD assertion on the supply of the cold point causes a destructive system reset. 1 Functional Reset Generated. LVD assertion on the supply of the cold point causes a functional system reset.

Table 1768. RES_HV1 field descriptions (continued)

Field	Description
14 RES14_IM	RES14_IM reset event select This bit defines whether a LVD400_IM VD, the VD for the high voltage IO mains 4.00 V supply generates a destructive reset or a functional reset by LVD14_IM bit. 0 Destructive Reset Generated. LVD assertion on the supply of the cold point causes a destructive system reset. 1 Functional Reset Generated. LVD assertion on the supply of the cold point causes a functional system reset.
21 RES13_IF	RES13_IF reset event select This bit defines whether a HVD400_IF VD, the VD for the high voltage IO Flexray 4.00 V supply generates a destructive reset or a functional reset by HVD13_IF bit. 0 Destructive Reset Generated. HVD assertion on the supply of the cold point causes a destructive system reset. 1 Functional Reset Generated. HVD assertion on the supply of the cold point causes a functional system reset.
22 RES13_IJ	RES13_IJ reset event select This bit defines whether a HVD400_IJ VD, the VD for the high voltage IO JTAG 4.00 V supply generates a destructive reset or a functional reset by HVD13_IJ bit. 0 Destructive Reset Generated. HVD assertion on the supply of the cold point causes a destructive system reset. 1 Functional Reset Generated. HVD assertion on the supply of the cold point causes a functional system reset.
23 RES13_C	RES13_C reset event select This bit defines whether a HVD400_C VD, the VD for the high voltage Core 4.00 V supply generates a destructive reset or a functional reset by HVD13_C bit. 0 Destructive Reset Generated. HVD assertion on the supply of the cold point causes a destructive system reset. 1 Functional Reset Generated. HVD assertion on the supply of the cold point causes a functional system reset.

67.5.19 Interrupt Enable Register (IE_HV1)

This configuration register contains a set of HV1 interrupt Enable bits that correspond to each of the HV1 Event Pending Registers (EPR_HV1). These bits indicate whether an interrupt is enabled waiting for a potential voltage event. A ‘0’ indicates that no interrupt is enabled, and a ‘1’ indicates that an interrupt is enabled. These bits are replicated as a read-only monitor in the register (IE_G) at address 0x0088. The IE_HV1 bits can be read or written at any time.

Offset: 0x006C Access: User read/write, Supervisor read/write, Test read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	LVDIE14_AD	LVDIE14_AS	0	LVDIE14_IF	LVDIE14_IJ	LVDIE14_IM	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	HVDIE13_IF	HVDIE13_IJ	HVDIE13_C	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1821. Interrupt Enable Register (IE_HV1)

Table 1769. IE_HV1 field descriptions

Field	Description
9 LVDIE14_AD	LVDIE14_AD Interrupt Enable This bit determines whether an interrupt is enabled or disabled for the LVD400_AD VD, the VD for the high voltage Sigma Delta ADC 4.00 V supply event occurs. 0 Interrupt disabled. 1 Interrupt enabled.
10 LVDIE14_AS	LVDIE14_AS Interrupt Enable This bit determines whether an interrupt is enabled or disabled for the LVD400_AS VD, the VD for the high voltage SAR ADC 4.00 V supply event occurs. 0 Interrupt disabled. 1 Interrupt enabled.
12 LVDIE14_IF	LVDIE14_IF Interrupt Enable This bit determines whether an interrupt is enabled or disabled for the LVD400_IF VD, the VD for the high voltage IO Flexray 4.00 V supply event occurs. 0 Interrupt disabled. 1 Interrupt enabled.
13 LVDIE14_IJ	LVDIE14_IJ Interrupt Enable This bit determines whether an interrupt is enabled or disabled for the LVD400_IJ VD, the VD for the high voltage IO JTAG 4.00 V supply event occurs. 0 Interrupt disabled. 1 Interrupt enabled.
14 LVDIE14_IM	LVDIE14_IM Interrupt Enable. This bit determines whether an interrupt is enabled or disabled for the LVD400_IM VD, the VD for the high voltage IO main 4.00 V supply event occurs. 0 Interrupt disabled. 1 Interrupt enabled.

Table 1769. IE_HV1 field descriptions (continued)

Field	Description
21 HVDIE13_IF	HVDIE13_IF Interrupt Enable This bit determines whether an interrupt is enabled or disabled for the HVD400_IF VD, the VD for the high voltage IO Flexray 4.00 V supply event occurs. 0 Interrupt disabled. 1 Interrupt enabled.
22 HVDIE13_IJ	HVDIE13_IJ Interrupt Enable This bit determines whether an interrupt is enabled or disabled for the HVD400_IJ VD, the VD for the high voltage IO JTAG 4.00 V supply event occurs. 0 Interrupt disabled. 1 Interrupt enabled.
23 HVDIE13_C	HVDIE13_C Interrupt Enable This bit determines whether an interrupt is enabled or disabled for the HVD400_C VD, the VD for the high voltage Core 4.00 V supply event occurs. 0 Interrupt disabled. 1 Interrupt enabled.

67.5.20 FCCU Event Enable Register (FEE_HV1)

This FCCU Event Enable Register HV1 controls whether the voltage detect signal events cause an event signal to be sent to the FCCU (Fault Collection and Control Unit). If the desired flag bit is enabled, the voltage detect event causes an FCCU event to be generated when the selected voltage passes the trigger event.

Offset: 0x0070 Access: User read/write, Supervisor read/write, Test read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	FEE14_AD	FEE14_AS	0	FEE14_IF	FEE14_IJ	FEE14_IM	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	FEE13_IF	FEE13_IJ	FEE13_C	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1822. FCCU Event Enable Register (FEE_HV1)

Table 1770. FEE_HV1 field descriptions

Field	Description
9 FEE14_AD	FEE14_AD FCCU event enable This bit defines whether a VD assertion of LV14 on LVD400_AD for the supply of the Sigma Delta ADC generates an FCCU event. 0 Disabled. A VD assertion of LV14 does not cause an FCCU event. 1 Enabled. A VD assertion of LV14 causes an FCCU event.
10 FEE14_AS	FEE14_AS FCCU event enable This bit defines whether a VD assertion of LV14 on LVD400_AS for the supply of the SAR ADC generates an FCCU event. 0 Disabled. A VD assertion of LV14 does not cause an FCCU event. 1 Enabled. A VD assertion of LV14 causes an FCCU event.
12 FEE14_IF	FEE14_IF FCCU event enable This bit defines whether a VD assertion of LV14 on LVD400_IF for the supply of the IO Flexray generates an FCCU event. 0 Disabled. A VD assertion of LV14 does not cause an FCCU event. 1 Enabled. A VD assertion of LV14 causes an FCCU event.
13 FEE14_IJ	FEE14_IJ FCCU event enable This bit defines whether a VD assertion of LV14 on LVD400_IJ for the supply of the IO JTAG generates an FCCU event. 0 Disabled. A VD assertion of LV14 does not cause an FCCU event. 1 Enabled. A VD assertion of LV14 causes an FCCU event.
14 FEE14_IM	FEE14_IM FCCU event enable This bit defines whether a VD assertion of LV14 on LVD400_IM for the supply of the IO main generates an FCCU event. 0 Disabled. A VD assertion of LV14 does not cause an FCCU event. 1 Enabled. A VD assertion of LV14 causes an FCCU event.
21 FEE13_IF	FEE13_IF FCCU event enable This bit defines whether a VD assertion of HV14 on HVD400_IF for the supply of the IO Flexray generates an FCCU event. 0 Disabled. A VD assertion of HV14 does not cause an FCCU event. 1 Enabled. A VD assertion of HV14 causes an FCCU event.
22 FEE13_IJ	FEE13_IJ FCCU event enable This bit defines whether a VD assertion of HV14 on HVD400_IJ for the supply of the IO JTAG generates an FCCU event. 0 Disabled. A VD assertion of HV14 does not cause an FCCU event. 1 Enabled. A VD assertion of HV14 causes an FCCU event.
23 FEE13_C	FEE13_C FCCU event enable This bit defines whether a VD assertion of HV14 on HVD400_C for the supply of the Core generates an FCCU event. 0 Disabled. A VD assertion of HV14 does not cause an FCCU event. 1 Enabled. A VD assertion of HV14 causes an FCCU event.

67.5.21 Supply Gauge Status Register (GR_S)

This status register contains the gauge of the indicated supply. This indicates the present state of the voltage detect events for each of the supplies. Each of the voltage detect signals for a given voltage are combined to form a single bit, refer to *Table 1771*.

Offset: 0x0080 Access: User read-only

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	TEMP_2	TEMP_1	TEMP_0	0	0	0	VD14_AD	VD14_AS	VD14_IF	VD14_IJ	VD14_IM	0	VD13_IF	VD13_IJ	VD13_C
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	VD11_AD	VD11_AS	VD11_IF	VD11_IJ	VD11_FL	VD11_C	0	VD6_C	0	VD3_SB	VD3_FL	VD3_C
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1823. Supply Gauge Status Register (GR_S)

Table 1771. GR_S field descriptions

Field	Description
1 TEMP_2	TEMP_2 flag This bit is the temperature status flag associated with the temperature for the temperature sensor point (165 °C). It is asserted when the temperature exceeds its corresponding threshold, and clears when the temperature falls below its corresponding threshold. 0 Currently no occurrence 1 Temperature occurrence detected
2 TEMP_1	TEMP_1 flag This bit is the temperature status flag associated with the temperature for the temperature sensor point (150 °C). It is asserted when the temperature exceeds its corresponding threshold, and clears when the temperature falls below its corresponding threshold. 0 Currently no occurrence 1 Temperature occurrence detected
3 TEMP_0	TEMP_0 flag This bit is the temperature status flag associated with the temperature for the temperature sensor point (-40 °C). It is asserted when the temperature falls below its corresponding threshold, and clears when the temperature is above its corresponding threshold. 0 Currently no occurrence 1 Temperature occurrence detected
7 VD14_AD	VD14_AD low-voltage detect flag This read-only bit is the low-voltage status flag associated with the voltage level detect LVD400_AD for the high voltage 4.00 V Sigma Delta ADC supply. It is asserted when any of the ADC supplies falls below the corresponding LVD threshold, and clears when all the ADC supplies are above the corresponding LVD threshold. All the 4.00 V LVDs are combined to form this bit. 0 Currently no occurrence 1 LVD occurrence detected on the high voltage 4.00 V supplies

Table 1771. GR_S field descriptions (continued)

Field	Description
8 VD14_AS	VD14_AS low-voltage detect flag This read-only bit is the low-voltage status flag associated with the voltage level detect LVD400_AS for the high voltage 4.00 V SAR ADC supply. It is asserted when any of the ADC supplies falls below the corresponding LVD threshold, and clears when all the ADC supplies are above the corresponding LVD threshold. All the 4.00 V LVDs are combined to form this bit. 0 Currently no occurrence 1 LVD occurrence detected on the high voltage 4.00 V supplies
9 VD14_IF	VD14_IF low-voltage detect flag This read-only bit is the low-voltage status flag associated with the voltage level detect LVD400_IF for the high voltage 4.00 V IO Flexray supply. It is asserted when the IO supplies falls below the corresponding LVD threshold, and clears when all the IO supplies are above the corresponding LVD threshold. All the 4.00 V LVDs are combined to form this bit. 0 Currently no occurrence 1 LVD occurrence detected on the high voltage 4.00 V supplies
10 VD14_IJ	VD14_IJ low-voltage detect flag This read-only bit is the low-voltage status flag associated with the voltage level detect LVD400_IJ for the high voltage 4.00 V IO JTAG supply. It is asserted when the IO supplies falls below the corresponding LVD threshold, and clears when all the IO supplies are above the corresponding LVD threshold. All the 4.00 V LVDs are combined to form this bit. 0 Currently no occurrence 1 LVD occurrence detected on the high voltage 4.00 V supplies
11 VD14_IM	VD14_IM low-voltage detect flag This read-only bit is the low-voltage status flag associated with the voltage level detect LVD400_IJ for the high voltage 4.00 V IO main supply. It is asserted when the IO supplies falls below the corresponding LVD threshold, and clears when all the IO supplies fall below the corresponding LVD threshold. All the 4.00 V LVDs are combined to form this bit. 0 Currently no occurrence 1 LVD occurrence detected on the high voltage 4.00 V supplies
13 VD13_IF	VD13_IF high-voltage detect flag This read-only bit is the high-voltage status flag associated with the voltage level detect HVD400_IF for the high voltage 4.00 V IO Flexray supply. It is asserted when the IO supply rises above its corresponding HVD threshold, and clears when the supply rises above its corresponding HVD threshold. 0 Currently no occurrence 1 HVD occurrence detected on the high voltage 4.00 V supply
14 VD13_IJ	VD13_IJ high-voltage detect flag This read-only bit is the high-voltage status flag associated with the voltage level detect HVD400_IJ for the high voltage 4.00 V IO JTAG supply. It is asserted when the IO supply rises above its corresponding HVD threshold, and clears when the supply rises above its corresponding HVD threshold. 0 Currently no occurrence 1 HVD occurrence detected on the high voltage 4.00 V supply

Table 1771. GR_S field descriptions (continued)

Field	Description
15 VD13_C	VD13_C high-voltage detect flag This read-only bit is the high-voltage status flag associated with the voltage level detect HVD400_C for the high voltage 4.00 V Core supply. It is asserted when the Core supply rises above its corresponding HVD threshold, and clears when the supply falls its corresponding HVD threshold. 0 Currently no occurrence 1 HVD occurrence detected on the high voltage 4.00 V supply
20 VD11_AD	VD11_AD low-voltage detect flag This read-only bit is the low-voltage status flag associated with the voltage level detect LVD290_AD for the high voltage 2.90 V Sigma Delta ADC supplies. It is asserted when any of the ADC supplies falls below the corresponding LVD threshold, and clears when all the ADC supplies rise above the corresponding LVD threshold. All of the 2.90 V LVDs are combined to form this bit. 0 Currently no occurrence 1 LVD occurrence detected on the high voltage 2.90 V supplies
21 VD11_AS	VD11_AS low-voltage detect flag This read-only bit is the low-voltage status flag associated with the voltage level detect LVD290_AD for the high voltage 2.90 V SAR ADC supplies. It is asserted when any of the ADC supplies falls below the corresponding LVD threshold, and clears when all the ADC supplies rise above the corresponding LVD threshold. All of the 2.90 V LVDs are combined to form this bit. 0 Currently no occurrence 1 LVD occurrence detected on the high voltage 2.90 V supplies
22 VD11_IF	VD11_IF low-voltage detect flag This read-only bit is the low-voltage status flag associated with the voltage level detect LVD290_IF for the high voltage 2.90 V IO Flexray supplies. It is asserted when any of the IO supplies falls below the corresponding LVD threshold, and clears when all the IO supplies rise above the corresponding LVD threshold. All of the 2.90 V LVDs are combined to form this bit. 0 Currently no occurrence 1 LVD occurrence detected on the high voltage 2.90 V supplies
23 VD11_IJ	VD11_IJ low-voltage detect flag This read-only bit is the low-voltage status flag associated with the voltage level detect LVD290_IJ for the high voltage 2.90 V IO JTAG supplies. It is asserted when any of the IO supplies falls below the corresponding LVD threshold, and clears when all the IO supplies rise above the corresponding LVD threshold. All of the 2.90 V LVDs are combined to form this bit. 0 Currently no occurrence 1 LVD occurrence detected on the high voltage 2.90 V supplies
24 VD11_FL	VD11_FL low-voltage detect flag This read-only bit is the low-voltage status flag associated with the voltage level detect LVD290_FL for the high voltage 2.90 V Flash supplies. It is asserted when any of the Flash supplies falls below the corresponding LVD threshold, and clears when all the Flash supplies rise above the corresponding LVD threshold. All of the 2.90 V LVDs are combined to form this bit. 0 Currently no occurrence 1 LVD occurrence detected on the high voltage 2.90 V supplies

Table 1771. GR_S field descriptions (continued)

Field	Description
25 VD11_C	VD11_C low-voltage detect flag This read-only bit is the low-voltage status flag associated with the voltage level detect LVD290_C for the high voltage 2.90 V Core supplies. It is asserted when any of the Core supplies falls below the corresponding LVD threshold, and clears when all the Core supplies rise above the corresponding LVD threshold. All of the 2.90 V LVDs are combined to form this bit. 0 Currently no occurrence 1 LVD occurrence detected on the high voltage 2.90 V supplies
27 VD6_C	VD6_C high-voltage detect flag This read-only bit is the high-voltage status flag associated with the voltage level detect HVD134_C for the high voltage 1.34 V Core supplies. It is asserted when any of the Core supplies rises above the corresponding HVD threshold, and clears when all the Core supplies fall below the corresponding HVD threshold. All the 1.34 V HVDs are combined to form this bit. 0 Currently no occurrence 1 HVD occurrence detected on the high voltage 1.34 V supply
29 VD3_SB	VD3_SB low-voltage detect flag This read-only bit is the low-voltage status flag associated with the voltage level detect LVD100_SB for the low voltage 1.00 V always on supplies. It is asserted when any of the always on ring supplies falls below the corresponding LVD threshold, and clears when the always on supply rises above the corresponding LVD threshold. 0 Currently no occurrence 1 LVD occurrence detected on the high voltage 1.00 V supplies
30 VD3_FL	VD3_FL low-voltage detect flag This read-only bit is the low-voltage status flag associated with the voltage level detect LVD100_FL for the high voltage 1.00 V Flash supplies. It is asserted when any of the Flash supplies falls below the corresponding LVD threshold, and clears when all the Flash supplies rise above the corresponding LVD threshold. 0 Currently no occurrence 1 LVD occurrence detected on the high voltage 1.00 V supplies
31 VD3_C	VD3_C low-voltage detect flag This read-only bit is the low-voltage status flag associated with the voltage level detect LVD100_C for the low voltage 1.00 V Core supplies. It is asserted when any of the Core supplies falls below the corresponding LVD threshold, and clears when all the Core supplies rise above the corresponding LVD threshold. 0 Currently no occurrence 1 LVD occurrence detected on the high voltage 1.00 V supplies

67.5.22 Pending Gauge Status Register (GR_P)

This configuration register contains a mirror of the contents of the Event Pending Registers, but in a format that includes one bit for each voltage detect signal. These bits indicate the state of the voltage detect events including whether a voltage level event has occurred at any time in the past but has not been cleared (via the EPR register). These bits are read only, and they are cleared by writing a one to the corresponding EPR register.

The Pending registers are only set on an active edge of an event, so a transition must be seen before they will become set.

The raw analog block voltage detect signals are also synchronized to the bus clock before being used for the pending registers.

Offset: 0x0084 Access: User read-only

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	VD14_AD	VD14_AS	VD14_IF	VD14_IJ	VD14_IM	0	VD13_IF	VD13_IJ	VD13_C
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	VD11_AD	VD11_AS	VD11_IF	VD11_IJ	VD11_FL	VD11_C	0	VD6_C	0	VD3_SB	VD3_FL	VD3_C	0	IRQ_ST	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1824. Pending Gauge Status Register (GR_P)

Table 1772. GR_P field descriptions

Field	Description
7 VD14_AD	VD14_AD Pending Monitor. Read-only register.
8 VD14_AS	VD14_AS Pending Monitor. Read-only register.
9 VD14_IF	VD14_IF Pending Monitor. Read-only register.
10 VD14_IJ	VD14_IJ Pending Monitor. Read-only register.
11 VD14_IM	VD14_IM Pending Monitor. Read-only register.
13 VD13_IF	VD13_IF Pending Monitor. Read-only register.
14 VD13_IJ	VD13_IJ Pending Monitor. Read-only register.
15 VD13_C	VD13_C Pending Monitor. Read-only register.
17 VD11_AD	VD11_AD Pending Monitor. Read-only register.
18 VD11_AS	VD11_AS Pending Monitor. Read-only register.
19 VD11_IF	VD11_IF Pending Monitor. Read-only register.

Table 1772. GR_P field descriptions (continued)

Field	Description
20 VD11_IJ	VD11_IJ Pending Monitor. Read-only register.
21 VD11_FL	VD11_FL Pending Monitor. Read-only register.
22 VD11_C	VD11_C Pending Monitor. Read-only register.
24 VD6_C	VD6_C Pending Monitor. Read-only register.
26 VD3_SB	VD3_SB Pending Monitor. Read-only register.
27 VD3_FL	VD3_FL Pending Monitor. Read-only register.
28 VD3_C	VD3_C Pending Monitor. Read-only register.
30 IRQ_ST	IRQ_ST user BIST Pending Monitor. Read-only register.

67.5.23 Global Interrupt Enable Register (IE_G)

This configuration register contains a set of Interrupt Enable bits that correspond to each of the Event Pending Registers (EPR_XX), but in a format similar to the Pending Gauge Status register (GR_P). These bits indicate whether an interrupt is enabled. A ‘0’ indicates that no interrupt is enabled, and a ‘1’ indicates that an interrupt is enabled. These bits are enabled via a write of ‘1’ to the MSB of this register (IE_EN). After this, these bits can be read or written at any time.

Offset: 0x0088 Access: User read/write, Supervisor read/write, Test read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	IE_EN	0	0	0	0	0	0	VD14IE_AD	VD14IE_AS	VD14IE_IF	VD14IE_IJ	VD14IE_IM	0	VD13IE_IF	VD13IE_IJ	VD13IE_C
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	VD11IE_AD	VD11IE_AS	VD11IE_IF	VD11IE_IJ	VD11IE_FL	VD11IE_C	0	VD6IE_C	0	VD3IE_SB	VD3IE_FL	VD3IE_C	0	IRQ_EN	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1825. Global Interrupt Enable Register (IE_G)

Table 1773. IE_G field descriptions

Field	Description
0 IE_EN	IE_EN. Interrupt Enable. This bit permits Interrupt Enable bit to be written. 0 No interrupt enables can be written. 1 Any interrupt enable can be written.
7 VD14IE_AD	VDIE14_AD enable interrupt monitor. Read-only register.
8 VD14IE_AS	VDIE14_AS enable interrupt monitor. Read-only register.
9 VD14IE_IF	VDIE14_IF enable interrupt monitor. Read-only register.
10 VD14IE_IJ	VDIE14_IJ enable interrupt monitor. Read-only register.
11 VD14IE_IM	VDIE14_IM enable interrupt monitor. Read-only register.
13 VD13IE_IF	VDIE13_IF enable interrupt monitor. Read-only register.
14 VD13IE_IJ	VDIE13_IJ enable interrupt monitor. Read-only register.
15 VD13IE_C	VDIE13_C enable interrupt monitor. Read-only register.
17 VD11IE_AD	VDIE11_AD enable interrupt monitor. Read-only register.
18 VD11IE_AS	VDIE11_AS enable interrupt monitor. Read-only register.
19 VD11IE_IF	VDIE11_IF enable interrupt monitor. Read-only register.
20 VD11IE_IJ	VDIE11_IJ enable interrupt monitor. Read-only register.
21 VD11IE_FL	VDIE11_FL enable interrupt monitor. Read-only register.
22 VD11IE_C	VDIE11_C enable interrupt monitor. Read-only register.
24 VD6IE_C	VDIE6_C enable interrupt monitor. Read-only register.
26 VD3IE_SB	VDIE3_SB enable interrupt monitor. Read-only register.
27 VD3IE_FL	VDIE3_FL enable interrupt monitor. Read-only register.

Table 1773. IE_G field descriptions (continued)

Field	Description
28 VD3IE_C	VDIE3_C enable interrupt monitor. Read-only register.
30 IRQ_EN	IRQ_EN user BIST enable interrupt monitor. Read-only register.

67.5.24 Voltage selection of IO (VSIO)

This register controls the pad operation in the different I/O supply segments. The configuration bits have to be set according to the supply voltage applied to the I/O segments in order to get the correct behavior of the input and output buffers.

Offset: 0x0104 Access: User read/write, Supervisor read/write, Test read/write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0			
W														VSIO_IM0_IM1	VSIO_IF	VSIO_IJ
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1

Figure 1826. Voltage selection of IO Register (VSIO)

Table 1774. VSIO field description

Field	Description
29 VSIO_IM0_IM1	Selects between 3.3/5 V for IM0 and IM1 IO ring. 0 Ring will operate at 3.3 V. 1 Ring will operate at 5 V.
30 VSIO_IF	Selects between 3.3/5 V for IO flexray ring. 0 Ring will operate at 3.3 V. 1 Ring will operate at 5 V.
31 VSIO_IJ	Selects between 3.3/5 V for IO JTAG ring. 0 Ring will operate at 3.3 V. 1 Ring will operate at 5 V.

67.5.25 PMOS/NMOS Regulator Status Register (PNREG_S)

This register shows the status of PMOS regulator and NMOS regulator. It also indicates internal or external regulator mode.

Offset: 0x0240 Access: User read-only

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	REG_MODE	0	FORCEON_PNMOS	PREG_ENB	NREG_ENB
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1827. PMOS/NMOS Regulator Status Register (PNREG_S)

Table 1775. PNREG_S field descriptions

Field	Description
27 REG_MODE	This bit indicates internal or external regulator mode. 0 Internal regulator mode 1 External regulator mode
29 FORCEON_PNMOS	This bit is used for the enabling of PMOS and NMOS regulator. 0 Both PMOS and NMOS regulator are off. There is no affect of PREG_ENB and NREG_ENB. 1 Both can be switched on depending on the enabling of PMOS and NMOS.
30 PREG_ENB	This bit is used to enable/disable PREG. 0 PREG is enabled (depending on FORCEON_PMOS_NMOS value). 1 PREG is off.
31 NREG_ENB	This bit is used to enable/disable NREG. 0 NREG is enabled (depending on FORCEON_PMOS_NMOS value). 1 NREG regulator is off.

67.5.26 HPREG SMPS SELECT STATUS (HPREG_SMPS_SEL_S)

This register shows the enabled/disabled status of HPREG and SMPS.

Offset: 0x0244 Access: User read-only

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	HPREG_ENB	SMPS_ENB
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

Figure 1828. HPREG SMPS SELECT STATUS register (HPREG_SMPS_SEL_S)

Table 1776. HPREG_SMPS_SEL_S field descriptions

Field	Description
30 HPREG_ENB	HPREG enable/disable status 0 HPREG enabled if HPREG_SEL pad programmed to 0 1 HPREG disabled irrespective of HPREG_SEL pad
31 SMPS_ENB	SMPS enable/disable status 0 SMPS enabled if HPREG_SEL pad is programmed to 1. 1 SMPS disabled irrespective of HPREG_SEL pad.

67.5.27 Miscellaneous CTRL REG (MISC_CTRL_REG)

This register provides the controlling of both LPRC regulator and oscillator.

Offset: 0x0248 Access: User read/write, Supervisor read/write, Test read/write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	LPRCREG_ENB	RCOSC1M_ENB	0	RESERVED(1)
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0

1. Software must keep this bit to 0.

Figure 1829. Miscellaneous CTRL REG (MISC_CTRL_REG)

Table 1777. MISC_CTRL_REG field descriptions

Field	Description
28 LPRCREG_ENB	Enabling/Disabling of LPRC regulator 0 Regulator enabled 1 Regulator disabled
29 RCOSC1M_ENB	Enabling/Disabling of LPRC oscillator (1024 KHz) 0 Oscillator enabled 1 Oscillator disabled

67.5.28 SSWU CTRL Register (SSWU_CTRL_REG)

This register provides the controlling of SSWU enable and its prescaler programming value.

Offset: 0x024C Access: User read/write, Supervisor read/write, Test read/write

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

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

Figure 1830. SSWU CTRL Register (SSWU_CTRL_REG)

Table 1778. SSWU_CTRL_REG field descriptions

Field	Description
25:31 SSWU_PRESCALER	Prescaler Value for SSWU The resultant TU prescaler is SSWU_PRESCALER + 1.

67.5.29 Event Pending Register (EPR_TD)

This Event Pending Register indicates the present and past state of the temperature sensor signals. If the temperature event has ever passed the trigger event since the last clearing event, the flag bit is set. In order to clear the flag, a ‘1’ must be written to the flag bit that needs to be cleared.

Offset: 0x0300 Access: User read/write, Supervisor read/write, Test read/write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	TEMP_2	TEMP_1	TEMP_0
W														w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1831. Event Pending Register (EPR_TD)

Table 1779. EPR_TD field descriptions

Field	Description
29 TEMP_2	TEMP_2 flag. This bit is the temperature status flag associated with the temperature for the temperature sensor point (165 °C). It is asserted when the temperature exceeds its corresponding threshold, and clears when the temperature falls below its corresponding threshold and a '1' is written to this bit location. *If the IETD_2 bit is also asserted, an interrupt is sent to the CPU*. If REE_TD[TEMP_2] is also asserted, a system reset is generated, *which clears IETD_2 and negate the interrupt request*. If REE_TD[TEMP_2] is asserted and a destructive reset is selected (via RES_TD[TEMP_2] being 0), then a destructive reset is generated. If RES_TD[TEMP_2] is set, then a functional reset is generated. 0 Currently no occurrence 1 Temperature occurrence detected
30 TEMP_1	TEMP_1 flag. This bit is the temperature status flag associated with the temperature for the temperature sensor point (150 °C). It is asserted when the temperature exceeds its corresponding threshold, and clears when the temperature falls below its corresponding threshold and a '1' is written to this bit location. *If the IETD_1 bit is also asserted, an interrupt is sent to the CPU*. If REE_TD[TEMP_1] is also asserted, a system reset is generated, *which clears IETD_1 and negate the interrupt request*. If REE_TD[TEMP_1] is asserted and a destructive reset is selected (via RES_TD[TEMP_1] being 0), then a destructive reset is generated. If RES_TD[TEMP_1] is set, then a functional reset is generated. 0 Currently no occurrence 1 Temperature occurrence detected

Table 1779. EPR_TD field descriptions (continued)

Field	Description
31 TEMP_0	TEMP_0 flag. This bit is the temperature status flag associated with the temperature for the *cold* temperature sensor point. It is asserted when the temperature exceeds its corresponding threshold, and clears when the temperature falls below its corresponding threshold and a '1' is written to this bit location. *If the IETD_0 bit is also asserted, an interrupt is sent to the CPU*. If REE_TD[TEMP_0] is also asserted, a system reset is generated, *which clears IETD_0 and negate the interrupt request*. If REE_TD[TEMP_0] is asserted and a destructive reset is selected (via RES_TD[TEMP_0] being 0), then a destructive reset is generated. If RES_TD[TEMP_0] is set, then a functional reset is generated. 0 Currently no occurrence 1 Temperature occurrence detected

67.5.30 Reset Event Enable Register (REE_TD)

This Reset Event Enable Register controls whether the temperature detect signal event causes a reset. If the desired flag bit is enabled, the temperature detect event causes a reset to be generated when the selected temperature passes the trigger event.

If the Temperature Sensor REE and RES bits are set to cause a destructive reset, then during that process the TS REE and RES bits will also get cleared by the destructive reset.

Offset: 0x0304 Access: User read/write, Supervisor read/write, Test read/write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	TEMP_2	TEMP_1	TEMP_0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1832. Reset Event Enable Register (REE_TD)

Table 1780. REE_TD field descriptions

Field	Description
29 TEMP_2	TEMP_2 reset enable. This bit defines whether a temperature detect assertion generates a system reset. The RES_TD[TEMP_2] bit determines whether the reset is functional or destructive. 0 Disabled. Temperature Sensor detect does not cause system reset. 1 Enabled. Temperature Sensor detect causes system reset.
30 TEMP_1	TEMP_1 reset enable. This bit defines whether a temperature detect assertion generates a system reset. The RES_TD[TEMP_1] bit determines whether the reset is functional or destructive. 0 Disabled. Temperature Sensor detect does not cause system reset. 1 Enabled. Temperature Sensor detect causes system reset.
31 TEMP_0	TEMP_0 reset enable. This bit defines whether a temperature detect assertion generates a system reset. The RES_TD[TEMP_0] bit determines whether the reset is functional or destructive. 0 Disabled. Temperature Sensor detect does not cause system reset. 1 Enabled. Temperature Sensor detect causes system reset.

67.5.31 Reset Event Select Register (RES_TD)

This Reset Event Select Register controls whether the temperature sensor detect signal event causes a destructive or functional reset. If the desired flag bit is enabled, the temperature sensor event causes either a destructive or functional reset to be generated when the selected temperature passes the trigger event.

If the Temperature Sensor REE and RES bits are set to cause a destructive reset, then during that process the TS REE and RES bits will also get cleared by the destructive reset.

Offset: 0x0308 Access: User read/write, Supervisor read/write, Test read/write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	TEMP_2	TEMP_1	TEMP_0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1833. Reset Event Select Register (RES_TD)

Table 1781. RES_TD field descriptions

Field	Description
29 TEMP_2	TEMP_2 reset event select. This bit defines whether a temperature detect assertion generates a destructive reset or a functional reset. The REE_TD[TEMP_2] bit determines whether the reset event is enabled. 0 Destructive Reset Generated. Temperature Sensor assertion causes a destructive system reset. 1 Functional Reset Generated. Temperature Sensor assertion causes a functional system reset.
30 TEMP_1	TEMP_1 reset event select. This bit defines whether a temperature detect assertion generates a destructive reset or a functional reset. The REE_TD[TEMP_1] bit determines whether the reset event is enabled. 0 Destructive Reset Generated. Temperature Sensor assertion causes a destructive system reset. 1 Functional Reset Generated. Temperature Sensor assertion causes a functional system reset.
31 TEMP_0	TEMP_0 reset event select. This bit defines whether an temperature detect assertion generates a destructive reset or a functional reset. The REE_TD[TEMP_0] bit determines whether the reset event is enabled. 0 Destructive Reset Generated. Temperature Sensor assertion causes a destructive system reset. 1 Functional Reset Generated. Temperature Sensor assertion causes a functional system reset.

67.5.32 Temperature Sensor Configuration Register (CTL_TD)

The Temperature Sensor Configuration Registers (CTL_TD) contains two enables that must both be enabled in order for the Temperature Sensor to be operable. It also includes two 5-bit signed trim adjustment register fields that are used to modify both TS trim values (one bus for over temperature and one bus for under temperature).

Offset: 0x030C Access: User read/write, Supervisor read/write, Test read/write

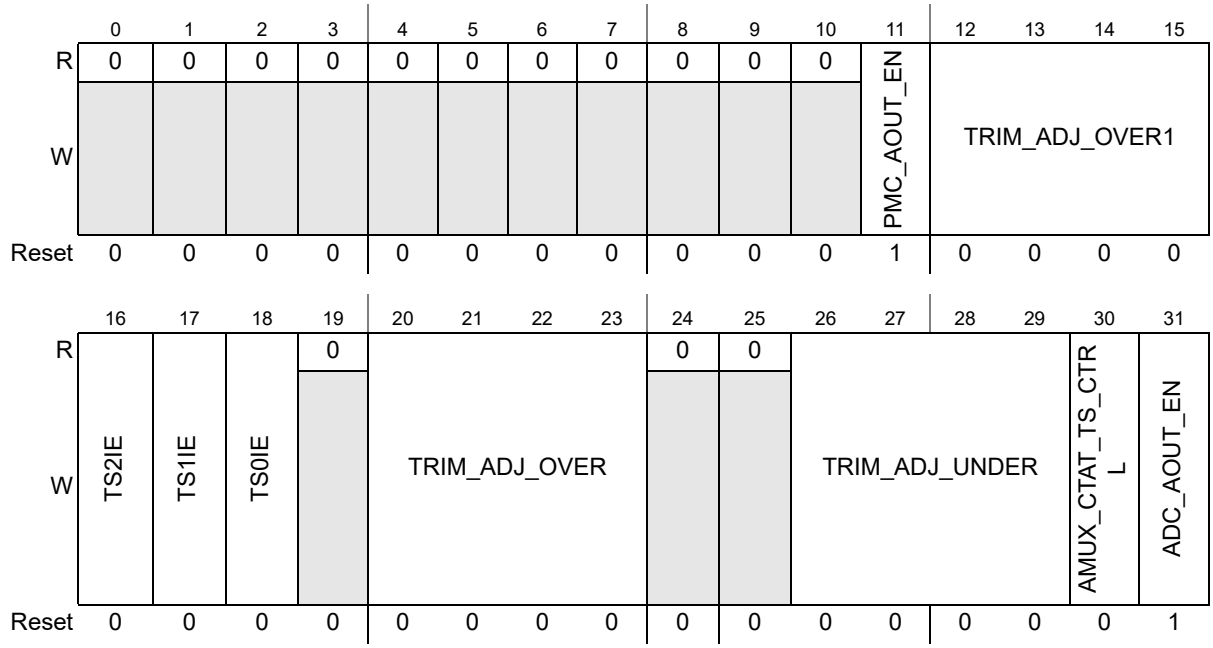


Figure 1834. Temperature Sensor Configuration Register (CTL_TD)

Table 1782. CTL_TD field descriptions

Field	Description
11 PMC_AOUT_EN	PMC_AOUT_EN. Temperature Sensor Enable for PMU. Refer to the Temperature Sensor chapter for details.
12:15 TRIM_ADJ_OVER1	TRIM_ADJ_OVER1[3:0]. Customer adjustable over trim register for PMU. These bits are a signed binary number that is added to the over temperature trim (NT_TD1) value. The TRIM value is protected from overflows and underflows due to this addition. Here MSB bit is sign bit and third bit is tied to zero, the actual trim values are on 2:0 bits. Refer to the Temperature Sensor chapter for details.
16 TS2IE	TS2IE Temperature Sensor input 2 Interrupt Enable. This bit determines whether an interrupt is seen by the system when the voltage detect event occurs. The MSB in IE_G must be set to access this bit. 0 No interrupt occurs. 1 An interrupt occurs.
17 TS1IE	TS1IE Temperature Sensor input 1 Interrupt Enable. This bit determines whether an interrupt is seen by the system when the voltage detect event occurs. The MSB in IE_G must be set to access this bit. 0 No interrupt occurs 1 An interrupt occurs
18 TS0IE	TS0IE Temperature Sensor input 0 Interrupt Enable. This bit determines whether an interrupt is seen by the system when the voltage detect event occurs. The MSB in IE_G must be set to access this bit. 0 No interrupt occurs 1 An interrupt occurs

Table 1782. CTL_TD field descriptions (continued)

Field	Description
20:23 TRIM_ADJ_OVER	TRIM_ADJ_OVER[3:0]. Customer adjustable over trim register for ADC. These bits are a signed binary number that is added to the over temperature trim (NT_TD2) value. The TRIM value is protected from overflows and underflows due to this addition. Here MSB bit is sign bit and third bit is tied to zero, the actual trim values are on 2:0 bits. Refer to the Temperature Sensor chapter for details.
26:29 TRIM_ADJ_UNDER	TRIM_ADJ_UNDER[3:0]. Customer adjustable under trim register. These bits are a signed binary number that is added to the Under temperature trim value (NT_TD0). The TRIM value is protected from overflows and underflows due to this addition. Here MSB bit is sign bit and third bit is tied to zero, the actual trim values are on 2:0 bits. Refer to the Temperature Sensor chapter for details.
30 AMUX_CTAT_TS_CTRL	AMUX_CTAT_TS_CTRL. Analog mux enable. Refer to the Temperature Sensor chapter for details.
31 ADC_AOUT_EN	ADC_AOUT_EN. Temperature Sensor Enable for ADC. Refer to the Temperature Sensor chapter for details.

67.5.33 Temperature Sensor FCCU Event Enable Register (FEE_TD)

This Temperature Sensor FCCU Event Enable Register controls whether the temperature sensor detect signal event causes an event signal to be sent to the FCCU (Fault Collection and Control Unit). If the desired flag bit (EPR_TD) is enabled, the temperature sensor detect assertion causes an FCCU event to be generated when the selected temperature exceeds the desired value.

Offset: 0x0318 Access: User read/write, Supervisor read/write, Test read/write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	FEE_TS2	FEE_TS1	FEE_TS0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1

Figure 1835. Temperature Sensor FCCU Event Enable Register (FEE_TD)

Table 1783. Temperature Sensor FEE_TD field descriptions

Field	Description
29 FEE_TS2	FEE_TS2 Temp Sensor FCCU event enable. This bit defines whether a temperature detect assertion generates an FCCU event. 0 Disabled. Temp Sensor detect does not cause an FCCU event. 1 Enabled. Temp Sensor detect causes an FCCU event.

Table 1783. Temperature Sensor FEE_TD field descriptions (continued)

Field	Description
30 FEE_TS1	FEE_TS1 Temp Sensor FCCU event enable. This bit defines whether a temperature detect assertion generates an FCCU event. 0 Disabled. Temp Sensor detect does not cause an FCCU event. 1 Enabled. Temp Sensor detect causes an FCCU event.
31 FEE_TS0	FEE_TS0 Temp Sensor FCCU event enable. This bit defines whether a temperature detect assertion generates an FCCU event. 0 Disabled. Temp Sensor detect does not cause an FCCU event. 1 Enabled. Temp Sensor detect causes an FCCU event.

67.5.34 BIST Mask Status register (BIST_MASK_STATUS)

This status register contains a monitor of the BIST MASK signal.

Offset: 0x03C8 Access: User read-only

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	BIST_MASK														
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	BIST_MASK															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1836. BIST Mask Status register (BIST_MASK_STATUS)

Table 1784. BIST_MASK_STATUS field descriptions

Field	Description
1:31 BIST_MASK	BIST mask status Bit 1: lvd100b_sb_1p2d_mask status Bit 2: mvd094b_fa_1p2d_mask status Bit 3: mvd094b_fb_1p2d_mask status Bit 4: mvd094b_c_1p2d_mask status Bit 5: lvd100b_c_1p2d_mask status Bit 6: lvd100b_fl_1p2d_mask status Bit 7: mvd270b_fl_1p2d_mask status Bit 8: mvd270b_c_main_1p2d_mask status Bit 9: mvd270b_c_stby_1p2d_mask status Bit 10: lvd290b_if_1p2d_mask status Bit 11: lvd290b_ij_1p2d_mask status Bit 12: lvd290b_fl_1p2d_mask status Bit 13: lvd290b_ad_1p2d_mask status Bit 14: lvd290b_as_1p2d_mask status Bit 15: lvd290b_c_1p2d_mask status Bit 16: lvd400b_if_1p2d_mask status Bit 17: lvd400b_ij_1p2d_mask status Bit 18: lvd400b_im_1p2d_mask status Bit 19: lvd400b_ad_1p2d_mask status Bit 20: lvd400b_as_1p2d_mask status Bit 21: mvd082b_b_1p2d_mask status Bit 22: hvd134b_c_1p2d_mask status Bit 23: hvd400b_c_1p2d_mask status Bit 24: hvd400b_if_1p2d_mask status Bit 25: hvd400b_ij_1p2d_mask status Bit 26: uvd140b_c_1p2d_mask status Bit 27: uvd140b_fl_1p2d_mask status Bit 28: uvd600b_c_1p2d_mask status Bit 29: uvd600b_fl_1p2d_mask status Bit 30: uvd600b_if_1p2d_mask status Bit 31: uvd600b_ij_1p2d_mask status

67.5.35 User BIST Flags Phase1 register (BIST_FLAGS_PHASE1)

This configuration register contains a monitor of the flags phase1 coming from the User BIST sub-block.

Offset: 0x03CC Access: User read-only

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	FLAGS_P1														
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	FLAGS_P1															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1837. User BIST Flags Phase1 register (BIST_FLAGS_PHASE1)

Table 1785. BIST_FLAGS_PHASE1 field descriptions

Field	Description
1:31 FLAGS_P1	The 31 bits of field FLAGS_P1 are the phase one flags of the User BIST that indicate the status of monitor when it is tripped: Bit 1 - UVD600B_IF Bit 2 - UVD600B_IJ Bit 3 - UVD600B_FL Bit 4 - UVD600B_C Bit 5 - HVD400B_IF Bit 6 - HVD400B_IJ Bit 7 - HVD400B_C Bit 8 - UVD140B_FL Bit 9 - UVD140B_C Bit 10 - HVD134B_C Bit 11 - MVD082B_B Bit 12 -LVD400B_IF Bit 13 - LVD400B_IJ Bit 14 -LVD400B_IM Bit 15 - LVD400B_AS Bit 16 - LVD400B_AD Bit 17 - LVD290B_IF Bit 18 - LVD290B_IJ Bit 19 - LVD290B_AS Bit 20 - LVD290B_AD Bit 21 - LVD290B_FL Bit 22 - LVD290B_C Bit 23 - MVD270B_FL Bit 24 - MVD270B_MAIN_C Bit 25 - MVD270B_C_SB Bit 26 - LVD100B_FL Bit 27 - LVD100B_C Bit 28 - MVD094B_FB Bit 29 - MVD094B_FA Bit 30 - MVD094B_C Bit 31 - LVD100B_SB

67.5.36 User BIST Flags Phase2 register (BIST_FLAGS_PHASE2)

This configuration register contains a monitor of the flags phase2 coming from the User BIST sub-block.

Offset: 0x03D0 Access: User read-only

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	FLAGS_P2														
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	FLAGS_P2															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1838. User BIST Flags Phase2 register (BIST_FLAGS_PHASE2)

Table 1786. BIST_FLAGS_PHASE2 field descriptions

Field	Description
1:31 FLAGS_P2	The 31 bits of field FLAGS_P2 are the phase two flags of the User BIST that indicate the status of monitor when it is lifted: Bit 1 - UVD600B_IF Bit 2 - UVD600B_IJ Bit 3 - UVD600B_FL Bit 4 - UVD600B_C Bit 5 - HVD400B_IF Bit 6 - HVD400B_IJ Bit 7 - HVD400B_C Bit 8 - UVD140B_FL Bit 9 - UVD140B_C Bit 10 - HVD134B_C Bit 11 - MVD082B_B Bit 12 -LVD400B_IF Bit 13 - LVD400B_IJ Bit 14 -LVD400B_IM Bit 15 - LVD400B_AS Bit 16 - LVD400B_AD Bit 17 - LVD290B_IF Bit 18 - LVD290B_IJ Bit 19 - LVD290B_AS Bit 20 - LVD290B_AD Bit 21 - LVD290B_FL Bit 22 - LVD290B_C Bit 23 - MVD270B_FL Bit 24 - MVD270B_MAIN_C Bit 25 - MVD270B_C_SB Bit 26 - LVD100B_FL Bit 27 - LVD100B_C Bit 28 - MVD094B_FB Bit 29 - MVD094B_FA Bit 30 - MVD094B_C Bit 31 - LVD100B_SB

67.5.37 User BIST Control register (BIST_CTRL)

The User BIST Control register (BIST_CTRL) contains both control bits and status bits to manage the User BIST.

Offset: 0x03D4 Access: User read/write, Supervisor read/write, Test read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	IRQST	0	0	0	IRQEN
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	NCFST				NCFEN	0	STATUS			0	0	0	START
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1839. User BIST Control register (BIST_CTRL)

Table 1787. BIST_CTRL field descriptions

Field	Description
11 IRQST	IRQST is the interrupt request pending status bit to generate a User BIST Interrupt. It is a R/W0 register. That means Set by HW and Cleared by SW. The destructive reset resets this bit. 0 No interrupt occured 1 An interrupt occured
15 IRQEN	IRQEN is the interrupt request enable bit to permit a User BIST Interrupt. It is a R/W register. The destructive reset resets this bit. 0 Interrupt is disabled 1 Interrupt enabled
19 NCFST	NCFST is the Not Critical Fault bit. The assertion of this bit will generate FCCU event once user BIST fails. It is set by HW and Cleared by SW. The destructive reset resets this bit. 0 No User BIST NCF occured 1 A User BIST NCF occured
23 NCFEN	NCFEN is the Not Critical Fault enable bit to permit a User BIST Not Critical Fault and Fault indication is sent to FCCU. It is a R/W register. The destructive reset resets these bits. 0 No User BIST NCF enabled 1 A User BIST NCF enabled

Table 1787. BIST_CTRL field descriptions (continued)

Field	Description
25:27 STATUS	STATUS register consists of 3 read-only bits, that according to the value has its own meaning. 000 BIST_IDLE 001 BIST_RUN 010 BIST_PASSED 011 BIST_FAILED 100 BIST_ABORT 101 Reserved 110 Reserved 111 Reserved
31 START	START User BIST bit. If set by SW a 1 is present in the START bit for a System clock cycle only, then the START bit register is clear by HW. The destructive reset resets this bit. 0 No start pulse. 1 A start pulse occurs of a 1 System clock cycle duration

67.5.38 User BIST Time1 and Time0 register (BIST_TIME10)

The User BIST Time1 and Time0 Register (BIST_TIME10) contains two time numbers Time1 and Time0 giving the System clock cycles number required by the User BIST timer1 and timer0 (refer to *Section 67.10* for more information regarding the User BIST timer registers).

Offset: 0x03D8 Access: User read/write, Supervisor read/write, Test read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	TIME_1											
W																
Reset	0	0	0	0	0	0	1	1	1	1	1	0	0	1	1	1

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	TIME_0											
W																
Reset	0	0	0	0	0	0	0	1	1	1	1	1	0	0	1	1

Figure 1840. User BIST Time1 and Time0 register (BIST_TIME10)

Table 1788. BIST_TIME10 field descriptions

Field	Description
4:15 TIME_1	TIME_1 is the system clock cycles number required by the TIMER 1 in the User BIST management. The destructive reset resets these bits.
20:31 TIME_0	TIME_0 is the system clock cycles number required by the TIMER 0 in the User BIST management. The destructive reset resets these bits.

67.5.39 User BIST Time3 and Time2 register (BIST_TIME32)

The User BIST Time3 and Time2 Register (BIST_TIME32) contains two time numbers Time3 and Time2 giving the system clock cycles number required by the User BIST timer3 and timer2 (refer to *Section 67.10* for more information regarding the User BIST timer registers).

Offset: 0x03DC Access: User read/write, Supervisor read/write, Test read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	TIME_3											
W																
Reset	0	0	0	0	0	0	1	1	1	1	1	0	0	1	1	1

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	TIME_2											
W																
Reset	0	0	0	0	0	0	1	1	1	1	1	0	0	1	1	1

Figure 1841. User BIST Time3 and Time2 register (BIST_TIME32)

Table 1789. BIST_TIME32 field descriptions

Field	Description
4:15 TIME_3	TIME_3 is the system clock cycles number required by the TIMER 3 in the User BIST management. The destructive reset resets these bits.
20:31 TIME_2	TIME_2 is the system clock cycles number required by the TIMER 2 in the User BIST management. The destructive reset resets these bits.

67.5.40 User BIST Time6 and Time5 register (BIST_TIME65)

The User BIST Time6 and Time5 Register (BIST_TIME65) contains two time numbers Time6 and Time5 giving the system clock cycles number required by the User BIST timer6 and timer5 (refer to *Section 67.10* for more information regarding the User BIST timer registers).

Offset: 0x03E0 Access: User read/write, Supervisor read/write, Test read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	TIME_6											
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	TIME_5											
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1

Figure 1842. User BIST Time6 and Time5 register (BIST_TIME65)

Table 1790. BIST_TIME65 field descriptions

Field	Description
4:15 TIME_6	TIME_6 is the system clock cycles number required by the TIMER 6 in the User BIST management. The destructive reset resets these bits.
20:31 TIME_5	TIME_5 is the system clock cycles number required by the TIMER 5 in the User BIST management. The destructive reset resets these bits.

67.5.41 User BIST VD Under Test Monitor register (BIST_DEBUG)

This VD Under Test Monitor register contains a monitor of the Voltage Detector Under Test of the USER BIST sub-block.

Note: The BIST_DEBUG register can only be read while BIST is running. Once BIST is finished the status can be obtained by checking the BIST_FLAGS_PHASE1 and BIST_FLAGS_PHASE2 registers.

Offset: 0x03E4 Access: User read-only

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	VD_MON					
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1843. User BIST VD Under Test Monitor register (BIST_DEBUG)

Table 1791. BIST_DEBUG field description

Field	Description
26:31 VD_MON	The six read only bits of field VD_MON give the current VD Under Test of the User BIST unit, among 31 VDs available, according to the following values: 0x1F: LVD100_SB 0x1E: MVD094_C 0x1D: MVD094_FA 0x1C: MVD094_FB 0x1B: LVD100_C 0x1A: LVD100_FL 0x16: MVD270_C_STBY 0x15: MVD270_C 0x14: MVD270_FL 0x13: LVD290_C 0x12: LVD290_FL 0x11: LVD290_AD 0x10: LVD290_AS 0x0F: LVD290_IJ 0x0E: LVD290_IF 0x0A: LVD400_AD 0x09: LVD400_AS 0x08: LVD400_IM 0x07: LVD400_IJ 0x06: LVD400_IF 0x01: MVD082_B 0x19: HVD134_C 0x18: UVD140_C 0x17: UVD140_FL 0x0D: HVD400_C 0x0C: HVD400_IJ 0x0B: HVD400_IF 0x05: UVD600_C 0x04: UVD600_FL 0x03: UVD600_IJ 0x02: UVD600_IF 0x00: No VD under Test

67.6 Standby entry/exit

Standby entry is an application initiated mode.The standby entry process starts once the application configures [MC_ME_MCTL] register for target mode as standby. For detailed description of entry/exit, refer to MC_PCU.

67.7 Temperature Sensor Interface Logic

The Temperature Sensor interface consists of an analog enable, a set of mode registers, and an overtemp threshold select control.

For more information regarding the Temperature Sensor function, refer to the *Chapter 43: Temperature Sensor*.

67.8 Power On Reset (MC_RGM Phase Gates)

The PMU digital logic interfaces to the Reset Generation Module in order to indicate the type and conditions of the resets from the PMU and Temperature Sensor Analog blocks.

There are two signals for both the PMU and Temp Sensor blocks; one is used to indicate a destructive reset, and one is used to indicate a functional reset.

The PMU digital logic also indicates when the RGM allows transition to the next POR phase.

The digital PMU also uses signals from the RGM to indicate when some of the reset signals should be masked during the POR sequence.

During the POR phase 3 exit, the PMU digital logic uses all the enabled Voltage Detects to determine if it is correct to exit from Phase 3. Once all the enabled LVDs and HVDs have become deasserted and the time from the LVD circuits to adjust to new trim values has elapsed, then the signal is sent to the RGM that the correct voltage levels have been achieved to exit from Phase 3.

67.9 Flash interface (DCF client usage)

The PMU digital is updated during POR from the Flash block via a DCF client. This load is used for several pieces of data, but most of the data is trim values being used for analog modules (PMU, Temperature Sensor, and ADC Bandgap).

These trim bits that are loaded from the Flash are not readable via the IP bus. These trim values loaded via the DCF client are not compacted (they are written individually, and not several at a time).

A DCF client is used to interface to the Flash for these early data accesses. This client also performs triple voting to keep the data safe from stray bit flips.

Another set of data that is loaded during the DCF single read process are updated values for the REE and RES bits. Refer to *Section 67.5.30: Reset Event Enable Register (REE_TD)* for more information regarding the REE/RES bits load from Flash.

If the REE bit is loaded via the DCF as set (enabled), then it is not possible to clear this REE bit. The reset value of the REE DCF bits are all low (thus to allow the bits to be rewriteable) and same is the reset value of the REE register.

The monitors for which reactions are enabled will be programmed in the corresponding bit flag of Reset Event Enable (REE_XX) register as “1” (reset enabled) and in the Reset Event Select (RES_XX) register as “0” (destructive reset selected).

The monitors for which reactions are not enabled will be programmed in the corresponding bit flag of Reset Event Enable (REE_XX) register as “0” (reset disabled) and in the Reset Event Select (RES_XX) register as “1” (functional reset selected).

The REE DCF bits loaded are compacted into a single DCF write (they do not match the IPS Bus register bit locations). The REE DCF loaded bits do not have the same limitation that they can only be written once and any further writes must be to the same value. The REE DCF bits can be rewritten to a new value during a new reset event.

When the REE DCF bits are updated, the REE and RES register bits are updated at the same time to the new value.

All of the DCF clients are reset via the POR reset signal.

For more information regarding the DCF records, refer to the DCF Records chapter.

The trim data is read out of the Flash's differential read space (slower access, but less issues with voltage variances). Once the differential reads are completed, the Flash indicates this to the PMU digital via the SSCM module.

Once the PMU trim values have been read, the analog circuit has been given time to update to the new trim values, and the LVDs have all deasserted, a signal is asserted to the Flash to indicate that the voltage is high enough for faster single access reads.

All of the LVD signals must indicate that a valid voltage has been reached before the voltage indication is sent to the Flash.

67.10 Voltage monitor BIST programming

The application has to program the BIST_TIMExx registers according to the following timings, respect to the configured PBRIDGE_CLK:

- BIST_TIME10[Time_0] = 40 µs.
- BIST_TIME10[Time_1] = 50 µs.
- BIST_TIME32[Time_2] = 50 µs.
- BIST_TIME32[Time_3] = 50 µs.
- BIST_TIME65[Time_5] = 200 ns.
- BIST_TIME65[Time_6] = 200 ns.

In case that aforementioned timings are not applied, some voltage monitor BIST could trigger fake faults.

The user can run a BIST mode by setting the BIST_CTRL[START] register bit. The following voltage detectors are then serially tested:

- LVD100_SB
- MVD094_C
- MVD094_FA
- MVD094_FB
- LVD100_C
- LVD100_FL
- MVD270B_SB
- MVD270_C
- MVD270_FL
- LVD290_C
- LVD290_FL
- LVD290_AD
- LVD290_AS
- LVD290_IJ
- LVD290_IF
- LVD400_AD
- LVD400_AS
- LVD400_IM
- LVD400_IJ
- LVD400_IF
- MVD082_B
- HVD134_C
- UVD140_C
- UVD140_FL
- HVD400_C
- HVD400_IJ
- HVD400_IF
- UVD600_C
- UVD600_FL
- UVD600_IJ
- UVD600_IF

Interrupt can be enabled by programming BIST_CTRL[IRQEN] and interrupt status is checked with BIST_CTRL[IRQST] on completion of BIST sequence.

FCCU event can be enabled by programming BIST_CTRL[NCFEN]. The BIST_CTRL[NCFST] status indicates BIST Fail on completion of BIST sequence.

BIST_DEBUG[VD_MON] register bits allow to know which voltage detector is under test.

The overall status of BIST is provided by the BIST_CTRL[STATUS] register bits.

The status of individual monitors is checked in flag registers BIST_FLAGS_PHASE1 (pass is all zero) and BIST_FLAGS_PHASE2 (pass is all one).

Note: When the SOC supply is 3.3 V, the LVD400_xx monitors are not lifted but when we run BIST all the monitors are masked (during this time the event is seen in PMC_DIG_EPR_HV1 register). When the SOC supply is 5 V, the HVD400_xx monitors are not lifted but when we run BIST, the event is seen in PMC_DIG_EPR_HV1 register.

68 Power Control Unit (MC_PCU)

68.1 Introduction

68.1.1 Overview

The power control unit (MC_PCU) is used to control the overall microcontroller power consumption. There are 4 power domains controlled by the MC_PCU which physically disconnects power to the domains when the specific operating mode allows this.

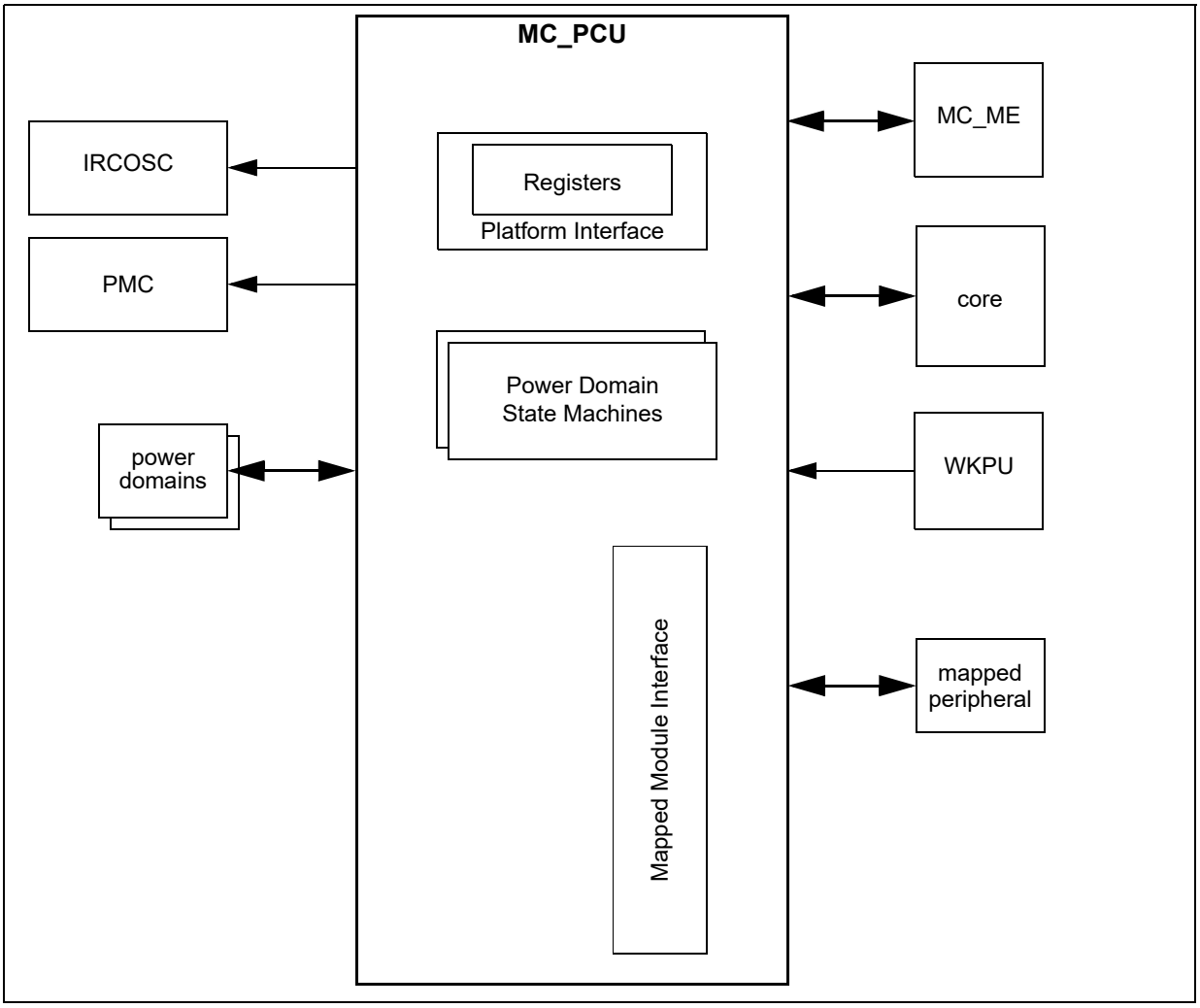
When a power domain is disconnected from the supply, the power consumption is reduced to zero in that domain. Any status information of such a power domain is lost. When re-connecting a power domain to the supply voltage, the domain draws an increased current until the power domain reaches its operational voltage.

For each mode, software can configure whether a power domain is connected to the supply voltage (power-up state) or disconnected (power-down state). Maximum power saving is reached by entering the STANDBY mode.

On each mode change request, the MC_PCU evaluates the power domain settings in the power domain configuration registers and initiates a power-down or a power-up sequence for each individual power domain. The power-up/down sequences are handled by finite state machines to ensure a smooth and safe transition from one power state to the other.

Figure 1844 depicts the MC_PCU block diagram.

Figure 1844. MC_PCU block diagram



68.1.2 Features

The MC_PCU includes:

- Support for 4 power domains
- Support for chip modes RESET, DRUN, SAFE, TEST, RUN0…3, HALT, STOP and STANDBY (for further mode details, refer to the MC_ME chapter)
- Power states updating on each mode change and on system wakeup
- A handshake mechanism for power state changes thus guaranteeing operable voltage

68.2 External Signal Description

The MC_PCU has no connections to any external pins.

68.3 Memory Map and Register Definition

68.3.1 Memory Map

Table 1792. MC_PCU Register Description

Address offset	Register name	Location
0x0000	Power Domain #0 Configuration (PCU_PCONF0)	*Section 68.3.2.1*
0x0004	Power Domain #1 Configuration (PCU_PCONF1)	*Section 68.3.2.2*
0x0008	Power Domain #2 Configuration (PCU_PCONF2)	*Section 68.3.2.3*
0x000C	Power Domain #3 Configuration (PCU_PCONF3)	*Section 68.3.2.4*
0x0010–0x003F	Reserved	
0x0040	Power Domain Status Register (PCU_PSTAT)	*Section 68.3.2.5*

Note: Any access to unused registers as well as write accesses to read-only registers will not change register content and will cause a transfer error

68.3.2 Register descriptions

All registers may be accessed as 32-bit words, 16-bit half-words, or 8-bit bytes. The bytes are ordered according to big endian. For example, the PD0 field of the PCU_PSTAT register may be accessed as a word at address offset 0x0040, as a half-word at address offset 0x0042, or as a byte at address offset 0x0043.

68.3.2.1 Power Domain #0 Configuration Register (PCU_PCONF0)

Offset: 0x0000 Access: User read, Supervisor read, Test read

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	STBY0	0	0	STOP	0	HALT	RUN3	RUN2	RUN1	RUN0	DRUN	SAFE	TEST	RST
W																
Reset	0	0	1	0	0	1	0	1	1	1	1	1	1	1	1	1

Figure 1845. Power Domain #0 Configuration Register (PCU_PCONF0)

This register defines for power domain #0 whether it is on or off in each chip mode. As power domain #0 is the always-on power domain (and includes the MC_PCU), none of its bits are programmable. This register is available for completeness reasons.

Table 1793. PCU_PCONF0 register field descriptions

Field	Description
31 RST	Power domain control during RESET mode 0 Power domain off 1 Power domain on
30 TEST	Power domain control during TEST mode 0 Power domain off 1 Power domain on
29 SAFE	Power domain control during SAFE mode 0 Power domain off 1 Power domain on
28 DRUN	Power domain control during DRUN mode 0 Power domain off 1 Power domain on
27 RUN0	Power domain control during RUN0 mode 0 Power domain off 1 Power domain on
26 RUN1	Power domain control during RUN1 mode 0 Power domain off 1 Power domain on
25 RUN2	Power domain control during RUN2 mode 0 Power domain off 1 Power domain on
24 RUN3	Power domain control during RUN3 mode 0 Power domain off 1 Power domain on
23 HALT	Power domain control during HALT mode 0 Power domain off 1 Power domain on
21 STOP	Power domain control during STOP mode 0 Power domain off 1 Power domain on
18 STBY0	Power domain control during STANDBY mode 0 Power domain off 1 Power domain on

68.3.2.2 Power Domain #1 Configuration Register (PCU_PCONF1)

Offset: 0x0004 Access: User read, Supervisor read, Test read

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	STBY0	0	0	STOP	0	HALT	RUN3	RUN2	RUN1	RUN0	DRUN	SAFE	TEST	RST
W																
Reset	0	0	0	0	0	1	0	1	1	1	1	1	1	1	1	1

Figure 1846. Power Domain #1 Configuration Register (PCU_PCONF1)

This register defines for power domain #1 whether it is on or off in each chip mode. The bit field description is the same as in *Table 1793*. As the platform, clock generation, and mode control reside in power domain #1, this power domain is only powered down during the STANDBY mode. Therefore, none of the bits are programmable. This register is available for completeness reasons.

The difference between PCU_PCONF0 and PCU_PCONF1 is the reset value of the STBY0 bit: during the STANDBY mode, power domain #1 is disconnected from the power supply, and therefore PCU_PCONF1.STBY0 is always ‘0’. Power domain #0 is always on, and therefore PCU_PCONF0.STBY0 is ‘1’.

For further details about STANDBY mode, refer to *Section 68.4.4.1: STANDBY Mode transition*. NO_FEATURE

68.3.2.3 Power Domain #2 Configuration Register (PCU_PCONF2)

Offset: 0x0008 Access: User read, Supervisor read, Test read

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	STBY0	0	0	STOP	0	HALT	RUN3	RUN2	RUN1	RUN0	DRUN	SAFE	TEST	RST
W																
Reset	0	0	0	0	0	1	0	1	1	1	1	1	1	1	1	1

Figure 1847. Power Domain #2 Configuration Register (PCU_PCONF2)

This register defines for power domain #2 whether it is on or off in each chip mode. The bit field description is the same as in *Table 1793*. This register controls the 120KB Standby RAM.

68.3.2.4 Power Domain #3 Configuration Register (PCU_PCONF3)

Offset: 0x000C Access: User read, Supervisor read, Test read

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	STBY0	0	0	STOP	0	HALT	RUN3	RUN2	RUN1	RUN0	DRUN	SAFE	TEST	RST
W																
Reset	0	0	0	0	0	1	0	1	1	1	1	1	1	1	1	1

Figure 1848. Power Domain #3 Configuration Register (PCU_PCONF3)

This register defines for power domain #3 whether it is on or off in each chip mode. The bit field description is the same as in *Table 1793*. This register controls the 128KB Standby RAM.

68.3.2.5 Power Domain Status Register (PCU_PSTAT)

Offset: 0x0040 Access: User read, Supervisor read, Test read

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	PD3	PD2	PD1	PD0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1

Figure 1849. Power Domain Status Register (PCU_PSTAT)

This register reflects the power status of all available power domains.

Table 1794. PCU_PSTAT field descriptions

Field	Description
28 PD3	Power status for power domain #3 0 Power domain is inoperable. 1 Power domain is operable.
29 PD2	Power status for power domain #2 0 Power domain is inoperable. 1 Power domain is operable.

Table 1794. PCU_PSTAT field descriptions (continued)

Field	Description
30 PD1	Power status for power domain #1 0 Power domain is inoperable. 1 Power domain is operable.
31 PD0	Power status for power domain #0 0 Power domain is inoperable. 1 Power domain is operable.

68.4 Functional description

68.4.1 General

The MC_PCU controls all available power domains on a chip mode basis. The PCU_PCONF*n* registers specify during which system/user modes a power domain is powered up. The power state for each individual power domain is reflected by the bits in the PCU_PSTAT register.

On a mode change, the MC_PCU evaluates which power domain(s) must change power state. The power state is controlled by a state machine (FSM) for each individual power domain which ensures a clean and safe state transition.

68.4.2 Reset / Power-On Reset

After any reset, the chip will transition to the RESET mode during which all power domains are powered up (refer to the MC_ME chapter). Once the reset sequence has been completed, the DRUN mode is entered and software can begin the MC_PCU configuration.

68.4.3 MC_PCU configuration

By default, all power domains are powered in all modes other than STANDBY. Software can change the configuration for each power domain on a mode basis by programming the PCU_PCONF*n* registers.

Each power domain which is powered down is held in a reset state. Read/write accesses to peripherals in those power domains will result in a transfer error.

68.4.4 Mode transitions

On a mode change requested by the MC_ME, the MC_PCU evaluates the power configurations for all power domains. It compares the settings in the PCU_PCONF*n* registers for the new mode with the settings for the current mode. If the configuration for a power domain differs between the modes, a power state change request is generated. These requests are handled by a finite state machine to ensure a smooth and safe transition from one power state to another.

68.4.4.1 STANDBY Mode transition

STANDBY offers the maximum power saving. The level of power saving is software-controllable via the settings in the PCU_PCONF*n* registers for power domain #2 onwards. Power domain #0 stays connected to the power supply while power domain #1 is

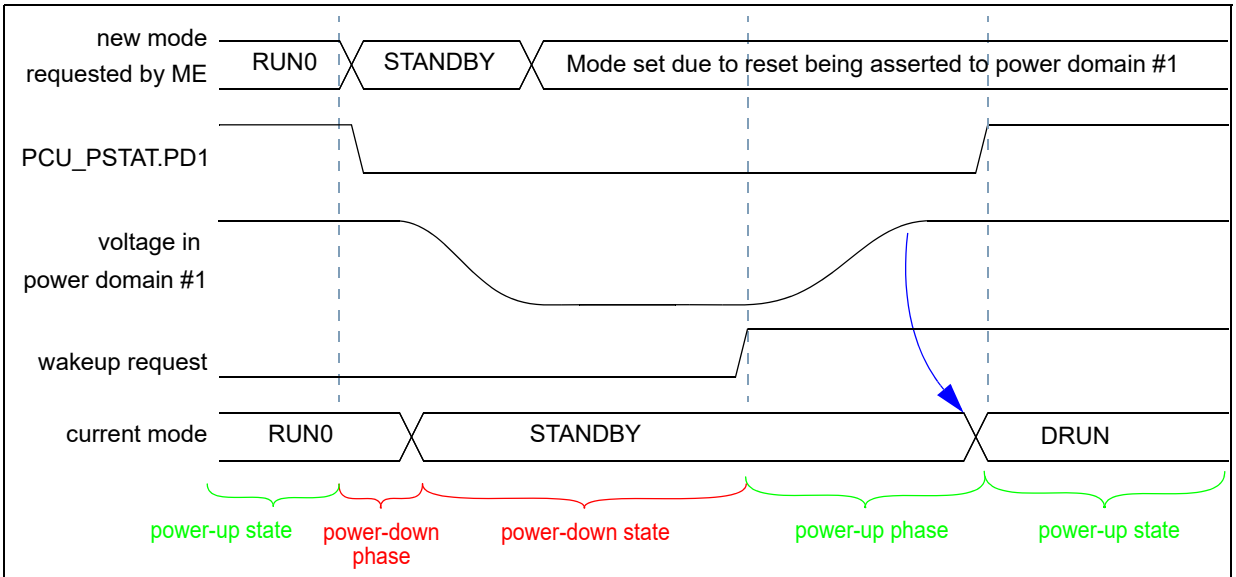
disconnected from the power supply. Amongst others power domain #1 contains the platform and the MC_ME. Therefore this mode differs from all other user/system modes.

Once STANDBY is entered it can only be left via a system wakeup. On exiting the STANDBY mode, all power domains are powered up according to the settings in the PCU_PCONF*n* registers, and the DRUN mode is entered. In DRUN mode, both domains #0 and #1 are powered.

Figure 1850 shows an example for a mode transition from RUN0 to STANDBY to DRUN. All power domains which have PCU_PCONF*n*.STBY cleared will enter power-down phase. In this example power domain #1 will be disabled during STANDBY mode.

When the MC_PCU receives the mode change request to STANDBY mode it starts the power down phase for power domain #1. During the power down phase, clocks are disabled and reset is asserted resulting in a loss of all information for this power domain. Then the power domain is disconnected from the power supply (power-down state).

Figure 1850. MC_PCU Events During Power Sequences (STANDBY mode)



Notes:

Not drawn to scale; PCONF1.RUN0 = 1; PCONF1.STBY0 = 0

When the MC_PCU receives a system wakeup request, it starts the power-up phase. The power domain is re-connected to the power supply and the voltage in power domain #1 will increase slowly. Once the voltage is in an operable range, clocks are enabled and the reset is be deasserted (power-up state).

Prior to standby entry, PCTL for WKPU should be disabled.

Note: It is possible that due to a wakeup request, power-up is requested before a power domain completed its power-down sequence. In this case, the information in that power domain is lost.

Exiting the STANDBY mode can only be done via a system wakeup event from:

- API/SSWU module (mapped on WKUP0 line),
- RTC timer (mapped on WKUP1 line),
- external wakeup pads (WKUPx line, where x>1).

Refer to the wakeup vector mapping table in the Device Configuration chapter for complete wakeup lines list and to SPC58xEx/SPC58xGx IO_Definition document for pad/wakeup lines mapping.

68.4.4.2 Power Saving for Memories During STANDBY Mode

All memories which are not powered down during STANDBY mode automatically enter a power saving state. No software configuration is required to enable this power saving state. While a memory is residing in this state an increased power saving is achieved. Data in the memories is retained.

68.5 Application information

68.5.1 STANDBY mode considerations

STANDBY offers maximum power saving possibility. But power is only saved during the time a power domain is disconnected from the supply. Increased power is required when a power domain is re-connected to the power supply. Additional power is required during restoring the information (for example in the platform). Care should be taken that the time during which the chip is operating in STANDBY mode is significantly longer than the required time for restoring the information.

69 Mode Entry Module (MC_ME)

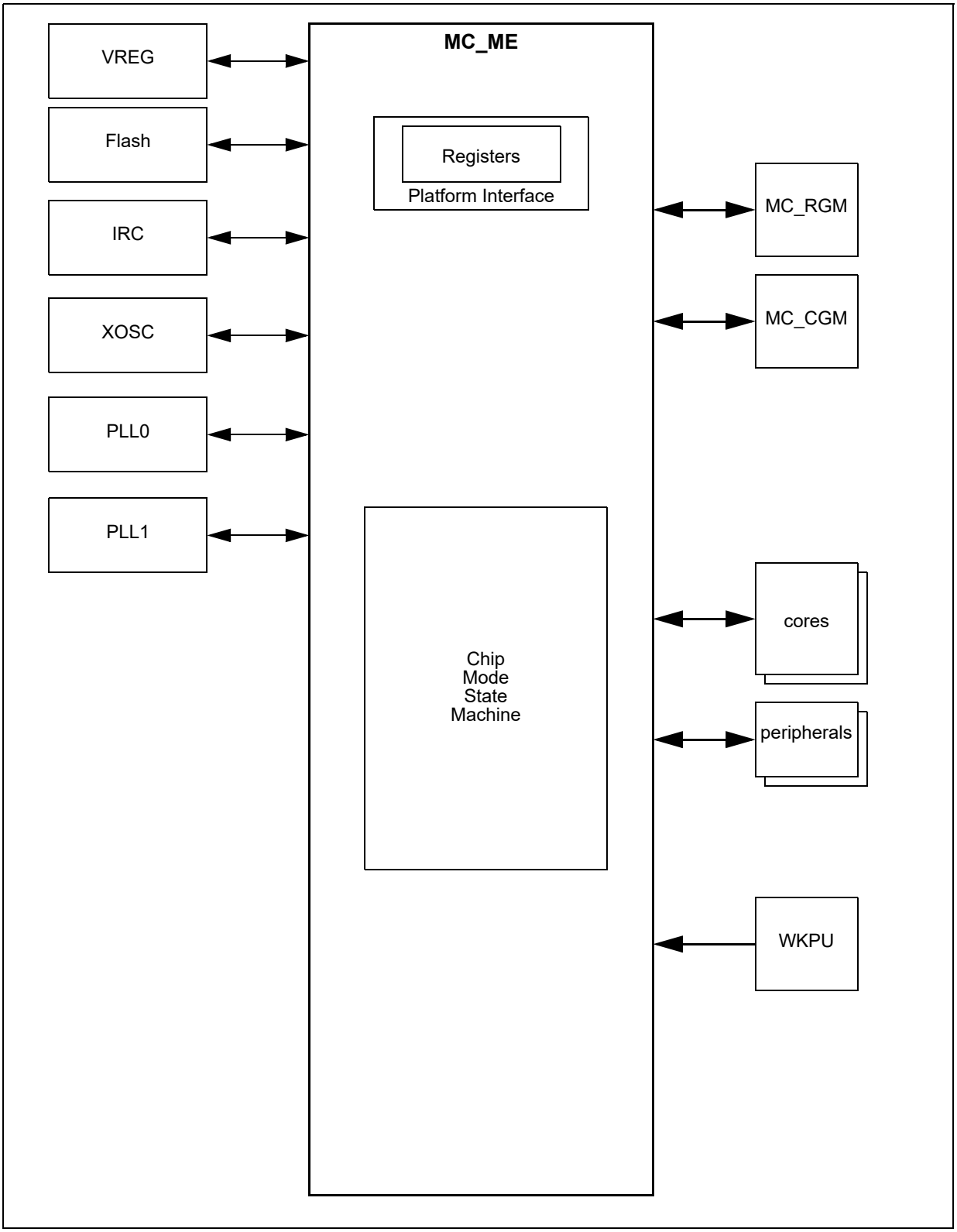
69.1 Introduction

69.1.1 Overview

The MC_ME controls the chip mode and mode transition sequences in all functional states. It also contains configuration, control and status registers accessible for the application.

Figure 1851 shows the MC_ME block diagram.

Figure 1851. MC_ME block diagram



69.1.2 Features

The MC_ME includes the following features:

- control of the available modes by the ME_ME register
- definition of various chip mode configurations by the ME_<mode>_MC registers
- control of the actual chip mode by the ME_MCTL register
- capture of the current mode and various resource status within the contents of the ME_GS register
- optional generation of various mode transition interrupts
- status bits for each cause of invalid mode transitions
- peripheral clock gating control based on the ME_RUN_PC0–7, ME_LP_PC0–7, and ME_PCTL*n* registers
- additional core clock gating and boot address control based on the ME_CCTL*n* and ME_CADDR*n* registers
- capture of current peripheral and additional core clock gated/enabled status
- progressive system clock switching when transitioning from a lower power consumption mode to a higher power consumption mode, and vice versa

69.1.3 Modes of operation

The MC_ME is based on several chip modes corresponding to different usage models of the chip. Each mode is configurable and can define a policy for energy and processing power management to fit particular system requirements. An application can easily switch from one mode to another depending on the current needs of the system. The operating modes controlled by the MC_ME are divided into system and user modes. The system modes are modes such as RESET, DRUN, SAFE, and TEST. These modes aim to ease the configuration and monitoring of the system. The user modes are modes such as RUN0–3, HALT0, STOP0, and STANDBY0 which can be configured to meet the application requirements in terms of energy management and available processing power. The modes DRUN, SAFE, TEST, and RUN0–3 are the chip software running modes.

Table 1795 describes the MC_ME modes.

Table 1795. MC_ME mode descriptions (continued)

Name	Description	Entry	Exit
RESET	This is a chip-wide virtual mode during which the application is not active. The system remains in this mode until all resources are available for the embedded software to take control of the chip. It manages hardware initialization of chip configuration, voltage regulators, clock sources, and flash modules.	system reset assertion from MC_RGM	system reset deassertion from MC_RGM
DRUN	This is the entry mode for the embedded software. It provides full accessibility to the system and enables the configuration of the system at startup. It provides the unique gate to enter user modes. BAF, when present, is executed in DRUN mode.	system reset deassertion from MC_RGM, software request from SAFE, TEST and RUN0–3, wakeup request from STANDBY0	system reset assertion, RUN0–3, TEST, STANDBY0 via software, SAFE via software or hardware failure.

Table 1795. MC_ME mode descriptions (continued)

Name	Description	Entry	Exit
SAFE	This is a chip-wide service mode which may be entered on the detection of a recoverable error. It forces the system into a pre-defined safe configuration from which the system may try to recover.	hardware failure, software request from DRUN, TEST, and RUN0–3	system reset assertion, DRUN via software
TEST	This is a chip-wide service mode which is intended to provide a control environment for chip software testing.	software request from DRUN	system reset assertion, DRUN via software
RUN0–3	These are software running modes where most processing activity is done. These various run modes allow to enable different clock & power configurations of the system with respect to each other.	software request from DRUN or other RUN0–3, interrupt event from HALT0, interrupt or wakeup event from STOP0	system reset assertion, SAFE via software or hardware failure, other RUN0–3 modes, HALT0, STOP0, STANDBY0 via software
HALT0	This is a reduced-activity low-power mode during which the clock to the core is disabled. It can be configured to switch off analog peripherals such as: clock sources, flash, main regulator, for efficient power management at the cost of higher wakeup latency.	software request from RUN0–3	system reset assertion, SAFE on hardware failure, RUN0–3 on interrupt event
STOP0	This is an advanced low-power mode during which the clock to the core is disabled. It may be configured to switch off most of the peripherals including clock sources for efficient power management at the cost of higher wakeup latency.	software request from RUN0–3	system reset assertion, SAFE on hardware failure, RUN0–3 on interrupt event or wakeup event
STANDBY0	This is a reduced-leakage low-power mode during which power supply is cut off from most of the chip. Wakeup from this mode takes a relatively long time, and content is lost or must be restored from backup.	software request from RUN0–3, DRUN modes	system reset assertion, DRUN on wakeup event

69.2 External signal description

The MC_ME has no connections to any external pins.

69.3 Memory map and register definition

The MC_ME contains registers for:

- mode selection and status reporting
- mode configuration
- mode transition interrupts status and mask control
- scalable number of peripheral sub-mode selection and status reporting
- scalable number of additional core enabling and disabling control and status reporting

69.3.1 Memory map

Table 1796. MC_ME register description (continued)

Address offset	Name	Description	Location
0x000	ME_GS	Global Status	*Section 69.3.2.1*
0x004	ME_MCTL	Mode Control	*Section 69.3.2.2*
0x008	ME_ME	Mode Enable	*Section 69.3.2.3*
0x00C	ME_IS	Interrupt Status	*Section 69.3.2.4*
0x010	ME_IM	Interrupt Mask	*Section 69.3.2.5*
0x014	ME_IMTS	Invalid Mode Transition Status	*Section 69.3.2.6*
0x018	ME_DMTS	Debug Mode Transition Status	*Section 69.3.2.7*
0x020	ME_RESET_MC	RESET Mode Configuration	*Section 69.3.2.8*
0x024	ME_TEST_MC	TEST Mode Configuration	*Section 69.3.2.9*
0x028	ME_SAFE_MC	SAFE Mode Configuration	*Section 69.3.2.10*
0x02C	ME_DRUN_MC	DRUN Mode Configuration	*Section 69.3.2.11*
0x030	ME_RUN0_MC	RUN0 Mode Configuration	*Section 69.3.2.12*
0x034	ME_RUN1_MC	RUN1 Mode Configuration	*Section 69.3.2.12*
0x038	ME_RUN2_MC	RUN2 Mode Configuration	*Section 69.3.2.12*
0x03C	ME_RUN3_MC	RUN3 Mode Configuration	*Section 69.3.2.12*
0x040	ME_HALT0_MC	HALT0 Mode Configuration	*Section 69.3.2.13*
0x048	ME_STOP0_MC	STOP0 Mode Configuration	*Section 69.3.2.14*
0x04C–0x053	Reserved		
0x054	ME_STANDBY0_MC	STANDBY0 Mode Configuration	*Section 69.3.2.15*
0x058–0x05F	Reserved		
0x060	ME_PS0	Peripheral Status 0	*Section 69.3.2.16*
0x064	ME_PS1	Peripheral Status 1	*Section 69.3.2.17*
0x068	ME_PS2	Peripheral Status 2	*Section 69.3.2.18*
0x06C	ME_PS3	Peripheral Status 3	*Section 69.3.2.19*
0x070	ME_PS4	Peripheral Status 4	*Section 69.3.2.20*
0x074	ME_PS5	Peripheral Status 5	*Section 69.3.2.21*
0x078	ME_PS6	Peripheral Status 6	*Section 69.3.2.22*
0x07C	ME_PS7	Peripheral Status 7	*Section 69.3.2.23*
0x080 + *n**0x4	ME_RUN_PCn	Run Peripheral Configuration *n* (n = 0 to 7)	*Section 69.3.2.24*
0x0A0 + *n**0x4	ME_LP_PC*n*	Low-Power Peripheral Configuration *n* (n = 0 to 7)	*Section 69.3.2.25*
0x0C0–0x0C8	ME_PCTL0–ME_PCTL8	Reserved	
0x0C9	ME_PCTL9	S_LFAST_0 Control	*Section 69.3.2.26*

Table 1796. MC_ME register description (continued)

Address offset	Name	Description	Location
0x0CA	ME_PCTL10	Reserved	
0x0CB	ME_PCTL11	S_SIPI_0 Control	*Section 69.3.2.26*
0x0CC–0x0CE	ME_PCTL12–ME_PCTL14	Reserved	
0x0CF	ME_PCTL15	S_SIUL2 Control	*Section 69.3.2.26*
0x0D0–0x0DD	ME_PCTL16–ME_PCTL29	Reserved	
0x0DE	ME_PCTL30	S_PIT_0 Control	*Section 69.3.2.26*
0x0DF–0x0E0	ME_PCTL31–ME_PCTL32	Reserved	
0x0E1	ME_PCTL33	S_PSI5-S_0 Control	*Section 69.3.2.26*
0x0E2–0x0E3	ME_PCTL34-ME_PCTL35	Reserved	
0x0E4	ME_PCTL36	S_DMAMUX_0/2/4 Control	*Section 69.3.2.26*
0x0E5	ME_PCTL37	Reserved	
0x0E6	ME_PCTL38	S_CRC_0 Control	*Section 69.3.2.26*
0x0E7–0x0F9	ME_PCTL39–ME_PCTL57	Reserved	
0x0FA	ME_PCTL58	S_SD_ADC_4 Control	*Section 69.3.2.26*
0x0FB	ME_PCTL59	S_SD_ADC_2 Control	*Section 69.3.2.26*
0x0FC	ME_PCTL60	S_SD_ADC_0 Control	*Section 69.3.2.26*
0x0FD	ME_PCTL61	Reserved	
0x0FE	ME_PCTL62	S_CCCU Control	*Section 69.3.2.26*
0x0FF–0x102	ME_PCTL63–ME_PCTL66	Reserved	
0x103	ME_PCTL67	S_CAN_SUB_0_MCAN_3 Control	*Section 69.3.2.26*
0x104	ME_PCTL68	S_CAN_SUB_0_MCAN_2 Control	*Section 69.3.2.26*
0x105	ME_PCTL69	S_CAN_SUB_0_MCAN_1 Control	*Section 69.3.2.26*
0x106–0x107	ME_PCTL70–ME_PCTL71	Reserved	
0x108	ME_PCTL72	S_CAN_SUB_0_M_TTCAN_0 Control	*Section 69.3.2.26*
0x109	ME_PCTL73	Reserved	
0x10A	ME_PCTL74	S_CAN_SUB_0_MESSAGE_RAM Control	*Section 69.3.2.26*
0x10B–0x113	ME_PCTL75–ME_PCTL83	Reserved	
0x114	ME_PCTL84	S_LINFlexD_16 Control	*Section 69.3.2.26*
0x115	ME_PCTL85	S_LINFlexD_14 Control	*Section 69.3.2.26*
0x116	ME_PCTL86	S_LINFlexD_12 Control	*Section 69.3.2.26*
0x117	ME_PCTL87	S_LINFlexD_10 Control	*Section 69.3.2.26*
0x118	ME_PCTL88	S_LINFlexD_8 Control	*Section 69.3.2.26*
0x119	ME_PCTL89	S_LINFlexD_6 Control	*Section 69.3.2.26*
0x11A	ME_PCTL90	S_LINFlexD_4 Control	*Section 69.3.2.26*

Table 1796. MC_ME register description (continued)

Address offset	Name	Description	Location
0x11B	ME_PCTL91	S_LINFlexD_2 Control	*Section 69.3.2.26*
0x11C	ME_PCTL92	S_LINFlexD_0 Control	*Section 69.3.2.26*
0x11D–0x11E	ME_PCTL93–ME_PCTL94	Reserved	
0x11F	ME_PCTL95	S_DSPI_8 Control	*Section 69.3.2.26*
0x120	ME_PCTL96	S_DSPI_6 Control	*Section 69.3.2.26*
0x121	ME_PCTL97	S_DSPI_4 Control	*Section 69.3.2.26*
0x122	ME_PCTL98	S_DSPI_2 Control	*Section 69.3.2.26*
0x123	ME_PCTL99	S_DSPI_0 Control	*Section 69.3.2.26*
0x124	ME_PCTL100	Reserved	
0x125	ME_PCTL101	S_IIC_0 Control	*Section 69.3.2.26*
0x126–0x127	ME_PCTL102–ME_PCTL103	Reserved	
0x128	ME_PCTL104	S_SENT_0 Control	*Section 69.3.2.26*
0x129–0x12A	ME_PCTL105–ME_PCTL106	Reserved	
0x12B	ME_PCTL107	S_FlexRay_0 Control	*Section 69.3.2.26*
0x12C–0x12E	ME_PCTL108–ME_PCTL110	Reserved	
0x12F	ME_PCTL111	S_PSI5_0 Control	*Section 69.3.2.26*
0x130	ME_PCTL112	S_SAR_ADC_12bit_B Control (SAR_ADC_12bit_SUPERVISOR)	*Section 69.3.2.26*
0x131	ME_PCTL113	Reserved	
0x132	ME_PCTL114	S_SAR_ADC_10bit_STDBY Control	*Section 69.3.2.26*
0x133–0x136	ME_PCTL115–ME_PCTL118	Reserved	
0x137	ME_PCTL119	S_SAR_ADC_10bit_0 Control	*Section 69.3.2.26*
0x138–0x13D	ME_PCTL120–ME_PCTL125	Reserved	
0x13E	ME_PCTL126	S_SAR_ADC_12bit_2 Control	*Section 69.3.2.26*
0x13F	ME_PCTL127	S_SAR_ADC_12bit_0 Control	*Section 69.3.2.26*
0x140	ME_PCTL128	S_GTMINT Control	*Section 69.3.2.26*
0x141	ME_PCTL129	S_SAR_ADC_12bit_B_seq Control (SAR_ADC_12bit_SUPERVISOR_seq)	*Section 69.3.2.26*
0x142	ME_PCTL130	S_SAR_ADC_10bit_STDBY_seq Control	*Section 69.3.2.26*
0x143–0x145	ME_PCTL131–ME_PCTL133	Reserved	
0x146	ME_PCTL134	S_SAR_ADC_10bit_1_seq Control	*Section 69.3.2.26*
0x147	ME_PCTL135	S_SAR_ADC_10bit_0_seq Control	*Section 69.3.2.26*
0x148–0x14B	ME_PCTL136–ME_PCTL139	Reserved	
0x14C	ME_PCTL140	S_SAR_ADC_12bit_3_seq Control	*Section 69.3.2.26*
0x14D	ME_PCTL141	S_SAR_ADC_12bit_1_seq Control	*Section 69.3.2.26*

Table 1796. MC_ME register description (continued)

Address offset	Name	Description	Location
0x14E	ME_PCTL142	S_SAR_ADC_12bit_2_seq Control	*Section 69.3.2.26*
0x14F	ME_PCTL143	S_SAR_ADC_12bit_0_seq Control	*Section 69.3.2.26*
0x150–0x15D	ME_PCTL144–ME_PCTL157	Reserved	
0x15E	ME_PCTL158	S_PIT_1 Control	*Section 69.3.2.26*
0x15F–0x163	ME_PCTL159–ME_PCTL163	Reserved	
0x164	ME_PCTL164	S_DMAMUX_1/3/5 Control	*Section 69.3.2.26*
0x165	ME_PCTL165	Reserved	
0x166	ME_PCTL166	S_CRC_1 Control	*Section 69.3.2.26*
0x167–0x179	ME_PCTL167–ME_PCTL185	Reserved	
0x17A	ME_PCTL186	S_SD_ADC_5 Control	*Section 69.3.2.26*
0x17B	ME_PCTL187	S_SD_ADC_3 Control	*Section 69.3.2.26*
0x17C	ME_PCTL188	S_SD_ADC_1 Control	*Section 69.3.2.26*
0x17D–0x182	ME_PCTL189–ME_PCTL194	Reserved	
0x183	ME_PCTL195	S_CAN_SUB_1_MCAN_4 Control	*Section 69.3.2.26*
0x184	ME_PCTL196	S_CAN_SUB_1_MCAN_3 Control	*Section 69.3.2.26*
0x185	ME_PCTL197	S_CAN_SUB_1_MCAN_2 Control	*Section 69.3.2.26*
0x186	ME_PCTL198	S_CAN_SUB_1_MCAN_1 Control	*Section 69.3.2.26*
0x187–0x189	ME_PCTL199–ME_PCTL201	Reserved	
0x18A	ME_PCTL202	S_CAN_SUB_1_MESSAGE_RAM Control	*Section 69.3.2.26*
0x18B–0x193	ME_PCTL203–ME_PCTL211	Reserved	
0x194	ME_PCTL212	S_LINFlexD_17 Control	*Section 69.3.2.26*
0x195	ME_PCTL213	S_LINFlexD_15 Control	*Section 69.3.2.26*
0x196	ME_PCTL214	S_LINFlexD_13 Control	*Section 69.3.2.26*
0x197	ME_PCTL215	S_LINFlexD_11 Control	*Section 69.3.2.26*
0x198	ME_PCTL216	S_LINFlexD_9 Control	*Section 69.3.2.26*
0x199	ME_PCTL217	S_LINFlexD_7 Control	*Section 69.3.2.26*
0x19A	ME_PCTL218	S_LINFlexD_5 Control	*Section 69.3.2.26*
0x19B	ME_PCTL219	S_LINFlexD_3 Control	*Section 69.3.2.26*
0x19C	ME_PCTL220	S_LINFlexD_1 Control	*Section 69.3.2.26*
0x19D–0x19E	ME_PCTL221–ME_PCTL222	Reserved	
0x19F	ME_PCTL223	S_DSPI_9 Control	*Section 69.3.2.26*
0x1A0	ME_PCTL224	S_DSPI_7 Control	*Section 69.3.2.26*
0x1A1	ME_PCTL225	S_DSPI_5 Control	*Section 69.3.2.26*
0x1A2	ME_PCTL226	S_DSPI_3 Control	*Section 69.3.2.26*

Table 1796. MC_ME register description (continued)

Address offset	Name	Description	Location
0x1A3	ME_PCTL227	S_DSPI_1 Control	*Section 69.3.2.26*
0x1A4–0x1A7	ME_PCTL228–ME_PCTL231	Reserved	
0x1A8	ME_PCTL232	S_SENT_1 Control	*Section 69.3.2.26*
0x1A9	ME_PCTL233	S_ETHERNET_1 Control	*Section 69.3.2.26*
0x1AA	ME_PCTL234	S_ETHERNET_0 Control	*Section 69.3.2.26*
0x1AB–0x1AE	ME_PCTL235–ME_PCTL238	Reserved	
0x1AF	ME_PCTL239	S_PSI5_1 Control	*Section 69.3.2.26*
0x1B0	ME_PCTL240	S_eMIOS_1 Control	*Section 69.3.2.26*
0x1B1	ME_PCTL241	S_eMIOS_0 Control	*Section 69.3.2.26*
0x1B2	ME_PCTL242	S_STDBY_CTU_0 Control	*Section 69.3.2.26*
0x1B3	ME_PCTL243	S_BODY_CTU_0 Control	*Section 69.3.2.26*
0x1B4–0x1B6	ME_PCTL244–ME_PCTL246	Reserved	
0x1B7	ME_PCTL247	S_SAR_ADC_10bit_1 Control	*Section 69.3.2.26*
0x1B8–0x1BD	ME_PCTL248–ME_PCTL253	Reserved	
0x1BE	ME_PCTL254	S_SAR_ADC_12bit_3 Control	*Section 69.3.2.26*
0x1BF	ME_PCTL255	S_SAR_ADC_12bit_1 Control	*Section 69.3.2.26*
0x1C0	ME_CS	Core Status	*Section 69.3.2.27*
0x1C4	ME_CCTL0	CORE0 Control	*Section 69.3.2.28*
0x1C6	ME_CCTL1	CORE1 Control	*Section 69.3.2.29*
0x1C8–0x1C9	Reserved		
0x1CA	ME_CCTL3	CORE3 Control	*Section 69.3.2.30*
0x1CC	ME_CCTL4	CORE4 Control	*Section 69.3.2.31*
0x1CE–0x1DF	Reserved		
0x1E0	ME_CADDR0	CORE0 Address	*Section 69.3.2.32*
0x1E4	ME_CADDR1	CORE1 Address	*Section 69.3.2.33*
0x1E8–0x1EB	Reserved		
0x1EC	ME_CADDR3	CORE3 Address	*Section 69.3.2.34*
0x1F0	ME_CADDR4	CORE4 Address	*Section 69.3.2.35*

Note: Any access to unused registers as well as write accesses to read-only registers will:
not change register content
cause a transfer error

69.3.2 Register description

Unless otherwise noted, all registers may be accessed as 32-bit words, 16-bit half-words, or 8-bit bytes. The bytes are ordered according to big endian. For example, the ME_RUN_PC0 register may be accessed as a word at address 0x080, as a half-word at address 0x082, or as a byte at address 0x083.

Some fields may be read-only, and their reset value of ‘1’ or ‘0’ and the corresponding behavior cannot be changed.

69.3.2.1 Global Status register (ME_GS)

This register contains global mode status.

Address: 0x000 Access: User read, Supervisor read, Test read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	S_CURRENT_MODE				S_MTRANS	0	0	0	S_PDO	0	0	S_MVR	0	0	S_FLA	
W																
Reset	0	0	0	0	1	0	0	0	0	0	0	1	0	0	1	1

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	S_PLL1	S_PLL0	S_XOSC	S_IRC	S_SYSCLK			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0

Figure 1852. Global Status register (ME_GS)

Table 1797. ME_GS field descriptions (continued)

Field	Description
0:3 S_CURRENT_MODE	Current chip mode status 0000 RESET 0001 TEST 0010 SAFE 0011 DRUN 0100 RUN0 0101 RUN1 0110 RUN2 0111 RUN3 1000 HALT0 1001 reserved 1010 STOP0 1011 Reserved 1100 Reserved 1101 STANDBY0 1110 Reserved 1111 Reserved
4 S_MTRANS	Mode transition status 0 Mode transition process is not active 1 Mode transition is ongoing
8 S_PDO	Output power-down status This bit specifies output power-down status of I/Os. This bit is asserted whenever outputs of pads are forced to high impedance state or the pads power sequence driver is switched off. 0 No automatic safe gating of I/Os used and pads power sequence driver is enabled. 1 In STOP0 mode, the slew rate control mechanism of the output pads is disabled to reduce power consumption, but the state of the output remains functional.
11 S_MVR	Main voltage regulator status 0 Main voltage regulator is not ready 1 Main voltage regulator is ready for use
14:15 S_FLA	Flash availability status 00 Flash is not available 01 Flash is in power-down mode 10 Flash is in low-power mode 11 Flash is in normal mode and available for use
24 S_PLL1	secondary PLL status 0 secondary PLL is not stable. 1 secondary PLL is providing a stable clock.
25 S_PLL0	primary PLL status 0 primary PLL is not stable. 1 primary PLL is providing a stable clock.
26 S_XOSC	external crystal oscillator status 0 external crystal oscillator is not stable. 1 external crystal oscillator is providing a stable clock.

Table 1797. ME_GS field descriptions (continued)

Field	Description
27 S_IRC	16 MHz internal RC oscillator status 0 16 MHz internal RC oscillator is not stable. 1 16 MHz internal RC oscillator is providing a stable clock.
28:31 S_SYSCLK	System clock switch status These bits specify the system clock currently used by the system. 0000 16 MHz int. RC osc. 0001 external crystal osc. 0010 primary PLL (PHI) 0011 reserved 0100 secondary PLL 0101 reserved 0110 reserved 0111 reserved 1000 reserved 1001 reserved 1010 reserved 1011 reserved 1100 reserved 1101 reserved 1110 reserved 1111 system clock is disabled

69.3.2.2 Mode Control register (ME_MCTL)

This register is used to trigger software-controlled mode changes. Depending on the modes as enabled by ME_ME register bits, configurations corresponding to unavailable modes are reserved and access to ME_<mode>_MC registers must respect this for successful mode requests.

Note: Byte and half-word write accesses are not allowed for this register as a predefined key is required to change its value.

Address: 0x004 Access: User read, Supervisor read/write, Test read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	TARGET_MODE				0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	1	0	1	0	0	1	0	1	0	0	0	0	1	1	1	1
W	KEY															
Reset	1	0	1	0	0	1	0	1	0	0	0	0	1	1	1	1

Figure 1853. Mode Control register (ME_MCTL)

Table 1798. ME_MCTL field descriptions

Field	Description
0:3 TARGET_MODE	Target chip mode These bits provide the target chip mode to be entered by software programming. The mechanism to enter into any mode by software requires the write operation twice: first time with key, and second time with inverted key. These bits are automatically updated by hardware while entering SAFE on hardware request. Also, while exiting from the HALT0 and STOP0 modes on hardware exit events, these are updated with the appropriate RUN0–3 mode value. 0000 RESET (triggers a 'functional' reset event) 0001 TEST 0010 SAFE 0011 DRUN 0100 RUN0 0101 RUN1 0110 RUN2 0111 RUN3 1000 HALT0 1001 reserved 1010 STOP0 1011 reserved 1100 reserved 1101 STANDBY0 1110 reserved 1111 RESET (triggers a 'destructive' reset event)
16:31 KEY	Control key These bits enable write access to this register. Any write access to the register with a value different from the keys is ignored. Read access will always return inverted key. KEY: 0101 1010 1111 0000 (0x5AF0) INVERTED KEY: 1010 0101 0000 1111 (0xA50F)

69.3.2.3 Mode Enable register (ME_ME)

This register allows a way to disable the chip modes which are not required for a given chip. RESET, SAFE, DRUN, and RUN0 modes are always enabled.

Address: 0x008 Access: User read, Supervisor read/write, Test read/write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	RESET_DEST	0	STANDBY0	0	0	STOP0	0	HALT0	RUN3	RUN2	RUN1	RUN0	DRUN	SAFE	TEST	RESET_FUNC
W																
Reset	1	0	1	0	0	0	0	0	0	0	0	1	1	1	0	1

Figure 1854. Mode Enable register (ME_ME)

Table 1799. ME_ME field descriptions (continued)

Field	Description
16 RESET_DEST	‘destructive’ RESET mode enable 1 ‘destructive’ RESET mode is enabled.
18 STANDBY0	STANDBY0 mode enable 0 STANDBY0 mode is disabled. 1 STANDBY0 mode is enabled.
21 STOP0	STOP0 mode enable 0 STOP0 mode is disabled. 1 STOP0 mode is enabled.
23 HALT0	HALT0 mode enable 0 HALT0 mode is disabled. 1 HALT0 mode is enabled.
24 RUN3	RUN3 mode enable 0 RUN3 mode is disabled. 1 RUN3 mode is enabled.
25 RUN2	RUN2 mode enable 0 RUN2 mode is disabled. 1 RUN2 mode is enabled.
26 RUN1	RUN1 mode enable 0 RUN1 mode is disabled. 1 RUN1 mode is enabled.
27 RUN0	RUN0 mode enable 1 RUN0 mode is enabled.
28 DRUN	DRUN mode enable 1 DRUN mode is enabled.
29 SAFE	SAFE mode enable 1 SAFE mode is enabled.

Table 1799. ME_ME field descriptions (continued)

Field	Description
30 TEST	TEST mode enable 0 TEST mode is disabled. 1 TEST mode is enabled.
31 RESET_FUNC	'functional' RESET mode enable 1 'functional' RESET mode is enabled.

69.3.2.4 Interrupt Status register (ME_IS)

This register provides the current interrupt status.

Address: 0x00C Access: User read, Supervisor read/write, Test read/write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	I_ICONF_CC	I_ICONF_CU	I_ICONF	I_IMODE	I_SAFE	I_MTC
W											w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1855. Interrupt Status register (ME_IS)

Table 1800. ME_IS field descriptions

Field	Description
26 I_ICONF_CC	Invalid mode configuration interrupt (core configuration) This bit is set if a write access to one of the ME_CCTL*n* registers is attempted while a mode transition is in progress. 0 No write to an ME_CADDR*n* register was attempted during an ongoing mode transition. 1 A write to an ME_CADDR*n* register was attempted during an ongoing mode transition.
27 I_ICONF_CU	Invalid mode configuration interrupt (Clock Usage) This bit is set during a mode transition if a clock which is required to be on by an enabled peripheral is configured to be turned off. It is cleared by writing a '1' to this bit. 0 No invalid mode configuration (clock usage) interrupt occurred. 1 Invalid mode configuration (clock usage) interrupt is pending.

Table 1800. ME_IS field descriptions

Field	Description
28 I_ICONF	Invalid mode configuration interrupt This bit is set whenever a write operation to ME_<*mode*>_MC registers with invalid mode configuration is attempted. It is cleared by writing a '1' to this bit. 0 No invalid mode configuration interrupt occurred. 1 Invalid mode configuration interrupt is pending.
29 I_IMODE	Invalid mode interrupt This bit is set whenever an invalid mode transition is requested. It is cleared by writing a '1' to this bit. 0 No invalid mode interrupt occurred. 1 Invalid mode interrupt is pending.
30 I_SAFE	SAFE mode interrupt This bit is set whenever the chip enters SAFE mode on hardware requests generated in the system. It is cleared by writing a '1' to this bit. 0 No SAFE mode interrupt occurred. 1 SAFE mode interrupt is pending.
31 I_MTC	Mode transition complete interrupt This bit is set whenever the mode transition process completes (S_MTRANS transits from 1 to 0). It is cleared by writing a '1' to this bit. This mode transition interrupt bit will not be set while entering low-power modes HALT0, STOP0, or STANDBY0. 0 No mode transition complete interrupt occurred. 1 Mode transition complete interrupt is pending.

69.3.2.5 Interrupt Mask register (ME_IM)

This register controls whether an event generates an interrupt or not.

Address: 0x010 Access: User read, Supervisor read/write, Test read/write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	M_ICONF_CC	M_ICONF_CU	M_ICONF	M_IMODE	M_SAFE	M_MTC
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1856. Interrupt Mask register (ME_IM)

Table 1801. ME_IM field descriptions

Field	Description
26 M_ICONF_CC	Invalid mode configuration (core configuration) interrupt mask 0 Invalid mode interrupt is masked. 1 Invalid mode interrupt is enabled.
27 M_ICONF_CU	Invalid mode configuration (clock usage) interrupt mask 0 Invalid mode interrupt is masked. 1 Invalid mode interrupt is enabled.
28 M_ICONF	Invalid mode configuration interrupt mask 0 Invalid mode interrupt is masked. 1 Invalid mode interrupt is enabled.
29 M_IMODE	Invalid mode interrupt mask 0 Invalid mode interrupt is masked. 1 Invalid mode interrupt is enabled.
30 M_SAFE	SAFE mode interrupt mask 0 SAFE mode interrupt is masked. 1 SAFE mode interrupt is enabled.
31 M_MTC	Mode transition complete interrupt mask 0 Mode transition complete interrupt is masked. 1 Mode transition complete interrupt is enabled.

69.3.2.6 Invalid Mode Transition Status register (ME_IMTS)

This register provides the status bits for the possible causes of an invalid mode interrupt.

Address: 0x014 Access: User read, Supervisor read/write, Test read/write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	S_MRIG	S_MTI	S_MRI	S_DMA	S_NMA	S_SEA
W											w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1857. Invalid Mode Transition Status register (ME_IMTS)

Table 1802. ME_IMTS field descriptions

Field	Description
26 S_MRIG	Mode Request Ignored status This bit is set whenever a new mode is requested while a transition to the SAFE mode is in progress, a transition to STOP0 mode is requested while a wakeup or interrupt request is active, or a transition to HALT0 mode is requested while an interrupt request is active, or a transition to STANDBY mode is requested while a wakeup request is active. It is cleared by writing a ‘1’ to this bit. 0 Mode transition requested is not ignored. 1 Mode transition requested is ignored.
27 S_MTI	Mode Transition Illegal status This bit is set whenever a new mode is requested while some other mode transition process is active (S_MTRANS is ‘1’). Please see *Section 69.4.5: Mode Transition Interrupts*, for the exceptions to this behavior. It is cleared by writing a ‘1’ to this bit. 0 Mode transition requested is not illegal. 1 Mode transition requested is illegal.
28 S_MRI	Mode Request Illegal status This bit is set whenever the target mode requested is not a valid mode with respect to current mode. It is cleared by writing a ‘1’ to this bit. 0 Target mode requested is not illegal with respect to current mode. 1 Target mode requested is illegal with respect to current mode.
29 S_DMA	Disabled Mode Access status This bit is set whenever the target mode requested is one of those disabled modes determined by ME_ME register. It is cleared by writing a ‘1’ to this bit. 0 Target mode requested is not a disabled mode. 1 Target mode requested is a disabled mode.
30 S_NMA	Non-existing Mode Access status This bit is set whenever the target mode requested is one of those non existing modes determined by ME_ME register. It is cleared by writing a ‘1’ to this bit. 0 Target mode requested is an existing mode. 1 Target mode requested is a non-existing mode.
31 S_SEA	SAFE Event Active status This bit is set whenever the chip is in SAFE mode, SAFE event bit is pending and a new mode requested other than RESET/SAFE modes. It is cleared by writing a ‘1’ to this bit. 0 No new mode requested other than RESET/SAFE while SAFE event is pending. 1 New mode requested other than RESET/SAFE while SAFE event is pending.

69.3.2.7 Debug Mode Transition Status register (ME_DMTS)

This register provides the status of different factors which influence mode transitions. It is used to give an indication of why a mode transition indicated by ME_GS.S_MTRANS may be taking longer than expected.

Note: The ME_DMTS register does not indicate whether a mode transition is ongoing. Therefore, some ME_DMTS bits may still be asserted after the mode transition has completed.

Address: 0x018 Access: User read, Supervisor read/write, Test read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	PREVIOUS_MODE				0	0	0	0	MPH_BUSY	0	0	PMC_PROG	DBG_MODE	CCKL_PROG	PCS_PROG	SMR
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	CDP_PRPH_0_255	VREG_CSRC_SC	CSRC_CSRC_SC	IRC_SC	SCSRC_SC	SYSCLK_SW	Reserved	FLASH_SC	CDP_PRPH_224_255	CDP_PRPH_192_223	CDP_PRPH_160_191	CDP_PRPH_128_159	CDP_PRPH_96_127	CDP_PRPH_64_95	CDP_PRPH_32_63	CDP_PRPH_0_31
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1858. Debug Mode Transition Status register (ME_DMTS)

Table 1803. ME_DMTS field descriptions

Field	Description
0:3 PREVIOUS_MODE	Previous chip mode These bits show the mode in which the chip was prior to the latest change to the current mode. 0000 RESET 0001 TEST 0010 SAFE 0011 DRUN 0100 RUN0 0101 RUN1 0110 RUN2 0111 RUN3 1000 HALT0 1001 Reserved 1010 STOP0 1011 Reserved 1100 Reserved 1101 STANDBY0 1110 Reserved 1111 Reserved
8 MPH_BUSY	MC_ME/MC_PCU Handshake Busy indicator This bit is set if the MC_ME has requested a mode change from the MC_PCU and the MC_PCU has not yet completed its power-up/down sequencing. It is cleared when the MC_PCU has responded. 0 Handshake is not busy. 1 Handshake is busy.

Table 1803. ME_DMTS field descriptions

Field	Description
11 PMC_PROG	MC_PCU Mode Change in Progress indicator This bit is set if the MC_PCU is in the process of powering up or down power domains. It is cleared when all power-up/down processes have completed. 0 Power-up/down transition is not in progress. 1 Power-up/down transition is in progress.
12 DBG_MODE	Debug mode indicator This bit is set while the chip is in debug mode. 0 The chip is not in debug mode. 1 The chip is in debug mode.
13 CCKL_PROG	Core Clock Enable/Disable in Progress This bit is set while any core's clock is in the process of being enabled or disabled. 0 No core clock is being enabled or disabled. 1 A core clock is being enabled or disabled.
14 PCS_PROG	Progressive System Clock Switching in Progress This bit is set while the progressive system clock switching process is in progress. 0 PCS is not in progress. 1 PCS is in progress.
15 SMR	SAFE mode request from MC_RGM is active indicator. This bit is set if a hardware SAFE mode request has been triggered. It is cleared when the hardware SAFE mode request has been cleared. 0 A SAFE mode request is not active. 1 A SAFE mode request is active.
16 CDP_PRPH_0_255	Clock Disable Process Pending status for Peripherals 0–255(1) This bit is set when any peripheral appearing has been requested to have its clock disabled. It is cleared when all the peripherals which have been requested to have their clocks disabled have entered the state in which their clocks may be disabled. 0 No peripheral clock disabling is pending. 1 Clock disabling is pending for at least one peripheral.
17 VREG_CSRC_SC	Main VREG dependent Clock Source State Change during mode transition indicator This bit is set when a clock source which depends on the main voltage regulator to be powered-up is requested to change its power up/down state. It is cleared when the clock source has completed its state change. 0 No state change is taking place. 1 A state change is taking place.
18 CSRC_CSRC_SC	(Other) Clock Source dependent Clock Source State Change during mode transition indicator This bit is set when a clock source which depends on another clock source to be powered-up is requested to change its power up/down state. It is cleared when the clock source has completed its state change. 0 No state change is taking place. 1 A state change is taking place.

Table 1803. ME_DMTS field descriptions

Field	Description
19 IRC_SC	IRC State Change during mode transition indicator This bit is set when the 16 MHz internal RC oscillator is requested to change its power up/down state. It is cleared when the 16 MHz internal RC oscillator has completed its state change. 0 No state change is taking place. 1 A state change is taking place.
20 SCSRC_SC	Secondary Clock Sources State Change during mode transition indicator This bit is set when a secondary clock source is requested to change its power up/down state. It is cleared when all secondary system clock sources have completed their state changes. (A secondary clock source is a clock source other than IRC.). 0 No state change is ta.king place 1 A state change is taking place
21 SYSCLK_SW	System Clock Switching pending status 0 No system clock source switching is pending. 1 A system clock source switching is pending.
23 FLASH_SC	FLASH State Change during mode transition indicator This bit is set when the FLASH is requested to change its power up/down state. It is cleared when the DFLASH has completed its state change. 0 No state change is taking place. 1 A state change is taking place.
24 CDP_PRPH_224_255	Clock Disable Process Pending status for Peripherals 224–255(1) This bit is set when any peripheral has been requested to have its clock disabled. It is cleared when all the peripherals which have been requested to have their clocks disabled have entered the state in which their clocks may be disabled. 0 No peripheral clock disabling is pending. 1 Clock disabling is pending for at least one peripheral.
25 CDP_PRPH_192_223	Clock Disable Process Pending status for Peripherals 192–223 This bit is set when any peripheral appearing in ME_PS6 has been requested to have its clock disabled. It is cleared when all these peripherals which have been requested to have their clocks disabled have entered the state in which their clocks may be disabled. 0 No peripheral clock disabling is pending. 1 Clock disabling is pending for at least one peripheral.
26 CDP_PRPH_160_191	Clock Disable Process Pending status for Peripherals 160–191 This bit is set when any peripheral appearing in ME_PS5 has been requested to have its clock disabled. It is cleared when all these peripherals which have been requested to have their clocks disabled have entered the state in which their clocks may be disabled. 0 No peripheral clock disabling is pending. 1 Clock disabling is pending for at least one peripheral.

Table 1803. ME_DMTS field descriptions

Field	Description
27 CDP_PRPH_128_159	Clock Disable Process Pending status for Peripherals 128–159 This bit is set when any peripheral appearing in ME_PS4 has been requested to have its clock disabled. It is cleared when all these peripherals which have been requested to have their clocks disabled have entered the state in which their clocks may be disabled. 0 No peripheral clock disabling is pending. 1 Clock disabling is pending for at least one peripheral.
28 CDP_PRPH_96_127	Clock Disable Process Pending status for Peripherals 96–127 This bit is set when any peripheral appearing in ME_PS3 has been requested to have its clock disabled. It is cleared when all these peripherals which have been requested to have their clocks disabled have entered the state in which their clocks may be disabled. 0 No peripheral clock disabling is pending. 1 Clock disabling is pending for at least one peripheral.
29 CDP_PRPH_64_95	Clock Disable Process Pending status for Peripherals 64–95 This bit is set when any peripheral appearing in ME_PS2 has been requested to have its clock disabled. It is cleared when all these peripherals which have been requested to have their clocks disabled have entered the state in which their clocks may be disabled. 0 No peripheral clock disabling is pending. 1 Clock disabling is pending for at least one peripheral.
30 CDP_PRPH_32_63	Clock Disable Process Pending status for Peripherals 32–63 This bit is set when any peripheral appearing in ME_PS1 has been requested to have its clock disabled. It is cleared when all these peripherals which have been requested to have their clocks disabled have entered the state in which their clocks may be disabled. 0 No peripheral clock disabling is pending. 1 Clock disabling is pending for at least one peripheral.
31 CDP_PRPH_0_31	Clock Disable Process Pending status for Peripherals 0–31 This bit is set when any peripheral appearing in ME_PS0 has been requested to have its clock disabled. It is cleared when all these peripherals which have been requested to have their clocks disabled have entered the state in which their clocks may be disabled. 0 No peripheral clock disabling is pending. 1 Clock disabling is pending for at least one peripheral.

1. Peripheral n corresponds to the ME_PCTL*n* register. Please refer to *Table 1796* for the ME_PCTL*n* locations actually occupied, which in turn indicates which peripherals are reported in the ME_DMTS register.

69.3.2.8 RESET Mode Configuration register (ME_RESET_MC)

This register configures system behavior during RESET mode. See *Table 1804* for details.

Note: The following configuration values are set according to the chip configuration:
XOSCON

Address: 0x020 Access: User read, Supervisor read/write, Test read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	PDO	0	0	MVRON	0	0	FLAON	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1	1

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	PLL1ON	PLL0ON	XOSCON	IRCON	SYSCLK			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0

Figure 1859. RESET Mode Configuration register (ME_RESET_MC)

69.3.2.9 TEST Mode Configuration register (ME_TEST_MC)

This register configures system behavior during TEST mode. See *Table 1804* for details.

Note: Byte write accesses are not allowed to this register.

Address: 0x024 Access: User read, Supervisor read/write, Test read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	PWRLVL(1)			0	0	0	0	PDO	0	0	MVRON	0	0	FLAON	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1	1

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	PLL1ON	PLL0ON	XOSCON	IRCON	SYSCLK			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0

1. Only bit [3] have an effect on progressive clock switching and bits [1:2] are ignored.

Figure 1860. TEST Mode Configuration register (ME_TEST_MC)

69.3.2.10 SAFE Mode Configuration register (ME_SAFE_MC)

This register configures system behavior during SAFE mode. See *Table 1804* for details.

Note: Byte write accesses are not allowed to this register.

Address: 0x028 Access: User read, Supervisor read/write, Test read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	PWRLVL(1)			0	0	0	0	PDO	0	0	MVRON	0	0	FLAON	
W																
Reset	0	0	0	0	0	0	0	0	1	0	0	1	0	0	1	1

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	PLL1ON	PLL0ON	XOSCON	IRCON	SYSCLK			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0

1. Only bit [3] have an effect on progressive clock switching and bits [1:2] are ignored.

Figure 1861. SAFE Mode Configuration register (ME_SAFE_MC)

69.3.2.11 DRUN Mode Configuration register (ME_DRUN_MC)

This register configures system behavior during DRUN mode. See *Table 1804* for details.

Note: Byte write accesses are not allowed to this register.

The following configuration values are set according to the chip configuration:

XOSCON

Address: 0x02C Access: User read, Supervisor read/write, Test read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	PWRLVL(1)			0	0	0	0	PDO	0	0	MVRON	0	0	FLAON	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1	1

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	PLL1ON	PLL0ON	XOSCON	IRCON	SYSCLK			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0

1. Only bit [3] have an effect on progressive clock switching and bits [1:2] are ignored.

Figure 1862. DRUN Mode Configuration register (ME_DRUN_MC)

69.3.2.12 RUN*n* Mode Configuration registers (ME_RUN*n*_MC)

This register configures system behavior during RUN0–3 modes. See *Table 1804* for details.

Note: Byte write accesses are not allowed to this register.

Address: 0x030 + n*0x4 (n = 0 to 3) Access: User read, Supervisor read/write, Test read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	PWRLVL(1)			0	0	0	0	PDO	0	0	MVRON	0	0	FLAON	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1	1

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	PLL1ON	PLL0ON	XOSCON	IRCON	SYSCLK			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0

1. Only bit [3] have an effect on progressive clock switching and bits [1:2] are ignored.

Figure 1863. RUN*n* Mode Configuration registers (ME_RUN*n*_MC)

69.3.2.13 HALT0 Mode Configuration register (ME_HALT0_MC)

This register configures system behavior during HALT0 mode. See *Table 1804* for details.

Note: Byte write accesses are not allowed to this register.

Address: 0x040 Access: User read, Supervisor read/write, Test read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	PDO	0	0	MVRON	0	0	FLAON	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	PLL1ON	PLL0ON	XOSCON	IRCON	SYSCLK			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0

Figure 1864. HALT0 Mode Configuration register (ME_HALT0_MC)

69.3.2.14 STOP0 Mode Configuration register (ME_STOP0_MC)

This register configures system behavior during STOP0 mode. See *Table 1804* for details.

Note: Byte write accesses are not allowed to this register.

Address: 0x048 Access: User read, Supervisor read/write, Test read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	PDO	0	0	MVRON	0	0	FLAON	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	PLL1ON	PLL0ON	XOSCON	IRCON	SYSCLK			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0

Figure 1865. STOP0 Mode Configuration register (ME_STOP0_MC)

69.3.2.15 STANDBY0 Mode Configuration register (ME_STANDBY0_MC)

Address: 0x054 Access: User read, Supervisor read/write, Test read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	PDO	0	0	MVRON	0	0	FLAON	
W																
Reset	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	1

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	PLL1ON	PLL0ON	XOSCON	IRCON	SYSCLK			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1

Figure 1866. STANDBY0 Mode Configuration register (ME_STANDBY0_MC)

This register configures system behavior during STANDBY0 mode. Please refer to *Table 1804* for details.

Note: Byte write accesses are not allowed to this register.

Table 1804. ME_<*mode*>_MC field descriptions (continued)

Field	Description
1:3 PWRLVL	Power level These bits indicate the relative power consumption level of this mode with respect to that of other modes. When switching between two modes with differing power levels, system clock progressive switching is enabled. **Note:** For ME_STANDBY0_MC, ME_RESET_MC, ME_HALT0_MC and ME_STOP0_MC registers this bit field is read-only and always 0.
8 PDO	I/O output power-down control This bit controls the output power-down of I/Os. 0 No automatic safe gating of I/Os used and pads power sequence driver is enabled. 1 In SAFE/TEST modes, outputs of pads are forced to high impedance state and pads power sequence driver is disabled. The inputs are level unchanged. In STOP0 mode, only the pad power sequence driver is disabled, but the state of the output remains functional. In STANDBY0 mode, power sequence driver and all pads except those mapped on wakeup lines are not powered and therefore high impedance. Wakeup line configuration remains unchanged.
11 MVRON	Main voltage regulator control This bit specifies whether main voltage regulator is switched off or not while entering this mode. 1 Main voltage regulator is switched on.
14:15 FLAON	Flash power-down control This bit specifies the operating mode of the code flash after entering this mode. 00 Reserved 01 Flash is in power-down mode. 10 Flash is in low-power mode. 11 Flash is in normal mode.
24 PLL1ON	secondary PLL control 0 secondary PLL is switched off. 1 secondary PLL is switched on.
25 PLL0ON	primary PLL control 0 primary PLL is switched off. 1 primary PLL is switched on.
26 XOSCON	external crystal oscillator control 0 external crystal oscillator is switched off. 1 external crystal oscillator is switched on.

Table 1804. ME_<*mode*>_MC field descriptions (continued)

Field	Description
27 IRCON	16 MHz internal RC oscillator control 0 16 MHz internal RC oscillator is switched off. 1 16 MHz internal RC oscillator is switched on.
28:31 SYSCLK	System clock switch control These bits specify the system clock to be used by the system. 0000 16 MHz int. RC osc. 0001 external crystal osc. 0010 primary PLL (PHI) 0011 reserved 0100 secondary PLL 0101 reserved 0110 reserved 0111 reserved 1000 reserved 1001 reserved 1010 reserved 1011 reserved 1100 reserved 1101 reserved 1110 reserved 1111 system clock is disabled in TEST mode, reserved in all other modes

69.3.2.16 Peripheral Status register 0 (ME_PS0)

This register provides the status of the peripherals. See *Table 1805* for details.

Address: 0x060 Access: User read, Supervisor read, Test read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	S_PIT_0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	S_SIUL	0	0	0	S_SIPI_0	0	S_LFAST_0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1867. Peripheral Status register 0 (ME_PS0)

69.3.2.17 Peripheral Status register 1 (ME_PS1)

This register provides the status of the peripherals. See *Table 1805* for details.

Address: 0x064 Access: User read, Supervisor read, Test read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	S_CCCU	0	S_SD_ADC_0	S_SD_ADC_2	S_SD_ADC_4	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	S_CRC_0	0	S_DMAMUX_0_2_4	0	0	S_PSI5_S_0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1868. Peripheral Status register 1 (ME_PS1)

69.3.2.18 Peripheral Status register 2 (ME_PS2)

This register provides the status of the peripherals. See *Table 1805* for details.

Address: 0x068 Access: User read, Supervisor read, Test read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	S_DSPI_8	0	0	S_LINFlexD_0	S_LINFlexD_2	S_LINFlexD_4	S_LINFlexD_6	S_LINFlexD_8	S_LINFlexD_10	S_LINFlexD_12	S_LINFlexD_14	S_LINFlexD_16	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	S_CANSUBSYS0_CAN_RAM_CTR	0	S_TTCAN	0	0	S_CANSUBSYS0_MCAN_1	S_CANSUBSYS0_MCAN_2	S_CANSUBSYS0_MCAN_3	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1869. Peripheral Status register 2 (ME_PS2)

69.3.2.19 Peripheral Status register 3 (ME_PS3)

This register provides the status of the peripherals. See *Table 1805* for details.

Address: 0x06C Access: User read, Supervisor read, Test read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	S_SAR_ADC_12BIT_0	S_SAR_ADC_12BIT_2	0	0	0	0	0	0	S_SAR_ADC_10BIT_0	0	0	0	0	S_SAR_ADC_10BIT_STDBY	0	S_SAR_ADC_12BIT_B
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	S_PSI5_0	0	0	0	S_FlexRay_0	0	0	S_SENT_0	0	0	S_IIC_0	0	S_DSPI_0	S_DSPI_2	S_DSPI_4	S_DSPI_6
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1870. Peripheral Status register 3 (ME_PS3)

69.3.2.20 Peripheral Status register 4 (ME_PS4)

This register provides the status of the peripherals. See *Table 1805* for details.

Address: 0x070 Access: User read, Supervisor read, Test read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	S_PIT_1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	S_SAR_ADC_12BIT_SEQ_0	S_SAR_ADC_12BIT_SEQ_2	S_SAR_ADC_12BIT_SEQ_1	S_SAR_ADC_12BIT_SEQ_3	0	0	0	0	S_SAR_ADC_10BIT_SEQ_0	S_SAR_ADC_10BIT_SEQ_1	0	0	0	S_SAR_ADC_SEQ_STDBY	S_SAR_ADC_SEQ_B	S_GTMINT
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1871. Peripheral Status register 4 (ME_PS4)

69.3.2.21 Peripheral Status register 5 (ME_PS5)

This register provides the status of the peripherals. See *Table 1805* for details.

Address: 0x074 Access: User read, Supervisor read, Test read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	S_SD_ADC_1	S_SD_ADC_3	S_SD_ADC_5	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	S_CRC_1	0	S_DMAMUX_1_3_5	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1872. Peripheral Status register 5 (ME_PS5)

69.3.2.22 Peripheral Status register 6 (ME_PS6)

This register provides the status of the peripherals. See *Table 1805* for details.

Address: 0x078 Access: User read, Supervisor read, Test read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	S_DSPI_9	0	0	S_LINFlexD_1	S_LINFlexD_3	S_LINFlexD_5	S_LINFlexD_7	S_LINFlexD_9	S_LINFlexD_11	S_LINFlexD_13	S_LINFlexD_15	S_LINFlexD_17	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	S_CANSUBSYS1_CAN_RAM_CTR	0	0	0	S_CANSUBSYS1_MCAN_1	S_CANSUBSYS1_MCAN_2	S_CANSUBSYS1_MCAN_3	S_CANSUBSYS1_MCAN_4	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1873. Peripheral Status register 6 (ME_PS6)

69.3.2.23 Peripheral Status register 7 (ME_PS7)

This register provides the status of the peripherals.

Address: 0x07C Access: User read, Supervisor read, Test read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	S_SAR_ADC_12BIT_1	S_SAR_ADC_12BIT_3	0	0	0	0	0	0	S_SAR_ADC_10BIT_1	0	0	0	S_BODY_CTU_0	S_STDBY_CTU_0	S_EMIOS_0	S_EMIOS_1
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	S_PSI5_1	0	0	0	0	S_ETHERNET_0	S_ETHERNET_1	S_SENT_1	0	0	0	0	S_DSPI_1	S_DSPI_3	S_DSPI_5	S_DSPI_7
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1874. Peripheral Status register 7 (ME_PS7)

Table 1805. ME_PS*n* field descriptions

Field	Description
S_<*periph*>	Peripheral status These bits specify the current status of each peripheral which is controlled by the MC_ME. 0 Peripheral is frozen. 1 Peripheral is active.

69.3.2.24 Run Peripheral Configuration registers (ME_RUN_PC*n*)

These registers configure eight different types of peripheral behavior during run modes.

Address: 0x080 +*n**0x4 (*n* = 0 to 7) Access: User read, Supervisor read/write, Test read/write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	RUN3	RUN2	RUN1	RUN0	DRUN	SAFE	TEST	RESET
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1875. Run Peripheral Configuration registers (ME_RUN_PC*n*)

Table 1806. ME_RUN_PC*n* field descriptions

Field	Description
24 RUN3	Peripheral control during RUN3 0 Peripheral is frozen with clock gated. 1 Peripheral is active.
25 RUN2	Peripheral control during RUN2 0 Peripheral is frozen with clock gated. 1 Peripheral is active.
26 RUN1	Peripheral control during RUN1 0 Peripheral is frozen with clock gated. 1 Peripheral is active.
27 RUN0	Peripheral control during RUN0 0 Peripheral is frozen with clock gated. 1 Peripheral is active.
28 DRUN	Peripheral control during DRUN 0 Peripheral is frozen with clock gated. 1 Peripheral is active.
29 SAFE	Peripheral control during SAFE 0 Peripheral is frozen with clock gated. 1 Peripheral is active.
30 TEST	Peripheral control during TEST 0 Peripheral is frozen with clock gated. 1 Peripheral is active.
31 RESET	Peripheral control during RESET 0 Peripheral is frozen with clock gated. 1 Peripheral is active.

69.3.2.25 Low-Power Peripheral Configuration registers (ME_LP_PC*n*)

These registers configure eight different types of peripheral behavior during non-run modes.

Address: 0x0A0 + *n**0x4 (*n* = 0 to 7) Access: User read, Supervisor read/write, Test read/write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	STANDBY0	0	0	STOP0	0	HALT0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1876. Low-Power Peripheral Configuration registers (ME_LP_PC*n*)

Table 1807. ME_LP_PC*n* field descriptions

Field	Description
18 STANDBY0	Peripheral control during STANDBY0 0 Peripheral is frozen with clock gated. 1 Peripheral is active.
21 STOP0	Peripheral control during STOP0 0 Peripheral is frozen with clock gated. 1 Peripheral is active.
23 HALT0	Peripheral control during HALT0 0 Peripheral is frozen with clock gated. 1 Peripheral is active.

69.3.2.26 Peripheral Control registers (ME_PCTL*n*)

These registers select the configurations during run and non-run modes for each peripheral. Please see *Table 1796* for information on which ME_PCTL*n* locations are actually occupied. The unoccupied locations contain a read-only byte value of 0x00.

Note: After modifying any of the ME_RUN_PC0–7, ME_LP_PC0–7, and ME_PCTLn registers, software must request a mode change and wait for the mode change to be completed before entering debug mode in order to have consistent behavior between the peripheral clock control process and the clock status reporting in the ME_PSn registers.

Address: 0x0C0 + *n**0x1 (*n* = 0 to 255) Access: User read, Supervisor read/write, Test read/write

	0	1	2	3	4	5	6	7
R	0	0(1)	LP_CFG			RUN_CFG		
W								
Reset	0	0	0	0	0	0	0	0

1. This bit is writable but has no impact.

Figure 1877. Peripheral Control registers (ME_PCTL*n*)

Table 1808. ME_PCTL*n* field descriptions

Field	Description
2:4 LP_CFG	Peripheral configuration select for non-run modes These bits associate a configuration as defined in the ME_LP_PC0–7 registers to the peripheral. 000 Selects ME_LP_PC0 configuration 001 Selects ME_LP_PC1 configuration 010 Selects ME_LP_PC2 configuration 011 Selects ME_LP_PC3 configuration 100 Selects ME_LP_PC4 configuration 101 Selects ME_LP_PC5 configuration 110 Selects ME_LP_PC6 configuration 111 Selects ME_LP_PC7 configuration
5:7 RUN_CFG	Peripheral configuration select for run modes These bits associate a configuration as defined in the ME_RUN_PC0–7 registers to the peripheral. 000 Selects ME_RUN_PC0 configuration 001 Selects ME_RUN_PC1 configuration 010 Selects ME_RUN_PC2 configuration 011 Selects ME_RUN_PC3 configuration 100 Selects ME_RUN_PC4 configuration 101 Selects ME_RUN_PC5 configuration 110 Selects ME_RUN_PC6 configuration 111 Selects ME_RUN_PC7 configuration

69.3.2.27 Core Status register (ME_CS)

This register provides the status of the additional cores.

Address: 0x1C0 Access: User read, Supervisor read, Test read

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	ChkINTC_Stat	ChkDMA1_Stat	ChkDMA0_Stat	0	0	S_CHK_CORE5	S_CORE4	S_CORE3	S_CHK_CORE2	S_CORE1	S_CORE0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

Figure 1878. Core Status register (ME_CS)

Table 1809. ME_CS field descriptions

Field	Description
21 ChkINTC_Stat	INTC checkered module This bit specifies the current status INTC checkered module. 0 INTC checker module is disabled. 1 INTC checker module is running.
22 ChkDMA1_Stat	DMA1 checkered module This bit specifies the current status DMA1 checkered module. 0 DMA1 checker module is disabled. 1 DMA1 checker module is running.
23 ChkDMA0_Stat	DMA0 checkered module This bit specifies the current status DMA0 checkered module. 0 DMA0 checker module is disabled. 1 DMA0 checker module is running.
26 S_CHK_CORE5	Core 5 status This bit specifies the current status of checker core_2. 0 Checker core_2 is disabled. 1 Checker core_2 is running.
27 S_CORE4	Core 4 status This bit specifies the current status of the HSM which is controlled by the ME_CCTL4 register. 0 The HSM is disabled. 1 The HSM is running.
28 S_CORE3	Core 3 status This bit specifies the current status of main core_1 which is controlled by the ME_CCTL3 register. 0 Main core_1 is disabled. 1 Main core_1 is running.
29 S_CHK_CORE2	Core 2 status This bit specifies the current status of checker core_0. 0 Checker core_0 is disabled. 1 Checker core_0 is running.
30 S_CORE1	Core 1 status This bit specifies the current status of main core_0 which is controlled by the ME_CCTL1 register. 0 Main core_0 is disabled. 1 Main core_0 is running.
31 S_CORE0	Core 0 status This bit specifies the current status of peripheral core_2 which is controlled by the ME_CCTL0 register. 0 Peripheral core_2 is disabled. 1 Peripheral core_2 is running.

69.3.2.28 Core Control register 0 (ME_CCTL0)

This register controls whether peripheral core_2 is disabled or running during run modes.

This register cannot be written after a mode change request has been made until the mode transition has completed (that is while ME_GS[S_MTRANS] = 1).

Address: 0x1C4 Access: User read, Supervisor read/write, Test read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	STANDBY0	0	0	STOP0	0	HALT0	RUN3	RUN2	RUN1	RUN0	DRUN	SAFE	TEST	RESET
W																
Reset	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	0

Figure 1879. Core Control register 0 (ME_CCTL0)

Table 1810. ME_CCTL0 field descriptions (continued)

Field	Description
2 STANDBY0	Core control during STANDBY0 The peripheral core_2 is always disabled during STANDBY0.
5 STOP0	Core control during STOP0 The peripheral core_2 is always disabled during STOP0.
7 HALT0	Core control during HALT0 The peripheral core_2 is always disabled during HALT0.
8 RUN3	Core control during RUN3 0 Peripheral core_2 is disabled with clock gated. 1 Peripheral core_2 is running.
9 RUN2	Core control during RUN2 0 Peripheral core_2 is disabled with clock gated. 1 Peripheral core_2 is running.
10 RUN1	Core control during RUN1 0 Peripheral core_2 is disabled with clock gated. 1 Peripheral core_2 is running.
11 RUN0	Core control during RUN0 0 Peripheral core_2 is frozen with clock gated. 1 Peripheral core_2 is running.
12 DRUN	Core control during DRUN 0 Peripheral core_2 is frozen with clock gated. 1 Peripheral core_2 is running.
13 SAFE	Core control during SAFE 0 Peripheral core_2 is frozen with clock gated. 1 Peripheral core_2 is running.
14 TEST	Core control during TEST 0 Peripheral core_2 is frozen with clock gated. 1 Peripheral core_2 is running.
15 RESET	Core control during RESET The peripheral core_2 is always disabled during RESET.

69.3.2.29 Core Control register (ME_CCTL1)

This register controls whether main core_0 is disabled or running during run modes.

This register cannot be written after a mode change request has been made until the mode transition has completed (that is while ME_GS[S_MTRANS] = 1).

Address: 0x1C6 Access: User read, Supervisor read/write, Test read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	STANDBY0	0	0	STOP0	0	HALT0	RUN3	RUN2	RUN1	RUN0	DRUN	SAFE	TEST	RESET
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1880. Core Control register (ME_CCTL1)

Table 1811. ME_CCTL1 field descriptions

Field	Description
2 STANDBY0	Core control during STANDBY0 The main core_0 is always disabled during STANDBY0.
5 STOP0	Core control during STOP0 The main core_0 is always disabled during STOP0.
7 HALT0	Core control during HALT0 The main core_0 is always disabled during HALT0.
8 RUN3	Core control during RUN3 0 Main core_0 is disabled with clock gated. 1 Main core_0 is running.
9 RUN2	Core control during RUN2 0 Main core_0 is disabled with clock gated. 1 Main core_0 is running.
10 RUN1	Core control during RUN1 0 Main core_0 is disabled with clock gated. 1 Main core_0 is running.
11 RUN0	Core control during RUN0 0 Main core_0 is frozen with clock gated. 1 Main core_0 is running.
12 DRUN	Core control during DRUN 0 Main core_0 is frozen with clock gated. 1 Main core_0 is running.
13 SAFE	Core control during SAFE 0 Main core_0 is frozen with clock gated. 1 Main core_0 is running.

Table 1811. ME_CCTL1 field descriptions (continued)

Field	Description
14 TEST	Core control during TEST 0 Main core_0 is frozen with clock gated. 1 Main core_0 is running.
15 RESET	Core control during RESET The main core_0 is always disabled during RESET.

69.3.2.30 Core Control register 3 (ME_CCTL3)

This register controls whether main core_1 is disabled or running during run modes.

This register cannot be written after a mode change request has been made until the mode transition has completed (that is while ME_GS[S_MTRANS] = 1).

Address: 0x1CA Access: User read, Supervisor read/write, Test read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	STANDBY0	0	0	STOP0	0	HALT0	RUN3	RUN2	RUN1	RUN0	DRUN	SAFE	TEST	RESET
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1881. Core Control register 3 (ME_CCTL3)

Table 1812. ME_CCTL3 field descriptions

Field	Description
2 STANDBY0	Core control during STANDBY0 The main core_1 is always disabled during STANDBY0.
5 STOP0	Core control during STOP0 The main core_1 is always disabled during STOP0.
7 HALT0	Core control during HALT0 The main core_1 is always disabled during HALT0.
8 RUN3	Core control during RUN3 0 Main core_1 is disabled with clock gated. 1 Main core_1 is running.
9 RUN2	Core control during RUN2 0 Main core_1 is disabled with clock gated. 1 Main core_1 is running.
10 RUN1	Core control during RUN1 0 Main core_1 is disabled with clock gated. 1 Main core_1 is running.
11 RUN0	Core control during RUN0 0 Main core_1 is frozen with clock gated. 1 Main core_1 is running.

Table 1812. ME_CCTL3 field descriptions (continued)

Field	Description
12 DRUN	Core control during DRUN 0 Main core_1 is frozen with clock gated. 1 Main core_1 is running.
13 SAFE	Core control during SAFE 0 Main core_1 is frozen with clock gated. 1 Main core_1 is running.
14 TEST	Core control during TEST 0 Main core_1 is frozen with clock gated. 1 Main core_1 is running.
15 RESET	Core control during RESET The main core_1 is always disabled during RESET.

69.3.2.31 Core Control Register (ME_CCTL4)

This register controls whether the HSM is disabled or running during run modes.

This register cannot be written after a mode change request has been made until the mode transition has completed (that is while ME_GS[S_MTRANS] = 1).

Address: 0x1CC Access: User read, Supervisor read/write, Test read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	STANDBY0	0	0	STOP0	0	HALT0	RUN3	RUN2	RUN1	RUN0	DRUN	SAFE	TEST	RESET
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1882. Core Control Register (ME_CCTL4)

Table 1813. ME_CCTL4 field descriptions

Field	Description
2 STANDBY0	Core control during STANDBY0 The HSM is always disabled during STANDBY0.
5 STOP0	Core control during STOP0 The HSM is always disabled during STOP0.
7 HALT0	Core control during HALT0 The HSM is always disabled during HALT0.
8 RUN3	Core control during RUN3 0 The HSM is disabled with clock gated. 1 The HSM is running.
9 RUN2	Core control during RUN2 0 The HSM is disabled with clock gated. 1 The HSM is running.

Table 1813. ME_CCTL4 field descriptions (continued)

Field	Description
10 RUN1	Core control during RUN1 0 The HSM is disabled with clock gated. 1 The HSM is running.
11 RUN0	Core control during RUN0 0 The HSM is frozen with clock gated. 1 The HSM is running.
12 DRUN	Core control during DRUN 0 The HSM is frozen with clock gated. 1 The HSM is running.
13 SAFE	Core control during SAFE 0 The HSM is frozen with clock gated. 1 The HSM is running.
14 TEST	Core control during TEST 0 The HSM is frozen with clock gated. 1 The HSM is running.
15 RESET	Core control during RESET The HSM is always disabled during RESET.

69.3.2.32 Core Address Register 0 (ME_CADDR0)

This register gives the boot address for peripheral core_2 and a bit for controlling whether peripheral core_2 is to be reset on the next mode change that has peripheral core_2 configured to be running in the target mode.

This register can be written only as a word and cannot be written after a mode change request has been made until the mode transition has completed (that is while ME_GS[S_MTRANS] = 1). A write access to this register during this time will result in the ME_IS[ICONF_CC] flag being asserted.

Note: The reset value of the ME_CADDR0[ADDR] field is determined by the chip configuration and boot mode.

Address: 0x1E0 Access: User read, Supervisor read/write, Test read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	ADDR															
Reset	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	ADDR														0	RMC
Reset	X	X	X	X	X	X	X	X	X	X	X	X	X	X	0	0

Figure 1883. Core Address Register 0 (ME_CADDR0)

Table 1814. ME_CADDR0 field descriptions

Field	Description
0:29 ADDR	Core Address This field is used by peripheral core_2 as the boot address (32-bit word aligned) when peripheral core_2 next exits reset.
31 RMC	Reset on Mode Change Peripheral core_2 will be reset on the next mode change that has peripheral core_2 configured to be running in the target mode. 0 Peripheral core_2 will not be reset on the next mode change. 1 Peripheral core_2 will be reset on the next mode change that has peripheral core_2 configured to be running in the target mode.

69.3.2.33 Core Address Register 1 (ME_CADDR1)

This register gives the boot address for main core_0 and a bit for controlling whether main core_0 is to be reset on the next mode change that has main core_0 configured to be running in the target mode.

This register can be written only as a word and cannot be written after a mode change request has been made until the mode transition has completed (that is while ME_GS[S_MTRANS] = 1). A write access to this register during this time will result in the ME_IS[ICONF_CC] flag being asserted.

Address: 0x1E4 Access: User read, Supervisor read/write, Test read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	ADDR															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	ADDR														0	RMC
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1884. Core Address Register 1 (ME_CADDR1)

Table 1815. ME_CADDR1 field descriptions

Field	Description
0:29 ADDR	Core Address This field is used by main core_0 as the boot address (32-bit word aligned) when main core_0 next exits reset.
31 RMC	Reset on Mode Change Main core_0 will be reset on the next mode change that has main core_0 configured to be running in the target mode. 0 Main core_0 will not be reset on the next mode change. 1 Main core_0 will be reset on the next mode change that has main core_0 configured to be running in the target mode.

69.3.2.34 Core Address Register 3 (ME_CADDR3)

This register gives the boot address for main core_1 and a bit for controlling whether main core_1 is to be reset on the next mode change that has main core_1 configured to be running in the target mode.

This register can be written only as a word and cannot be written after a mode change request has been made until the mode transition has completed (that is while ME_GS[S_MTRANS] = 1). A write access to this register during this time will result in the ME_IS[ICONF_CC] flag being asserted.

Address: 0x1EC Access: User read, Supervisor read/write, Test read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	ADDR															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	ADDR														0	RMC
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1885. Core Address Register 3 (ME_CADDR3)

Table 1816. ME_CADDR3 field descriptions

Field	Description
0:29 ADDR	Core Address This field is used by main core_1 as the boot address (32-bit word aligned) when main core_1 next exits reset.
31 RMC	Reset on Mode Change Main core_1 will be reset on the next mode change that has main core_1 configured to be running in the target mode. 0 Main core_1 will not be reset on the next mode change. 1 Main core_1 will be reset on the next mode change that has main core_1 configured to be running in the target mode.

69.3.2.35 Core Address Register 4 (ME_CADDR4)

This register gives the boot address for the HSM and a bit for controlling whether the HSM is to be reset on the next mode change that has the HSM configured to be running in the target mode.

This register can be written only as a word and cannot be written after a mode change request has been made until the mode transition has completed (that is while ME_GS[S_MTRANS] = 1). A write access to this register during this time will result in the ME_IS[ICONF_CC] flag being asserted.

Note: The reset value of the ADDR field of the ME_CADDR4 is determined by the chip configuration.

Address: 0x1F0 Access: User read, Supervisor read/write, Test read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	ADDR															
Reset	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	ADDR														0	RMC
Reset	X	X	X	X	X	X	X	X	X	X	X	X	X	X	0	0

Figure 1886. Core Address Register 4 (ME_CADDR4)

Table 1817. ME_CADDR4 field descriptions

Field	Description
0:29 ADDR	Core Address This field is used by the HSM as the boot address (32-bit word aligned) when main core_0 next exits reset.
31 RMC	Reset on Mode Change The HSM will be reset on the next mode change that has the HSM configured to be running in the target mode. 0 The HSM will not be reset on the next mode change. 1 The HSM will be reset on the next mode change that has the HSM configured to be running in the target mode.

69.4 Functional description

69.4.1 Mode transition request

The transition from one mode to another mode is normally handled by software by accessing the mode control register ME_MCTL. But the in case of special events, the mode transition can be automatically managed by hardware. In order to switch from one mode to another, the application should access the ME_MCTL register twice by writing

- the first time with the value of the key (0x5AF0) into the KEY bit field and the required target mode into the TARGET_MODE bit field,
- and the second time with the inverted value of the key (0xA50F) into the KEY bit field and the required target mode into the TARGET_MODE bit field.

Once a valid mode transition request is detected, the target mode configuration information is loaded from the corresponding ME_<mode>_MC register.The mode transition request may require a number of cycles depending on the programmed configuration, and software should check the S_CURRENT_MODE bit field and the S_MTRANS bit of the global status register ME_GS to verify when the mode has been correctly entered and the transition process has completed. For a description of valid mode requests, please see *Section 69.4.5: Mode Transition Interrupts*.

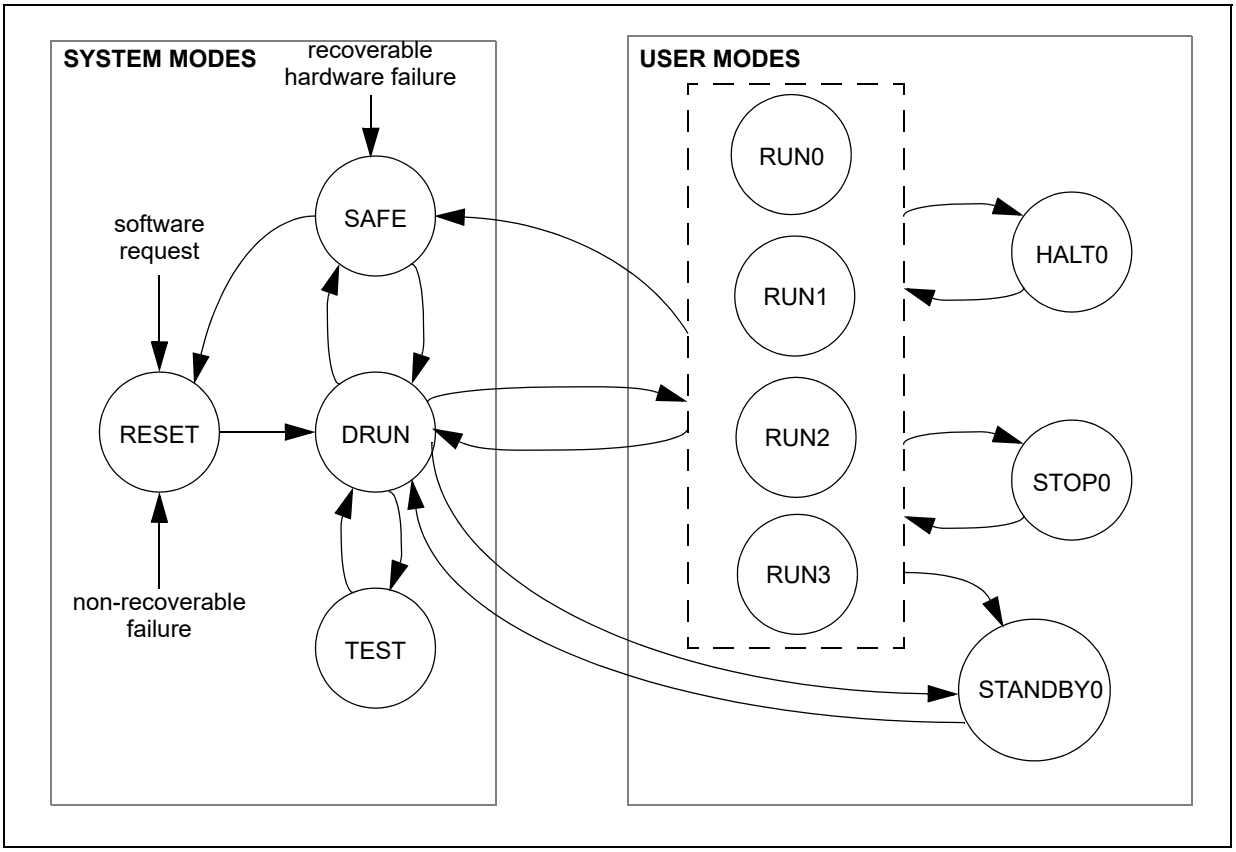
Any modification of the mode configuration register of the currently selected mode will not be taken into account immediately but on the next request to enter this mode. This means that transition requests such as RUN0–3 → RUN0–3, DRUN → DRUN, SAFE → SAFE, and

TEST → TEST are considered valid mode transition requests. As soon as the mode request is accepted as valid, the S_MTRANS bit is set till the status in the ME_GS register matches the configuration programmed in the respective ME_<mode>_MC register.

Note: It is recommended that software polls the S_MTRANS bit in the ME_GS register after requesting a transition to HALT0, STOP0, or STANDBY0 modes.

If CMU reveals a loss of PLL/XOSC clock before a transition request, such transition is not completed. If failure removal and CMU flags are reset, then transition is executed correctly.

Figure 1887. MC_ME Mode Diagram



69.4.2 Modes details

69.4.2.1 RESET mode

The chip enters this mode on the following events:

- from SAFE, DRUN, RUN0–3, or TEST mode when the TARGET_MODE bit field of the ME_MCTL register is written with either “0000” for a ‘functional’ reset or “1111” for a ‘destructive’ reset
- from any mode due to a system reset by the MC_RGM because of some non-recoverable hardware failure in the system (see the MC_RGM chapter for details)

Transition to this mode is instantaneous, and the system remains in this mode until the reset sequence is finished. The mode configuration information for this mode is provided by the ME_RESET_MC register. This mode has a pre-defined configuration, and the 16 MHz int. RC osc. is selected as the system clock.

69.4.2.2 DRUN mode

The chip enters this mode on the following events:

- automatically from RESET mode after completion of the reset sequence
- from RUN0–3, SAFE, or TEST mode when the TARGET_MODE bit field of the ME_MCTL register is written with “0011”
- from the STANDBY0 mode after an external wakeup event or internal wakeup alarm.

As soon as any of the above events has occurred, a DRUN mode transition request is generated. The mode configuration information for this mode is provided by the ME_DRUN_MC register. In this mode, the flash, all clock sources, and the system clock configuration can be controlled by software as required. After system reset, the software execution starts with the default configuration selecting the 16 MHz int. RC osc. as the system clock. The clock source configuration except system clock source 0 can be chosen by preloaded information in flash memory. This information is loaded into the ME_DRUN_MC register during PHASE3 of the reset sequence.

This mode is intended to be used by software

- to initialize all registers as per the system needs
- to execute small routines in a ‘ping-pong’ with the STANDBY0 mode

When this mode is entered from STANDBY0 after a wakeup event, the ME_DRUN_MC register content is restored to its pre-STANDBY0 values, and the mode starts in that configuration.

Note: the exit from STANDBY0 after a wakeup event,*Software must ensure that the code executes from RAM before changing to this mode if the flash is configured to be in the low-power or power-down state in this mode.*

69.4.2.3 SAFE mode

The chip enters this mode on the following events:

- from any mode except RESET when the TARGET_MODE bit field of the ME_MCTL register is written with “0010”
- from any mode except RESET due to a SAFE mode request generated by the MC_RGM because of some potentially recoverable hardware failure in the system (see *Chapter 64: Reset Generation Module (MC_RGM)*, for details)

Note: If a hardware SAFE mode request occurs during RESET, depending on the timing of the SAFE mode request, SAFE mode may be entered immediately after the normal completion of the reset sequence or several system clock cycles after DRUN entry. The SAFE mode request does not have any influence on the execution of the reset sequence itself.

As soon as any of the above events has occurred, a SAFE mode transition request is generated. The mode configuration information for this mode is provided by the ME_SAFE_MC register. This mode has a pre-defined configuration, and the 16 MHz int. RC osc. is selected as the system clock.

If the SAFE mode is requested by hardware, a SAFE mode interrupt is generated. See *Section 69.4.5.3: SAFE Mode Transition Interrupt* for details.

If the SAFE mode is requested by software while some other mode transition process is ongoing, the new target mode becomes the SAFE mode regardless of other pending requests or new requests during the mode transition. No new mode request made during a transition to the SAFE mode will cause an invalid mode interrupt.

Note: If software requests to change to the SAFE mode and then requests to change back to the parent mode before the mode transition is completed, the chip's final mode after mode transition will be the SAFE mode, and an invalid mode transition interrupt will be generated.

As long as a SAFE event is active, the system remains in the SAFE mode, and any software mode request during this time is ignored and lost.

This mode is intended to be used by software

- to assess the severity of the cause of failure and then to either
  - re-initialize the chip via the DRUN mode, or
  - completely reset the chip via the RESET mode.

If the outputs of the system I/Os need to be forced to a high impedance state upon entering this mode, the PDO bit of the ME_SAFE_MC register should be set. The input levels remain unchanged.

69.4.2.4 TEST mode

The chip enters this mode on the following event:

- from the DRUN mode when the TARGET_MODE bit field of the ME_MCTL register is written with "0001"

As soon as any of the above event has occurred, a TEST mode transition request is generated. The mode configuration information for this mode is provided by the ME_TEST_MC register. Except for the main voltage regulator, all resources of the system are configurable in this mode. The system clock to the whole system can be stopped by programming the SYSCLK bit field to "1111", and in this case, the only way to exit this mode is via a chip reset.

This mode is intended to be used by software

- to execute software test routines

Note: Software must ensure that the code executes from RAM before changing to this mode if the flash is configured to be in the low-power or power-down state in this mode.

69.4.2.5 RUN0–3 modes

The chip enters one of these modes on the following events:

- from the DRUN, SAFE, or another RUN0–3 mode when the TARGET_MODE bit field of the ME_MCTL register is written with "0100–0111"
- from the HALT0 mode due to an interrupt event
- from the STOP0 mode due to an interrupt or wakeup event

As soon as any of the above events has occurred, a RUN0–3 mode transition request is generated. The mode configuration information for these modes is provided by the ME_RUN0–3_MC registers. In these modes, the flash, all clock sources, and the system clock configuration can be controlled by software as required.

These modes are intended to be used by software

- to execute application routines

Note: Software must ensure that the code executes from RAM before changing to this mode if the flash is configured to be in the low-power or power-down state in this mode.

69.4.2.6 HALT0 mode

The chip enters this mode on the following event:

- from one of the RUN0–3 modes when the TARGET_MODE bit field of the ME_MCTL register is written with “1000”.

As soon as any of the above event has occurred, a HALT0 mode transition request is generated. The mode configuration information for this mode is provided by ME_HALT0_MC register. This mode is quite configurable, and the ME_HALT0_MC register should be programmed according to the system needs. The flash can be put in low-power or power-down mode as needed. If there is a HALT0 mode request while an interrupt request is active, the transition to HALT0 is aborted with the resultant mode being the current mode, SAFE (on SAFE mode request), or DRUN (on reset), and an invalid mode interrupt is not generated.

This mode is intended as a first-level low-power mode with

- the core clocks frozen
- the additional core clocks frozen
- only a few peripherals running

and to be used by software

- to wait until it is required to do something and then to react quickly (that is within a few system clock cycles of an interrupt event)

The normal exit of the HALT0 mode is from an interrupt event. Wakeup events will not cause HALT0 mode exit.

Note: It is good practice for software to ensure that the S_MTRANS bit in the ME_GS register has been cleared on HALT0 mode exit to ensure that the previous RUN0–3 mode configuration has been fully restored before executing critical code.

69.4.2.7 STOP0 mode

The chip enters this mode on the following event:

- from one of the RUN0–3 modes when the TARGET_MODE bit field of the ME_MCTL register is written with “1010”.

As soon as any of the above event has occurred, a STOP0 mode transition request is generated. The mode configuration information for this mode is provided by the ME_STOP0_MC register. This mode is fully configurable, and the ME_STOP0_MC register should be programmed according to the system needs. The following clock sources are switched off in this mode:

- the primary PLL
- the secondary PLL

The flash can be put in power-down mode as needed. If there is a STOP0 mode request while any interrupt or wakeup event is active, the transition to STOP0 is aborted with the resultant mode being the current mode, SAFE (on SAFE mode request), or DRUN (on reset), and an invalid mode interrupt is not generated.

This can be used as an advanced low-power mode with the core clock frozen and almost all peripherals stopped.

This mode is intended as an advanced low-power mode with

- the core clock frozen
- the additional core clocks frozen
- almost all peripherals stopped

and to be used by software

- to wait until it is required to do something with no need to react quickly (for example, allow for system clock source to be re-started).

The normal exit from the STOP0 mode is by an interrupt or a wakeup event.

This mode can be used to stop all clock sources and thus preserve the chip status. When exiting the STOP0 mode, the 16 MHz internal RC oscillator clock is selected as the system clock until the target clock is available.

Note: It is good practice for software to ensure that the S_MTRANS bit in the ME_GS register has been cleared on STOP0 mode exit to ensure that the previous RUN0–3 mode configuration has been fully restored before executing critical code.

69.4.2.8 STANDBY0 Mode

The chip enters this mode on the following event:

- from the DRUN or one of the RUN0–3 modes when the TARGET_MODE bit field of the ME_MCTL register is written with “1101”.

As soon as any of the above event occur, a STANDBY0 mode transition request is generated. The mode configuration information for this mode is provided by the ME_STANDBY0_MC register. In this mode, the power supply is turned off for most of the chip. The only parts of the chip that are still powered during this mode are pads mapped on wakeup lines and power domain #0 which contains the MC_RGM, MC_PCU, WKPU, PASS, OSC32k, RTC, LPROSC, SSWU, ADC-STDBY, PMC_DIG. The IRC can be optionally switched off. This is the lowest power consumption mode possible on the chip.

This mode is intended as an extreme low-power mode with

- the cores, the flashes, and almost all peripherals and memories powered down and to be used by software
- to wait until it is required to do something with no need to react quickly (that is to allow for system power-up and system clock source to be re-started)

The exit sequence of this mode is similar to the reset sequence, and the resets to all but power domain #0 are asserted with timing that is the same as reset PHASE1 through PHASE3. Device will boot from the address specified on ME_CADDR0 register. In case that boot from RAM is requested, ME_CADDR0 should address standby RAM (link to SRAM table). In this case, it is also possible to keep flash in power-down by writing "01" to FLAON field in the ME_DRUN_MC register..

If there is a STANDBY0 mode request while any wakeup event is active, the chip mode does not change.

69.4.3 Mode transition process

The process of mode transition follows the following steps in a pre-defined manner depending on the current chip mode and the requested target mode. In many cases of mode transition, not all steps need to be executed based on the mode control information, and some steps may not be applicable according to the mode definition itself.

Since the MC_ME ensures that all resources are in the proper state during the mode transition process, the following, for example, do not need to be checked by software.

- When the XOSC is enabled in a mode configuration register, a mode transition to that mode will not complete until the End Of Count timeout occurs for the oscillator. Software does not need to interrogate the End Of Count timeout status bit.
- When a PLL is enabled in a mode configuration register, a mode transition to that mode will not complete until the PLL is locked. Software does not need to interrogate the PLL's lock status bit.

69.4.3.1 Target mode request

The target mode is requested by accessing the ME_MCTL register with the required keys. This mode transition request by software must be a valid request satisfying a set of pre-defined rules to initiate the process. If the request fails to satisfy these rules, it is ignored, and the TARGET_MODE bit field is not updated. An optional interrupt can be generated for invalid mode requests. See *Section 69.4.5: Mode Transition Interrupts*, for details.

In the case of mode transitions occurring because of hardware events such as a reset, a SAFE mode request, or interrupt requests and wakeup events to exit from low-power modes, the TARGET_MODE bit field of the ME_MCTL register is automatically updated with the appropriate target mode. The mode change process start is indicated by the setting of the mode transition status bit S_MTRANS of the ME_GS register.

A RESET mode requested via the ME_MCTL register is passed to the MC_RGM, which generates a global system reset and initiates the reset sequence. The RESET mode request has the highest priority, and the MC_ME is kept in the RESET mode during the entire reset sequence.

The SAFE mode request has the next highest priority after reset. It can be generated either by software via the ME_MCTL register from all software running modes or by the MC_RGM after the detection of system hardware failures, which may occur in any mode.

69.4.3.2 Target mode configuration loading

On completion of the *Target mode request* step, the target mode configuration from the ME_<target mode>_MC register is loaded to start the resources (such as: voltage sources, clock sources, flash, pads) control process.

An overview of resource control possibilities for each mode is shown in *Table 1818*. A 'X' indicates that a given resource is configurable for a given mode.

Table 1818. MC_ME Resource Control Overview

Resource	Mode							
	RESET	TEST	SAFE	DRUN	RUN0–3	HALT0	STOP0	STANDBY0
IRC	on	X on	on	on	on	on	on	X on
XOSC	off	X off	off	X off	X off	X off	X off	off
PLL0	off	X off	off	X off	X off	X off	off	off

Table 1818. MC_ME Resource Control Overview (continued)

Resource	Mode							
	RESET	TEST	SAFE	DRUN	RUN0–3	HALT0	STOP0	STANDBY0
PLL1	off	X off	off	X off	X off	X off	off	off
FLASH	normal	X normal	normal	X normal	X normal	X low-power	X power-down	power-down
MVREG	on	on	on	on	on	on	on	off
PDO	off	X off	X on	off	off	off	X off	on

69.4.3.3 Peripheral clocks disable

On completion of the *Target mode request* step and the system clock frequency ramp-down of the *System clock switching* step, the MC_ME requests each peripheral to enter its stop mode when:

- the peripheral is configured to be disabled via the target mode, the peripheral configuration registers ME_RUN_PC0–7 and ME_LP_PC0–7, and the peripheral control registers ME_PCTL*n*

Note: The MC_ME automatically requests peripherals to enter their stop modes if the power domains in which they are residing are to be turned off due to a mode change. However, it is good practice for software to ensure that those peripherals that are to be powered down are configured in the MC_ME to be frozen.

Each peripheral acknowledges its stop mode request after closing its internal activity. The MC_ME then disables the corresponding clock(s) to this peripheral.

In the case of a SAFE mode transition request, the MC_ME does not wait for the peripherals to acknowledge the stop requests. The SAFE mode clock gating configuration is applied immediately regardless of the status of the peripherals' stop acknowledges.

Please see *Section 69.4.6: Peripheral Clock Gating*, for more details.

Each peripheral that may block or disrupt a communication bus to which it is connected ensures that these outputs are forced to a safe or recessive state when the chip enters the SAFE mode.

69.4.3.4 Processor low-power mode entry

If, on completion of the *Peripheral clocks disable* step, the mode transition is to the HALT0 mode, the MC_ME requests the processor to enter its halted state. The processor acknowledges its halt state request after completing all outstanding bus transactions.

If, on completion of the *Peripheral clocks disable* step, the mode transition is to the STOP0 or STANDBY0 mode, the MC_ME requests the processor to enter its stopped state. The processor acknowledges its stop state request after completing all outstanding bus transactions.

69.4.3.5 Processor clocks disable

If, on completion of the *Processor low-power mode entry* step, and all processors which are configured in the ME_CCTL*n* registers to be disabled are in their appropriate stopped state, the MC_ME disables the applicable processor clocks bits to achieve further power saving.

The clocks to processors which are configured to be running in both the current and target modes or disabled in both the current and target modes are unaffected while transitioning between non-low-power modes.

Caution: Clocks to the whole chip including the processor and system memory can be disabled in TEST mode.

69.4.3.6 System clock disable

If, on completion of the *Processor clocks disable* step, all processor clocks have been disabled, the system clock used for non-processor system functions and memory is disabled.

69.4.3.7 Clock sources (Main Voltage Regulator Independent) switch-on

On completion of the *Processor low-power mode entry* step, the MC_ME switches on all clock sources, which do not need the main voltage regulator to be on, based on the <clock source>ON bits of the ME_<current mode>_MC and ME_<target mode>_MC registers. The following clock sources are switched on at this step:

- the 16 MHz internal RC oscillator
- the external crystal oscillator
- the primary PLL
- the secondary PLL

Note: Clock sources which need the main voltage regulator to be stable are not controlled by this step.

The clock sources that are required by the target mode are switched on. The duration required for the output clocks to be stable depends on the type of source, and all further steps of mode transition depending on one or more of these clocks waits for the stable status of the respective clocks. The availability status of these clocks is updated in the S_<clock source> bits of ME_GS register.

The clock sources which need to be switched off are unaffected during this process in order to not disturb the system clock which might require one of these clocks before switching to a different target clock.

69.4.3.8 Main Voltage Regulator Switch-On

On completion of the *Target mode request* step, if the main voltage regulator needs to be switched on from its off state based on the MVRON bit of the ME_<current mode>_MC and ME_<target mode>_MC registers, the MC_ME requests the MC_PCU to power-up the regulator and waits for the output voltage stable status in order to update the S_MVR bit of the ME_GS register.

This step is required only during the exit of the low-power modes HALT0 and STOP0. In this step, the 16 MHz internal RC oscillator is switched on regardless of the target mode configuration, as the main voltage regulator requires the 16 MHz int. RC osc. during power-up in order to generate the voltage status.

During the STANDBY0 exit sequence, the MC_PCU alone manages the power-up of the main voltage regulator, and the MC_ME is kept in RESET or shut off (depending on the power domain #1 status).

69.4.3.9 Flash memory module switch-on

On completion of the *Main Voltage Regulator Switch-On* step, if the flash memory needs to be switched to normal mode from its low-power or power-down mode based on the FLAON bit field of the ME_<current mode>_MC and ME_<target mode>_MC registers, the MC_ME requests the flash memory to exit from its low-power/power-down mode. When the flash memory is available for access, the S_FLA bit field of the ME_GS register is updated to “11” by hardware.

Caution: It is illegal to switch the FLASH or the DFLASH from low-power mode to power-down mode and from power-down mode to low-power mode. The MC_ME, however, does not prevent this nor does it flag it.

69.4.3.10 Clock Sources (Main Voltage Regulator Dependent) Switch-On

On completion of the *Clock sources (Main Voltage Regulator Independent) switch-on* and *Main Voltage Regulator Switch-On*, the MC_ME controls all clock sources, which need the main voltage regulator to be on, based on the <clock source>ON bits of the ME_<current mode>_MC and ME_<target mode>_MC registers. The following clock sources are switched on at this step:

69.4.3.11 Pad outputs-on

On completion of the *Main Voltage Regulator Switch-On* step, if the PDO bit of the ME_<target mode>_MC register is cleared, then

- all pad outputs are enabled to return to their previous state
- slew rate control mechanism is switched on

69.4.3.12 Peripheral clocks enable

Based on the current and target chip modes, the peripheral configuration registers ME_RUN_PC0–7, ME_LP_PC0–7, and the peripheral control registers ME_PCTL*n*, the MC_ME enables the clocks for selected modules as required. If the value of the PWRLVL field of the ME_<target mode>_MC register is less than that of the ME_<current mode>_MC register’s, this step is executed only after the *System clock switching* process is completed.

69.4.3.13 System and processor clocks enable

On completion of the *Flash memory module switch-on* step, the MC_ME enables the non-processor system clock and the clocks of all processors which are configured in the ME_CCTL*n* registers to be disabled in the current mode and to be running in the target mode.

The MC_ME initiates the resetting of all processors which are configured in the ME_CCTL*n* registers to be running in the target mode and which have their RMC bits in the ME_CADDR*n* registers set to ‘1’. The core_rst_b outputs remain asserted until the end of the mode transition as indicated by the S_MTRANS bit of the MW_GS register at which time they are deasserted.

If the value of the PWRLVL field of the ME_<target mode>_MC register is different from that of the ME_<current mode>_MC register’s, the mode change handshake will not be initiated

until after the system clock frequency ramp-down step of the *System clock switching* process is completed.

69.4.3.14 Processor low-power mode exit

On completion of the *System and processor clocks enable* step, the MC_ME requests all processors which are configured in the ME_CCTL*n* registers to be disabled in the current mode and to be running in the target mode by deasserting the corresponding core_stop bits.

69.4.3.15 System clock switching

Based on the SYSCLK bit field of the ME_<current mode>_MC and ME_<target mode>_MC registers, if the target and current system clock configurations differ, the following method is implemented for clock switching.

- The target clock configuration for the 16 MHz int. RC osc. takes effect only after the S_IRC bit of the ME_GS register is set by hardware (that is the 16 MHz internal RC oscillator has stabilized).
- The target clock configuration for the external crystal osc. takes effect only after the S_IRC bit of the ME_GS register is set by hardware (that is the 16 MHz internal RC oscillator has stabilized).
- The target clock configuration for the primary PLL (PHI) takes effect only after the S_XOSC bit of the ME_GS register is set by hardware (that is the external crystal oscillator has stabilized).
- The target clock configuration for the secondary PLL takes effect only after the S_PLL0 bit of the ME_GS register is set by hardware (that is the primary PLL has stabilized).
- If the clock is to be disabled, the SYSCLK bit field should be programmed with “1111”. This is possible only in the TEST mode. In the STANDBY0 mode, the clock configuration is fixed, and the system clock is automatically forced to ‘0’. .

The current system clock configuration can be observed by reading the S_SYSCLK bit field of the ME_GS register, which is updated after every system clock switching. Until the target clock is available, the system uses the previous clock configuration.

System clock switching starts only after

- the *Clock sources (Main Voltage Regulator Independent) switch-on* process has completed if the target system clock source is one of the following:
  - the 16 MHz internal RC oscillator
  - the external crystal oscillator
  - the primary PLL
- the *Peripheral clocks disable* process has completed in order not to change the system clock frequency before peripherals close their internal activities
- the *Peripheral clocks enable* process has completed

An overview of system clock source selection possibilities for each mode is shown in *Table 1819*. A ‘X’ indicates that a given clock source is selectable for a given mode.

Table 1819. MC_ME System Clock Selection Overview

System Clock Source	Mode							
	RESET	TEST	SAFE	DRUN	RUN0–3	HALT0	STOP0	STANDBY0
16 MHz int. RC osc.		X (default)		X (default)	X (default)	X (default)		
external crystal osc.		X		X	X	X		
primary PLL (PHI)		X		X	X	X		
secondary PLL		X		X	X	X		
system clock is disabled		X(1)						

1. Disabling the system clock during TEST mode will require a reset in order to exit TEST mode

69.4.3.16 Pad switch-off

If the PDO bit of the ME_<target mode>_MC register is ‘1’ then

- the outputs of the pads are forced to the high impedance state if the target mode is SAFE or TEST

In STANDBY0 mode, the power sequence driver and all pads except the external reset and those mapped on wakeup lines are not powered and therefore high impedance. The wakeup line configuration remains unchanged.

This step is executed only after the *Peripheral clocks disable* process has completed.

69.4.3.17 Clock sources switch-off

Based on the chip mode and the <clock source>ON bits of the ME_<mode>_MC registers, if a given clock source is to be switched off, the MC_ME requests the clock source to power down and updates its availability status bit S_<clock source> of the ME_GS register to ‘0’. The following clock sources switched off at this step:

- the 16 MHz internal RC oscillator
- the external crystal oscillator
- the primary PLL
- the secondary PLL

For the clock sources related to the system clock, this step is executed only after the *System clock switching* process has completed.

69.4.3.18 Flash memory switch-off

Based on the FLAON bit field of the ME_<current mode>_MC and ME_<target mode>_MC registers, if the flash is to be put in its low-power or power-down mode, the MC_ME requests the flash to enter the corresponding power mode and waits for the flash to acknowledge. The exact power mode status of the flash is updated in the S_FLA bit field of the ME_GS register. This step is executed only when the *Processor clocks disable* and *System clock disable* processes have completed.

69.4.3.19 Main Voltage Regulator Switch-Off

Based on the MVRON bit of the ME_<current mode>_MC and ME_<target mode>_MC registers, if the main voltage regulator is to be switched off, the MC_ME requests it to power down and clears the availability status bit S_MVR of the ME_GS register.

This step is executed o nly after completing the following processes :

- *Clock sources switch-off*
- *Flash memory switch-off*

If the target mode is STANDBY0, the main voltage regulator is not switched off by the MC_ME and the STANDBY0 request is asserted after the above processes have completed upon which the MC_PCU takes control of the main regulator. As the MC_PCU needs the 16 MHz int. RC osc., the 16 MHz internal RC oscillator remains active until all the STANDBY0 steps are executed by the MC_PCU after which it may be switched off depending on the IRCON bit of the ME_STANDBY0_MC register.

69.4.3.20 Current mode update

The current mode status bit field S_CURRENT_MODE of the ME_GS register is updated with the target mode bit field TARGET_MODE of the ME_MCTL register when :

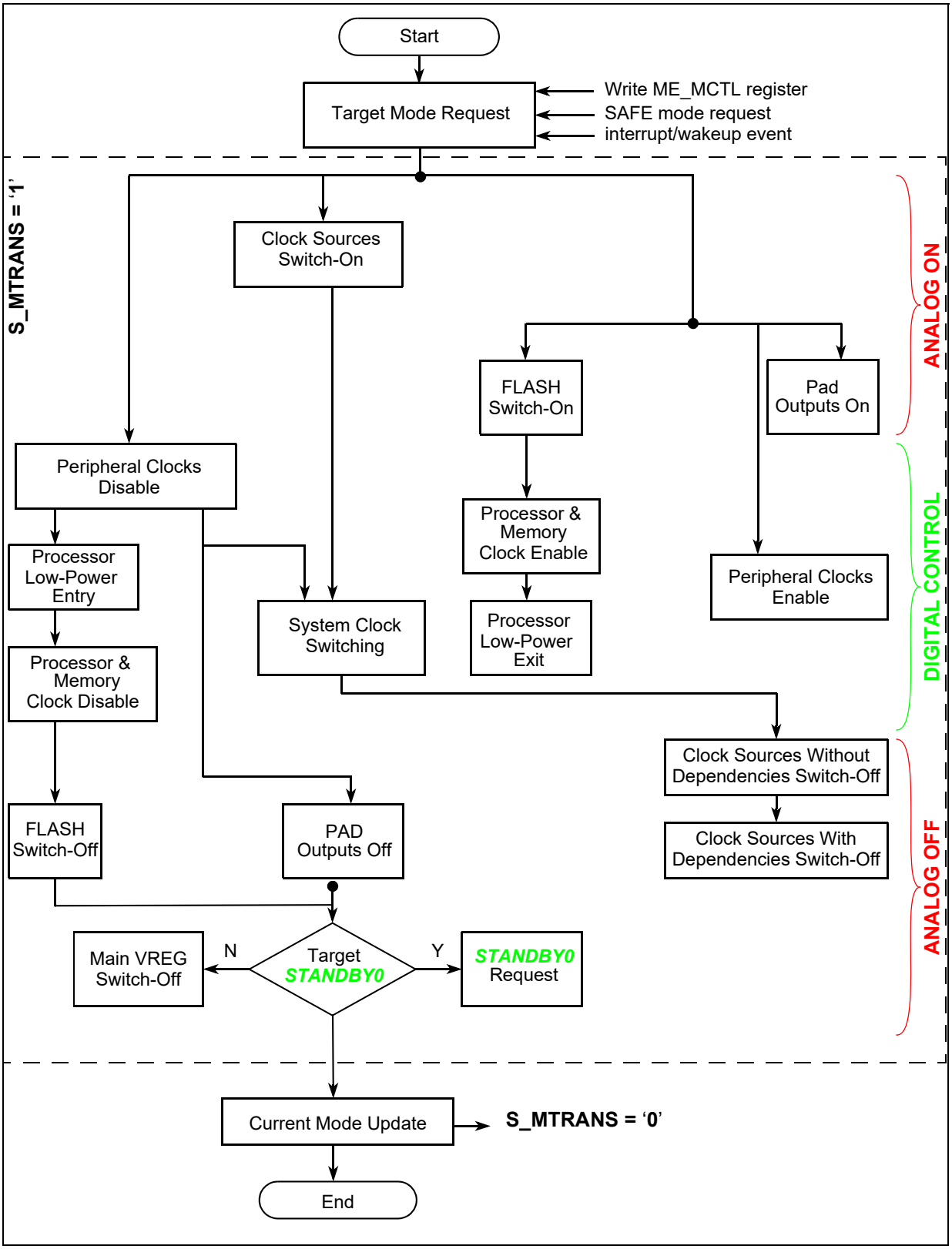
- all the updated status bits in the ME_GS register match the configuration specified in the ME_<target mode>_MC register
- power sequences are done
- clock disable/enable process is finished
- processor low-power mode (stop) entry and exit processes are finished

Note: SAFE mode entry does not wait for the clock disable/enable process to finish. It only waits for the ME_GS.S_RC bit to be set. This is to ensure that the SAFE mode is entered as quickly as possible.

Software can monitor the mode transition status by reading the S_MTRANS bit of the ME_GS register. The mode transition latency can differ from one mode to another depending on the resources' availability before the new mode request and the target mode's requirements.

If a mode transition is taking longer to complete than is expected, the ME_DMTS register can indicate which process is still in progress.

Figure 1888. MC_ME Transition Diagram



69.4.4 Protection of Mode Configuration Registers

While programming the mode configuration registers ME_<mode>_MC, the following rules must be respected. Otherwise, the write operation is ignored and an invalid mode configuration interrupt may be generated.

- If the 16 MHz int. RC osc. is selected as the system clock, IRC must be on.
- If the external crystal osc. clock is selected as the system clock, XOSC must be on.
- If the primary PLL (PHI) clock is selected as the system clock, PLL0 must be on.
- If the secondary PLL clock is selected as the system clock, PLL1 must be on.

Note: Software must ensure that clock sources with dependencies other than those mentioned above are switched on as needed. There is no automatic protection mechanism to check this in the MC_ME.

- Configuration “00” for the FLAON bit field is reserved.
- System clock configurations marked as ‘reserved’ may not be selected.
- Configuration “1111” for the SYSCLK bit field is allowed for the TEST mode and STANDBY mode, and only in this case may all system clock sources be turned off.

Caution: If the system clock is stopped during TEST mode, the chip can exit only via a system reset.

69.4.5 Mode Transition Interrupts

The MC_ME provides interrupts for incorrectly configuring a mode, requesting an invalid mode transition, indicating a SAFE mode transition not due to a software request, and indicating when a mode transition has completed.

69.4.5.1 Invalid Mode Configuration Interrupt

Whenever a write operation is attempted to the ME_<mode>_MC registers violating the protection rules mentioned in the *Section 69.4.4: Protection of Mode Configuration Registers*, the interrupt pending bit I_ICONF of the ME_IS register is set and an interrupt request is generated if the mask bit M_ICONF of the ME_IM register is ‘1’.

In addition, during a mode transition, if a clock source has been configured in the ME_<target mode>_MC register to be off and a peripheral requiring this clock source to be on has been enabled via the ME_RUN_PC0–7/ME_LP_PC0–7 and ME_PCTL*n* registers, the interrupt pending bit I_ICONF_CU of the ME_IS register is set and an interrupt request is generated if the mask bit M_ICONF_CU of the ME_IM register is ‘1’.

If an attempt to write to one of the ME_CADDR*n* registers is made after a mode change request has been made via the second write to the ME_CTL register and before the completion of that mode transition as indicated by the S_MTRANS bit in the ME_GS register deasserting, the interrupt pending bit I_ICONF_CC of the ME_IS register is set and an interrupt request is generated if the mask bit M_ICONF_CC of the ME_IM register is ‘1’.

69.4.5.2 Invalid Mode Transition Interrupt

The mode transition request is considered invalid under the following conditions:

- If the system is in the SAFE mode and the SAFE mode request from MC_RGM is active, and if the target mode requested is other than RESET or SAFE, then this new

mode request is considered to be invalid, and the S_SEA bit of the ME_IMTS register is set.

- If the TARGET_MODE bit field of the ME_MCTL register is written with a value different from the specified mode values (that is a non-existing mode), an invalid mode transition event is generated. When such a non existing mode is requested, the S_NMA bit of the ME_IMTS register is set. This condition is detected regardless of whether the proper key mechanism is followed while writing the ME_MCTL register.
- If some of the chip modes are disabled as programmed in the ME_ME register, their respective configurations are considered reserved, and any access to the ME_MCTL register with those values results in an invalid mode transition request. When such a disabled mode is requested, the S_DMA bit of the ME_IMTS register is set. This condition is detected regardless of whether the proper key mechanism is followed while writing the ME_MCTL register.
- If the target mode is not a valid mode with respect to the current mode, the mode request illegal status bit S_MRI of the ME_IMTS register is set. This condition is detected only when the proper key mechanism is followed while writing the ME_MCTL register. Otherwise, the write operation is ignored.
- If further new mode requests occur while a mode transition to a mode other than SAFE mode is in progress (the S_MTRANS bit of the ME_GS register is ‘1’), the mode transition illegal status bit S_MTI of the ME_IMTS register is set. This condition is detected only when the proper key mechanism is followed while writing the ME_MCTL register. Otherwise, the write operation is ignored.
- If a new mode request occurs while a mode transition to the SAFE mode is in progress (the S_MTRANS bit of the ME_GS register is ‘1’), the mode request ignored status bit S_MRIG of the ME_IMTS register is set. This condition is detected only when the proper key mechanism is followed while writing the ME_MCTL register. Otherwise, t he write operation is ignored.
- If a HALT0 mode request occurs while an interrupt request is active, the mode request ignored status bit S_MRIG of the ME_IMTS register is set. This condition is detected only when the proper key mechanism is followed while writing the ME_MCTL register. Otherwise, t he write operation is ignored.
- If a STOP0 mode request occurs while a wakeup or interrupt request is active, the mode request ignored status bit S_MRIG of the ME_IMTS register is set. This condition is detected only when the proper key mechanism is followed while writing the ME_MCTL register. Otherwise, t he write operation is ignored.

Note: As the causes of invalid mode transitions may overlap at the same time, the priority implemented for invalid mode transition status bits of the ME_IMTS register in the order from highest to lowest is S_SEA, S_NMA, S_DMA, S_MRI, and S_MTI.

As an exception, the mode transition request is not considered as invalid under the following conditions:

- A new request is allowed to enter the RESET or SAFE mode irrespective of the mode transition status.
- As the exit of HALT0 and STOP0 modes depends on the interrupts of the system which can occur at any instant, these requests to return to RUN0–3 modes are always valid.
- In order to avoid any unwanted lockup of the chip modes, software can abort a mode transition by requesting the parent mode if, for example, the mode transition has not completed after a software determined ‘reasonable’ amount of time for whatever reason. The parent mode is the chip mode before a valid mode request was made.

Warning: Depending on the cause of the mode transition not completing, aborting via a request back to the parent mode may not be able to recover. In this case, software should request a change to either SAFE or RESET mode, since the cause of the hang-up is most likely due to a hardware fault.

- Self-transition requests (for example, RUN0 → RUN0) are not considered as invalid even when the mode transition process is active (that is S_MTRANS is '1'). During the low-power mode exit process, if the system is not able to enter the respective RUN0–3 mode properly (that is all status bits of the ME_GS register match with configuration bits in the ME_<mode>_MC register), then software can only request the SAFE or RESET mode. It is not possible to request any other mode or to go back to the low-power mode again.

Whenever an invalid mode request is detected, the interrupt pending bit I_IMODE of the ME_IS register is set, and an interrupt request is generated if the mask bit M_IMODE of the ME_IM register is '1'.

69.4.5.3 SAFE Mode Transition Interrupt

Whenever the system enters the SAFE mode as a result of a SAFE mode request from the MC_RGM due to a hardware failure, the interrupt pending bit I_SAFE of the ME_IS register is set, and an interrupt is generated if the mask bit M_SAFE of ME_IM register is '1' .

The SAFE mode interrupt pending bit can be cleared only when the SAFE mode request is deasserted by the MC_RGM (see the MC_RGM chapter for details on how to clear a SAFE mode request). If the system is already in SAFE mode, any new SAFE mode request by the MC_RGM also sets the interrupt pending bit I_SAFE. However, the SAFE mode interrupt pending bit is not set when the SAFE mode is entered by a software request (by programming the ME_MCTL register).

69.4.5.4 Mode Transition Complete interrupt

Whenever the system fully completes a mode transition (that is the S_MTRANS bit of ME_GS register transits from '1' to '0'), the interrupt pending bit I_MTC of the ME_IS register is set, and an interrupt request is generated if the mask bit M_MTC of the ME_IM register is '1'. The interrupt bit I_MTC is not set when entering low-power modes HALT0 and STOP0 in order to avoid the same event requesting the immediate exit of these low-power modes.

69.4.6 Peripheral Clock Gating

During all chip modes, each peripheral can be associated with a particular clock gating policy determined by two groups of peripheral configuration registers.

The run peripheral configuration registers ME_RUN_PC0–7 are chosen only during the software running modes DRUN, TEST, SAFE, and RUN0–3. All configurations are programmable by software according to the needs of the application. Each configuration register contains a mode bit which determines whether or not a peripheral clock is to be gated. Run configuration selection for each peripheral is done by the RUN_CFG bit field of the ME_PCTL*n* registers.

The low-power peripheral configuration registers ME_LP_PC0–7 are chosen only during the low-power modes HALT0, STOP0, and STANDBY0. All configurations are programmable by software according to the needs of the application. Each configuration register contains a mode bit which determines whether or not a peripheral clock is to be gated. Low-power configuration selection for each peripheral is done by the LP_CFG bit field of the ME_PCTL*n* registers.

Any modifications to the ME_RUN_PC0–7, ME_LP_PC0–7, and ME_PCTL*n* registers do not affect the clock gating behavior until a new mode transition request is generated.

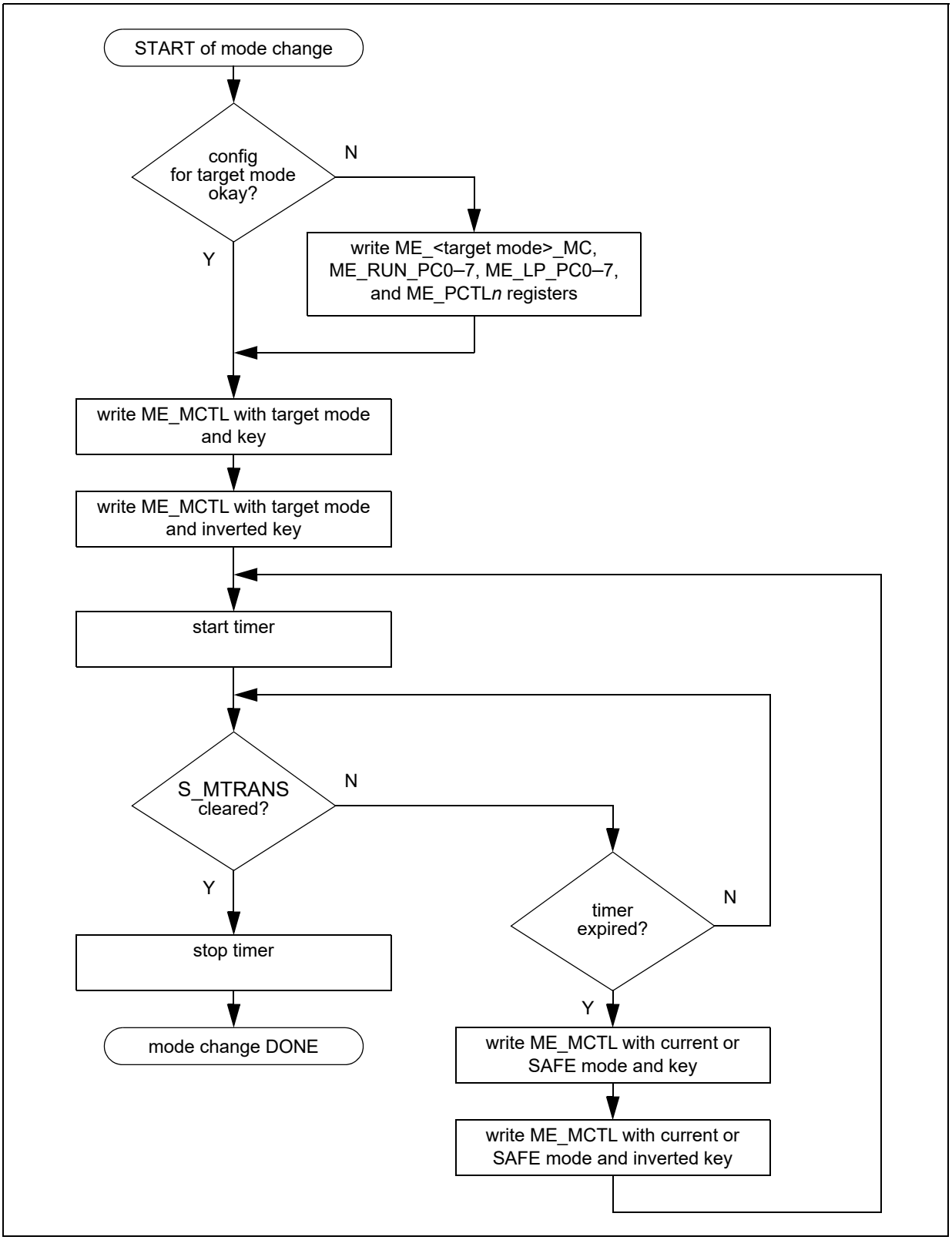
Whenever the processor enters a debug session during any mode, the following occurs for each peripheral:

- The clock is gated if the DBG_F bit of the associated ME_PCTL*n* register is set. Otherwise, the peripheral clock gating status depends on the RUN_CFG and LP_CFG bits. Any further modifications of the ME_RUN_PC0–7, ME_LP_PC0–7, and ME_PCTL*n* registers during a debug session will take affect immediately without requiring any new mode request.

69.4.7 Application Example

Figure 1889 shows an example application flow for requesting a mode change and then waiting until the mode transition has completed.

Figure 1889. MC_ME Application Example Flow Diagram



70 e200z425n3 Core Debug Support

This chapter describes the debug features of the e200z425n3 core.

70.1 Overview

Internal debug support in the e200z425n3 core allows for software and hardware debug by providing debug functions, such as instruction and data breakpoints and program trace modes. For software based debugging, debug facilities consisting of a set of software accessible debug registers and interrupt mechanisms are provided. These facilities are also available to a hardware-based debugger, which communicates using a modified IEEE 1149.1 Test Access Port (TAP) controller and pin interface. When hardware debug is enabled, the debug facilities controlled by hardware are protected from software modification.

Software debug facilities are defined as part of PowerISA 2.06. e200z425n3 supports a subset of these defined facilities. In addition to the facilities defined in PowerISA 2.06, e200z425n3 provides additional flexibility and functionality in the form of additional instruction breakpoints, linked instruction and data breakpoints, and extended range and value masking. These features are also available to a hardware-based debugger.

When not being used for debugging purposes, a portion of the debug facilities may be configured to provide a stack limit checking mechanism.

70.1.1 Software debug facilities

e200z425n3 provides debug facilities to enable hardware and software debug functions, such as instruction and data breakpoints and program single stepping. The debug facilities consist of a set of debug control registers (DBCR0–2,4–8, EDBRAC0), a set of address compare registers (IAC1–8, DAC1–4), a set of 64-bit data value compare registers (DVC1, DVC2), a Debug Status Register (DBSR) for enabling and recording various kinds of debug events, a Debug Data Effective Address Register (DDEAR), and a special Debug interrupt type built into the interrupt mechanism.

The debug facilities also provide a mechanism for software-controlled processor reset, and debug mode entry. In addition, the Performance Monitor may be configured to utilize the debug interrupt type. Also, the Memory Protection Unit (MPU) may be configured to generate debug events using region descriptors that are not utilized for performing a protection function.

Software debug facilities are enabled by setting the internal debug mode bit in Debug Control register 0 ($DBCR0_{IDM}$). When internal debug mode is enabled, debug events can occur, and can be enabled to record exceptions in the Debug Status register (DBSR). If enabled by MSR_{DE}, these recorded exceptions cause Debug interrupts to occur. When $DBCR0_{IDM}$ is cleared (and $EDBCR0_{EDM}$ is cleared as well), no debug events occur, and no status flags are set in DBSR unless already set. In addition, when $DBCR0_{IDM}$ is cleared (or is overridden by $EDBCR0_{EDM}$ being set and EDBRAC0 indicating no resource is “owned” by software) no Debug interrupts will occur, regardless of the contents of DBSR. A software Debug interrupt handler may access all system resources and perform necessary functions appropriate for system debug.

70.1.1.1 PowerISA 2.06 compatibility

The e200z425n3 core implements a subset of the PowerISA 2.06 internal debug features. The following restrictions on functionality are present:

- Instruction address compares do not support compare on physical (real) addresses.
- Data address compares do not support compare on physical (real) addresses.

70.1.2 Additional debug facilities

In addition to the debug functionality defined in PowerISA 2.06, e200z425n3 provides capability to link instruction and data breakpoints, provides additional instruction and data breakpoints, extended range and value masking, multiple watchpoint outputs, provides capability for the Performance Monitor to generate debug events, and allows for sharing of debug resources between software and a hardware debugger. (See *Section 70.1.4: Sharing debug resources by software/hardware*.) A data trace port is also provided.

70.1.2.1 Data trace port

The data trace port interface is provided to assist in implementing extended debug watchpoint/breakpoint/trace capture capability with logic external to the e200z425n3 core. This port provides information corresponding to each read or write access completed without error by the CPU. In order to report data accesses, the Nexus 3 DTC register DI control bit must be set to trace data accesses, not instruction accesses (that is the normal setting). See the “Data Trace Control Register (DTC)” section in the Core (e200z425n3) Nexus 3 Module chapter. The data trace port will report aligned accesses, and most misaligned accesses as one access, unless the two portions of a misaligned access are non-contiguous, such as when the second portion of the misaligned access is to address 0. Also, for accesses generated by a LSP load/store instruction using circular addressing mode that crosses a buffer boundary, the two portions of the misaligned access will be reported separately, since otherwise it is not possible to correctly correlate the data with the initial access address.

70.1.3 Hardware debug facilities

The e200z425n3 core contains facilities that allow for external test and debugging. A modified IEEE 1149.1 control interface is used to communicate with the core resources. This interface is implemented through a standard 1149.1 TAP (test access port) controller.

By using public instructions, the external debugger can freeze or halt the e200z425n3 core, read and write internal state and debug facilities, single-step instructions, and resume normal execution.

Hardware Debug is enabled by setting the External Debug mode enable bit in External Debug Control register 0 ($EDBCR0_{EDM}$), which is also aliased to $DBCR0_{EDM}$. Setting $EDBCR0_{EDM}$ overrides the Internal Debug mode enable bit $DBCR0_{IDM}$ unless resources are provided back to software via the settings in EDBRAC0. When the Hardware Debug facility is enabled, software is blocked from modifying the “hardware-owned” debug facilities. In addition, since resources are “owned” by the hardware debugger, inconsistent values may be present if software attempts to read “hardware-owned” debug-related resources.

When hardware debug is enabled by setting $EDBCR0_{EDM}$=1, the control registers and resources described in *Section 70.3: Debug registers* are reserved for use by the external debugger. The same events described in *Section 70.2: Software debug events and*

exceptions are also used for external debugging, but exceptions are not generated to running software. Hardware-owned debug events enabled in the respective DBCR0–8 registers are recorded in the EDBSR0 register (not the DBSR) regardless of MSR_{DE}, and no debug interrupts are generated unless a) the resource is granted back to software via EDBRAC0 settings, and b) debug mode entry is not masked by the corresponding event bit in EDBSRMSK0. Instead, the CPU will enter debug mode when an enabled event causes a EDBSR0 bit to become set. $EDBCR0_{EDM}$, EDBSR0, EDBSRMSK0, EDDEAR, and EDBRAC0 may only be written through the OnCE port.

Access to most debug resources (registers) requires that the CPU clock (**m_clk**) be running in order to perform write accesses from the external hardware debugger.

70.1.4 Sharing debug resources by software/hardware

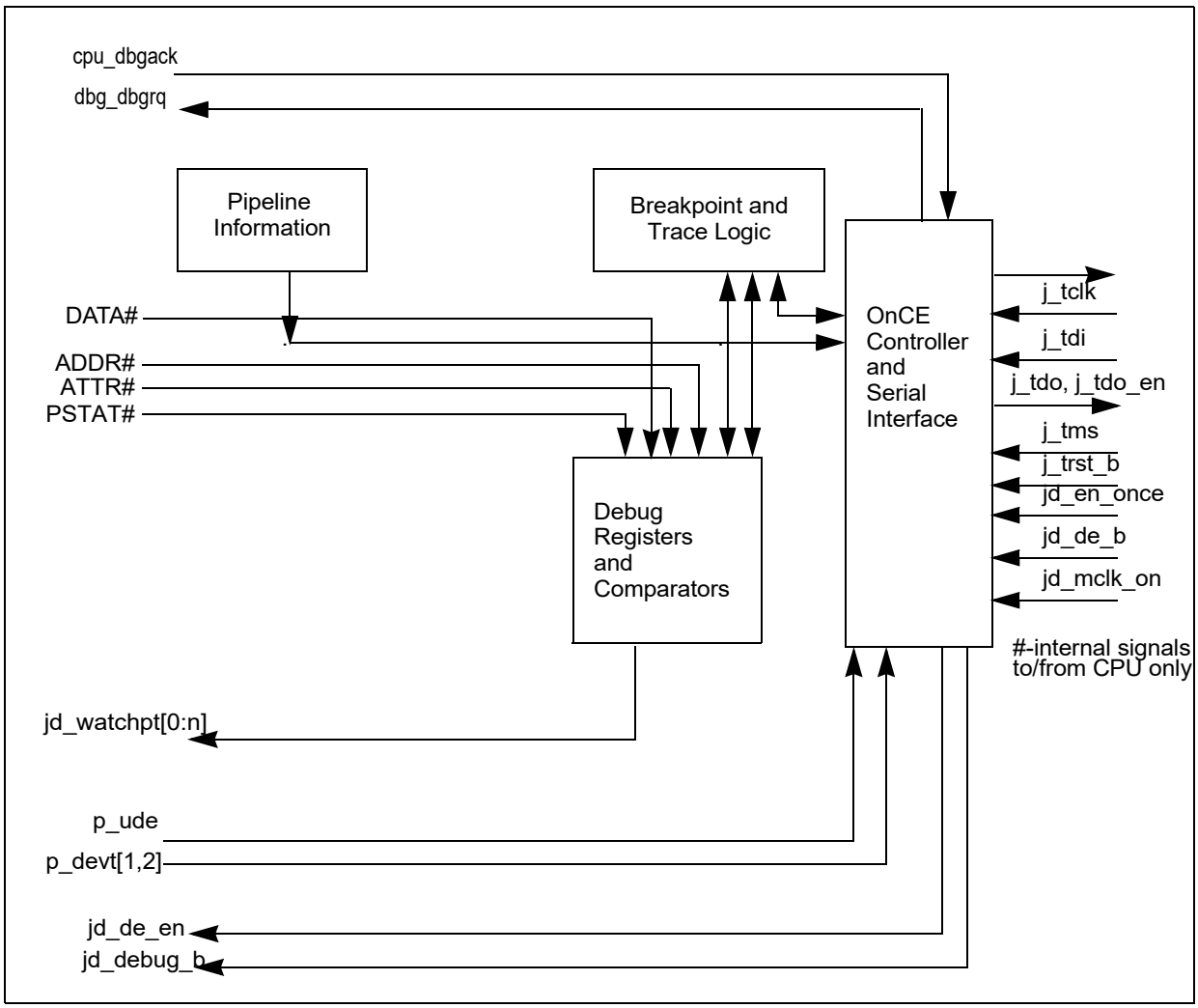
Debug resources may be shared by a hardware debugger and software debug based on the settings of debug control register EDBRAC0. When $EDBCR0_{EDM}$ is set, EDBRAC0 settings determine which debug resources are allocated to software and which resources remain under exclusive hardware control. Software-owned resources that set DBSR bits when $DBCR0_{IDM}$=1 will cause a debug interrupt to occur when enabled with MSR_{DE}. Hardware-owned resources that set EDBSR0 bits when $EDBCR0_{EDM}$=1 will cause an entry into debug mode if the event is not masked in EDBSRMSK0. EDBRAC0 is read-only by software. When resource sharing is enabled, ($EDBCR0_{EDM}$=1 and $EDBRAC0_{IDM}$=1), only software-owned resources may be modified by software. Hardware always has full access to all registers and all register fields through the OnCE register access mechanism, and it is up to the debug firmware to properly implement modifications to these registers with read-modify-write operations to implement any control sharing with software. Hardware-owned resources will set status bits in the EDBSR0 register instead of in DBSR, and will report the effective address of data breakpoints in EDDEAR instead of DDEAR. Settings in EDBRAC0 should be considered by the debug firmware in order to preserve software settings of control and status registers as appropriate when hardware modifications to the debug registers are performed.

70.1.4.1 Simultaneous hardware and software debug event handing

Since it is possible that a “hardware-owned” resource can produce a debug event in conjunction with a software-owned resource producing a different debug event simultaneously, a priority ordering mechanism is implemented that guarantees that the hardware event is handled as soon as possible, while preserving the recognition of the software event. The CPU will give highest priority to the software event initially in order to reach a recoverable boundary, and then will give highest priority to the hardware event in order to enter debug mode as near the point of event occurrence as possible. This is implemented by allowing software exception handing to begin internal to the CPU and to reach the point where the current program counter and MSR values have been saved into DSRR0/1, and the new PC pointing to the debug interrupt handler, along with the new MSR updates. At this point, hardware priority takes over, and the CPU enters debug mode.

Figure 1890 shows the e200z425n3 debug resources.

Figure 1890. e200z425n3 debug resources



70.2 Software debug events and exceptions

Software debug events and exceptions are available when internal debug mode is enabled ($DBCR0_{IDM}$=1) and not overridden by external debug mode ($EDBCR0_{EDM}$ must either be cleared or corresponding resources must be allocated to software debug by the settings in EDBRAC0). When enabled, debug events cause debug exceptions to be recorded in the Debug Status Register. Specific event types are enabled by the Debug Control Registers (DBCR0–8). The Unconditional Debug Event (UDE) is an exception to this rule; it is always enabled. Once a Debug Status Register (DBSR) bit is set by a debug resource that is “owned” by software (other than MRR, DAC_OFST, or VLES), if Debug interrupts are enabled by MSR_{DE}, a Debug interrupt will be generated. The debug interrupt handler is responsible for ensuring that multiple repeated debug interrupts do not occur by clearing the DBSR as appropriate.

Certain debug events are not allowed to occur when MSR_{DE}=0 and $DBCR0_{IDM}$=1. In such situations, no debug exception occurs and thus no DBSR bit is set. Other debug events may

cause debug exceptions and set DBSR bits regardless of the state of MSR_{DE}. A Debug interrupt will be delayed until MSR_{DE} is later set to '1'.

When a Debug Status Register bit is set while MSR_{DE}=0, an Imprecise Debug Event flag ($DBSR_{IDE}$) will also be set to indicate that an exception bit in the Debug Status Register was set while Debug interrupts were disabled. Debug interrupt handler software can use this bit to determine whether the address recorded in Debug Save/Restore Register 0 is an address associated with the instruction causing the debug exception, or the address of the instruction that enabled a delayed Debug interrupt by setting the MSR_{DE} bit. A **mtmsr** or **mtdbcr0** that causes both MSR_{DE} and $DBCR0_{IDM}$ to become set, enabling precise debug mode, may cause an Imprecise (Delayed) Debug exception to be generated due to an earlier recorded event in the Debug Status register.

There are eight types of debug events defined by PowerISA 2.06:

1. Instruction Address Compare debug events
2. Data Address Compare debug events
3. Trap debug events
4. Branch Taken debug events
5. Instruction Complete debug events
6. Interrupt/Critical Interrupt Taken debug events
7. Return/Critical Return debug events
8. Unconditional debug events

In addition, e200z425n3 defines additional debug events:

- The External debug events DEVT1 and DEVT2, which are described in *Section 70.2.11: External debug event*.
- The Performance Monitor Interrupt event PMI, which is described in *Section 70.2.13: Performance Monitor Interrupt debug event*.

The e200z425n3 debug configuration supports most of these event types. Unsupported PowerISA 2.06 defined functionality is as follows:

- Instruction Address Compare and Data Address Compare *Real address* mode is not supported

A brief description of each of the event types follows.

70.2.1 Instruction Address Compare event

Instruction Address Compare debug events occur when enabled and execution is attempted of an instruction at an address that meets the criteria specified in the DBCR0, DBCR1, DBCR5, DBCR6, and IAC1–8 Registers. Instruction Address compares may specify user/supervisor mode, along with an effective address, masked effective address, or range of effective addresses for comparison. This event can occur and be recorded in DBSR regardless of the setting of MSR_{DE}. IAC events will not occur when an instruction would not have normally begun execution due to a higher priority exception at an instruction boundary.

IAC compares perform a 31-bit compare for VLE instruction storage accesses. Each halfword fetched by the instruction fetch unit will be marked with a set of bits indicating whether an Instruction Address Compare occurred on that halfword. Debug exceptions will occur if enabled and a 16-bit instruction, or the first halfword of a 32-bit instruction, is tagged with an IAC hit.

70.2.2 Data Address Compare event

Data Address Compare debug events occur when enabled and execution of a load or store class instruction results in a data access that meets the criteria specified in the DBCR0, DBCR2, DBCR4, DBCR7–8, DAC1–4, DVC1, and DVC2 Registers. Data address compares may specify user/supervisor mode, along with an effective address, masked effective address, or range of effective addresses for comparison. This event can occur and be recorded in DBSR regardless of the setting of MSR_{DE}. Four address compare values (DAC1–4) are provided.

The effective address of the load or store operation is recorded in the DDEAR register when a data address compare event is recorded in DBSR if the previous values of the $DBSR_{DAC\{R,W\}}$ bits are zero. Once a DDEAR update is performed, these bits must be cleared by software prior to another DDEAR hardware update occurring. A subsequent DAC event will not update the DDEAR register if either of the $DBSR_{DAC\{R,W\}}$ bits are set.

Note: In contrast to the PowerISA 2.06 definition, Data Address Compare events on e200z425n3 do not prevent the load or store class instruction from completing. If a load or store class instruction completes successfully without a Data Storage interrupt, Data Address Compare exceptions are reported at the completion of the instruction. If the exception results in a precise Debug interrupt, the address value saved in DSRR0 is the address of the instruction following the load or store class instruction. For DVC DAC events, the exception can be imprecisely reported even further past the load or store class instruction generating the event (without necessarily affecting $DBSR_{IDE}$) and the saved address value can point to a subsequent instruction past the next instruction. This occurrence is indicated in the $DBSR_{DAC_OFST}$ field.

If a load or store class instruction does not complete successfully due to a Data Storage exception or a machine check condition for the load or store, and a Data Address Compare debug exception also occurs, the result is an imprecise Debug interrupt, the address value saved in DSRR0 is the address of the load or store class instruction, and the $DBSR_{IDE}$ bit will be set. In addition to occurring when $DBCR0_{IDM}$=1, this circumstance can also occur when $EDBCR0_{EDM}$=1 and the event is hardware-owned, in which case $EDBSR0_{IDE}$ will be set.

DAC events will not be recorded if a load multiple word or store multiple word instruction is interrupted prior to completion by a critical input or external input interrupt.

DAC events are not signaled on the second portion of a misaligned load or store that is broken up into two separate accesses.

*DAC events are not signaled on the **mpure** or **mpuwe** MPU instructions.*

*DAC[1,2] events are not signaled if DVC[1,2]M is non-zero and a DSI exception occurs on the load or store, since the load or store access is not performed. For a **lmw** or **stmw** transfer however, if a DVC successfully occurs on a transfer and a later transfer encounters a DSI exception, the DAC event will be reported, since a successful data value compare took place.*

*Due to internal pipeline status, for a reported DAC event on a **lmw** or **stmw** transfer, the value stored in the DDEAR/EDDEAR will be the effective address of the first transfer of the sequence, and not necessarily the precise transfer that caused the DAC event.*

70.2.2.1 Data Address Compare event status updates

Data Address Compare debug events with Data Value compares can be reported ambiguously in several circumstances involving issuing a sequence of load or store class instructions. Due to the CPU pipeline and the delay in performing the data value compare

following completion of the access, if the first load or store class instruction generates a DVC DAC, a second and possibly third load or store class instruction may also generate a DAC or DVC DAC event, or may generate a DSI exception with or without a simultaneous DAC event.

Also, since non-load/store instructions may be dual-issued in combination with a load/store instruction, the actual number of additional instructions that are completed following a recognized DVC DAC on a load/store instruction may vary from 0 to 5. This value will be reported in the $DBSR_{DAC_OFST}$ field when the DVC DAC status is recorded.

Table 1820 outlines the settings of the DBSR, DSRR0 saved value, and potential updating of the ESR register for various exception cases on sequences of load/store class instructions. Not all exception combinations are covered in the table, such as IAC, ISI, or Alignment exceptions on subsequent instructions. In general these exceptions will cause further instruction issue to be halted, execution of the excepting instruction to be aborted, and reporting of these exceptions will be masked. The saved DSRR0 value will point to this excepting instruction, and the exception(s) may be regenerated after returning from the debug interrupt handler and attempting to re-execute the instruction pointed to by DSRR0. In addition, in the examples in *Table 1820*, the DAC_OFST and DSRR0 values assume no dual issue occurs. If dual-issue occurs with the first, second, or third column, then the DAC_OFST and DSRR0 values will point beyond the values shown.

Table 1820. DAC events and resultant updates

1st load/store class instruction	2nd instruction (load/store class unless otherwise specified)	3rd instruction (load/store class unless otherwise specified)	Result
DSI, no DAC	—	—	Take DSI exception, no DBSR update. Update ESR.
DSI, with DACx	—	—	Take Debug exception, DBSR update setting DACx and IDE, DAC_OFST not set. DSRR0 points to 1st load/store class instruction. No ESR update.
DACx	—	—	Take Debug exception, DBSR update setting DACx, DAC_OFST not set. DSRR0 points to 2nd load/store class instruction. No ESR update.
DVC DACx	No exceptions, any instruction	No exceptions, Non-ldst instruction	Take Debug exception, DBSR update setting DACx, DAC_OFST set to 3'b001. DSRR0 points to 3rd instruction. No ESR update.
DVC DACx	No exceptions	No exceptions, Ldst instruction	Take Debug exception, DBSR update setting DACx, DAC_OFST set to 3'b010. DSRR0 points to instruction after 3rd instruction. No ESR update.
DVC DACx	DSI, no DAC	—	Take Debug exception, DBSR update setting DACx, DAC_OFST not set. DSRR0 points to 2nd load/store class instruction. No ESR update. **Note:** in this case the 2nd ld/st exception is masked. This behavior is implementation dependent and may differ on other CPUs.

Table 1820. DAC events and resultant updates (continued)

1st load/store class instruction	2nd instruction (load/store class unless otherwise specified)	3rd instruction (load/store class unless otherwise specified)	Result
DVC DACx	DSI, with DACy	—	Take Debug exception, DBSR update setting DACx. DAC_OFST not set. DSRR0 points to 2nd load/store class instruction. No ESR update. **Note:** in this case the 2nd ld/st exception is masked. This behavior is implementation dependent and may differ on other CPUs.
DVC DACx	DACy	—	Take Debug exception, DBSR update setting DACx, DACy. DAC_OFST set to 3'b001. DSRR0 points to 3rd instruction. **Note:** in this case if x==y, then the resultant state of DBSR and DSRR0 may be indistinguishable from the "no DACy" case.
DVC DACx	DVC DACy, Normal Ldst	Non-Ldst instruction	Take Debug exception, DBSR update setting DACx, DACy. DAC_OFST set to 3'b001. DSRR0 points to the 3rd instruction. **Note:** in this case if x==y, then the resultant state of DBSR and DSRR0 may be indistinguishable from the "no DACy" case.
DVC DACx	DVC DACy, Normal Ldst	Ldst instruction, no exception	Take Debug exception, DBSR update setting DACx, DACy. DAC_OFST set to 3'b010. DSRR0 points to instruction after the 3rd load/store class instruction. **Note:** in this case if x==y, then the resultant state of DBSR and DSRR0 may be indistinguishable from the "no DACy" case.
DVC DACx	DVC DACy, Normal Ldst	DSI Error, with or without DAC	Take Debug exception, DBSR update setting DACx, DACy. DAC_OFST set to 3'b001. No ESR update. DSRR0 points to 3rd instruction. **Note:** in this case if x==y, then the resultant state of DBSR and DSRR0 may be indistinguishable from the "no DACy" case. **Note:** in this case the 3rd ld/st exception is masked. This behavior is implementation dependent and may differ on other CPUs.
DVC DACx	DVC DACy, Normal Ldst	DACy, or DVC DAC_y Normal Ldst or multiple word Ldst	Take Debug exception, DBSR update setting DACx, DACy. DAC_OFST set to 3'b010. DSRR0 points to instruction after the 3rd load/store class instruction. **Note:** in this case if x==y, then the resultant state of DBSR and DSRR0 may be indistinguishable from the "no DACy" case.
DVC DACx	DVC DACy, Ldst multiple (lmw, stmw)	Any instruction including ld/st	Take Debug exception, DBSR update setting DACx, DACy. DAC_OFST set to 3'b001. DSRR0 points to the 3rd instruction. **Note:** in this case if x==y, then the resultant state of DBSR and DSRR0 may be indistinguishable from the "no DACy" case.

Table 1820. DAC events and resultant updates (continued)

1st load/store class instruction	2nd instruction (load/store class unless otherwise specified)	3rd instruction (load/store class unless otherwise specified)	Result
DVC DACx	Any instruction (no exception)	DSI, with or without DAC, Normal Ldst or multiple word Ldst	Take Debug exception, DBSR update setting DACx. DAC_OFST set to 3'b001. DSRR0 points to the 3rd instruction. No ESR update. **Note:** in this case the 3rd ld/st exception is masked. This behavior is implementation dependent and may differ on other CPUs.
DVC DACx	Any instruction (no exception)	DACy, or DVC DAC_y Normal Ldst or multiple word Ldst	Take Debug exception, DBSR update setting DACx, DACy. DAC_OFST set to 3'b010. DSRR0 points to instruction after the 3rd class instruction. **Note:** in this case if x==y, then the resultant state of DBSR and DSRR0 may be indistinguishable from the "no DACy" case.

Table 1821 through *Table 1824* show some example updates for specific code sequences of dual issuing of load/store class instructions with non-load/store class instructions and the results of DAC and DVC events on selected ones of the load/store instructions.

Table 1821. DAC events and resultant updates, dual-issue case 1

Instruction sequence:	Event(s)	Result
The following pairs dual-issue: (1) load/store (2) alu (3) load/store (4) alu (5) load/store (6) alu	Instruction (1): DSI, no DAC	Take DSI exception, no DBSR update, Update ESR.
	Instruction (1): DSI, with DACx	Take Debug exception, DBSR update setting DACx and IDE, DAC_OFST set to 3'b000. DSRR0 points to instruction (1). No ESR update.
	Instruction (1): DACx	Take Debug exception, DBSR update setting DACx, DAC_OFST set to 3'b000. DSRR0 points to instruction (2). No ESR update.
	Instruction (1): DVC DACx No other exceptions	Take Debug exception, DBSR update setting DACx, DAC_OFST set to 3'b100. DSRR0 points to instruction (6). No ESR update. No debug event updates for instruction (6)
	Instruction (1): DVC DACx Instruction (3): DSI, with or without DAC	Take Debug exception, DBSR update setting DACx, DAC_OFST set to 3'b001. DSRR0 points to instruction (3). No ESR update. No debug event updates for instructions (3)–(6). **Note:** in this case the 2nd ld/st exception is masked. This behavior is implementation dependent and may differ on other CPUs.
	Instruction (1): DVC DACx Instruction (3): DACy	Take Debug exception, DBSR update setting DACx, DACy. DAC_OFST set to 3'b010. DSRR0 points to instruction (4). No debug event updates for instructions (4)–(6). **Note:** in this case if x==y, then the resultant state of DBSR and DSRR0 may be indistinguishable from the "no DACy" case.
	Instruction (1): DVC DACx Instruction (3): DVC DACy	Take Debug exception, DBSR update setting DACx, DAC_OFST set to 3'b100. DSRR0 points to instruction (6). No ESR update. No debug event updates for instruction (6). **Note:** in this case if x==y, then the resultant state of DBSR and DSRR0 may be indistinguishable from the "no DACy" case.
	Instruction (1): DVC DACx Instruction (3): DVC DACy Instruction (5): DSI, with or without DAC	Take Debug exception, DBSR update setting DACx, DACy. DAC_OFST set to 3'b010. No ESR update. DSRR0 points to instruction (4). No debug event updates for instructions (4)–(6). **Note:** in this case if x==y, then the resultant state of DBSR and DSRR0 may be indistinguishable from the "no DACy" case. **Note:** in this case the 3rd ld/st exception is masked. This behavior is implementation dependent and may differ on other CPUs.
	Instruction (1): DVC DACx Instruction (3): DVC DACy Instruction (5): DACy or DVC DACy	Take Debug exception, DBSR update setting DACx, DACy. DAC_OFST set to 3'b100. No ESR update. DSRR0 points to instruction (6). **Note:** in this case if x==y, then the resultant state of DBSR and DSRR0 may be indistinguishable from the "no DACy" case.

Table 1822. DAC events and resultant updates, dual-issue case 2

Instruction Sequence:	Event(s)	Result
The following pairs dual-issue: (1) load/store (2) alu (3) load/store (4) alu (5) alu (6) load/store	Instruction (1): DSI, no DAC	Take DSI exception, no DBSR update, Update ESR.
	Instruction (1): DSI, with DACx	Take Debug exception, DBSR update setting DACx and IDE, DAC_OFST set to 3'b000. DSRR0 points to instruction (1). No ESR update.
	Instruction (1): DACx	Take Debug exception, DBSR update setting DACx, DAC_OFST set to 3'b000. DSRR0 points to instruction (2). No ESR update.
	Instruction (1): DVC DACx No other exceptions	Take Debug exception, DBSR update setting DACx, DAC_OFST set to 3'b101. DSRR0 points to instruction after instruction (6). No ESR update.
	Instruction (1): DVC DACx Instruction (3): DSI, with or without DAC	Take Debug exception, DBSR update setting DACx, DAC_OFST set to 3'b001. DSRR0 points to instruction (3). No ESR update. No debug event updates for instructions (3)–(6). **Note:** in this case the 2nd ld/st exception is masked. This behavior is implementation dependent and may differ on other CPUs.
	Instruction (1): DVC DACx Instruction (3): DACy	Take Debug exception, DBSR update setting DACx, DACy. DAC_OFST set to 3'b010. DSRR0 points to instruction (4). No debug event updates for instructions (4)–(6). **Note:** in this case if x==y, then the resultant state of DBSR and DSRR0 may be indistinguishable from the "no DACy" case.
	Instruction (1): DVC DACx Instruction (3): DVC DACy	Take Debug exception, DBSR update setting DACx, DAC_OFST set to 3'b101. DSRR0 points to instruction (7). No ESR update. **Note:** in this case if x==y, then the resultant state of DBSR and DSRR0 may be indistinguishable from the "no DACy" case.
	Instruction (1): DVC DACx Instruction (3): DVC DACy Instruction (6): DSI, with or without DAC	Take Debug exception, DBSR update setting DACx, DACy. DAC_OFST set to 3'b010. No ESR update. DSRR0 points to instruction (4). No debug event updates for instruction (4). **Note:** in this case if x==y, then the resultant state of DBSR and DSRR0 may be indistinguishable from the "no DACy" case. **Note:** in this case the 3rd ld/st exception is masked. This behavior is implementation dependent and may differ on other CPUs.
	Instruction (1): DVC DACx Instruction (3): DVC DACy Instruction (6): DACy or DVC DACy	Take Debug exception, DBSR update setting DACx, DACy. DAC_OFST set to 3'b101. No ESR update. DSRR0 points to instruction (7). No debug event updates for instruction (7). **Note:** In this case if x==y, then the resultant state of DBSR and DSRR0 may be indistinguishable from the "no DACy" case.

Table 1823. DAC events and resultant updates, dual-issue case 3

Instruction Sequence	Event(s)	Result
The following pairs dual-issue: (1) load/store (2) alu (3) alu (4) alu (5) load/store (6) alu	Instruction (1): DSI, no DAC	Take DSI exception, no DBSR update, Update ESR.
	Instruction (1): DSI, with DACx	Take Debug exception, DBSR update setting DACx and IDE, DAC_OFST set to 3'b000. DSRR0 points to instruction (1). No ESR update.
	Instruction (1): DACx	Take Debug exception, DBSR update setting DACx, DAC_OFST set to 3'b000. DSRR0 points to instruction (2). No ESR update.
	Instruction (1): DVC DACx No other exceptions	Take Debug exception, DBSR update setting DACx, DAC_OFST set to 3'b100. DSRR0 points to instruction (6). No ESR update. No debug event updates for instruction (6).
	Instruction (1): DVC DACx Instruction (5): DSI, with or without DAC	Take Debug exception, DBSR update setting DACx, DAC_OFST set to 3'b011. DSRR0 points to instruction (5). No ESR update. No debug event updates for instructions (5)–(6). **Note:** in this case the 2nd ld/st exception is masked. This behavior is implementation dependent and may differ on other CPUs.
	Instruction (1): DVC DACx Instruction (5): DACy	Take Debug exception, DBSR update setting DACx, DACy. DAC_OFST set to 3'b100. DSRR0 points to instruction (6). No debug event updates for instruction (6). **Note:** in this case if x==y, then the resultant state of DBSR and DSRR0 may be indistinguishable from the "no DACy" case.
	Instruction (1): DVC DACx Instruction (5): DVC DACy	Take Debug exception, DBSR update setting DACx, DAC_OFST set to 3'b100. DSRR0 points to instruction (6). No ESR update. No debug event updates for instruction (6) **Note:** in this case if x==y, then the resultant state of DBSR and DSRR0 may be indistinguishable from the "no DACy" case.

Table 1824. DAC events and resultant updates, dual-issue case 4

Instruction Sequence:	Event(s)	Result
The following pairs dual-issue: (1) load/store (2) alu (3) load/store (4) alu (5) alu (6) alu	Instruction (1): DSI, no DAC	Take DSI exception, no DBSR update, Update ESR.
	Instruction (1): DSI, with DACx	Take Debug exception, DBSR update setting DACx and IDE, DAC_OFST set to 3'b000. DSRR0 points to instruction (1). No ESR update.
	Instruction (1): DACx	Take Debug exception, DBSR update setting DACx, DAC_OFST set to 3'b000. DSRR0 points to instruction (2). No ESR update.
	Instruction (1): DVC DACx No other exceptions	Take Debug exception, DBSR update setting DACx, DAC_OFST set to 3'b011. DSRR0 points to instruction (5). No ESR update. No debug event updates for instructions (5)–(6)
	Instruction (1): DVC DACx Instruction (3): DSI, with or without DAC	Take Debug exception, DBSR update setting DACx, DAC_OFST set to 3'b001. DSRR0 points to instruction (3). No ESR update. No debug event updates for instructions (3)–(6). **Note:** in this case the 2nd ld/st exception is masked. This behavior is implementation dependent and may differ on other CPUs.
	Instruction (1): DVC DACx Instruction (3): DACy	Take Debug exception, DBSR update setting DACx, DACy. DAC_OFST set to 3'b010. DSRR0 points to instruction (4). No debug event updates for instructions (4)–(6). **Note:** in this case if x==y, then the resultant state of DBSR and DSRR0 may be indistinguishable from the "no DACy" case.
	Instruction (1): DVC DACx Instruction (3): DVC DACy	Take Debug exception, DBSR update setting DACx, DAC_OFST set to 3'b011. DSRR0 points to instruction (5). No ESR update. No debug event updates for instructions (5)–(6). **Note:** in this case if x==y, then the resultant state of DBSR and DSRR0 may be indistinguishable from the "no DACy" case.

70.2.3 Linked Instruction Address Compare and Data Address Compare events

Data Address Compare 1, 2, 3, and 4 debug events may be 'linked' with an Instruction Address Compare event by setting the DAC[1–4]LNK control bits in DBCR2 and DBCR8 to further refine when a Data Address Compare debug event is generated. DAC1 may be linked with IAC1, DAC2 (when not used as a mask or range bounds register) may be linked with IAC3. DAC3 may be linked with IAC5, DAC4 (when not used as a mask or range bounds register) may be linked with IAC7. When linked, a DAC debug event occurs when the same instruction that generates the DAC 'hit' also generates a corresponding linked IAC 'hit'. When linked, the IAC event is not recorded in the Debug Status register, regardless of whether a corresponding linked DAC event occurs, or whether the IAC event enable is set.

When enabled and execution of a load or store class instruction results in a data access with an address that meets the criteria specified in the DBCRx, DACx, and DVCx Registers, and the instruction also meets the criteria for generating an Instruction Address Compare event, a Linked Data Address Compare debug event occurs. This event can occur and be recorded in DBSR regardless of the setting of MSR_{DE}. The normal DAC1 status bit in the

DBSR is used for recording these events. The IAC status bit is not set if the corresponding Instruction Address Compare register is linked.

Linking is enabled using control bits in DBCR2 and DBCR8. Note that linking is only available in EDM or IDM. Attempts to use linking otherwise are ignored.

Note: Linked DAC events will not be recorded if a load multiple word or store multiple word type instruction is interrupted prior to completion by a critical input or external input interrupt.

70.2.4 Trap debug event

A Trap debug event (TRAP) occurs if Trap debug events are enabled ($DBCR0_{TRAP}$=1), a Trap instruction (**tw**) is executed, and the conditions specified by the instruction for the trap are met. This event can occur and be recorded in DBSR regardless of the setting of MSR_{DE}. When a Trap debug event occurs, the $DBSR_{TRAP}$ bit is set to 1 to record the debug exception.

70.2.5 Branch Taken debug event

A Branch Taken debug event (BRT) occurs if Branch Taken debug events are enabled ($DBCR0_{BRT}$=1) and execution is attempted of a branch instruction that will be taken (either an unconditional branch, or a conditional branch whose branch condition is true), and MSR_{DE}=1. Branch Taken debug events are not recognized if MSR_{DE}=0 at the time of execution of the branch instruction and thus $DBSR_{IDE}$ can not be set by a Branch Taken debug event. When a Branch Taken debug event is recognized, the $DBSR_{BRT}$ bit is set to 1 to record the debug exception, and the address of the branch instruction will be recorded in DSRR0.

70.2.6 Instruction Complete debug event

An Instruction Complete debug event (ICMP) occurs if Instruction Complete debug events are enabled ($DBCR0_{ICMP}$=1), execution of any instruction is completed, and MSR_{DE}=1. If execution of an instruction is suppressed due to the instruction causing some other exception that is enabled to generate an interrupt, then the attempted execution of that instruction does not cause an Instruction Complete debug event. The ***se_sc*** instruction does not fall into the category of an instruction whose execution is suppressed, since the instruction actually executes and then generates a System Call interrupt. In this case, the Instruction Complete debug exception will also be set. When an Instruction Complete debug event is recognized, $DBSR_{ICMP}$ is set to 1 to record the debug exception and the address of the next instruction to be executed will be recorded in DSRR0.

Instruction Complete debug events are not recognized if MSR_{DE}=0 at the time of execution of the instruction, thus $DBSR_{IDE}$ is not generally set by an ICMP debug event.

One circumstance may cause the $DBSR_{ICMP}$ and $DBSR_{IDE}$ bits to be set. This occurs when a EFPU Round exception occurs. Since the instruction is by definition completed (SRR0 points to the following instruction), this interrupt takes higher priority than the Debug interrupt so as not to be lost, and $DBSR_{IDE}$ is set to indicate the imprecise recognition of a Debug interrupt. In this case, the Debug interrupt will be taken with SRR0 pointing to the instruction following the instruction that generated the EFPU Round exception, and DSRR0 will point to the Round exception handler. In addition to occurring when $DBCR0_{IDM}$=1, this circumstance can also occur when $EDBCR0_{EDM}$=1 and the event is hardware-owned, in which case $EDBSR0_{IDE}$ will be set.

Note: Instruction complete debug events are not generated by the execution of an instruction that sets MSR_{DE} to '1' while $DBCR0_{ICMP}$=1, nor by the execution of an instruction that sets $DBCR0_{ICMP}$ to '1' while MSR_{DE}=1.

70.2.7 Interrupt Taken debug event

An Interrupt Taken debug event (IRPT) occurs if Interrupt Taken debug events are enabled ($DBCR0_{IRPT}$=1) and a base-class interrupt occurs. Only base-class interrupts (an interrupt using SRR0/1) cause an Interrupt Taken debug event. This event can occur and be recorded in DBSR regardless of the setting of MSR_{DE}. When an Interrupt Taken debug event occurs, the $DBSR_{IRPT}$ bit is set to 1 to record the debug exception. The value saved in DSRR0 will be the address of the non-critical interrupt handler.

70.2.8 Critical Interrupt Taken debug event

A Critical Interrupt Taken debug event (CIRPT) occurs if Critical Interrupt Taken debug events are enabled ($DBCR0_{CIRPT}$=1) and a critical interrupt occurs. Only critical class interrupts (an interrupt using CSRR0/1) cause a Critical Interrupt Taken debug event. This event can occur and be recorded in DBSR regardless of the setting of MSR_{DE}. When a Critical Interrupt Taken debug event occurs, the $DBSR_{CIRPT}$ bit is set to 1 to record the debug exception. The value saved in DSRR0 will be the address of the critical interrupt handler.

70.2.9 Return debug event

A Return debug event (RET) occurs if Return debug events are enabled ($DBCR0_{RET}$=1) and an attempt is made to execute an **se_rfi** instruction. This event can occur and be recorded in DBSR regardless of the setting of MSR_{DE}. When a Return debug event occurs, the $DBSR_{RET}$ bit is set to 1 to record the debug exception.

If MSR_{DE}=0 at the time of the execution of the **se_rfi** (that is before the MSR is updated by the **se_rfi**), then $DBSR_{IDE}$ is also set to 1 to record the imprecise debug event.

If MSR_{DE}=1 at the time of the execution of the **se_rfi**, a Debug interrupt will occur provided there exists no higher priority exception that is enabled to cause an interrupt. Debug Save/Restore Register 0 will be set to the address of the **se_rfi** instruction.

70.2.10 Critical Return Debug Event

A Critical Return debug event (CRET) occurs if Critical Return debug events are enabled ($DBCR0_{CRET}$=1) and an attempt is made to execute an **se_rfci** instruction. This event can occur and be recorded in DBSR regardless of the setting of MSR_{DE}. When a Critical Return debug event occurs, the $DBSR_{CRET}$ bit is set to 1 to record the debug exception.

If MSR_{DE}=0 at the time of the execution of the **se_rfci** (that is before the MSR is updated by the **se_rfci**), then $DBSR_{IDE}$ is also set to 1 to record the imprecise debug event.

If MSR_{DE}=1 at the time of the execution of the **se_rfci**, a Debug interrupt will occur provided there exists no higher priority exception that is enabled to cause an interrupt. Debug Save/Restore Register 0 will be set to the address of the **se_rfci** instruction.

70.2.11 External debug event

An External debug event (DEVT1, DEVT2) occurs if External debug events are enabled ($DBCR0_{DEVT1}$=1 or $DBCR0_{DEVT2}$=1), and the respective **p_devt1** or **p_devt2** input signal

transitions to the asserted state while the CPU is not in the Stopped state. This event can occur and be recorded in DBSR regardless of the setting of MSR_{DE}. When an External debug event occurs, $DBSR_{DEVT\{1,2\}}$ is set to '1' to record the debug exception. This debug event is an asynchronous event, but is only sampled when the CPU **m_clk** is active.

70.2.12 Unconditional debug event

An Unconditional debug event (UDE) occurs when the Unconditional Debug Event (**p_ude**) input transitions to the asserted state. The Unconditional debug event is the only debug event that does not have a corresponding enable bit for the event in DBCR0. This event can occur and be recorded in DBSR regardless of the setting of MSR_{DE}. When an Unconditional debug event occurs, the $DBSR_{UDE}$ bit is set to '1' to record the debug exception. This debug event is an asynchronous event.

70.2.13 Performance Monitor Interrupt debug event

A Performance Monitor Interrupt debug event (PMI) occurs if Performance Monitor Interrupt debug events are enabled ($PMGC0_{UDI}$=1), and a performance monitor interrupt event occurs. This event can occur and be recorded in DBSR regardless of the setting of MSR_{DE}. When a Performance Monitor Interrupt debug event occurs, $DBSR_{PMI}$ is set to '1' to record the debug exception. This debug event is an asynchronous event.

70.3 Debug registers

This section describes debug-related registers that are software accessible. These registers are intended for use by special debug tools and debug software, not by general application code.

Access to these registers (other than DBSR) by software is conditioned by the External Debug mode control bit ($EDBCR0_{EDM}$) and the settings of debug control register EDBRAC0, which can be set by the hardware debug port. If $EDBCR0_{EDM}$ is set and if the bit in EDBRAC0 corresponding to the resource is cleared, software is prevented from modifying debug register values other than in DBSR, since the resource is not "owned" by software. Software always has ownership of DBSR. Execution of a **mtspr** instruction targeting a debug register or register field not "owned" by software will not cause modifications to occur, and no exception will be signaled. In addition, since the external debugger hardware may be manipulating debug register values, the state of these registers or register fields not "owned" by software is not guaranteed to be consistent if accessed (read) by software with a **mfspr** instruction, other than the $DBCR0_{EDM}$ bit itself and the EDBRAC0 register. Hardware always has full access to all registers and all register fields through the OnCE register access mechanism, and it is up to the debug firmware to properly implement modifications to these registers with read-modify-write operations to implement any control sharing with software. Settings in EDBRAC0 should be considered by the debug firmware in order to preserve software settings of control registers as appropriate when hardware modifications to the debug registers is performed.

70.3.1 Debug Address and Value registers

Instruction Address Compare registers IAC1–8 are used to hold instruction addresses for address comparison purposes. In addition, IAC2 and IAC4 may hold mask information for IAC1 and IAC3 respectively and IAC6 and IAC8 may hold mask information for IAC5 and IAC7 respectively, when *Address Bit Match* compare modes are selected. Note that when

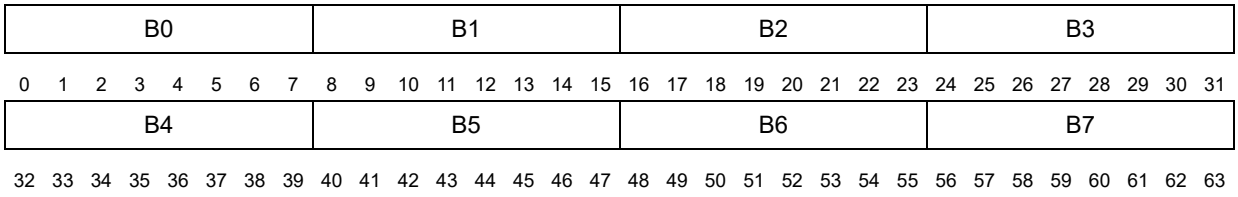
performing instruction address compares, the low order bit of the instruction address and the corresponding IAC register is ignored for VLE instructions.

Data Address Compare registers DAC1–4 are used to hold data access addresses for address comparison purposes. In addition, DAC2 and DAC4 may hold mask information for DAC1 and DAC3 respectively when *Address Bit Match* compare modes are selected.

Data Value Compare registers DVC1 and DVC2 are used to hold data values for data comparison purposes. DVC1 and DVC2 are 64-bit registers. Data value comparisons are used to qualify Data Address compare 1 and 2 debug events. Data value comparisons are not available for Data address compare 3–4 events. DVC1 is associated with DAC1, and DVC2 is associated with DAC2. The most significant byte of the DVC1(2) register (labeled B0 in *Figure 1891*) corresponds to the byte data value transferred to/from memory byte offset 0, 8, ..., and the least significant byte of the register (labeled B7 in *Figure 1891*) corresponds to byte offset 7, F, ... When enabled for performing data value comparisons, each enabled byte in DVC1(2) is compared with the memory value transferred on the corresponding active byte lane of the data memory interface to determine if a match occurs. Inactive byte lanes do not participate in the comparison; they are implicitly masked. The Byte Strobe Assertion for Transfers table in the Core (e200z425n3) Core Complex Overview chapter shows active byte lanes for data transfers. Software must also program the DVC1(2) register byte positions based on the alignment of the access. Misaligned accesses are not fully supported, since the data address and data value comparisons are only performed on the initial access in the case of a misaligned access; thus, accesses that cross a 64-bit boundary cannot be fully matched. For address and size combinations that involve two transfers, only the initial transfer is used for data address and value matching.

DVC1 and DVC2 may be read or written using **mtspr** and **mfspr** instructions. The DVC1U and DVC2U SPR numbers (601,602) are used for accessing the upper 32-bit portion of the DVC register. The DVC1 and DVC2 SPR numbers (318,319) are used for accessing the lower 32-bit portion of the DVC register.

Figure 1891. DVC1, DVC2 registers



SPR 601 (DVC1U), 318 (DVC1), 602 (DVC2U) 319 (DVC2); Read/Write; Reset: Unaffected

70.3.2 Debug Control and Status registers

Debug Control Registers (DBCR0–8 and EDBRAC0) are used to enable debug events, reset the processor, and set the debug mode of the processor. The Debug Status register (DBSR) records debug exceptions while Internal Debug mode is enabled. The Debug Data Effective Address register (DDEAR) records the effective address of a Data Address Compare event while Internal Debug mode is enabled.

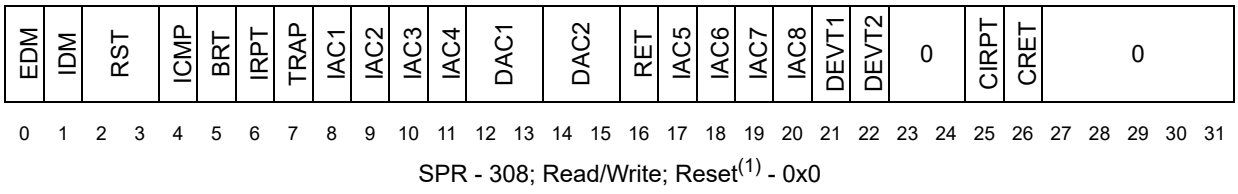
e200z425n3 requires that a context synchronizing instruction follow a **mtspr** DBCR0–8 or DBSR to ensure that any alterations enabling/disabling debug events are effective. The context synchronizing instruction may or may not be affected by the alteration. Typically, an **se_isync** instruction is used to create a synchronization boundary beyond which it can be guaranteed that the newly written control values are in effect.

For watchpoint generation, configuration settings contained in DBCR1–8 are used, even though the corresponding event(s) may be disabled (via DBCR0) from setting DBSR flags.

70.3.2.1 Debug Control Register 0 (DBCR0)

Debug Control Register 0 is used to enable debug modes and controls which debug events are allowed to set DBSR or EDBSR0 flags. e200z425n3 adds some implementation specific bits to this register, as seen in *Figure 1892*.

Figure 1892. Debug Control Register 0 (DBCR0) register



SPR - 308; Read/Write; Reset(1) - 0x0

1. $DBCR0_{EDM}$ is affected by **j_trst_b** or **m_por** assertion, and remains reset while in the Test_Logic_Reset state, but is not affected by **p_reset_b**. All other bits are reset by processor reset **p_reset_b** if $DBCR0_{EDM}$=0, as well as unconditionally by **m_por**. If $DBCR0_{EDM}$=1, EDBRAC0 masks off hardware-owned resources (other than RST) from reset by **p_reset_b,** and only software-owned resources indicated by EDBRAC0 and the $DBCR0_{RST}$ field will be reset by **p_reset_b**. The $DBCR0_{RST}$ field will always be reset by **p_reset_b** regardless of the value of $DBCR0_{EDM}$.

Table 1825 provides bit definitions for Debug Control Register 0.

Table 1825. DBCR0 field descriptions

Bit	Name	Description
0	EDM	External Debug mode. This bit is read-only by software. When external debug mode is enabled, hardware-owned resources in debug registers are not affected by processor reset **p_reset_b**. This allows the debugger to set up hardware debug events that remain active across a processor reset. 0 External debug mode disabled. Internal debug events not mapped into external debug events. 1 External debug mode enabled. Hardware-owned debug events will not cause the CPU to vector to interrupt code. Software is not permitted to write to debug registers {DBCR0–8, IAC1–8, DAC1–4, DVC1–2[U]} unless permitted by settings in EDBRAC0. Hardware-owned events will set status bits in EDBSR0.
		Programming Notes: It is recommended that debug status bits in the Debug Status Registers be cleared before disabling external debug mode to avoid any internal imprecise debug interrupts. Software may use this bit to determine if external debug has control over the debug registers. The hardware debugger must set the EDM bit to ‘1’ before other bits in this register (and other debug registers) may be altered. On the initial setting of this bit to ‘1’, all other bits are unchanged. This bit is only writable through the OnCE port.
1	IDM	Internal Debug mode 0 Debug exceptions are disabled. Debug events do not affect DBSR. 1 Debug exceptions are enabled. Enabled debug events owned by software will update the DBSR. If MSR_{DE}=1, the occurrence of a debug event, or the recording of an earlier debug event in the Debug Status Register when MSR_{DE} was cleared, will cause a Debug interrupt.

Table 1825. DBCR0 field descriptions (continued)

Bit	Name	Description
2:3	RST	Reset Control 00 No function 01 **p_dbrstc[1]** pin asserted by Debug Reset Control. Allows external device to initiate processor or system reset 10 **p_dbrstc[0]** pin asserted by Debug Reset Control. Allows external device to initiate processor or system reset. 11 Reserved
4	ICMP	Instruction Complete Debug Event Enable 0 ICMP debug events are disabled 1 ICMP debug events are enabled
5	BRT	Branch Taken Debug Event Enable 0 BRT debug events are disabled 1 BRT debug events are enabled
6	IRPT	Interrupt Taken Debug Event Enable 0 IRPT debug events are disabled 1 IRPT debug events are enabled
7	TRAP	Trap Taken Debug Event Enable 0 TRAP debug events are disabled 1 TRAP debug events are enabled
8	IAC1	Instruction Address Compare 1 Debug Event Enable 0 IAC1 debug events are disabled 1 IAC1 debug events are enabled
9	IAC2	Instruction Address Compare 2 Debug Event Enable 0 IAC2 debug events are disabled 1 IAC2 debug events are enabled
10	IAC3	Instruction Address Compare 3 Debug Event Enable 0 IAC3 debug events are disabled 1 IAC3 debug events are enabled
11	IAC4	Instruction Address Compare 4 Debug Event Enable 0 IAC4 debug events are disabled 1 IAC4 debug events are enabled
12:13	DAC1	Data Address Compare 1 Debug Event Enable 00 DAC1 debug events are disabled 01 DAC1 debug events are enabled only for store-type data storage accesses 10 DAC1 debug events are enabled only for load-type data storage accesses 11 DAC1 debug events are enabled for load-type or store-type data storage accesses
14:15	DAC2	Data Address Compare 2 Debug Event Enable 00 DAC2 debug events are disabled 01 DAC2 debug events are enabled only for store-type data storage accesses 10 DAC2 debug events are enabled only for load-type data storage accesses 11 DAC2 debug events are enabled for load-type or store-type data storage accesses
16	RET	Return Debug Event Enable 0 RET debug events are disabled 1 RET debug events are enabled

Table 1825. DBCR0 field descriptions (continued)

Bit	Name	Description
17	IAC5	Instruction Address Compare 5 Debug Event Enable 0 IAC5 debug events are disabled 1 IAC5 debug events are enabled
18	IAC6	Instruction Address Compare 6 Debug Event Enable 0 IAC6 debug events are disabled 1 IAC6 debug events are enabled
19	IAC7	Instruction Address Compare 7 Debug Event Enable 0 IAC7 debug events are disabled 1 IAC7 debug events are enabled
20	IAC8	Instruction Address Compare 8 Debug Event Enable 0 IAC8 debug events are disabled 1 IAC8 debug events are enabled
21	DEVT1	External Debug Event 1 Enable 0 DEVT1 debug events are disabled 1 DEVT1 debug events are enabled
22	DEVT2	External Debug Event 2 Enable 0 DEVT2 debug events are disabled 1 DEVT2 debug events are enabled
23:24	—	Reserved
25	CIRPT	Critical Interrupt Taken Debug Event Enable 0 CIRPT debug events are disabled 1 CIRPT debug events are enabled
26	CRET	Critical Return Debug Event Enable 0 CRET debug events are disabled 1 CRET debug events are enabled
27:31	—	Reserved

70.3.2.2 Debug Control Register 1 (DBCR1)

Debug Control Register 1 is used to configure Instruction Address Compare operation. The DBCR1 register is shown in *Figure 1893*.

Figure 1893. DBCR1 register

IAC1US	IAC1ER	IAC2US	IAC2ER	IAC12M	0	IAC3US	IAC3ER	IAC4US	IAC4ER	IAC34M	0

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

SPR - 309; Read/Write; Reset(1) - 0x0

1. Reset by processor reset **p_reset_b** if $EDBCR0_{EDM}$=0, as well as unconditionally by **m_por**. If $EDBCR0_{EDM}$=1, EDBRAC0 masks off hardware-owned resources from reset by **p_reset_b** and only software-owned resources indicated by EDBRAC0 will be reset by **p_reset_b.**

Table 1826 provides bit definitions for Debug Control Register 1.

Table 1826. DBCR1 field descriptions

Bit	Name	Description
0:1	IAC1US	Instruction Address Compare 1 User/Supervisor Mode 00 IAC1 debug events not affected by MSR_{PR} 01 Reserved 10 IAC1 debug events can only occur if MSR_{PR}=0 (Supervisor mode) 11 IAC1 debug events can only occur if MSR_{PR}=1. (User mode)
2:3	IAC1ER	Instruction Address Compare 1 Effective/Real Mode 00 IAC1 debug events are based on effective address 01 Unimplemented (Book E real address compare), no match can occur 10 IAC1 debug events are based on effective address and can only occur if MSR_{IS}=0 11 IAC1 debug events are based on effective address and can only occur if MSR_{IS}=1
4:5	IAC2US	Instruction Address Compare 2 User/Supervisor Mode 00 IAC2 debug events not affected by MSR_{PR} 01 Reserved 10 IAC2 debug events can only occur if MSR_{PR}=0 (Supervisor mode) 11 IAC2 debug events can only occur if MSR_{PR}=1. (User mode)
6:7	IAC2ER	Instruction Address Compare 2 Effective/Real Mode 00 IAC2 debug events are based on effective address 01 Unimplemented (Book E real address compare), no match can occur 10 IAC2 debug events are based on effective address and can only occur if MSR_{IS}=0 11 IAC2 debug events are based on effective address and can only occur if MSR_{IS}=1
8:9	IAC12M	Instruction Address Compare 1/2 Mode 00 Exact address compare. IAC1 debug events can only occur if the address of the instruction fetch is equal to the value specified in IAC1. IAC2 debug events can only occur if the address of the instruction fetch is equal to the value specified in IAC2. 01 Address bit match. IAC1 debug events can occur only if the address of the instruction fetch, ANDed with the contents of IAC2 are equal to the contents of IAC1, also ANDed with the contents of IAC2. IAC2 debug events do not occur. IAC1US and IAC1ER settings are used. 10 Inclusive address range compare. IAC1 debug events can occur only if the address of the instruction fetch is greater than or equal to the value specified in IAC1 and less than the value specified in IAC2. IAC2 debug events do not occur. IAC1US and IAC1ER settings are used. 11 Exclusive address range compare. IAC1 debug events can occur only if the address of the instruction fetch is less than the value specified in IAC1 or is greater than or equal to the value specified in IAC2. IAC2 debug events do not occur. IAC1US and IAC1ER settings are used.
10:15	—	Reserved
16:17	IAC3US	Instruction Address Compare 3 User/Supervisor Mode 00 IAC3 debug events not affected by MSR_{PR} 01 Reserved 10 IAC3 debug events can only occur if MSR_{PR}=0 (Supervisor mode) 11 IAC3 debug events can only occur if MSR_{PR}=1 (User mode)
18:19	IAC3ER	Instruction Address Compare 3 Effective/Real Mode 00 IAC3 debug events are based on effective address 01 Unimplemented (Book E real address compare), no match can occur 10 IAC3 debug events are based on effective address and can only occur if MSR_{IS}=0 11 IAC3 debug events are based on effective address and can only occur if MSR_{IS}=1

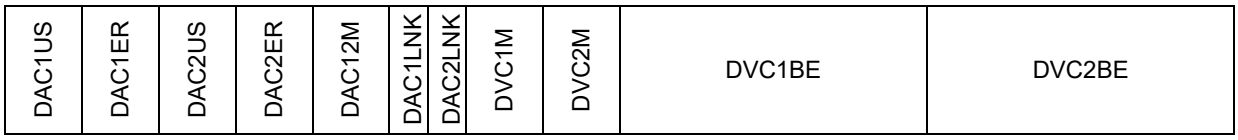
Table 1826. DBCR1 field descriptions (continued)

Bit	Name	Description
20:21	IAC4US	Instruction Address Compare 4 User/Supervisor Mode 00 IAC4 debug events not affected by MSR_{PR} 01 Reserved 10 IAC4 debug events can only occur if MSR_{PR}=0 (Supervisor mode). 11 IAC4 debug events can only occur if MSR_{PR}=1. (User mode)
22:23	IAC4ER	Instruction Address Compare 4 Effective/Real Mode 00 IAC4 debug events are based on effective address 01 Unimplemented (Book E real address compare), no match can occur 10 IAC4 debug events are based on effective address and can only occur if MSR_{IS}=0 11 IAC4 debug events are based on effective address and can only occur if MSR_{IS}=1
24:25	IAC34M	Instruction Address Compare 3/4 Mode 00 Exact address compare. IAC3 debug events can only occur if the address of the instruction fetch is equal to the value specified in IAC3. IAC4 debug events can only occur if the address of the instruction fetch is equal to the value specified in IAC4. 01 Address bit match. IAC3 debug events can occur only if the address of the instruction fetch, ANDed with the contents of IAC4 are equal to the contents of IAC3, also ANDed with the contents of IAC4. IAC4 debug events do not occur. IAC3US and IAC3ER settings are used. 10 Inclusive address range compare. IAC3 debug events can occur only if the address of the instruction fetch is greater than or equal to the value specified in IAC3 and less than the value specified in IAC4. IAC4 debug events do not occur. IAC3US and IAC3ER settings are used. 11 Exclusive address range compare. IAC3 debug events can occur only if the address of the instruction fetch is less than the value specified in IAC3 or is greater than or equal to the value specified in IAC4. IAC4 debug events do not occur. IAC3US and IAC3ER settings are used.
26:31	—	Reserved

70.3.2.3 Debug Control Register 2 (DBCR2)

Debug Control Register 2 is used to configure Data Address Compare and Data Value Compare operation.The DBCR2 register is shown in *Figure 1894*.

Figure 1894. DBCR2 register



SPR - 310; Read/Write; Reset(1) - 0x0

1. Reset by processor reset **p_reset_b** if $EDBCR0_{EDM}$=0, as well as unconditionally by **m_por**. If $EDBCR0_{EDM}$=1, EDBRAC0 masks off hardware-owned resources from reset by **p_reset_b** and only software-owned resources indicated by EDBRAC0 will be reset by **p_reset_b**.

Table 1827 provides bit definitions for Debug Control Register 2.

Table 1827. DBCR2 field descriptions

Bit	Name	Description
0:1	DAC1US	Data Address Compare 1 User/Supervisor Mode 00 DAC1 debug events not affected by MSR_{PR} 01 Reserved 10 DAC1 debug events can only occur if MSR_{PR}=0 (Supervisor mode) 11 DAC1 debug events can only occur if MSR_{PR}=1 (User mode)
2:3	DAC1ER	Data Address Compare 1 Effective/Real Mode 00 DAC1 debug events are based on effective address 01 Unimplemented (Book E real address compare), no match can occur 10 DAC1 debug events are based on effective address and can only occur if MSR_{DS}=0 11 DAC1 debug events are based on effective address and can only occur if MSR_{DS}=1
4:5	DAC2US	Data Address Compare 2 User/Supervisor Mode 00 DAC2 debug events not affected by MSR_{PR} 01 Reserved 10 DAC2 debug events can only occur if MSR_{PR}=0 (Supervisor mode) 11 DAC2 debug events can only occur if MSR_{PR}=1. (User mode)
6:7	DAC2ER	Data Address Compare 2 Effective/Real Mode 00 DAC2 debug events are based on effective address 01 Unimplemented (Book E real address compare), no match can occur 10 DAC2 debug events are based on effective address and can only occur if MSR_{DS}=0 11 DAC2 debug events are based on effective address and can only occur if MSR_{DS}=1

Table 1827. DBCR2 field descriptions (continued)

Bit	Name	Description
8:9	DAC12M	Data Address Compare 1/2 Mode 00 Exact address compare. DAC1 debug events can only occur if the address of the data access is equal to the value specified in DAC1. DAC2 debug events can only occur if the address of the data access is equal to the value specified in DAC2. 01 Address bit match. DAC1 debug events can occur only if the address of the data access ANDed with the contents of DAC2, are equal to the contents of DAC1 also ANDed with the contents of DAC2. DAC2 debug events do not occur. DAC1US and DAC1ER settings are used. 10 Inclusive address range compare. DAC1 debug events can occur only if the address of the data access is greater than or equal to the value specified in DAC1 and less than the value specified in DAC2. DAC2 debug events do not occur. DAC1US and DAC1ER settings are used. 11 Exclusive address range compare. DAC1 debug events can occur only if the address of the data access is less than the value specified in DAC1 or is greater than or equal to the value specified in DAC2. DAC2 debug events do not occur. DAC1US and DAC1ER settings are used.
10	DAC1LNK	Data Address Compare 1 Linked. When linked to IAC1, DAC1 debug events are conditioned based on whether the instruction also generated an IAC1 debug event. Note that linking is only available in EDM or IDM. 0 No effect 1 DAC1 debug events are linked to IAC1 debug events. IAC1 debug events do not affect DBSR
11	DAC2LNK	Data Address Compare 2 Linked. When linked to IAC3, DAC2 debug events are conditioned based on whether the instruction also generated an IAC3 debug event. DAC2 can only be linked if DAC12M specifies *Exact Address Compare* since DAC2 debug events are not generated in the other compare modes. Note that linking is only available in EDM or IDM. 0 No effect 1 DAC 2 debug events are linked to IAC3 debug events. IAC3 debug events do not affect DBSR.

Table 1827. DBCR2 field descriptions (continued)

Bit	Name	Description
12:13	DVC1M	Data Value Compare 1 Mode When $DBCR4_{DVC1C}$=0: 00 DAC1 debug events not affected by data value compares. 01 DAC1 debug events can only occur when all bytes specified in the DVC1BE field match the corresponding data byte values for active byte lanes of the memory access. 10 DAC1 debug events can only occur when any byte specified in the DVC1BE field matches the corresponding data byte value for active byte lanes of the memory access. 11 DAC1 debug events can only occur when all bytes specified in the DVC1BE field within at least one of the halfwords of the data value of the memory access matches the corresponding DVC1 value. **Note:** Inactive byte lanes of the memory access are automatically masked. When $DBCR4_{DVC1C}$=1: 00 Reserved 01 DAC1 debug events can only occur when any byte specified in the DVC1BE field does not match the corresponding data byte value for active byte lanes of the memory access. If all active bytes match, then no event will be generated. 10 DAC1 debug events can only occur when all bytes specified in the DVC1BE field do not match the corresponding data byte values for active byte lanes of the memory access. If any active byte match occurs, no event will be generated. 11 Reserved **Note:** Inactive byte lanes of the memory access are automatically masked.
14:15	DVC2M	Data Value Compare 2 Mode When $DBCR4_{DVC2C}$=0: 00 DAC2 debug events not affected by data value compares. 01 DAC2 debug events can only occur when all bytes specified in the DVC2BE field match the corresponding data byte values for active byte lanes of the memory access. 10 DAC2 debug events can only occur when any byte specified in the DVC2BE field matches the corresponding data byte value for active byte lanes of the memory access. 11 DAC2 debug events can only occur when all bytes specified in the DVC2BE field within at least one of the halfwords of the data value of the memory access matches the corresponding DVC2 value. **Note:** Inactive byte lanes of the memory access are automatically masked. When $DBCR4_{DVC2C}$=1: 00 Reserved 01 DAC2 debug events can only occur when any byte specified in the DVC2BE field does not match the corresponding data byte value for active byte lanes of the memory access. If all active bytes match, then no event will be generated. 10 DAC2 debug events can only occur when all bytes specified in the DVC2BE field do not match the corresponding data byte values for active byte lanes of the memory access. If any active byte match occurs, no event will be generated. 11 Reserved **Note:** Inactive byte lanes of the memory access are automatically masked.

Table 1827. DBCR2 field descriptions (continued)

Bit	Name	Description
16:23	DVC1BE	Data Value Compare 1 Byte Enables Specifies which bytes in the aligned doubleword value associated with the memory access are compared to the corresponding bytes in DVC1. Inactive byte lanes of a memory access smaller than 64 bits are automatically masked by hardware. If all bits in the DVC1BE field are clear, then a match will occur regardless of the data. Misaligned accesses that cross a doubleword boundary are not fully supported. 1*xxxxxxx* Byte lane 0 is enabled for comparison with the value in bits 0:7 of DVC1. *x*1*xxxxxx* Byte lane 1 is enabled for comparison with the value in bits 8:15 of DVC1. *xx*1*xxxxx* Byte lane 2 is enabled for comparison with the value in bits 16:23 of DVC1. *xxx*1*xxxx* Byte lane 3 is enabled for comparison with the value in bits 24:31 of DVC1. *xxxx*1*xxx* Byte lane 4 is enabled for comparison with the value in bits 32:39 of DVC1. *xxxxx*1*xx* Byte lane 5 is enabled for comparison with the value in bits 40:47 of DVC1. *xxxxxx*1*x* Byte lane 6 is enabled for comparison with the value in bits 48:55 of DVC1. *xxxxxxx*1 Byte lane 7 is enabled for comparison with the value in bits 56:63 of DVC1.
24:31	DVC2BE	Data Value Compare2 Byte Enables Specifies which bytes in the aligned doubleword value associated with the memory access are compared to the corresponding bytes in DVC2. Inactive byte lanes of a memory access smaller than 64 bits are automatically masked by hardware. If all bits in the DVC1BE field are clear, then a match will occur regardless of the data. Misaligned accesses that cross a doubleword boundary are not fully supported. 1*xxxxxxx* Byte lane 0 is enabled for comparison with the value in bits 0:7 of DVC2. *x*1*xxxxxx* Byte lane 1 is enabled for comparison with the value in bits 8:15 of DVC2. *xx*1*xxxxx* Byte lane 2 is enabled for comparison with the value in bits 16:23 of DVC2. *xxx*1*xxxx* Byte lane 3 is enabled for comparison with the value in bits 24:31 of DVC2. *xxxx*1*xxx* Byte lane 4 is enabled for comparison with the value in bits 32:39 of DVC2. *xxxxx*1*xx* Byte lane 5 is enabled for comparison with the value in bits 40:47 of DVC2. *xxxxxx*1*x* Byte lane 6 is enabled for comparison with the value in bits 48:55 of DVC2. *xxxxxxx*1 Byte lane 7 is enabled for comparison with the value in bits 56:63 of DVC2.

70.3.2.4 Debug Control Register 4 (DBCR4)

Debug Control Register 4 is used to extend data address and value compare matching functionality. DBCR4 is shown in *Figure 1895*.

Figure 1895. DBCR4 register

0	DVC1C	0	DVC2C	0	0	DAC1XMH	0	DAC2XMH	DAC1XM	DAC2XM	DAC1CFG	DAC2CFG
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31

SPR - 563; Read/Write; Reset(1) - 0x0

1. DBCR4 is reset by processor reset **p_reset_b** if $EDBCR0_{EDM}$=0, as well as unconditionally by **m_por**. If $EDBCR0_{EDM}$=1, EDBRAC0 masks off hardware-owned resources from reset by **p_reset_b** and only software-owned resources indicated by EDBRAC0 will be reset by **p_reset_b**.

Table 1828 provides bit definitions for Debug Control Register 4.

Table 1828. DBCR4 field descriptions

Bit	Name	Description
0	—	Reserved
1	DVC1C	Data Value Compare 1 Control. DVC1C controls whether DVC1 data value comparisons utilize the normal compare operation, or an alternate “inverted compare” operation. In inverted polarity mode, data value compares perform a not-equal comparison. See details in the DBCR2 register definition. 0 Normal DVC1 operation. 1 Inverted polarity DVC1 operation
2	—	Reserved
3	DVC2C	Data Value Compare 2 Control. DVC2C controls whether DVC2 data value comparisons utilize the normal compare operation, or an alternate “inverted compare” operation. In inverted polarity mode, data value compares perform a not-equal comparison. See details in the DBCR2 register definition 0 Normal DVC2 operation. 1 Inverted polarity DVC2 operation
4:12	—	Reserved
13	DAC1XMH	Data Address Compare 1 Extended Mask Control High. DAC1XMH extends the range of the DAC1XM field 0 DAC1XM masks 0–15 low-order address bits 1 DAC1XM masks 16–31 low-order address bits
14	—	Reserved
15	DAC2XMH	Data Address Compare 2 Extended Mask Control High. DAC2XMH extends the range of the DAC2XM field. 0 DAC2XM masks 0–15 low-order address bits 1 DAC2XM masks 16–31 low-order address bits
16:19	DAC1XM	Data Address Compare 1 Extended Mask Control. DAC1XM allows for binary power of 2 address range compares for DAC1 without requiring the use of DAC2. Value of DAC1XMH || DAC1XM: 00000 No additional masking when DBCR2[DAC12M] = 00 00001–11111 Exact Match Bit Mask. One to 31 low-order bits are masked in DAC1 when comparing the storage address with the value in DAC1 for exact address compare (DBCR2[DAC12M] = 00). Address ranges of 2 bytes to 2GB are supported.
20:23	DAC2XM	Data Address Compare 2 Extended Mask Control Value of DAC2XMH || DAC2XM: 00000 No additional masking when DBCR2[DAC12M] = 00 00001–11111 Exact Match Bit Mask. One to 31 low-order bits are masked in DAC2 when comparing the storage address with the value in DAC2 for exact address compare (DBCR2[DAC12M] = 00). Address ranges of 2 bytes to 2GB are supported. DAC2XM allows for binary power of 2 address range compares for DAC2.

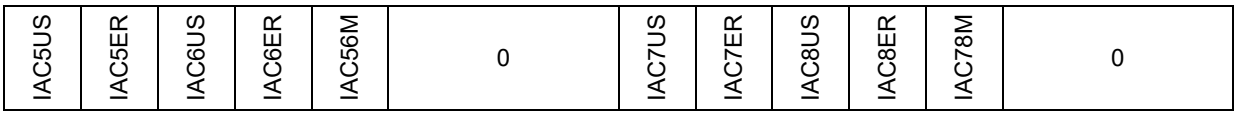
Table 1828. DBCR4 field descriptions (continued)

Bit	Name	Description
24:27	DAC1CFG	Data Address Compare 1 Configuration 0000 DAC1 debug watchpoints are enabled for load-type or store-type data storage accesses when $DBCR0_{DAC1}$=00 0001 DAC1 debug watchpoints are enabled only for store-type data storage accesses when $DBCR0_{DAC1}$=00 0010 DAC1 debug watchpoints are enabled only for load-type data storage accesses when $DBCR0_{DAC1}$=00 0011 Reserved 01*xx* Reserved 1000 DAC1 address comparisons are used for stack limit checking. DAC1 address comparisons are qualified with use of GPR R1 in <EA> calculation for most load or store instructions. No debug events occur for DAC1. When a qualified DAC1 match occurs, a machine check exception is generated for supervisor-mode accesses or a DSI is generated for user-mode accesses, and a DAC1 watchpoint is generated. $DBCR0_{DAC1}$ and $DBCR2_{DVC1M}$ settings are ignored. 1001 – 1111 Reserved DAC1CFG controls whether DAC1 data address comparisons utilize the normal PowerISA operation for generating both debug events and watchpoints, a watchpoint-only function, or a stack limit checking function (with watchpoint). **Note:** Unlike the DAC3–4CFG fields, debug event enabling for DAC1 is controlled by the DAC1 field in DBCR0. The DAC1CFG encodings 0000–0010 are used to control watchpoint generation when $DBCR0_{DAC1}$=0; when $DBCR0_{DAC1}$!=00, DAC1 watchpoints will fire whenever a DAC1 debug event occurs.
28:31	DAC2CFG	Data Address Compare 2 Configuration 0000 DAC2 debug watchpoints are enabled for load-type or store-type data storage accesses when $DBCR0_{DAC2}$=00 0001 DAC2 debug watchpoints are enabled only for store-type data storage accesses when $DBCR0_{DAC2}$=00 0010 DAC2 debug watchpoints are enabled only for load-type data storage accesses when $DBCR0_{DAC2}$=00 0011 Reserved 01*xx* Reserved 1*xxx* Reserved DAC2CFG controls whether DAC2 data address comparisons utilize the normal compare operation for generating both debug events and watchpoints, or a watchpoint-only function. **Note:** Unlike the DAC3–4CFG fields, debug event enabling for DAC2 is controlled by the DAC2 field in DBCR0. The DAC2CFG encodings 0000–0010 are used to control watchpoint generation when $DBCR0_{DAC2}$=00. When $DBCR0_{DAC2}$!=00, DAC2 watchpoints will fire whenever a DAC2 debug event occurs.

70.3.2.5 Debug Control Register 5 (DBCR5)

Debug Control Register 5 is used to configure Instruction Address Compare operation for IAC5–8. The DBCR5 register is shown in *Figure 1896*

Figure 1896. DBCR5 register



SPR - 564; Read/Write; Reset(1) - 0x0

1. Reset by processor reset **p_reset_b** if $EDBCR0_{EDM}$=0, as well as unconditionally by **m_por**. If $EDBCR0_{EDM}$=1, EDBRAC0 masks off hardware-owned resources from reset by **p_reset_b** and only software-owned resources indicated by EDBRAC0 will be reset by **p_reset_b**.

Table 1829 provides bit definitions for Debug Control Register 5.

Table 1829. DBCR5 field descriptions

Bit	Name	Description
0:1	IAC5US	Instruction Address Compare 5 User/Supervisor Mode 00 IAC5 debug events not affected by MSR_{PR} 01 Reserved 10 IAC5 debug events can only occur if MSR_{PR}=0 (Supervisor mode) 11 IAC5 debug events can only occur if MSR_{PR}=1. (User mode)
2:3	IAC5ER	Instruction Address Compare 5 Effective/Real Mode 00 IAC5 debug events are based on effective address 01 Unimplemented (Book E real address compare), no match can occur 10 IAC5 debug events are based on effective address and can only occur if MSR_{IS}=0 11 IAC5 debug events are based on effective address and can only occur if MSR_{IS}=1
4:5	IAC6US	Instruction Address Compare 6 User/Supervisor Mode 00 IAC6 debug events not affected by MSR_{PR} 01 Reserved 10 IAC6 debug events can only occur if MSR_{PR}=0 (Supervisor mode) 11 IAC6 debug events can only occur if MSR_{PR}=1. (User mode)
6:7	IAC6ER	Instruction Address Compare 6 Effective/Real Mode 00 IAC6 debug events are based on effective address 01 Unimplemented (Book E real address compare), no match can occur 10 IAC6 debug events are based on effective address and can only occur if MSR_{IS}=0 11 IAC6 debug events are based on effective address and can only occur if MSR_{IS}=1
8:9	IAC56M	Instruction Address Compare 5/6 Mode 00 Exact address compare. IAC5 debug events can only occur if the address of the instruction fetch is equal to the value specified in IAC5. IAC6 debug events can only occur if the address of the instruction fetch is equal to the value specified in IAC6. 01 Address bit match. IAC5 debug events can occur only if the address of the instruction fetch, ANDed with the contents of IAC6 are equal to the contents of IAC5, also ANDed with the contents of IAC6. IAC6 debug events do not occur. IAC5US and IAC5ER settings are used. 10 Inclusive address range compare. IAC5 debug events can occur only if the address of the instruction fetch is greater than or equal to the value specified in IAC5 and less than the value specified in IAC6. IAC6 debug events do not occur. IAC5US and IAC5ER settings are used. 11 Exclusive address range compare. IAC5 debug events can occur only if the address of the instruction fetch is less than the value specified in IAC5 or is greater than or equal to the value specified in IAC6. IAC6 debug events do not occur. IAC5US and IAC5ER settings are used.
10:15	—	Reserved

Table 1829. DBCR5 field descriptions (continued)

Bit	Name	Description
16:17	IAC7US	Instruction Address Compare 7 User/Supervisor Mode 00 IAC7 debug events not affected by MSR_{PR} 01 Reserved 10 IAC7 debug events can only occur if MSR_{PR}=0 (Supervisor mode) 11 IAC7 debug events can only occur if MSR_{PR}=1 (User mode)
18:19	IAC7ER	Instruction Address Compare 7 Effective/Real Mode 00 IAC7 debug events are based on effective address 01 Unimplemented (Book E real address compare), no match can occur 10 IAC7 debug events are based on effective address and can only occur if MSR_{IS}=0 11 IAC7 debug events are based on effective address and can only occur if MSR_{IS}=1
20:21	IAC8US	Instruction Address Compare 8 User/Supervisor Mode 00 IAC8 debug events not affected by MSR_{PR} 01 Reserved 10 IAC8 debug events can only occur if MSR_{PR}=0 (Supervisor mode). 11 IAC8 debug events can only occur if MSR_{PR}=1. (User mode)
22:23	IAC8ER	Instruction Address Compare 8 Effective/Real Mode 00 IAC8 debug events are based on effective address 01 Unimplemented (Book E real address compare), no match can occur 10 IAC8 debug events are based on effective address and can only occur if MSR_{IS}=0 11 IAC8 debug events are based on effective address and can only occur if MSR_{IS}=1
24:25	IAC78M	Instruction Address Compare 7/8 Mode 00 Exact address compare. IAC7 debug events can only occur if the address of the instruction fetch is equal to the value specified in IAC7. IAC8 debug events can only occur if the address of the instruction fetch is equal to the value specified in IAC8. 01 Address bit match. IAC7 debug events can occur only if the address of the instruction fetch, ANDed with the contents of IAC8 are equal to the contents of IAC7, also ANDed with the contents of IAC8. IAC8 debug events do not occur. IAC7US and IAC7ER settings are used. 10 Inclusive address range compare. IAC7 debug events can occur only if the address of the instruction fetch is greater than or equal to the value specified in IAC7 and less than the value specified in IAC8. IAC8 debug events do not occur. IAC7US and IAC7ER settings are used. 11 Exclusive address range compare. IAC7 debug events can occur only if the address of the instruction fetch is less than the value specified in IAC7 or is greater than or equal to the value specified in IAC8. IAC8 debug events do not occur. IAC7US and IAC7ER settings are used.
26:31	—	Reserved

70.3.2.6 Debug Control Register 6 (DBCR6)

Debug Control Register 6 is used to extend instruction address compare matching functionality. DBCR6 is shown in *Figure 1897*.

Figure 1897. DBCR6 register

IAC1XM	IAC2XM	IAC3XM	IAC4XM	IAC5XM	IAC6XM	IAC7XM	IAC8XM

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

SPR - 603; Read/Write; Reset(1) - 0x0

1. DBCR6 is reset by processor reset **p_reset_b** if $EDBCR0_{EDM}$=0, as well as unconditionally by **m_por**. If $EDBCR0_{EDM}$=1, EDBRAC0 masks off hardware-owned resources from reset by **p_reset_b** and only software-owned resources indicated by EDBRAC0 will be reset by **p_reset_b**.

Table 1830 provides bit definitions for Debug Control Register 6.

Table 1830. DBCR6 field descriptions

Bit	Name	Description
0:3	IAC1XM	Instruction Address Compare 1 Extended Mask Control. IAC1XM allows for binary power of 2 address range compares for IAC1 without requiring the use of IAC2. 0000 No additional masking when DBCR1[IAC12M] = 00 0001–1100 Exact Match Bit Mask. Number of low order bits masked in IAC1 when comparing the storage address with the value in IAC1 for exact address compare (DBCR1[IAC12M] = 00). Ranges up to 4 KB are supported. 1101–1111 = Reserved
4:7	IAC2XM	Instruction Address Compare 2 Extended Mask Control. IAC2XM allows for binary power of 2 address range compares for IAC2 without requiring the use of IAC1. 0000 No additional masking when DBCR1[IAC12M] = 00 0001–1100 Exact Match Bit Mask. Number of low order bits masked in IAC2 when comparing the storage address with the value in IAC2 for exact address compare (DBCR1[IAC12M] = 00). Ranges up to 4 KB are supported. 1101–1111 Reserved
8:11	IAC3XM	Instruction Address Compare 3 Extended Mask Control. IAC3XM allows for binary power of 2 address range compares for IAC1 without requiring the use of IAC2. 0000 No additional masking when DBCR1[IAC34M] = 00 0001–1100 Exact Match Bit Mask. Number of low order bits masked in IAC3 when comparing the storage address with the value in IAC3 for exact address compare (DBCR1[IAC34M] = 00). Ranges up to 4 KB are supported. 1101–1111 Reserved
12:15	IAC4XM	Instruction Address Compare 4 Extended Mask Control. IAC4XM allows for binary power of 2 address range compares for IAC4 without requiring the use of IAC3. 0000 No additional masking when DBCR1[IAC34M] = 00 0001–1100 Exact Match Bit Mask. Number of low order bits masked in IAC4 when comparing the storage address with the value in IAC4 for exact address compare (DBCR1[IAC34M] = 00). Ranges up to 4 KB are supported. 1101–1111 Reserved
16:19	IAC5XM	Instruction Address Compare 5 Extended Mask Control. IAC5XM allows for binary power of 2 address range compares for IAC5 without requiring the use of IAC6. 0000 No additional masking when DBCR5[IAC56M] = 00 0001–1100 Exact Match Bit Mask. Number of low order bits masked in IAC5 when comparing the storage address with the value in IAC5 for exact address compare (DBCR5[IAC56M] = 00). Ranges up to 4 KB are supported. 1101–1111 Reserved
20:23	IAC6XM	Instruction Address Compare 6 Extended Mask Control. IAC6XM allows for binary power of 2 address range compares for IAC6 without requiring the use of IAC5. 0000 No additional masking when DBCR5[IAC56M] = 00 0001–1100 Exact Match Bit Mask. Number of low order bits masked in IAC6 when comparing the storage address with the value in IAC6 for exact address compare (DBCR5[IAC56M] = 00). Ranges up to 4 KB are supported. 1101–1111 Reserved

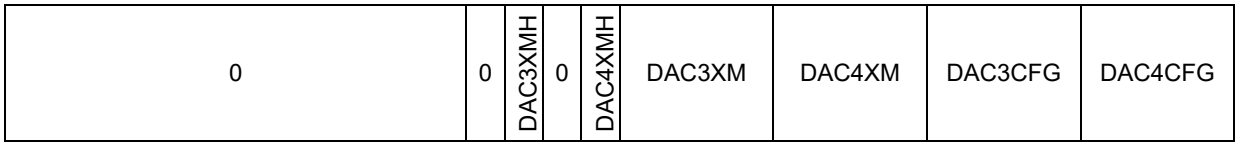
Table 1830. DBCR6 field descriptions (continued)

Bit	Name	Description
24:27	IAC7XM	Instruction Address Compare 7 Extended Mask Control. IAC7XM allows for binary power of 2 address range compares for IAC7 without requiring the use of IAC8. 0000 No additional masking when DBCR5[IAC78M] = 00 0001–1100 Exact Match Bit Mask. Number of low order bits masked in IAC7 when comparing the storage address with the value in IAC7 for exact address compare (DBCR5[IAC78M] = 00). Ranges up to 4 KB are supported. 1101 – 1111 = Reserved
28:31	IAC8XM	Instruction Address Compare 8 Extended Mask Control. IAC8XM allows for binary power of 2 address range compares for IAC8 without requiring the use of IAC7. 0000 No additional masking when DBCR5[IAC78M] = 00 0001–1100 Exact Match Bit Mask. Number of low order bits masked in IAC8 when comparing the storage address with the value in IAC8 for exact address compare (DBCR5[IAC78M] = 00). Ranges up to 4 KB are supported. 1101–1111 Reserved

70.3.2.7 Debug Control Register 7 (DBCR7)

Debug Control Register 7 is used to enable and configure Data Address Compare 3 and 4 functionality. DBCR7 is shown in *Figure 1898*.

Figure 1898. DBCR7 register



SPR - 596; Read/Write; Reset(1) - 0x0

1. DBCR7 is reset by processor reset **p_reset_b** if $EDBCR0_{EDM}$=0, as well as unconditionally by **m_por**. If $EDBCR0_{EDM}$=1, EDBRAC0 masks off hardware-owned resources from reset by **p_reset_b** and only software-owned resources indicated by EDBRAC0 will be reset by **p_reset_b**.

Table 1831 provides bit definitions for Debug Control Register 7.

Table 1831. DBCR7 field descriptions

Bit	Name	Description
0:12	—	Reserved
13	DAC3XMH	Data Address Compare 3 Extended Mask Control High. DAC3XMH extends the range of the DAC3XM field. 0 DAC3XM masks 0–15 low-order address bits 1 DAC3XM masks 16–31 low-order address bits
14	—	Reserved
15	DAC4XMH	Data Address Compare 4 Extended Mask Control High. DAC4XMH extends the range of the DAC4XM field. 0 DAC4XM masks 0–15 low-order address bits 1 DAC4XM masks 16–31 low-order address bits

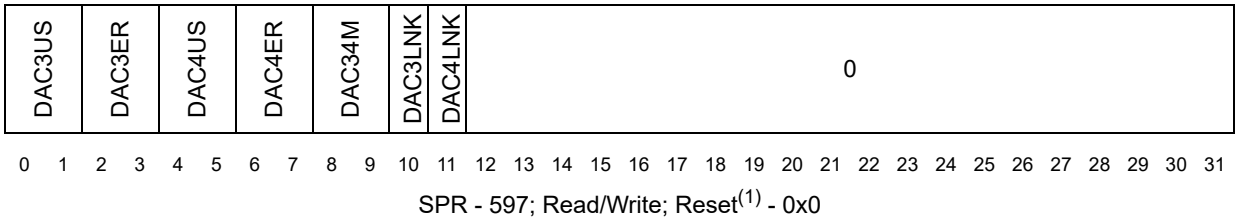
Table 1831. DBCR7 field descriptions (continued)

Bit	Name	Description
16:19	DAC3XM	Data Address Compare 3 Extended Mask Control. DAC3XM allows for binary power of 2 address range compares for DAC3 without requiring the use of DAC4. Value of DAC3XMH || DAC3XM: 00000 No additional masking when DBCR8[DAC34M] = 00 00001–11111 Exact Match Bit Mask. One to 31 low-order bits are masked in DAC3 when comparing the storage address with the value in DAC3 for exact address compare (DBCR8[DAC34M] = 00). Address ranges of 2 bytes to 2GB are supported.
20:23	DAC4XM	Data Address Compare 4 Extended Mask Control. DAC4XM allows for binary power of 2 address range compares for DAC4 without requiring the use of DAC3. Value of DAC4XMH || DAC4XM: 00000 No additional masking when DBCR8[DAC34M] = 00 00001–11111 Exact Match Bit Mask. One to 31 low-order bits are masked in DAC4 when comparing the storage address with the value in DAC4 for exact address compare (DBCR8[DAC34M] = 00). Address ranges of 2 bytes to 2GB are supported.
24:27	DAC3CFG	Data Address Compare 3 Configuration. DAC3CFG controls whether DAC3 data address comparisons utilize the normal compare operation for generating both debug events and watchpoints, a watchpoint-only function, or a stack limit checking function (with watchpoint). 0000 DAC3 debug watchpoints are enabled for load-type or store-type data storage accesses 0001 DAC3 debug watchpoints are enabled only for store-type data storage accesses 0010 DAC3 debug watchpoints are enabled only for load-type data storage accesses 0011 Reserved 0100 Reserved 0101 DAC3 debug events and watchpoints are enabled only for store-type data storage accesses 0110 DAC3 debug events and watchpoints are enabled only for load-type data storage accesses 0111 DAC3 debug events and watchpoints are enabled for load-type or store-type data storage accesses 1000 DAC3 address comparisons are used for stack limit checking. DAC3 address comparisons are qualified with use of GPR R1 in <EA> calculation for most load or store instructions. No debug events occur for DAC3. When a qualified match occurs, a machine check exception is generated for supervisor-mode accesses or a DSI is generated for user-mode accesses, and a DAC3 watchpoints is generated. 1001–1111 Reserved
28:31	DAC4CFG	Data Address Compare 4 Configuration. DAC4CFG controls whether DAC4 data address comparisons utilize the normal compare operation for generating both debug events and watchpoints, or a watchpoint-only function. 0000 DAC4 debug watchpoints are enabled for load-type or store-type data storage accesses 0001 DAC4 debug watchpoints are enabled only for store-type data storage accesses 0010 DAC4 debug watchpoints are enabled only for load-type data storage accesses 0011 Reserved 0100 Reserved 0101 DAC4 debug events and watchpoints are enabled only for store-type data storage accesses 0110 DAC4 debug events and watchpoints are enabled only for load-type data storage accesses 0111 DAC4 debug events and watchpoints are enabled for load-type or store-type data storage accesses 1xxx Reserved

70.3.2.8 Debug Control Register 8 (DBCR8)

Debug Control Register 8 is used to configure Data Address Compare 3 and 4 operation. The DBCR8 register is shown in *Figure 1899*.

Figure 1899. DBCR8 register



SPR - 597; Read/Write; Reset(1) - 0x0

1. Reset by processor reset **p_reset_b** if $EDBCR0_{EDM}$=0, as well as unconditionally by **m_por**. If $EDBCR0_{EDM}$=1, EDBRAC0 masks off hardware-owned resources from reset by **p_reset_b** and only software-owned resources indicated by EDBRAC0 will be reset by **p_reset_b**.

Table 1832 provides bit definitions for Debug Control Register 8.

Table 1832. DBCR8 field descriptions

Bit	Name	Description
0:1	DAC3US	Data Address Compare 3 User/Supervisor Mode 00 DAC3 debug events not affected by MSR_{PR} 01 Reserved 10 DAC3 debug events can only occur if MSR_{PR}=0 (Supervisor mode) 11 DAC3 debug events can only occur if MSR_{PR}=1 (User mode)
2:3	DAC3ER	Data Address Compare 3 Effective/Real Mode 00 DAC3 debug events are based on effective address 01 Unimplemented (Book E real address compare), no match can occur 10 DAC3 debug events are based on effective address and can only occur if MSR_{DS}=0 11 DAC3 debug events are based on effective address and can only occur if MSR_{DS}=1
4:5	DAC4US	Data Address Compare 4 User/Supervisor Mode. 00 DAC4 debug events not affected by MSR_{PR} 01 Reserved 10 DAC4 debug events can only occur if MSR_{PR}=0 (Supervisor mode) 11 DAC4 debug events can only occur if MSR_{PR}=1 (User mode)
6:7	DAC4ER	Data Address Compare 4 Effective/Real Mode 00 DAC4 debug events are based on effective address 01 Unimplemented (Book E real address compare), no match can occur 10 DAC4 debug events are based on effective address and can only occur if MSR_{DS}=0 11 DAC4 debug events are based on effective address and can only occur if MSR_{DS}=1

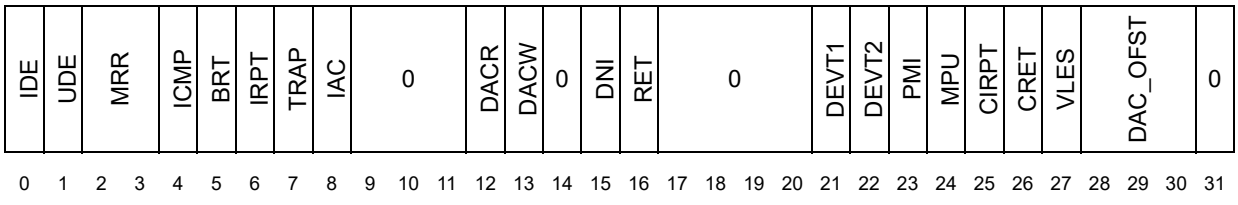
Table 1832. DBCR8 field descriptions (continued)

Bit	Name	Description
8:9	DAC34M	Data Address Compare 3/4 Mode 00 Exact address compare. DAC3 debug events can only occur if the address of the data access is equal to the value specified in DAC3. DAC4 debug events can only occur if the address of the data access is equal to the value specified in DAC4. 01 Address bit match. DAC3 debug events can occur only if the address of the data access ANDed with the contents of DAC4, are equal to the contents of DAC3 also ANDed with the contents of DAC4. DAC4 debug events do not occur. DAC3US and DAC3ER settings are used. 10 Inclusive address range compare. DAC3 debug events can occur only if the address of the data access is greater than or equal to the value specified in DAC3 and less than the value specified in DAC4. DAC4 debug events do not occur. DAC3US and DAC3ER settings are used. 11 Exclusive address range compare. DAC3 debug events can occur only if the address of the data access is less than the value specified in DAC3 or is greater than or equal to the value specified in DAC4. DAC4 debug events do not occur. DAC3US and DAC3ER settings are used.
10	DAC3LNK	Data Address Compare 3 Linked 0 No effect 1 DAC3 debug events are linked to IAC5 debug events. IAC5 debug events do not affect DBSR When linked to IAC5, DAC3 debug events are conditioned based on whether the instruction also generated an IAC5 debug event. Note that linking is only available in EDM or IDM.
11	DAC4LNK	Data Address Compare 4 Linked 0 No effect 1 DAC4 debug events are linked to IAC7 debug events. IAC7 debug events do not affect DBSR When linked to IAC7, DAC4 debug events are conditioned based on whether the instruction also generated an IAC7 debug event. Note that linking is only available in EDM or IDM.
12:31	—	Reserved

70.3.2.9 Debug Status Register (DBSR)

The Debug Status Register (DBSR) contains status on debug events and the most recent processor reset. The Debug Status Register is set via hardware, and read and cleared via software. Bits in the Debug Status Register can be cleared using **mtspr** *DBSR,RS*. Clearing is done by writing to the Debug Status Register with a 1 in any bit position that is to be cleared and 0 in all other bit positions. The write data to the Debug Status Register is not direct data, but a mask. A '1' causes the bit to be cleared, and a '0' has no effect. Debug Status bits are set by Debug events only while Internal Debug mode is enabled ($DBCR0_{IDM}$=1). When debug interrupts are enabled (MSR_{DE}=1 $DBCR0_{IDM}$=1 and $EDBCR0_{EDM}$=0, or MSR_{DE}=1, $DBCR0_{IDM}$=1, $EDBCR0_{EDM}$=1 and software is allocated resource(s) via EDBRAC0), a set bit in DBSR other than MRR, DAC_OFST, or VLES will cause a debug interrupt to be generated. The debug interrupt handler is responsible for clearing DBSR bits prior to returning to normal execution. When resource sharing is enabled, ($EDBCR0_{EDM}$=1 and $EDBRAC0_{IDM}$=1), only software-owned resources may be modified by software, and status bits associated with hardware-owned resources will not be set by hardware in DBSR. The DBSR register is shown in *Figure 1900*.

Figure 1900. Debug Status Register (DBSR)



SPR - 304; Read/Clear; Reset - 0x1000_0000

Table 1833 provides bit definitions for the Debug Status Register.

Table 1833. DBSR field descriptions

Bit	Name	Description
0	IDE	Imprecise Debug Event Set to 1 if MSR_{DE}=0 and $DBCR0_{IDM}$=1 and a debug event causes its respective Debug Status Register bit to be set to 1. It may also be set to '1' if an imprecise debug event occurs due to a DAC event on a load or store that is terminated with error, or if an ICMP event occurs in conjunction with a EFPU FP round exception.
1	UDE	Unconditional Debug Event Set to 1 if an Unconditional debug event occurred.
2:3	MRR	Most Recent Reset. 00 No reset occurred since these bits were last cleared by software 01 A hard reset occurred since these bits were last cleared by software 10 Reserved 11 Reserved
4	ICMP	Instruction Complete Debug Event Set to 1 if an Instruction Complete debug event occurred.
5	BRT	Branch Taken Debug Event Set to 1 if an Branch Taken debug event occurred.
6	IRPT	Interrupt Taken Debug Event Set to 1 if an Interrupt Taken debug event occurred.
7	TRAP	Trap Taken Debug Event Set to 1 if a Trap Taken debug event occurred.
8	IAC	Instruction Address Compare Debug Event Set to 1 if an IAC debug event occurred.
9:11	—	Reserved
12	DACR	Data Address Compare Read Debug Event Set to 1 if a read-type DAC debug event occurred
13	DACW	Data Address Compare Write Debug Event Set to 1 if a write-type DAC debug event occurred
14	—	Reserved
15	DNI	DNI Debug Event Set to 1 if a DNI debug event occurred

Table 1833. DBSR field descriptions (continued)

Bit	Name	Description
16	RET	Return Debug Event Set to 1 if a Return debug event occurred
17:20	—	Reserved
21	DEVT1	External Debug Event 1 Debug Event Set to 1 if a DEVT1 debug event occurred
22	DEVT2	External Debug Event 2 Debug Event Set to 1 if a DEVT2 debug event occurred
23	PMI	Performance Monitor Interrupt Debug Event Set to 1 if a Performance Monitor Interrupt event occurred with $PMGC0_{UDI}$=1
24	MPU	Memory Protection Unit Debug Event Set to 1 if a MPU debug event occurred. For MPU debug events, the IAC, DACR, or DACW status bit will also be set, depending on the type of access that matched a region descriptor enabled to generate debug events, in order to further define the type of MPU debug event. Note that software MPU debug events on data accesses normally occur after the data access is performed and the instruction completes, thus act like a normal DAC debug event. The instruction may not complete if a DSI occurs. In this case, IDE will be set to indicate this occurrence.
25	CIRPT	Critical Interrupt Taken Debug Event Set to 1 if a Critical Interrupt Taken debug event occurred.
26	CRET	Critical Return Debug Event Set to 1 if a Critical Return debug event occurred
27	VLES	VLE Status Set to 1 if an ICMP, BRT, TRAP, RET, CRET, IAC, or DAC debug event occurred on a PowerISA VLE Instruction. Undefined for IRPT, CIRPT, DEVT[1,2], and UDE events
28:30	DAC_OFST	Data Address Compare Offset Indicates offset-1 of saved DSRR0 value from the address of the load or store instruction that took a DAC Debug exception, unless a simultaneous DSI error occurs, in which case this field is set to 3‘b000 and $DBSR_{IDE}$ is set to 1. Normally set to 3‘b000 by a non-DVC DAC. A DVC DAC may set this field to any value.
31	—	Reserved

70.3.2.10 Debug Data Effective Address Register (DDEAR)

The Debug Data Effective Address Register (DDEAR) contains address information for data address compare debug events (DAC or MPU DAC). DDEAR is update by hardware with the effective address of the load or store operation when a data address compare event is recorded in DBSR if the previous values of the $DBSR_{DAC\{R,W\}}$ bits are zero. Once a DDEAR update is performed, these bits must be cleared by software prior to another DDEAR hardware update occurring. A subsequent DAC or MPU DAC event will not update the DDEAR register if either of the $DBSR_{DAC\{R,W\}}$ bits are set, in order to capture the first event address.

The DDEAR register is shown in *Figure 1901*.

Figure 1901. Debug Data Effective Address Register (DDEAR)

Data Effective Address

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

SPR - 600; Read/Write; Reset - unaffected

70.3.3 External Debug Resource Allocation Control (EDBRAC0) register

The External Debug Resource Allocation Control Register (EDBRAC0) controls resource allocation when $EDBCR0_{EDM}$ is set to '1'. EDBRAC0 provides a mechanism for the hardware debugger to share certain debug resources with software. Individual resources are allocated based on the settings of EDBRAC0 when $EDBCR0_{EDM}$=1. EDBRAC0 settings are ignored when $EDBCR0_{EDM}$=0.

Hardware-owned resources that generate debug events update EDBSR0 instead of DBSR and cause entry into debug mode if the event is not masked in EDBSRMSK0, while software-owned resources that generate debug events if $DBCR0_{IDM}$=1 update DBSR, causing debug interrupts to occur if MSR_{DE}=1. EDBRAC0 is controlled via the OnCE port hardware, and is read-only to software.

The DBSR status register is always owned by software. Debug status bits in DBSR are set by software-owned debug events only while Internal Debug mode is enabled. When debug interrupts are enabled (MSR_{DE}=1 $DBCR0_{IDM}$=1 and $EDBCR0_{EDM}$=0, or MSR_{DE}=1, $DBCR0_{IDM}$=1 and $EDBCR0_{EDM}$=1 and software is allocated resource(s) via EDBRAC0), a set bit in DBSR by an event that is software-owned (other than MRR, DAC_OFST, or VLES) will cause a debug interrupt to be generated.

Debug status bits in EDBSR0 are set by hardware-owned debug events only while External Debug mode is enabled ($EDBCR0_{EDM}$=1). When $EDBCR0_{EDM}$=1, a set bit in EDBSR0 by an event that is hardware-owned (other than IDE, DAC_OFST, or VLES) will cause entry into debug mode unless entry is masked via EDBSRMSK0.

If $EDBCR0_{EDM}$=1, DBSR status bits corresponding to hardware-owned debug events are masked from being set by hardware.

Software-owned resources may be modified by software, but only the corresponding control bits in DBCR0–8 or MPU0CSR0 are affected by execution of a **mtspr**, thus only a portion of these registers may be affected, depending on the allocation settings in EDBRAC0. The debug interrupt handler is still responsible for clearing DBSR bits for software-owned resources prior to returning to normal execution. Hardware always has full access to all registers and register fields through the OnCE register access mechanism, and it is up to the debug firmware to properly implement modifications to these registers with read-modify-write operations to implement any control sharing with software. Settings in EDBRAC0 should be considered by the debug firmware in order to preserve software settings of control and status registers as appropriate when hardware modifications to the debug registers is performed.

The EDBRAC0 register is shown in *Figure 1902*.

Figure 1902. External Debug Resource Allocation Control (EDBRAC0) register

0	IDM	RST	UDE	ICMP	BRT	IRPT	TRAP	IAC1	IAC2	IAC3	IAC4	DAC1	DAC34	DAC2	0	RET	IAC5	IAC6	IAC7	IAC8	DEVT1	DEVT2	PMI	MPU	CIRPT	CRET	DNI	DQM	0
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31

SPR - 638; Read-only by Software; Reset - Unaffected by **p_reset_b**, reset to 0x00000180 by **m_por** or while in the test-logic-reset OnCE controller state

Table 1833 provides bit definitions for the Debug External Resource Control Register. Note that EDBRAC0 controls are disabled when EDBCR0EDM=0.

Table 1834. EDBRAC0 field descriptions

Bit	Name	Description
0	—	Reserved
1	IDM	Internal Debug Mode control. When $EDBRAC0_{IDM}$=1, software writes to hardware-owned bits in DBCR0–8 via **mtspr** are ignored. 0 Internal Debug mode may not be enabled by software. $DBCR0_{IDM}$ is owned exclusively by hardware. **mtspr** DBCR0–8 and other debug registers is always ignored. No resource sharing occurs, regardless of the settings of other fields in EDBRAC0. Hardware exclusively owns all resources. 1 Internal Debug mode may be enabled by software. $DBCR0_{IDM}$ is owned by software. $DBCR0_{IDM}$ is software readable/writable.
2	RST	Reset Field Control 0 $DBCR0_{RST}$ owned exclusively by hardware debug. No **mtspr** access by software to $DBCR0_{RST}$ field. 1 $DBCR0_{RST}$ accessible by software debug. $DBCR0_{RST}$ is software readable/writable.
3	UDE	Unconditional Debug Event 0 Event owned by hardware debug. 1 Event owned by software debug.
4	ICMP	Instruction Complete Debug Event 0 Event owned by hardware debug. No **mtspr** access by software to $DBCR0_{ICMP}$ field. 1 Event owned by software debug. $DBCR0_{ICMP}$ is software readable/writable.
5	BRT	Branch Taken Debug Event 0 Event owned by hardware debug. No **mtspr** access by software to $DBCR0_{BRT}$ field. 1 Event owned by software debug. $DBCR0_{BRT}$ is software readable/writable.
6	IRPT	Interrupt Taken Debug Event 0 Event owned by hardware debug. No **mtspr** access by software to $DBCR0_{IRPT}$ field. 1 Event owned by software debug. $DBCR0_{IRPT}$ is software readable/writable.
7	TRAP	Trap Taken Debug Event 0 Event owned by hardware debug. No **mtspr** access by software to $DBCR0_{TRAP}$ field. 1 Event owned by software debug. $DBCR0_{TRAP}$ is software readable/writable.
8	IAC1	Instruction Address Compare 1 Debug Event 0 Event owned by hardware debug. No **mtspr** access by software to IAC1 control and status fields. 1 Event owned by software debug. IAC1 control fields are software readable/writable.

Table 1834. EDBRAC0 field descriptions (continued)

Bit	Name	Description
9	IAC2	Instruction Address Compare 2 Debug Event 0 Event owned by hardware debug. No **mtspr** access by software to IAC2 control and status fields. 1 Event owned by software debug. IAC2 control fields are software readable/writable.
10	IAC3	Instruction Address Compare 3 Debug Event 0 Event owned by hardware debug. No **mtspr** access by software to IAC3 control and status fields. 1 Event owned by software debug. IAC3 control fields are software readable/writable.
11	IAC4	Instruction Address Compare 4 Debug Event 0 Event owned by hardware debug. No **mtspr** access by software to IAC4 control and status fields. 1 Event owned by software debug. IAC4 control fields are software readable/writable.
12	DAC1	Data Address Compare 1 Debug Event 0 Event owned by hardware debug. No **mtspr** access by software to DAC1 control and status fields. 1 Event owned by software debug. DAC1 control fields are software readable/writable.
13	DAC34	Data Address Compare 3 and 4 Debug Events 0 Events owned by hardware debug. No **mtspr** access by software to DAC3 and DAC4 control and status fields. 1 Event owned by software debug. DAC3 and DAC4 control fields are software readable/writable.
14	DAC2	Data Address Compare 2 Debug Event 0 Event owned by hardware debug. No **mtspr** access by software to DAC2 control and status fields. 1 Event owned by software debug. DAC2 control fields are software readable/writable.
15	—	Reserved
16	RET	Return Debug Event 0 Event owned by hardware debug. No **mtspr** access by software to $DBCR0_{RET}$ field. 1 Event owned by software debug. $DBCR0_{RET}$ is software readable/writable.
17	IAC5	Instruction Address Compare 5 Debug Event 0 Event owned by hardware debug. No **mtspr** access by software to IAC5 control and status fields. 1 Event owned by software debug. IAC5 control fields are software readable/writable.
18	IAC6	Instruction Address Compare 6 Debug Event 0 Event owned by hardware debug. No **mtspr** access by software to IAC6 control and status fields. 1 Event owned by software debug. IAC6 control fields are software readable/writable.
19	IAC7	Instruction Address Compare 7 Debug Event 0 Event owned by hardware debug. No **mtspr** access by software to IAC7 control and status fields. 1 Event owned by software debug. IAC7 control fields are software readable/writable.
20	IAC8	Instruction Address Compare 8 Debug Event 0 Event owned by hardware debug. No **mtspr** access by software to IAC8 control and status fields. 1 Event owned by software debug. IAC8 control are software readable/writable.

Table 1834. EDBRAC0 field descriptions (continued)

Bit	Name	Description
21	DEVT1	External Debug Event Input 1 Debug Event 0 Event owned by hardware debug. No **mtspr** access by software to $DBCR0_{DEVT1}$ field. 1 Event owned by software debug. $DBCR0_{DEVT1}$ is software readable/writable.
22	DEVT2	External Debug Event Input 2 Debug Event 0 Event owned by hardware debug. No **mtspr** access by software to $DBCR0_{DEVT2}$ field. 1 Event owned by software debug. $DBCR0_{DEVT2}$ is software readable/writable.
23	PMI	Performance Monitor Interrupt Debug Event 0 Event owned by hardware debug. No **mtpmr** access by software to the PMRs. Performance monitor interrupts set $EDBSR0_{PMI}$ regardless of the setting of $PMGC0_{UDI}$ 1 Event owned by software debug. PMRs are software readable/writable. **Note:** This bit is reset to '1'.
24	MPU	Memory Protection Unit Debug Event 0 Event owned by hardware debug. No **mpuwe** access by software to region descriptors that have the DEBUG control bit set to '1' or to the $MPU0CSR0_{DRDEN,DWDEN,IDEN}$ control bits, unless the CPU is in a debug session (**jd_debug_b** is asserted). MPU debug events set $EDBSR0_{MPU}$, and if not masked by $EDBSRMSK0_{MPU}$, one of $EDBSR0_{IAC}$, $EDBSR0_{DACR}$, or $EDBSR0_{DACW}$. MPU flash invalidates do not affect DEBUG=1 entries unless the CPU is in a debug session (**jd_debug_b** is asserted). 1 Event owned by software debug. All region descriptors and the $MPU0CSR0_{DRDEN,DWDEN,IDEN}$ control bits are software readable/writable. **Note:** This bit is reset to '1'.
25	CIRPT	Critical Interrupt Taken Debug Event 0 Event owned by hardware debug. No **mtspr** access by software to $DBCR0_{CIRPT}$ field. 1 Event owned by software debug. $DBCR0_{CIRPT}$ is software readable/writable.
26	CRET	Critical Return Debug Event 0 Event owned by hardware debug. No **mtspr** access by software to $DBCR0_{CRET}$ field. 1 Event owned by software debug. $DBCR0_{CRET}$ is software readable/writable.
27	DNI	DNI Instruction Debug Control 0 DNI resource owned by hardware debug. Execution of an **e_dni** or **se_dni** instruction results in entry into debug mode. 1 DNI resource owned by software debug. Execution of an **e_dni** or **se_dni** instruction results in either a debug interrupt ($DBCR0_{IDM}$=1 and MSR_{DE}=1) or a nop ($DBCR0_{IDM}$=0 or MSR_{DE}=0). **Note:** DNI events are not blocked during a debug session
28	DQM	Data Acquisition Messaging Registers 0 $DEVENT_{DQTAG}$ and DDAM register are exclusively owned by hardware debug. No **mtspr** access by software to $DEVENT_{DQTAG}$ field or DDAM register. Attempted access by software is ignored. 1 $DEVENT_{DQTAG}$ and DDAM register are owned by software. Software has read/write access to $DEVENT_{DQTAG}$ field and DDAM register.
29:31	—	Reserved

Table 1835 shows which resources are controlled by EDBRAC0 settings.

Table 1835. EDBRAC0 resource control

$EDBCR0_{EDM}$	$EDBRAC0_{IDM}$	$EDBRAC0_{RST}$	$EDBRAC0_{UDE}$	$EDBRAC0_{ICMP}$	$EDBRAC0_{BRT}$	$EDBRAC0_{IRPT}$	$EDBRAC0_{TRAP}$	$EDBRAC0_{IAC1}$	$EDBRAC0_{IAC2}$	$EDBRAC0_{IAC3}$	$EDBRAC0_{IAC4}$	$EDBRAC0_{IAC5}$	$EDBRAC0_{IAC6}$	$EDBRAC0_{IAC7}$	$EDBRAC0_{IAC8}$	$EDBRAC0_{DAC1}$	$EDBRAC0_{DAC34}$	$EDBRAC0_{DAC2}$	$EDBRAC0_{RET}$	$EDBRAC0_{DEVT1}$	$EDBRAC0_{DEVT2}$	$EDBRAC0_{PMI}$	$EDBRAC0_{MPU}$	$EDBRAC0_{CIRT}$	$EDBRAC0_{CRET}$	$EDBRAC0_{DNI}$	$EDBRAC0_{DCM}$	Software Accessible via mtspr, affected by p_reset_b
0	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	All debug registers
1	1	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	$DBCR0_{IDM}$
1	1	1	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	$DBCR0_{RST}$
1	1	—	1	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	$DBCR0_{UDE}$
1	1	—	—	1	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	$DBCR0_{ICMP}$
1	1	—	—	—	1	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	$DBCR0_{BRT}$
1	1	—	—	—	—	1	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	$DBCR0_{IRPT}$
1	1	—	—	—	—	—	1	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	$DBCR0_{TRAP}$
1	1	—	—	—	—	—	—	1	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	IAC1, $DBCR0_{IAC1}$, $DBCR1_{IAC1US,IAC1ER}$, $DBCR6_{IAC1XM}$
1	1	—	—	—	—	—	—	—	1	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	IAC2, $DBCR0_{IAC2}$, $DBCR1_{IAC2US,IAC2ER}$, $DBCR6_{IAC2XM}$
1	1	—	—	—	—	—	—	—	—	1	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	$DBCR1_{IAC12M}$

Table 1835. EDBRAC0 resource control (continued)

$EDBCR0_{EDM}$	$EDBRAC0_{IDM}$	$EDBRAC0_{RST}$	$EDBRAC0_{UDE}$	$EDBRAC0_{ICMP}$	$EDBRAC0_{BRT}$	$EDBRAC0_{IRPT}$	$EDBRAC0_{TRAP}$	$EDBRAC0_{IAC1}$	$EDBRAC0_{IAC2}$	$EDBRAC0_{IAC3}$	$EDBRAC0_{IAC4}$	$EDBRAC0_{IAC5}$	$EDBRAC0_{IAC6}$	$EDBRAC0_{IAC7}$	$EDBRAC0_{IAC8}$	$EDBRAC0_{DAC1}$	$EDBRAC0_{DAC34}$	$EDBRAC0_{DAC2}$	$EDBRAC0_{RET}$	$EDBRAC0_{DEVT1}$	$EDBRAC0_{DEVT2}$	$EDBRAC0_{PMI}$	$EDBRAC0_{MPU}$	$EDBRAC0_{CIRT}$	$EDBRAC0_{CRET}$	$EDBRAC0_{DNI}$	$EDBRAC0_{DCM}$	Software Accessible via mtspr, affected by p_reset_b
1	1	—	—	—	—	—	—	—	—	1	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	IAC3, $DBCR0_{IAC3}$, $DBCR1_{IAC3US,IAC3ER}$, $DBCR6_{IAC3XM}$
1	1	—	—	—	—	—	—	—	—	—	1	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	IAC4, $DBCR0_{IAC4}$, $DBCR1_{IAC4US,IAC4ER}$, $DBCR6_{IAC4XM}$
1	1	—	—	—	—	—	—	—	—	1	1	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	$DBCR1_{IAC34M}$
1	1	—	—	—	—	—	—	—	—	—	—	1	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	IAC5, $DBCR0_{IAC5}$, $DBCR5_{IAC5US,IAC5ER}$, $DBCR6_{IAC5XM}$
1	1	—	—	—	—	—	—	—	—	—	—	—	1	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	IAC6, $DBCR0_{IAC6}$, $DBCR5_{IAC6US,IAC6ER}$, $DBCR6_{IAC6XM}$
1	1	—	—	—	—	—	—	—	—	—	—	1	1	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	$DBCR5_{IAC56M}$
1	1	—	—	—	—	—	—	—	—	—	—	—	—	1	—	—	—	—	—	—	—	—	—	—	—	—	—	—	IAC7, $DBCR0_{IAC7}$, $DBCR5_{IAC7US,IAC7ER}$, $DBCR6_{IAC7XM}$
1	1	—	—	—	—	—	—	—	—	—	—	—	—	—	1	—	—	—	—	—	—	—	—	—	—	—	—	—	IAC8, $DBCR0_{IAC8}$, $DBCR5_{IAC8US,IAC8ER}$, $DBCR6_{IAC8XM}$
1	1	—	—	—	—	—	—	—	—	—	—	—	—	1	1	—	—	—	—	—	—	—	—	—	—	—	—	—	$DBCR5_{IAC78M}$

Table 1835. EDBRAC0 resource control (continued)

$EDBCR0_{EDM}$	$EDBRAC0_{IDM}$	$EDBRAC0_{RST}$	$EDBRAC0_{UDE}$	$EDBRAC0_{ICMP}$	$EDBRAC0_{BRT}$	$EDBRAC0_{IRPT}$	$EDBRAC0_{TRAP}$	$EDBRAC0_{IAC1}$	$EDBRAC0_{IAC2}$	$EDBRAC0_{IAC3}$	$EDBRAC0_{IAC4}$	$EDBRAC0_{IAC5}$	$EDBRAC0_{IAC6}$	$EDBRAC0_{IAC7}$	$EDBRAC0_{IAC8}$	$EDBRAC0_{DAC1}$	$EDBRAC0_{DAC34}$	$EDBRAC0_{DAC2}$	$EDBRAC0_{RET}$	$EDBRAC0_{DEVT1}$	$EDBRAC0_{DEVT2}$	$EDBRAC0_{PMI}$	$EDBRAC0_{MPU}$	$EDBRAC0_{CIRT}$	$EDBRAC0_{CRET}$	$EDBRAC0_{DNI}$	$EDBRAC0_{DCM}$	Software Accessible via mtspr, affected by p_reset_b
1	1	—	—	—	—	—	—	—	—	—	—	—	—	—	—	1	—	—	—	—	—	—	—	—	—	—	—	—	DAC1, DVC1, DVC1U $DBCR0_{DAC1}$, $DBCR2_{DAC1US,DAC1ER}$, $DBCR2_{DVC1M,DVC1BE}$ $DBCR4_{DVC1C,DAC1XM}$
1	1	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	1	—	—	—	—	—	—	—	—	—	—	—	DAC3, DAC4 $DBCR7_{DAC\{3,4\},}$ $_{DAC\{3,4\}CFG,}$ $_{DAC\{3,4\}XM,}$ $_{DAC\{3,4\}XMH}$ $DBCR8_{DAC\{3,4\}US,}$ $_{DAC\{3,4\}ER, DAC\{3,4\}M}$
1	1	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	1	—	—	—	—	—	—	—	—	—	—	DAC2, DVC2, DVC2U $DBCR0_{DAC2}$, $DBCR2_{DAC2US,DAC2ER}$, $DBCR2_{DVC2M,DVC2BE}$ $DBCR4_{DVC2C,DAC2XM}$
1	1	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	
1	1	—	—	—	—	—	—	—	—	—	—	—	—	—	—	1	—	1	—	—	—	—	—	—	—	—	—	—	$DBCR2_{DAC12M}$
1	1	—	—	—	—	—	—	1	—	—	—	—	—	—	—	1	—	—	—	—	—	—	—	—	—	—	—	—	$DBCR2_{DAC1LNK}$
1	1	—	—	—	—	—	—	—	—	1	—	—	—	—	—	—	—	1	—	—	—	—	—	—	—	—	—	—	$DBCR2_{DAC2LNK}$
1	1	—	—	—	—	—	—	—	—	—	—	1	—	—	—	—	1	—	—	—	—	—	—	—	—	—	—	—	$DBCR8_{DAC3LNK}$
1	1	—	—	—	—	—	—	—	—	—	—	—	1	—	—	—	1	—	—	—	—	—	—	—	—	—	—	—	$DBCR8_{DAC4LNK}$

Table 1835. EDBRAC0 resource control (continued)

$EDBCR0_{EDM}$	$EDBRAC0_{IDM}$	$EDBRAC0_{RST}$	$EDBRAC0_{UDE}$	$EDBRAC0_{ICMP}$	$EDBRAC0_{BRT}$	$EDBRAC0_{IRPT}$	$EDBRAC0_{TRAP}$	$EDBRAC0_{IAC1}$	$EDBRAC0_{IAC2}$	$EDBRAC0_{IAC3}$	$EDBRAC0_{IAC4}$	$EDBRAC0_{IAC5}$	$EDBRAC0_{IAC6}$	$EDBRAC0_{IAC7}$	$EDBRAC0_{IAC8}$	$EDBRAC0_{DAC1}$	$EDBRAC0_{DAC34}$	$EDBRAC0_{DAC2}$	$EDBRAC0_{RET}$	$EDBRAC0_{DEVT1}$	$EDBRAC0_{DEVT2}$	$EDBRAC0_{PMI}$	$EDBRAC0_{MPU}$	$EDBRAC0_{CIRT}$	$EDBRAC0_{CRET}$	$EDBRAC0_{DNI}$	$EDBRAC0_{DQM}$	Software Accessible via mtspr, affected by p_reset_b
1	1	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	1	—	—	—	—	—	—	—	—	DBCR0$_{RET}$
1	1	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	1	—	—	—	—	—	—	—	DBCR0$_{DEVT1}$
1	1	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	1	—	—	—	—	—	—	DBCR0$_{DEVT2}$
1	1	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	1	—	—	—	—	—	All PMRs
1	1	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	1	—	—	—	—	MPU entries with DEBUG bit set, MPU0CSR0$_{DRDEN,}$ DWDEN,IDEN
1	1	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	1	—	—	—	DBCR0$_{CIRPT}$
1	1	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	1	—	—	DBCR0$_{CRET}$
1	1	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	1	—	
1	— (1)	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	1	DEVENT$_{DQTAG}$, DDAM

1. IDM not required to be set to enable software access.

EDBRAC0 also controls which bits or fields in DBCR0–8, MPU0CSR0, and the PMRs are reset by assertion of **p_reset_b** when $EDBCR0_{EDM}$=1. Only software-owned bits or fields as shown in *Table 1835* are affected in this case, except that $DBCR0_{RST}$ and $DBSR_{MRR}$ are updated by assertion of **p_reset_b** regardless of the value of $EDBCR0_{EDM}$ or EDBRAC0.

70.3.4 Debug Event Select register (DEVENT) register

The Debug Event Select Register allows instrumented software to internally generate signals when a **mtspr** instruction is executed and this register is accessed. The values written to this register determine which of the **p_devnt_out[0:7]** processor output signals are asserted upon access. Writing a '1' to any of these bit positions will cause a one clock pulse to be generated on the corresponding output. For **p_devnt_out[0:3]**, a corresponding **jd_watchpt[x]** output is asserted as well to indicate a watchpoint has occurred. These signals may be used for internal core debug resources as well as for SoC level cross-triggering.

The $DEVENT_{DEVNT}$ register field value is undefined on a read; it may or may not remain set to the last value written. Since it is unconditionally shared by hardware debug and software, software should not rely on any value remaining.

The upper 8 bits of the DEVENT register also provide the DQTAG used to identify channels within Data Acquisition Messages. See the "Data Acquisition ID Tag Field" section in the Core (e200z425n3) Nexus 3 Module chapter for more detail on the DQTAG.

The DEVENT register is shown in *Figure 1903*.

Figure 1903. Debug Event Select register (DEVENT) register

DQTAG	0	DEVNT

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

SPR 975; Reset(1): 0x0

1. Reset by processor reset **p_reset_b** if $EDBCR0_{EDM}$=0, as well as unconditionally by **m_por**. If $EDBCR0_{EDM}$=1, EDBRAC0 masks off hardware-owned resources from reset by **p_reset_b** and only software-owned resources indicated by EDBRAC0 will be reset by **p_reset_b.** Note that DEVNT field is shared by hardware and software but is always reset by **p_reset_b**.

Table 1836 provides bit definitions for the Debug Event Register.

Table 1836. DEVENT field descriptions

Bit	Name	Description
0:7	DQTAG	Data Acquisition Message IDTAG channel identifier (supplied to Nexus 3)
8:23	—	Reserved, should be cleared.
24:31	DEVNT	Debug Event Signals 00000000 No signal is asserted *xxxxxxx*1 **p_devnt_out[0]** and **jd_watchpt[12]** are asserted for one clock *xxxxxx*1*x* **p_devnt_out[1]** and **jd_watchpt[13]** are asserted for one clock *xxxxx*1*xx* **p_devnt_out[2]** and **jd_watchpt[20]** are asserted for one clock *xxxx*1*xxx* **p_devnt_out[3]** and **jd_watchpt[21]** are asserted for one clock *xxx*1*xxxx* **p_devnt_out[4]** is asserted for one clock *xx*1*xxxxx* **p_devnt_out[5]** is asserted for one clock *x*1*xxxxxx* **p_devnt_out[6]** is asserted for one clock 1*xxxxxxx* **p_devnt_out[7]** is asserted for one clock

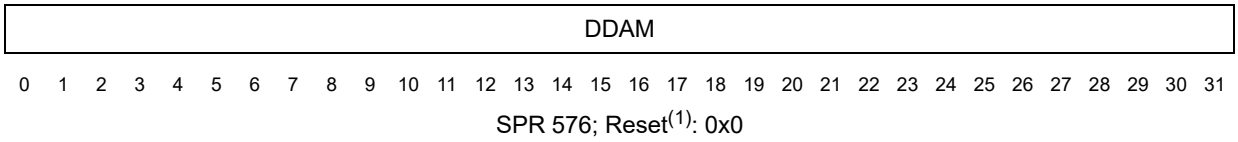
70.3.5 Debug Data Acquisition Message (DDAM) register

The Debug Data Acquisition Message Register allows instrumented software to generate real-time Data Acquisition Messages (as defined by Nexus 3) via a **mtspr** instruction to this

register. See the “Data Acquisition Messaging” section in the Core (e200z425n3) Nexus 3 Module chapter for details.

The DDAM register is shown in *Figure 1904*.

Figure 1904. Debug Data Acquisition Message (DDAM) register



1. Reset by processor reset **p_reset_b** if $EDBCR0_{EDM}$=0, as well as unconditionally by **m_por**. If $EDBCR0_{EDM}$=1, EDBRAC0 masks off hardware-owned resources from reset by **p_reset_b** and only software-owned resources indicated by EDBRAC0 will be reset by **p_reset_b**.

Table 1837 provides bit definitions for the Debug Data Acquisition Message Register.

Table 1837. DDAM field descriptions

Bit	Name	Description
0:31	DDAM	Value to be transmitted in a Data Acquisition Message (DQM) (supplied to Nexus 3 with strobe)

70.4 Using debug resources for stack limit checking

The DAC1,2 and DAC3,4 resources can be used for stack overflow/underflow detection when not being used as a hardware or software debug resource. Stack limit checking is available regardless of EDM or IDM mode, and when resources used for stack limit checking are owned by software, will utilize a DSI or machine check exception. Software-owned stack limit checking does not require IDM to be set. Hardware owned stack limit checking requires EDM to be set.

When stack limit checking is enabled, and DAC resources used for stack limit checking are owned by software, DAC events are not generated for resources configured to perform stack limit checking, and no DBSR DAC status flag will be set due to a detected stack limit violation. Instead, depending on the processor mode, a data storage interrupt or a machine check exception is signaled. When stack limit checking is enabled, and DAC resources used for stack limit checking are owned by hardware, DAC events will be generated for resources configured to perform stack limit checking, and the EDBSR0 DAC status flag will be set due to a detected stack limit violation, causing entry into debug halted mode in the same way as a DAC exception normally does. The only difference is that qualification of the access address is performed as discussed in the next paragraph.

Stack limit checking is implemented in the same way as range compares using DAC1,2 or 3,4, or extended masking using DAC1 or DAC3, but qualify a load or store access address with the use of GPR R1 as the base or index register used to compute an effective address when a load or store instruction is executed. No stack limit checking is performed for other instructions that may calculate an EA using GPR R1 such as cache, MMU/MPU management instructions, or instructions that indicate GPR R1 is used to hold a decoration value (for decorated load/store type instructions) or a modify specifier (for LSP load/store instructions w/modify). When DAC resources configured to perform stack limit checking are not owned by hardware, if a stack limit violation occurs when performing the load or store, the access is aborted, and an error report machine check is generated, with MCSRR0

pointing to the address of the load or store access that generated the stack overflow/underflow. If DAC resources configured to perform stack limit checking are owned by hardware, then a normal DAC event is generated (but qualified with use of GPR R1), and debug mode entry will occur in the same manner as for a non-stack limit DAC event.

Enabling of this functionality is described in more detail in *Table 1828*. Note that IDM is not required to be set for software-owned stack limit checking.

In contrast to the normal case of DAC address matching resulting in a debug event, the stack limit check address compare logic operates differently for stack limit checking. When using resources for stack limit checking, a DACn hit to a resource enabled for performing stack limit checking on a stack access indicates a stack limit violation.

When stack limit checking is enabled by the setting of the $DBCR4_{DAC1CFG}$ field to '1000' or the $DBCR8_{DAC3CFG}$ field to '1000' (or both), and the corresponding DACn resources are owned by software, DACn events are not generated and the DBSR DAC status flag will not be set due to a detected stack limit hit. Instead, if stack limit checking is enabled for supervisor mode stack accesses in DAC1 or DAC3, and a compare hit occurs for a supervisor mode stack access (A load or store using GPR R1 in the <EA> calculation), a machine check exception is signaled. If stack limit checking is enabled for user mode accesses, a DSI exception is signaled when a stack limit checking enabled DAC1 or DAC3 compare hit occurs for a user mode access. A watchpoint for DAC1 or DAC3 will be generated when the stack limit violation is detected, even though the instruction does not complete. Stack limit checking for supervisor mode stack accesses is considered enabled when either the $DBCR4_{DAC1CFG}$ field is set to '1000' with $DBCR2_{DAC1US}$ set to '00' or '10', or the $DBCR7_{DAC3CFG}$ field is set to '1000' with $DBCR8_{DAC3US}$ set to '00' or '10', or both. Stack limit checking for user mode stack accesses is considered enabled when either the $DBCR4_{DAC1CFG}$ field is set to '1000' with $DBCR2_{DAC1US}$ set to '00' or '01', or the $DBCR7_{DAC3CFG}$ field is set to '1000' with $DBCR8_{DAC3US}$ set to '00' or '01', or both. Note that unlike regular debug DAC events, both halves of a misaligned access are checked for limit violations.

When stack limit checking is enabled for a stack access, and DACn resources are owned by hardware, the EDBSR0 DAC status flag will be set due to a detected stack limit violation, to cause entry into debug halted mode or to generate a watchpoint, or both, in the same way as a DAC event normally does, that is after the access has completed. The only difference is that qualification of the access address is performed as discussed in the next paragraph. If the access is aborted due to a DSI or other exception such as machine check condition, the $EDBSR0_{IDE}$ status bit will also be set to indicate that the data access instruction was not completed.

Stack limit checking is implemented in the same way as address compares using DACn, but qualify a load or store access address with the use of GPR R1 as the base or index register used to compute an effective address when a load or store instruction is executed. No stack limit checking is performed for other instructions that may calculate an EA using GPR R1 such as cache, MMU/MPU management instructions, or instructions that indicate GPR R1 is used to hold a decoration value (for decorated load/store type instructions) or a modify specifier (for LSP load/store instructions w/modify). When DACn resources are not owned by hardware, if a stack limit violation occurs (a hit to a stack limit enabled DAC is detected) when performing the load or store, the access request is aborted and the access is not performed. Instead, for supervisor mode accesses, an error report machine check exception is generated, with MCSRR0 pointing to the address of the load or store instruction that generated the stack overflow/underflow, and for user mode accesses, a DSI exception is generated, with SRR0 pointing to the address of the load or store instruction that generated the stack overflow/underflow. If all stack limit check enabled DACn resources are owned by

hardware, then a DAC event is generated (but qualified with use of GPR R1), and debug mode entry will occur in the same manner as for a non-stack limit DAC event. In this case, the instruction and access will normally have completed, in the same manner as a normal DAC event.

Independent limit checks for supervisor and user accesses may be implemented by allocating independent DACn resources to each, or a single limit may be applied using a single DACn resource. Typically, a DAC1,2 pair operating in exclusive address range mode is utilized for stack limit checking for a single shared limit. For separate limits, DAC3,4 may also be utilized. If more than one DACn resource is utilized, a DAC hit on any resource utilized for stack limit checking will cause the corresponding stack limit exception action to occur. If both a hardware-owned and a software-owned resource generate a stack limit exception for a given load or store, the software resource will have priority, since it is detected prior to completion of the access, and the access is aborted, thus the hardware event will not occur.

Enabling of this functionality is described in more detail in the description of the DACnCFG fields in *Table 1828* and *Table 1831* in *Section 70.3.2: Debug Control and Status registers*. Note that for DAC1 and DAC2, access type (read, write) control is part of DBCR0.

70.5 External debug support

External debug support is supplied through the OnCE controller serial interface, which allows access to internal CPU registers and other system state while the CPU is halted in debug mode. All debug resources including DBCR0–8, DBSR, IAC1–8, DAC1–4, and DVC1–2 are accessible through the serial OnCE interface in external debug mode. Setting the $EDBCR0_{EDM}$/$DBCR0_{EDM}$ bit to '1' through the OnCE interface enables external debug mode, and unless otherwise permitted by the settings in EDBRAC0, disables software updates to the debug control registers. When $EDBCR0_{EDM}$ is set, debug events enabled to set respective status bits will also cause the CPU to enter Debug mode if the event is not masked in EDBSRMSK0, as opposed to generating Debug Interrupts, unless the specific events are allocated to software via the settings in EDBRAC0. In Debug mode, the CPU is halted at a recoverable boundary, and an external Debug Control Module may control CPU operation through the On-Chip Emulation logic (OnCE).

Note that the descriptions of events in the subsections of *Section 70.2: Software debug events and exceptions* refer to setting DBSR status bits, however, when resources are owned by hardware, the events for those resources set the respective status bits in EDBSR0 instead of DBSR.

Note: On the initial setting of $EDBCR0_{EDM}$ to '1', other bits in DBCR0 will remain unchanged. After $EDBCR0_{EDM}$ has been set, all debug register resources may be subsequently controlled through the OnCE interface. The CPU should be placed into debug mode via the OCR_{DR} control bit prior to writing EDM to '1'. This gives the debugger the opportunity to cleanly write to the DBCRx registers and the DBSR to clear out any residual state / control information that could cause unintended operation.

*It is intended for the CPU to remain in external debug mode ($EDBCR0_{EDM}$=1) in order to single step or perform other debug mode entry/ reentry via the OCR_{DR}, by performing go+noexit commands, or by assertion of the **jd_de_b** signal.*

*$EDBCR0_{EDM}$ operation will be blocked if OnCE operation is disabled (**jd_en_once** negated) regardless of whether it is set or cleared. This means that if $EDBCR0_{EDM}$ was previously set, and then **jd_en_once** is negated (this should not occur), entry into debug mode will be blocked, and all hardware debug events are blocked. Watchpoints are not blocked.*

Due to clock domain design, the CPU clock (**m_clk**) must be active in order to perform writes to debug registers other than the OnCE Command register (OCMD), the OnCE Control register (OCR), External Debug Control Register 0 (EDBCR0), External Debug Status register 0 (EDBSR0), External Debug Status Register Mask 0 (EDBSRMSK0), or the $EDBCR0_{EDM}$ bit. Register read data is synchronized back to the **j_tclk** clock domain. The OnCE Control register provides the capability of signaling the system level clock controller that the CPU clock should be activated if not already active.

Updates to the DBCRx, DBSR, and most other debug registers via the OnCE interface should be performed with the CPU in debug mode to guarantee proper operation. Due to the various points in the CPU pipeline where control is sampled and event handshaking is performed, it is possible that modifications to these registers while the CPU is running may result in early or late entry into debug mode, and may have incorrect status posted in the DBSR register.

If resource sharing is enabled via EDBRAC0, updates to the EDBRAC0, DBCRx, and DBSR registers must be performed with the CPU in debug mode, since simultaneous updates of register portions could otherwise be attempted, and such updates are not guaranteed to properly occur. The results of such an attempt are undefined.

70.5.1 External debug registers

The external debug registers are used for controlling several debug aspects of the core and reporting status while in External Debug mode.

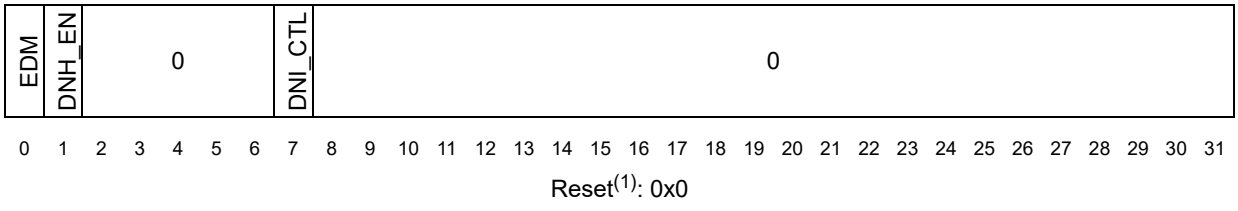
70.5.1.1 External Debug Control Register 0 (EDBCR0)

EDBCR0 is a control register accessible to an external debugger through the OnCE/Jtag port. An external development tool can write to this register in order to enable external debug mode or to enable Debugger Notify Halt instructions (**e_dnh**, **se_dnh**), as well as to control aspects of Debugger Notify Interrupt instructions (**e_dni**, **se_dni**).

EDBCR0 is not accessible by software, However, the state of $EDBCR0_{EDM}$ is reflected as a read-only bit in $DBCR0_{EDM}$ to software. There is only one physical EDM bit implemented; it is reflected in both the DBCR0 and EDBCR0 registers, and may be written and read using either register by the hardware debugger. For future compatibility, EDBCR0 updates are preferred.

EDBCR0 is shown in *Figure 1905*.

Figure 1905. External Debug Control Register 0 (EDBCR0)



1. EDBCR0 is affected (reset) by **j_trst_b** or **m_por** assertion, and remains reset while in the Test_Logic_Reset state, but is not affected by **p_reset_b**.

Table 1838 provides bit definitions for External Debug Control Register 0.

Table 1838. EDBCR0 field descriptions

Bit	Name	Description
0	EDM	External Debug Mode. This bit is also reflected in DBCR0. When external debug mode is enabled, hardware-owned resources in debug registers are not affected by processor reset **p_reset_b**. This allows the debugger to set up hardware debug events that remain active across a processor reset. 0 External debug mode disabled. Internal debug events not mapped into external debug events. 1 External debug mode enabled. Hardware-owned events will not cause the CPU to vector to interrupt code. Software is not permitted to write to debug registers {DBCRx, IAC1–8, DAC1–4, DVC1–2[U]} unless permitted by settings in EDBRAC0.
1	DNH_EN	**dnh** Instruction Enable 0 Execution of **e_dnh** and **se_dnh** instructions cause illegal instruction exceptions to occur. 1 Execution of **e_dnh** and **se_dnh** instructions cause entry into debug mode and a debug halt occurs, regardless of the value of EDM.
2:6	—	Reserved
7	DNI_CTL	**dni** Instruction Control 0 When the **dni** resource is owned by hardware, the MSR_{DE} bit is cared, and when MSR_{DE}=0, execution of **e_dni** and **se_dni** instructions are nop'ed and no entry into debug mode occurs. 1 When the **dni** resource is owned by hardware, the MSR_{DE} bit is don't-cared, and execution of **e_dni** and **se_dni** instructions cause entry into debug mode and a debug halt occurs, regardless of the value of MSR_{DE}. **Note:** This control bit is only used when the dni resource is owned by hardware via control in $EDBRAC0_{DNI}$, and thus also only when EDM=1.
8:31	—	Reserved

70.5.1.2 External Debug Status Register 0 (EDBSR0)

The External Debug Status Register 0 (EDBSR0) contains status on debug events owned by hardware. The External Debug Status Register 0 is set via hardware, and read and cleared via OnCE access by the debugger. Clearing is done by writing to the External Debug Status Register via the OnCE port, with a '1' in any bit position that is to be cleared and '0' in all other bit positions. The write data to EDBSR0 is not direct data, but a mask. A '1' causes the bit to be cleared, and a '0' has no effect. The EDBSR0 register is shown in *Figure 1906*.

Figure 1906. External Debug Status Register 0 (EDBSR0)

IDE	UDE	DNH	0	ICMP	BRT	IRPT	TRAP	IAC	0	DACR	DACW	0	DNI	RET	0	DEVT1	DEVT2	PMI	MPU	CIRPT	CRET	VLES	DAC_OFST	0
0	1	2	3	4	5	6	7	8	9 10 11	12	13	14	15	16	17 18 19 20	21	22	23	24	25	26	27	28 29 30	31

Read/Write; Reset(1) - 0x0000_0000

1. Reset by **j_trst_b** or **m_por** assertion, and remains reset while in the Test_Logic_Reset state or while $EDBCR0_{EDM}$=0.

Table 1839 provides bit definitions for External Debug Status Register 0.

Table 1839. EDBSR0 field descriptions

Bit	Name	Description
0	IDE	Imprecise Debug Event Set to 1 if $EDBCR0_{EDM}$=1 and an imprecise debug event occurs for a hardware-owned DAC event due to a load or store that is terminated with error, or if a hardware-owned ICMP event occurs in conjunction with a EFPU round exception. This bit will not be set for imprecise debug events that are masked via settings in EDBSRMSK0.
1	UDE	Unconditional Debug Event Set to 1 if a hardware-owned Unconditional debug event occurred.
2	DNH	Debugger Notify Halt Event Set to 1 if a debugger notify halt instruction was executed and caused a debug halt.
3	—	Reserved
4	ICMP	Instruction Complete Debug Event Set to 1 if a hardware-owned Instruction Complete debug event occurred.
5	BRT	Branch Taken Debug Event Set to 1 if a hardware-owned Branch Taken debug event occurred.
6	IRPT	Interrupt Taken Debug Event Set to 1 if a hardware-owned Interrupt Taken debug event occurred.
7	TRAP	Trap Taken Debug Event Set to 1 if a hardware-owned Trap Taken debug event occurred.
8	IAC	Instruction Address Compare 1 Debug Event Set to 1 if a hardware-owned IAC debug event occurred.
9:11	—	Reserved
12	DACR	Data Address Compare Read Debug Event Set to 1 if a hardware-owned read-type DAC debug event occurred
13	DACW	Data Address Compare Write Debug Event Set to 1 if a hardware-owned write-type DAC1 debug event occurred
14	—	Reserved
15	DNI	DNI Debug Event Set to 1 if a hardware-owned DNI debug event occurred
16	RET	Return Debug Event Set to 1 if a hardware-owned Return debug event occurred
17:20	—	Reserved
21	DEVT1	External Debug Event 1 Debug Event Set to 1 if a hardware-owned DEVT1 debug event occurred
22	DEVT2	External Debug Event 2 Debug Event Set to 1 if a hardware-owned DEVT2 debug event occurred
23	PMI	Performance Monitor Interrupt Debug Event Set to 1 if a Performance Monitor Interrupt event occurred with $EDBRAC0_{PMI}$=0 regardless of the setting of $PMGC0_{UDI}$

Table 1839. EDBSR0 field descriptions (continued)

Bit	Name	Description
24	MPU	Memory Protection Unit Debug Event Set to 1 if a hardware-owned MPU debug event occurred. For MPU debug events, if $EDBSRMSK0_{MPU}$ is cleared, the IAC, DACR, or DACW status bit will also be set, depending on the type of access that matched a region descriptor enabled to generate debug events, in order to further define the type of MPU debug event. If $EDBSRMSK0_{MPU}$ is set at the time a MPU debug event occurs, the IAC, DACR, and DACW status bits will not be set by MPU debug events, in order to properly mask the MPU debug event. Note that hardware MPU debug events on data accesses normally occur after the data access is performed and the instruction completes, thus act like a normal DAC debug event. The instruction may not complete if a DSI occurs. In this case, IDE will be set to indicate this occurrence.
25	CIRPT	Critical Interrupt Taken Debug Event Set to 1 if a hardware-owned Critical Interrupt Taken debug event occurred.
26	CRET	Critical Return Debug Event Set to 1 if a hardware-owned Critical Return debug event occurred
27	VLES	VLE Status Set to 1 if a hardware-owned ICMP, BRT, TRAP, RET, CRET, IAC, or DAC debug event occurred on a PowerISA VLE Instruction. Also set for execution of an e_dnh or se_dnh instruction when enabled by $EDBCR0_{DNH_EN}$. Undefined for IRPT, CIRPT, DEVT[1,2], and UDE events
28:30	DAC_OFST	Data Address Compare Offset Indicates offset-1 of saved DSRR0 value from the address of the load or store instruction that took a hardware-owned DAC Debug exception, unless a simultaneous DSI error occurs, in which case this field is set to 3‘b000 and $EDBSR0_{IDE}$ is set to 1. Normally set to 3‘b000 by a non-DVC DAC. A DVC DAC may set this field to any value.
31	—	Reserved

70.5.1.3 External Debug Status Register Mask 0 (EDBSRMSK0)

The External Debug Status Register Mask 0 (EDBSRMSK0) is used to mask debug events set in EDBSR0 from causing entry into debug halted mode. A ‘1’ stored in any mask bit prevents debug mode entry caused by the corresponding bit being set in EDBSR0. The mask has no effect on DBSR actions or on the setting of EDBSR0 status bits by hardware-owned events, except that the IDE bit will not be set by imprecise hardware-owned debug events that are masked. EDBSRMSK0 may be used to allow debug events owned by hardware to be configured for watchpoint generation purposes without causing debug mode entry when the watchpoint occurs. EDBSRMSK0 is read and written via OnCE access by the debugger. No software access is provided. The EDBSRMSK0 register is shown in *Figure 1907*.

Figure 1907. External Debug Status Register Mask 0 (EDBSRMSK0)

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
0	UDE	DNH	0	ICMP	BRT	IRPT	TRAP	IAC	0			DACR	DACW	0	DNI	RET	0				DEVT1	DEVT2	PMI	MPU	CIRPT	CRET	0				

Read/Write; Reset(1) - 0x0000_0000

1. Reset by **j_trst_b** or **m_por** assertion, and remains reset while in the Test_Logic_Reset state or while $EDBCR0_{EDM}$=0.

Table 1840 provides bit definitions for External Debug Status Register Mask 0.

Table 1840. EDBSRMSK0 field descriptions

Bit	Name	Description
0	—	Reserved
1	UDE	Unconditional Debug Event Set to 1 to mask debug mode entry by $EDBSR0_{UDE}$
2	DNH	Debugger Notify Halt Event Set to 1 to mask debug mode entry by $EDBSR0_{DNH}$
3	—	Reserved
4	ICMP	Instruction Complete Debug Event Set to 1 to mask debug mode entry by $EDBSR0_{ICMP}$
5	BRT	Branch Taken Debug Event Set to 1 to mask debug mode entry by $EDBSR0_{BRT}$
6	IRPT	Interrupt Taken Debug Event Set to 1 to mask debug mode entry by $EDBSR0_{IRPT}$
7	TRAP	Trap Taken Debug Event Set to 1 to mask debug mode entry by $EDBSR0_{TRAP}$
8	IAC	Instruction Address Compare Debug Event Set to 1 to mask debug mode entry by $EDBSR0_{IAC}$
9:11	—	Reserved
12	DACR	Data Address Compare 1 Read Debug Event Set to 1 to mask debug mode entry by $EDBSR0_{DACR}$
13	DACW	Data Address Compare 1 Write Debug Event Set to 1 to mask debug mode entry by $EDBSR0_{DACW}$
14	—	Reserved
15	DNI	DNI Debug Event Set to 1 to mask debug mode entry by $EDBSR0_{DNI}$
16	RET	Return Debug Event Set to 1 to mask debug mode entry by $EDBSR0_{RET}$
17:20	—	Reserved
21	DEVT1	External Debug Event 1 Debug Event Set to 1 to mask debug mode entry by $EDBSR0_{DEVT1}$
22	DEVT2	External Debug Event 2 Debug Event Set to 1 to mask debug mode entry by $EDBSR0_{DEVT2}$
23	PMI	Performance Monitor Interrupt Debug Event Set to 1 to mask debug mode entry by $EDBSR0_{PMI}$
24	MPU	Memory Protection Unit Debug Event Set to 1 to mask debug mode entry by $EDBSR0_{MPU}$

Table 1840. EDBSRMSK0 field descriptions (continued)

Bit	Name	Description
25	CIRPT	Critical Interrupt Taken Debug Event Set to 1 to mask debug mode entry by $EDBSR0_{CIRPT}$
26	CRET	Critical Return Debug Event Set to 1 to mask debug mode entry by $EDBSR0_{CRET}$
22:31	—	Reserved

70.5.1.4 External Debug Data Effective Address Register (EDDEAR)

The External Debug Data Effective Address Register (EDDEAR) contains address information for hardware-owned data address compare debug events, including MPU DAC events. EDDEAR is update by hardware with the effective address of the load or store operation when an unmasked data address compare event is recorded in EDBSR0 if the previous values of $EDBSR0_{DAC\{R,W\}}$ bits that are unmasked in EDBSRMSK0 are zero. Once an EDDEAR update is performed, these unmasked bits must be cleared by the hardware debugger prior to another EDDEAR hardware update occurring. A subsequent hardware-owned unmasked DAC event will not update the EDDEAR register if either of the $EDBSR0_{DAC\{R,W\}}$ bits are set and are not masked by EDBSRMSK0, in order to capture the first unmasked event address. EDDEAR is read and written via OnCE access by the debugger. No software access is provided.

The EDDEAR register is shown in *Figure 1908*.

Figure 1908. External Debug Data Effective Address Register (EDDEAR)

Data Effective Address

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

Read/Write; Reset: unaffected

70.5.2 OnCE introduction

The e200z425n3 on-chip emulation circuitry (OnCE™/Nexus Class 1 interface) provides a means of interacting with the e200z425n3 core and integrated system so that a user may examine registers, memory, or on-chip peripherals facilitating hardware/software development. OnCE operation is controlled via an industry standard IEEE 1149.1 TAP controller. By using public instructions, the external hardware debugger can freeze or halt the CPU, read and write internal state, and resume normal execution. The core does not contain IEEE 1149.1 standard boundary cells on its interface, as it is a building block for further integration. It does not support the JTAG related boundary scan instruction functionality, although JTAG public instructions may be decoded and signaled to external logic.

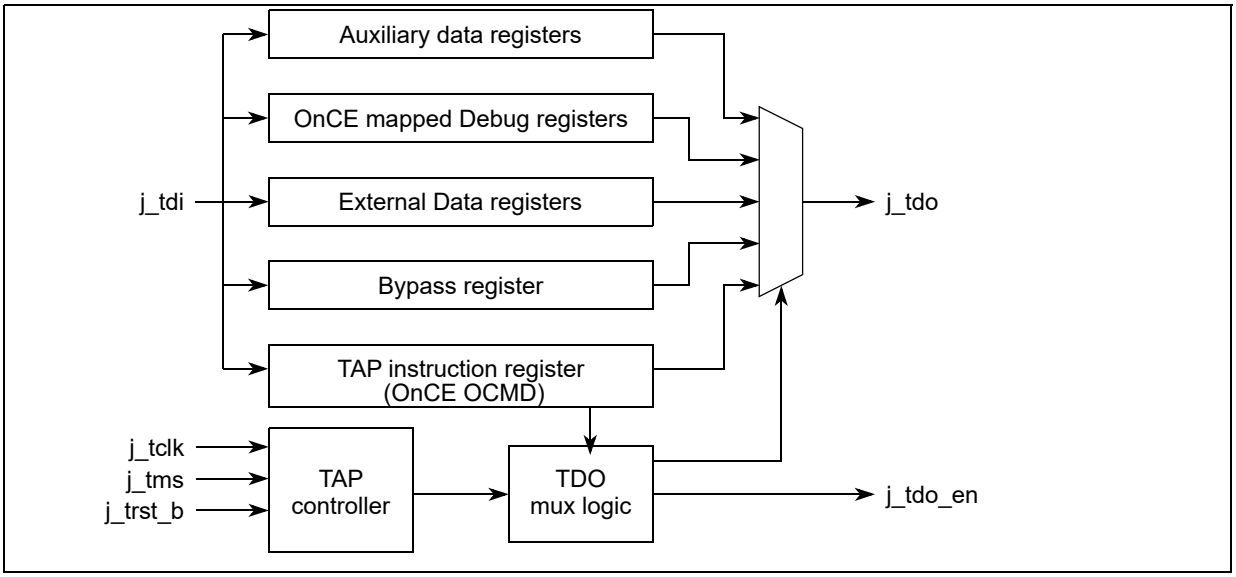
The OnCE logic provides for Nexus Class 1 static debug capability (utilizing the same set of resources available to software while in internal debug mode).

In order to enable full OnCE operation, the **jd_enable_once** input signal must be asserted. In some system integrations, this is automatic, since the input will be tied asserted. Other integrations may require the execution of the Enable OnCE command via the TAP and appropriate entry of serial data. Exact requirements will be documented by the integrated

product specification. The **jd_enable_once** input signal should not change state during a debug session, or undefined activity may occur.

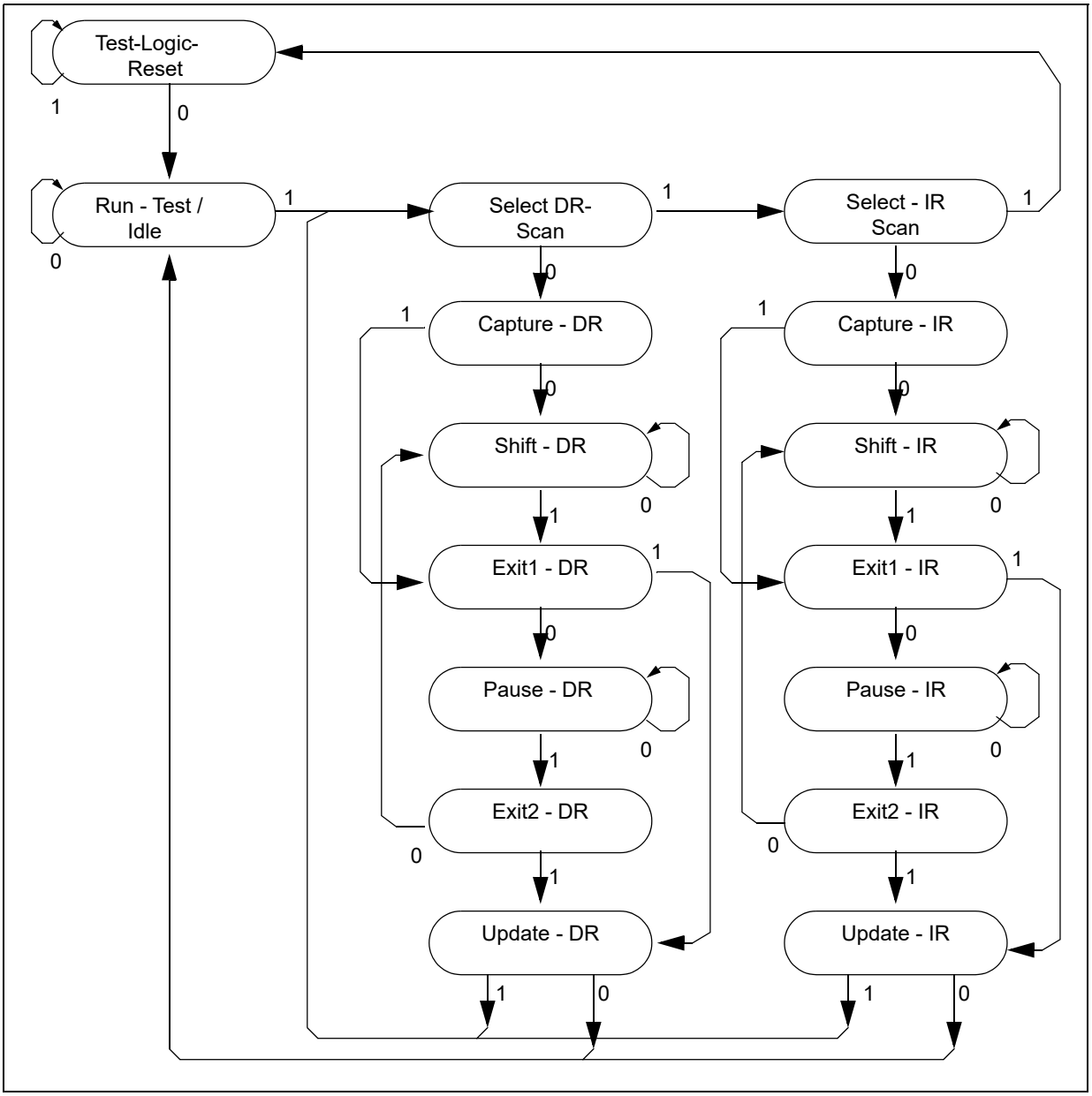
The following figures show the TAP controller state model and the TAP registers implemented by the OnCE logic.

Figure 1909. OnCE TAP controller and registers



The OnCE controller is implemented as a 16-state FSM, with a one-to-one correspondence to the states defined for the JTAG TAP controller.

Figure 1910. OnCE controller implementation



Access to processor registers and the contents of memory locations are performed by enabling external debug mode (setting $EDBCR0_{EDM}$ to '1'), placing the processor into debug mode, followed by scanning instructions and data into and out of the CPU Scan Chain (CPUSCR); execution of scanned instructions by the CPU is used as the method to access required data. Memory locations may be read by scanning a load instruction into the CPU core, which will reference the desired memory location, executing the load instruction, and then scanning out the result of the load. Other resources are accessed in a similar manner.

The initial entry by the CPU into the debug state (or mode) from normal, Waiting, Stopped, or Halted states (all indicated via the OnCE Status Register (OSR), *Section 70.5.6.1: OnCE Status register*) by assertion of one or more debug requests, begins a *debug session*. The **jd_debug_b** output signal indicates that a debug session is in progress, and the OSR will

indicate the CPU is in the debug state. Instructions may the be single-stepped by scanning new values into the CPUSCR, and performing a OnCE go+noexit command (See *Section 70.5.6.2: OnCE Command (OCMD) register*). The CPU will then temporarily exit the debug state (but not the debug session) to execute the instruction, and will then return to the debug state (again indicated via the OnCE Status Register (OSR)). The debug session remains in force until the final OnCE go+exit command is executed, at which time the CPU will return to the previous state it was in (unless a new debug request is pending). A scan into the CPUSCR is required prior to executing each go+exit or go+noexit OnCE command.

70.5.3 JTAG/OnCE pins

The JTAG/OnCE pin interface is used to transfer OnCE instructions and data to the OnCE control block. Depending on the particular resource being accessed, the CPU may need to be placed in the Debug mode. For resources outside of the CPU block and contained in the OnCE block, the processor is not disturbed, and may continue execution. If a processor resource is required, an internal debug request (**dbg_dbgrq**) may be asserted to the CPU by the OnCE controller, and causes the CPU to finish the current instruction being executed, save the instruction pipeline information, enter Debug mode, and wait for further commands. Asserting **dbg_dbgrq** will cause the chip to temporarily exit the Waiting, Stopped or Halted power management states.

Table 1841 details the primary JTAG/OnCE interface signals.

Table 1841. JTAG/OnCE primary interface signals

Signal name	Type	Description
j_trst_b	I	JTAG test reset
j_tclk	I	JTAG test clock
j_tms	I	JTAG test mode select
j_tdi	I	JTAG test data input
j_tdo	O	Test data out to master controller or pad
j_tdo_en(1)	O	Enables TDO output buffer

1. j_tdo_en is asserted when the TAP controller is in the shift_DR or shift_IR state.

A full description of JTAG pins is provided in the JTAG Support Signals section of the Core (e200z425n3) Core Complex Overview.

70.5.4 OnCE internal interface signals

The following paragraphs describe the OnCE interface signals to other internal blocks associated with the OnCE controller.

70.5.4.1 CPU Debug Request (dbg_dbgrq)

The **dbg_dbgrq** signal is asserted by the OnCE control logic to request the CPU to enter the debug state. It may be asserted for a number of different conditions, and causes the CPU to finish the current instruction being executed, save the instruction pipeline information, enter the debug mode, and wait for further commands.

70.5.4.2 CPU Debug Acknowledge (cpu_dbgack)

The **cpu_dbgack** signal is asserted by the CPU upon entering the debug state. This signal is used as part of the handshake mechanism between the OnCE control logic and the rest of the CPU. The CPU core may enter debug mode either through a software or hardware event.

70.5.4.3 CPU address and attributes

The CPU address and attribute information are used by the Nexus3 unit with information for real-time address trace information.

70.5.4.4 CPU data

The CPU data buses are used to supply the Nexus3 debug unit with information for real-time data trace capability.

70.5.5 OnCE interface signals

The following paragraphs describe additional OnCE interface signals to other external blocks such as a Nexus controller and external blocks that may need information pertaining to debug operation.

70.5.5.1 OnCE Enable (jd_en_once)

The OnCE enable signal **jd_en_once** is used to enable the OnCE controller to allow certain instructions and operations to be executed. Assertion of this signal will enable the full OnCE command set, as well as operation of control signals and OnCE Control register functions. When this signal is disabled, only the Bypass, ID and Enable_OnCE commands are executed by the OnCE unit, and all other commands default to a “Bypass” command. The OnCE Status register (OSR) is not visible when OnCE operation is disabled. In addition, OnCE Control register (OCR) functions are disabled, as is the operation of the **jd_de_b** input. Secure systems may choose to leave the **jd_en_once** signal negated until a security check has been performed. Other systems should tie this signal asserted to enable full OnCE operation. The **j_en_once_regsel** output signal is provided to assist external logic performing security checks.

The **jd_en_once** input must only change state during the Test-Logic-Reset, Run-Test/Idle, or Update_DR TAP states. A new value will take affect after one additional **j_tclk** cycle of synchronization. In addition, **jd_enable_once** input signal must not change state during a debug session, or undefined activity may occur.

70.5.5.2 OnCE Debug Request/Event (jd_de_b, jd_de_en)

If implemented at the SoC level, a system level bidirectional open drain debug event pin **DE_b** (not part of the e200z425n3 interface) provides a fast means of entering the Debug mode of operation from an external command controller (when input) as well as a fast means of acknowledging the entering of the Debug mode of operation to an external command controller (when output). The assertion of this pin by a command controller causes the CPU core to finish the current instruction being executed, save the instruction pipeline information, enter Debug mode, and wait for commands to be entered. If **DE_b** was used to enter the Debug mode then **DE_b** must be negated after the OnCE controller responds with an acknowledge and before sending the first OnCE command. The assertion of this pin by the CPU Core acknowledges that it has entered the Debug mode and is waiting for commands to be entered.

To support operation of this system pin, the OnCE logic supplies the **jd_de_en** output and samples the **jd_de_b** input when OnCE is enabled (**jd_en_once** asserted). Assertion of **jd_de_b** will cause the OnCE logic to place the CPU into Debug mode. Once Debug mode has been entered, the **jd_de_en** output will be asserted for three **j_tclk** periods to signal an acknowledge. **jd_de_en** can be used to enable the open-drain pulldown of the system level **DE_b** pin.

For systems that do not implement a system level bidirectional open drain debug event pin **DE_b**, the **jd_de_en** and **jd_de_b** signals may still be used to handshake debug entry.

70.5.5.3 OnCE Debug output (jd_debug_b)

The OnCE Debug output **jd_debug_b** is used to indicate to on-chip resources that a debug session is in progress. Peripherals and other units may use this signal to modify normal operation for the duration of a debug session, which may involve the CPU executing a sequence of instructions solely for the purpose of visibility/system control that are not part of the normal instruction stream the CPU would have executed had it not been placed in debug mode. This signal is asserted the first time the CPU enters the debug state, and remains asserted until the CPU is released by a write to the OnCE Command Register with the GO and EX bits set, and a register specified as either “No Register Selected” or the CPUSCR. This signal will remain asserted even though the CPU may enter and exit the debug state for each instruction executed under control of the OnCE controller. See *Section 70.5.6.2: OnCE Command (OCMD) register* for more information on the function of the GO and EX bits. This signal is not normally used by the CPU.

70.5.5.4 CPU Clock On input (jd_mclk_on)

The CPU Clock On input **jd_mclk_on** is used to indicate that the CPU’s **m_clk** input is active. This input signal is expected to be driven by system logic external to the e200z425n3 core, is synchronized to the **j_tclk** (scan clock) clock domain, and is presented as a status flag on the **j_tdo** output during the Shift_IR state. External firmware may use this signal to ensure proper scan sequences will occur to access debug resources in the **m_clk** clock domain.

70.5.5.5 Watchpoint events (jd_watchpt[0:31])

The **jd_watchpt[0:31]** signals may be asserted by the OnCE control logic to signal that a watchpoint condition has occurred. Watchpoints do not directly cause the CPU to be affected. They are provided to allow external visibility only or for triggering purposes. Watchpoint events are conditioned by the settings in the DBCRxx registers, as well as by the DEVENT register, the DTC/DTSA/DTEA registers, the Performance Monitor control register settings, and generation of MPU debug events. Refer to *Table 1848* for details of the signal assignments. Note that assertion of most watchpoint outputs is conditioned on being in EDM or IDM. The Performance monitor, DEVENT, and DTC watchpoints are not conditioned however, and may assert regardless of the state of EDM or IDM. In addition, DAC1 or DAC3 watchpoints may be generated for stack limit violation occurrences when those resources are configured to perform stack limit checking, regardless of the state of EDM or IDM.

70.5.5.6 Update DR w/go+exit (j_ocmd_go_exit)

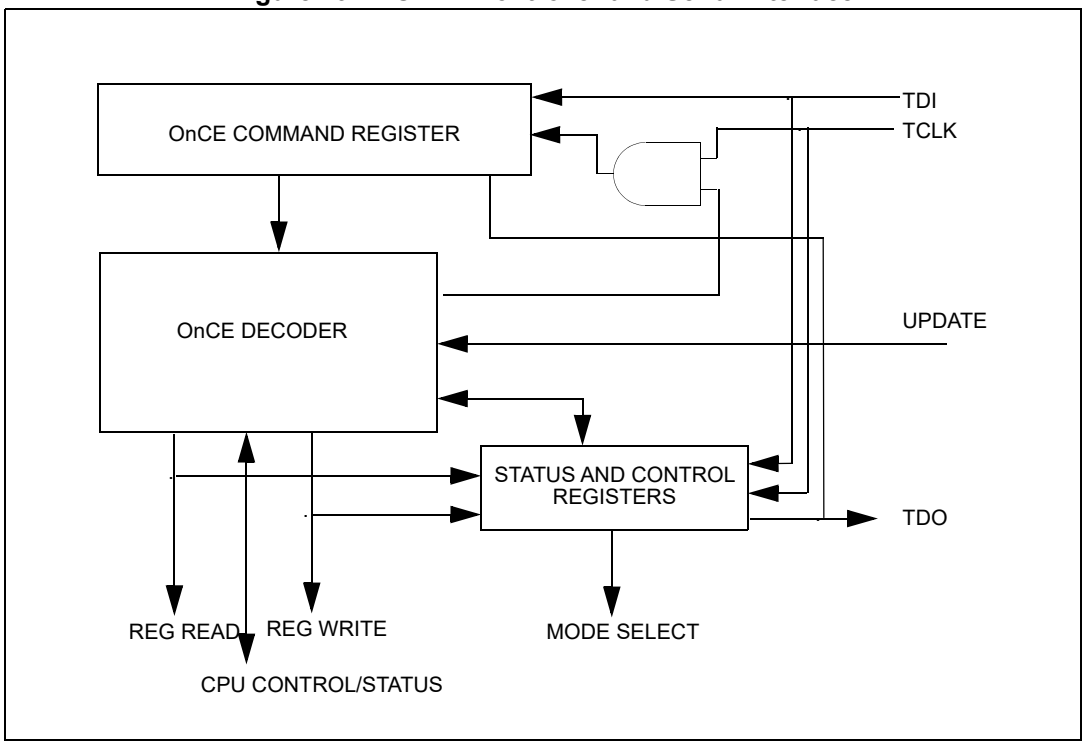
This signal indicates the TAP controller is in the Update_DR state and that the go and exit bits in the OnCE Command register are high, and RS indicates “no register selected”. This signal will assert regardless of whether the CPU is currently in debug mode. It may be

monitored by external logic to cause a synchronous exit from debug mode of other modules (but not other CPUs) in the system. A debugger can place all of the CPU tap controllers in parallel, scan in a go+exit command with “no register selected”, and sequence through the Update_DR state to cause any CPU currently in debug mode to exit. CPUs that are not in debug mode will ignore the command.

70.5.6 OnCE controller and serial interface

The OnCE Controller contains the OnCE command register, the OnCE decoder, and the status/control register. *Figure 1911* is a block diagram of the OnCE controller. In operation, the OnCE Command register acts as the IR for the TAP controller, and all other OnCE resources are treated as data registers (DR) by the TAP controller. The Command register is loaded by serially shifting in commands during the TAP controller Shift-IR state, and is loaded during the Update-IR state. The Command register selects a resource to be accessed as a data register (DR) during the TAP controller Capture-DR, Shift-DR, and Update-DR states.

Figure 1911. OnCE Controller and Serial Interface



70.5.6.1 OnCE Status register

Status information regarding the state of the CPU is latched into the OnCE Status register when the OnCE controller state machine enters the Capture-IR state. When OnCE operation is enabled, this information is provided on the **j_tdo** output in serial fashion when the Shift_IR state is entered following a Capture-IR. Information is shifted out least significant bit first.

Figure 1912. OnCE Status register

MCLK	ERR	0	RESET	HALT	STOP	DEBUG	WAIT	0	1
0	1	2	3	4	5	6	7	8	9

Table 1842 provides bit definitions for the OnCE Status register.

Table 1842. OnCE Status register field descriptions

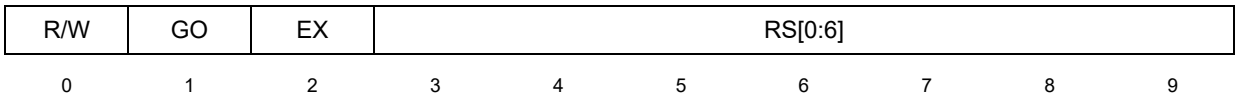
Bit	Name	Description
0	MCLK	MCLK **m_clk** Status Bit 0 Inactive state 1 Active state This status bit reflects the logic level on the **jd_mclk_on** input signal after capture by **j_tclk**.
1	ERR	ERROR This bit is used to indicate that an error condition occurred during attempted execution of the last single-stepped instruction (GO+NoExit with CPUSCR or No Register Selected in OCMD), and that the instruction may not have been properly executed. This could occur if an Interrupt (all classes including such as External, Critical, machine check, Storage, Alignment, Program) occurred while attempting to perform the instruction single step. In this case, the CPUSCR will contain information related to the first instruction of the Interrupt handler, and no portion of the handler will have been executed.
2	—	Reserved, set to zero
3	RESET	RESET Mode This bit reflects the inverted logic level on the CPU **p_reset_b** input after capture by **j_tclk**.
4	HALT	HALT Mode This bit reflects the logic level on the CPU **p_halted** output after capture by **j_tclk**.
5	STOP	STOP Mode This bit reflects the logic level on the CPU **p_stopped** output after capture by **j_tclk**.
6	DEBUG	Debug Mode This bit is asserted once the CPU is in debug mode. It is negated once the CPU exits debug mode (even during a debug session)
7	WAIT	Waiting Mode This bit reflects the logic level on the CPU **p_waiting** output after capture by **j_tclk**.
8	0	Reserved, set to 0 for 1149.1 compliance
9	1	Reserved, set to 1 for 1149.1 compliance

70.5.6.2 OnCE Command (OCMD) register

The OnCE Command (OCMD) register is a 10-bit shift register that receives its serial data from the TDI pin and serves as the instruction register (IR). It holds the 10-bit commands to be used as input for the OnCE Decoder. The Command Register is shown in *Figure 1913*. The OCMD is updated when the TAP controller enters the Update-IR state. It contains fields for controlling access to a resource, as well as controlling single-step operation and exit from OnCE mode.

Although the OCMD is updated during the Update-IR TAP controller state, the corresponding resource is accessed in the DR scan sequence of the TAP controller, and as such, the Update-DR state must be transitioned through in order for an access to occur. In addition, the Update-DR state must also be transitioned through in order for the single-step and/or exit functionality to be performed, even though the command appears to have no data resource requirement associated with it.

Figure 1913. OnCE Command (OCMD) register



Reset - 10'b1000000010 on assertion of **j_trst_b** or **m_por**, or while in the Test_Logic_Reset state

Table 1843 provides bit definitions for the OCMD.

Table 1843. OCMD field descriptions

Bit	Name	Description
0	R/W	Read/Write Command Bit The R/W bit specifies the direction of data transfer. The table below describes the options defined by the R/W bit. **Note:** The R/W bit generally ignored for read-only or write-only registers, although the PC FIFO pointer is only guaranteed to be update when R/W=1. In addition, it is ignored for all bypass operations. When performing writes, most registers are sampled in the Capture-DR state into a 32-bit shift register, and subsequently shifted out on **j_tdo** during the first 32 clocks of Shift-DR. 0 Write the data associated with the command into the register specified by RS[0:6] 1 Read the data contained in the register specified by RS[0:6]
1	GO	Go Command Bit If the GO bit is set, and the CPU is currently in debug mode, the CPU will execute the instruction that resides in the IR register in the CPUSCR. To execute the instruction, the processor leaves the debug mode, executes the instruction, and if the EX bit is cleared, returns to the debug mode immediately after executing the instruction. The processor goes on to normal operation if the EX bit is set, and no other debug request source is asserted. The GO command is executed only if the operation is a read/write to CPUSCR or a read/write to "No Register Selected". Otherwise the GO bit is ignored.The processor will leave the debug mode after the TAP controller Update-DR state is entered. On a GO+NoExit operation, returning to debug mode is treated as a debug event, thus exceptions such as machine checks and interrupts may take priority and prevent execution of the intended instruction. Debug firmware should mask these exceptions as appropriate. The OSR_{ERR} bit indicates such an occurrence. If the CPU is not currently in debug mode, then the GO command will be ignored. **Note:** Asynchronous interrupts are blocked on a GO+Exit operation until the first instruction to be executed begins execution. See *Section 70.5.9.6: Exiting Debug mode and interrupt blocking*. 0 Inactive (no action taken) 1 Execute instruction in IR

Table 1843. OCMD field descriptions (continued)

Bit	Name	Description
2	EX	Exit Command Bit 0 Remain in debug mode 1 Leave debug mode If the EX bit is set, the processor will leave the debug mode and resume normal operation until another debug request is generated. The Exit command is executed only if the Go command is issued, and the operation is a read/write to CPUSCR or a read/write to “No Register Selected”. Otherwise the EX bit is ignored. The processor will leave the debug mode after the TAP controller Update-DR state is entered. **Note:** If the DR bit in the OnCE control register is set or remains set, or if a bit in EDBSR0 is set and $EDBCR0_{EDM}$=1 (external debug mode is enabled), or if another debug request source is asserted, then the processor may return to the debug mode without execution of an instruction, even though the EX bit was set. **Note:** Asynchronous interrupts are blocked on a GO+Exit operation until the first instruction to be executed begins execution. See *Section 70.5.9.6: Exiting Debug mode and interrupt blocking*.
3:9	RS	Register Select The Register Select bits define which register is source (destination) for the read (write) operation. *Table 1844* indicates the OnCE register addresses. Attempted writes to read-only registers are ignored.

Table 1844 indicates the OnCE register addresses.

Table 1844. OnCE register addressing

RS[0:6]	Register Selected
000 0000	Reserved
000 0001	Reserved
000 0010	JTAG ID (read-only)
000 0011–000 1111	Reserved
001 0000	CPU Scan Register (CPUSCR)
001 0001	No Register Selected (Bypass)
001 0010	OnCE Control Register (OCR)
001 0011	Reserved
001 0100–001 0111	Reserved
001 1000	Data Address Compare 3 (DAC3)
001 1001	Data Address Compare 4 (DAC4)
001 1010–001 1111	Reserved
010 0000	Instruction Address Compare 1 (IAC1)
010 0001	Instruction Address Compare 2 (IAC2)
010 0010	Instruction Address Compare 3 (IAC3)
010 0011	Instruction Address Compare 4 (IAC4)

Table 1844. OnCE register addressing (continued)

RS[0:6]	Register Selected
010 0100	Data Address Compare 1 (DAC1)
010 0101	Data Address Compare 2 (DAC2)
010 0110	Data Value Compare 1 (DVC1) — *all 64 bits of the DVC register are accessed
010 0111	Data Value Compare 2 (DVC2) — *all 64 bits of the DVC register are accessed
010 1000	Instruction Address Compare 5 (IAC5)
010 1001	Instruction Address Compare 6 (IAC6)
010 1010	Instruction Address Compare 7 (IAC7)
010 1011	Instruction Address Compare 8 (IAC8)
010 1100	Debug Data Effective Address (DDEAR)
010 1101	External Debug Data Effective Address (EDDEAR)
010 1110	External Debug Control Register 0 (EDBCR0)
010 1111	External Debug Status Register 0 (EDBSR0)
011 0000	Debug Status Register (DBSR)
011 0001	Debug Control Register 0 (DBCR0)
011 0010	Debug Control Register 1 (DBCR1)
011 0011	Debug Control Register 2 (DBCR2)
011 0100	Reserved (do not access)
011 0101	Debug Control Register 4 (DBCR4)
011 0110	Debug Control Register 5 (DBCR5)
011 0111	Debug Control Register 6 (DBCR6)
011 1000	Debug Control Register 7 (DBCR7)
011 1001	Debug Control Register 8 (DBCR8)
011 1010	Reserved (do not access)
011 1011	Reserved (do not access)
011 1100	External Debug Status Register MASK 0 (EDBSRMSK0)
011 1101	Debug Data Acquisition Message Register (DDAM)
011 1110	Debug Event Control (DEVENT)
011 1111	External Debug Resource Allocation Control 0 (EDBRAC0)
100 0000	Reserved for Debug L1 Cache Control/Status 0 (DBL1CCSR0)
100 0001–110 1100	Reserved (do not access)
110 1101	MPU0 Control/Status 0 (MPU0CSR0)
110 1110	Performance Monitor Register Access
110 1111	Reserved for Shared Nexus Control Register Select
111 0000–111 1001	General Purpose register selects [0:9]
111 1010	Cache Debug Access Control Register (CDACNTL)

Table 1844. OnCE register addressing (continued)

RS[0:6]	Register Selected
111 1011	Cache Debug Access Data Register (CDADATA)
111 1100	Nexus3 Access (see *Chapter 83: e200z420n3 Nexus 3 Module*)
111 1101	LSRL Select (see Test Specification)
111 1110	Enable_OnCE(1)
111 1111	Bypass

1. Causes assertion of the j_en_once_regsel output.

The OnCE Decoder receives as input the 10-bit command from the OCMD, and status signals from the processor, and generates all the strobes required for reading and writing the selected OnCE registers.

Single-stepping of instructions is performed by placing the CPU in debug mode, scanning in appropriate information into the CPUSCR, and setting the Go bit (with the EX bit cleared) with the RS field indicating either the CPUSCR or No Register Selected. After executing a single instruction, the CPU will re-enter debug mode and await further commands. During single-stepping, exception conditions may occur if not properly masked by debug firmware (such as interrupts, machine checks, bus error conditions) and may prevent the desired instruction from being successfully executed. The OSR_{ERR} bit is set to indicate this condition. In these cases, values in the CPUSCR will correspond to the first instruction of the exception handler.

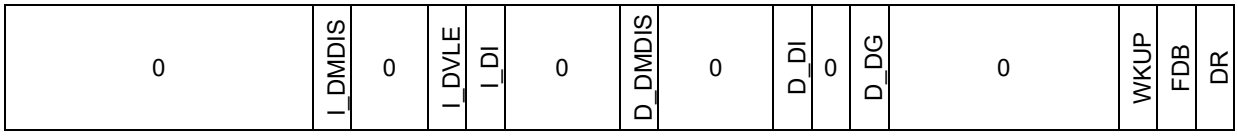
Additionally, the $EDBCR0_{EDM}$ bit is forced to '1' internally while single-stepping to prevent Debug events from generating Debug interrupts. Also, during a debug session, the DBSR register is frozen from updates due to debug events other than execution of a DNI-type instruction, regardless of $EDBCR0_{EDM}$. DBSR may still be modified during a debug session via a single-stepped **mtspr** instruction, or via OnCE access.

If the CPU is not currently in debug mode, go+exit and go+noexit commands are ignored, although for a go+exit command with "No Register Selected", the j_ocmd_go_exit output will be asserted.

70.5.6.3 OnCE Control Register (OCR)

The OnCE Control Register is a 32-bit register used to force the e200z425n3 core into debug mode and to enable / disable sections of the OnCE control logic. It also provides control over the MPU during a debug session (see *Section 70.7: MPU operation during debug*). The control bits are read/write. These bits are only effective while OnCE is enabled (**jd_en_once** asserted). The OCR is shown in *Figure 1914*.

Figure 1914. OnCE Control Register (OCR)



Reset - 0xo000_0000 on **m_por**, **j_trst_b**, or entering Test_logic_Reset state

Table 1845 provides bit definitions for the OnCE Control Register.

Table 1845. OCR field descriptions

Bit	Name	Description
0:7	—	Reserved
8	I_DMDIS	Instruction Side Debug MPU Disable Control Bit This bit may be used to control whether the MPU is enabled normally, or whether the MPU is disabled during a debug session for Instruction Accesses. When enabled, the MPU functions normally. When disabled, for Instruction Accesses, the I bit is taken from the OCR bit I_DI. The SX and UX access permission control bits are set to‘1’ to allow full access. When disabled, no MPU exceptions are generated for Instruction accesses. External access errors can still occur. MPU entries with DEBUG=1 are not affected by this control bit. 0 MPU not disabled for debug sessions 1 MPU disabled for debug sessions
9:10	—	Reserved
11	I_DVLE	Instruction Side Debug ‘VLE’ Attribute Bit This bit is used to provide the ‘VLE’ attribute bit to be used during a debug session. **Note:** This bit is ignored, since VLE is always enabled.
12	I_DI	Instruction Side Debug ‘I’ Attribute Bit This bit is used to provide the ‘I’ attribute bit to be used for Instruction accesses for Instruction accesses during a debug session.
13:15	—	Reserved
16	D_DMDIS	Data Side Debug MPU Disable Control Bit This bit may be used to control whether the MPU is enabled normally, or whether the MPU is disabled during a debug session for Data Accesses. When enabled, the MPU functions normally. When disabled, for Data Accesses, the MPU I and G bits are taken from the OCR bits D_DI and D_DG. The SR, SW, UR, and UW access permission control bits are set to‘1’ to allow full access. When disabled, no MPU exceptions are generated for Data accesses. External access errors can still occur. MPU entries with DEBUG=1 are not affected by this control bit. 0 MPU not disabled for debug sessions 1 MPU disabled for debug sessions
17:19	—	Reserved
20	D_DI	Data Side Debug ‘I’ Attribute Bit This bit is used to provide the ‘I’ attribute bit to be used for Data accesses during a debug session.
21	—	Reserved
22	D_DG	Data Side Debug ‘G’ Attribute Bit This bit is used to provide the ‘G’ attribute bit to be used for Data accesses during a debug session.
23:28	—	Reserved
29	WKUP	Wakeup Request Bit This control bit may be used to force the **p_wakeup** output signal to be asserted. This control function may be used by debug firmware to request that the chip-level clock controller restore the **m_clk** input to normal operation regardless of whether the CPU is in a low power state to ensure that debug resources may be properly accessed by external hardware through scan sequences.

Table 1845. OCR field descriptions (continued)

Bit	Name	Description
30	FDB	Force Breakpoint Debug Mode Bit This control bit is used to determine whether the processor is operating in breakpoint debug enable mode or not. The processor may be placed in breakpoint debug enable mode by setting this bit. In breakpoint debug enable mode, execution of the '**bkpt**' pseudo-instruction will cause the processor to enter debug mode, as if the **jd_de_b** input had been asserted. This bit is qualified with $EDBCR0_{EDM}$, which must be set for FDB to take effect. **Note:** This bit has no effect on **e_dnh** or **se_dnh** instruction operation.
31	DR	CPU Debug Request Control Bit This control bit is used to unconditionally request the CPU to enter Debug mode. The CPU will indicate that Debug mode has been entered via the data scanned out in the shift-IR state. 0 No Debug mode request 1 Unconditional Debug mode request When the DR bit is set, the processor will enter Debug mode at the next instruction boundary.

70.5.7 Access to debug resources

Resources contained in the OnCE Module that do not require the processor core to be halted for access may be accessed while the e200z425n3 core is running, and will not interfere with processor execution. Accesses to other resources such as the CPUSCR require the e200z425n3 core to be placed in debug mode to avoid synchronization hazards. Debug firmware may ensure that it is safe to access these resources by determining the state of the e200z425n3 core prior to access. Note that a scan operation to update the CPUSCR is required prior to exiting debug mode if debug mode has been entered.

Some cases of write accesses other than accesses to the OnCE Command and Control registers, or the EDM bit of DBCR0 require **m_clk** to be running for proper operation. The OnCE control register provides a means of signaling this need to a system level clock control module via the OCRWKUP control bit.

In addition, since the CPU may cause multiple bits of certain registers to change state, reads of certain registers while the CPU is running (such as DBSR) may not have consistent bit settings unless read twice with the same value indicated. In order to guarantee that the contents are consistent, the CPU should be placed into debug mode, or multiple reads should be performed until consistent values have been obtained on consecutive reads.

Table 1846 provides a list of access requirements for OnCE registers.

Table 1846. OnCE register access requirements

Register name	Access requirements					Notes
	Requires jd_en_once to be asserted	Requires $EDBCR0_{EDM}$ = 1 for write access	Requires m_clk active for write access	Requires CPU to be halted for read access	Requires CPU to be halted for write access	
Enable_OnCE	N	N	N	N	—	—
Bypass	N	N	N	N	N	—
CPUSCR	Y	Y	Y	Y	Y	—
DAC1–4	Y	Y	Y	N	*(1)	—
DBCR0	Y	Y	Y	N	*(1)	*$EDBCR0_{EDM}$ access only requires **jd_en_once** asserted
DBCR1–8	Y	Y	Y	N	*(1)	—
DEVENT	Y	Y	Y	N	*(1)	—
EDBRAC0 (DBERC0)	Y	N	Y	N	*(1)	—
DBSR	Y	Y	Y	N(2)	*(1)	—
EDBCR0	Y	N	N	N	N	—
EDBSR0	Y	N	Y	N	N	—
EDBSRMSK0	Y	N	N	N	N	—
IAC1–8	Y	Y	Y	N	*(1)	—
JTAG ID	N	N	—	N	—	Read-only
MPU0CSR0	Y	N	Y	N	N	—
OCR	Y	N	N	N	N	—
OSR	Y	N	—	N	—	Read-only, accessed by scanning out IR while **jd_en_once** is asserted
Cache Debug Access Control (CDACNTL)	Y	N	Y	N	N	CPU gives lowest priority to a Cache access request when it is not debug halted
Cache Debug Access Data (CDADATA)	Y	N	Y	N	N	CPU gives lowest priority to a Cache access request when it is not debug halted
Nexus3-Access	Y	N	N	N	N	—
PMR-Access	Y	N	N	N	N	—
PMR Registers	Y	Y	Y	N	N	—
External GPRs	Y	N	N	N	N	—
LSRL Select	Y	N	—	—	—	System Test logic implementation determines LSRL functionality

1. Writes to these registers while the CPU is running may have unpredictable results due to the pipelined nature of operation, and the fact that updates are not synchronized to a particular clock, instruction, or bus cycle boundary, therefore it is strongly recommended to ensure the processor is first placed into debug mode before updates to these registers are performed.
2. Reads of these registers while the CPU is running may not give data that is self-consistent due to synchronization across clock domains.

70.5.8 Methods of entering Debug mode

The OnCE Status Register indicates that the CPU has entered the debug mode via the DEBUG status bit. The following sections describe how Debug mode is entered assuming the OnCE circuitry has been enabled. OnCE operation is enabled by the assertion of the **jd_en_once** input See *Section 70.5.5.1: OnCE Enable (jd_en_once)*.

70.5.8.1 External debug request during RESET

Holding the **jd_de_b** signal asserted during the assertion of **p_reset_b**, and continuing to hold it asserted following the negation of **p_reset_b** causes the e200z425n3 core to enter Debug mode. After receiving an acknowledge via the OnCE Status Register DEBUG bit, the external command controller should negate the **jd_de_b** signal before sending the first command. Note that in this case the e200z425n3 core does not execute an instruction before entering Debug mode, although the first instruction to be executed will be fetched prior to entering Debug mode in order to properly initialize the CPUSCR.

70.5.8.2 Debug request during RESET

Asserting a debug request by setting the DR bit in the OCR during the assertion of **p_reset_b** and then negating **p_reset_b** causes the chip to enter debug mode. In this case the CPU will fetch the first instruction of the reset exception handler in order to properly initialize the CPUSCR, but does not execute an instruction before entering debug mode.

70.5.8.3 Debug request during normal activity

Asserting a debug request by setting the DR bit in the OCR during normal chip activity causes the chip to finish the execution of the current instruction and then enter the debug mode. Note that in this case the chip completes the execution of the current instruction and stops after the newly fetched instruction enters the CPU instruction register. This process is the same for any newly fetched instruction including instructions fetched by the interrupt processing, or those that will be aborted by the interrupt processing.

70.5.8.4 Debug request during Waiting, Halted, or Stopped state

Asserting a debug request by setting the DR bit in the OCR when the chip is in the Waiting state (**p_waiting** asserted), Halted state (**p_halted** asserted) or Stopped state (**p_stopped** asserted) causes the CPU to exit the state and enter the debug mode once the CPU clock **m_clk** has been restored. Note that in this case, the CPU will negate the **p_waiting, p_halted** and **p_stopped** outputs. Once the debug session has ended, the CPU will return to the state it was in prior to entering debug mode.

To signal the chip-level clock generator to re-enable **m_clk**, the **p_wakeup** output will be asserted whenever the debug block is asserting a debug request to the CPU due to OCR_{DR} being set, or **jd_de_b** assertion, and will remain set from then until the debug session ends (**jd_debug_b** goes from asserted to negated). In addition, the status of the **jd_mclk_on** input (after synchronization to the **j_tclk** clock domain) may be sampled along with other status bits from the **j_tdo** output during the Shift_IR TAP controller state. This status may be

used if necessary by external debug firmware to ensure proper scan sequences occur to registers in the **m_clk** clock domain.

70.5.8.5 Software request during normal activity

Upon executing a '**bkpt'** pseudo-instruction (for e200z425n3, defined to be an all 0's instruction opcode) when the OCR register's (FDB) bit is set (debug mode enable control bit is true), and $EDBCR0_{EDM}$=1, the CPU enters the debug mode after the instruction following the '**bkpt'** pseudo-instruction has entered the instruction register.

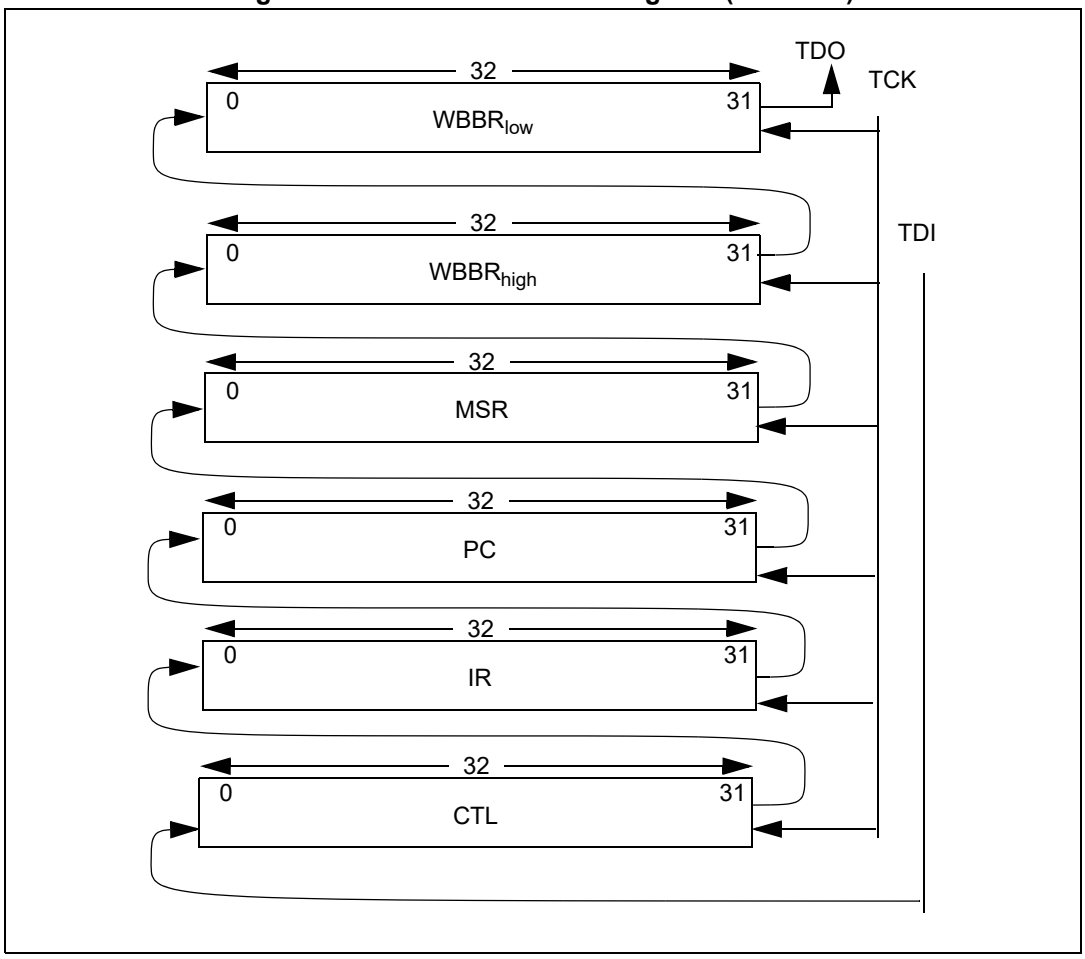
70.5.8.6 Debug notify halt instructions

The **e_dnh** and **se_dnh** instructions allow software to transition the core from a running state to a debug halted state if enabled by $EDBCR0_{DNH_EN}$, and provide the external debugger with bits reserved in the instruction itself to pass additional information. Entry into debug mode is ***not*** conditioned on $EDBCR0_{EDM}$, allowing for debug of software debug handlers as well as other software. For e200z425n3, when the CPU enters a debug halted state due to a **e_dnh** or **se_dnh** instruction, the instruction will be stored in the CPUSCR[IR] portion, and the CPUSCR[PC] value will point to the instruction. The external debugger should update the CPUSCR prior to exiting the debug halted state to point past the **e_dnh** or **se_dnh** instruction.

70.5.9 CPU Status and Control Scan Chain Register (CPUSCR)

A number of on-chip registers store the CPU pipeline status and are configured in a single scan chain for access by the OnCE controller. The CPUSCR register contains these processor resources, which are used to restore the pipeline and resume normal chip activity upon return from the debug mode, as well as a mechanism for the emulator software to access processor and memory contents. *Figure 1915* shows the block diagram of the pipeline information registers contained in the CPUSCR. Once debug mode has been entered, it is required to scan in and update this register prior to exiting debug mode.

Figure 1915. CPU Scan Chain Register (CPUSCR)



70.5.9.1 Instruction Register (IR)

The Instruction Register (IR) provides a mechanism for controlling the debug session by serving as a means for forcing in selected instructions, and then causing them to be executed in a controlled manner by the debug control block. The opcode of the next instruction to be executed when entering debug mode is contained in this register when the scan-out of this chain begins. This value should be saved for later restoration if continuation of the normal instruction stream is desired.

On scan-in, in preparation for exiting debug mode, this register is filled with an instruction opcode selected by debug control software. By selecting appropriate instructions and controlling the execution of those instructions, the results of execution may be used to examine or change memory locations and processor registers. The debug control module external to the processor core will control execution by providing a single-step capability. Once the debug session is complete and normal processing is to be resumed, this register may be loaded with the value originally scanned out.

70.5.9.2 Control State Register (CTL)

The Control State Register (CTL) is a 32-bit register that stores the value of certain internal CPU state variables before the debug mode is entered. This register is affected by the operations performed during the debug session and should normally be restored by the

external command controller when returning to normal mode. In addition to saved internal state variables, two of the bits are used by emulation firmware to control the debug process. In certain circumstances, emulation firmware must modify the content of this register as well as the PC and IR values in the CPUSCR before exiting debug mode. These cases are described below.

Figure 1916. Control State Register (CTL)

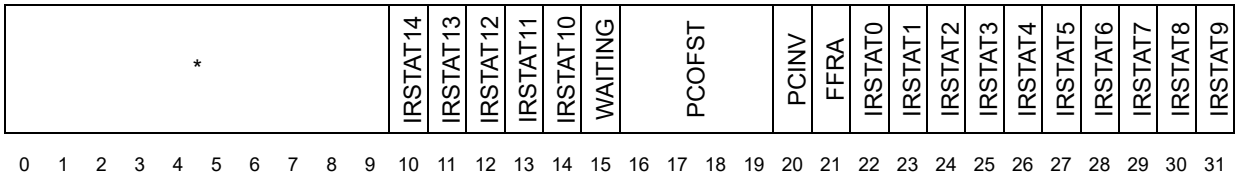


Table 1847. CTL field descriptions

Bit	Name	Description
0:9	*	Internal State Bits These control bits represent internal processor state and should be restored to their original value after a debug session is completed, i.e when a OnCE command is issued with the GO and EX bits set and not ignored. When performing instruction execution during a debug session. See *Section 70.5.5.3: OnCE Debug output (jd_debug_b)*, which is not part of the normal program execution flow, these bits should be set to a 0.
10	IRSTAT14	IR Status Bit 14 This control bit indicates an MPU Instruction Address Compare event status for the IR. 0 No MPU Instruction Address Compare event occurred on the fetch of this instruction. 1 An MPU Instruction Address Compare event occurred on the fetch of this instruction.
11	IRSTAT13	IR Status Bit 13 This control bit indicates an Instruction Address Compare 8 event status for the IR. 0 No Instruction Address Compare 8 event occurred on the fetch of this instruction. 1 An Instruction Address Compare 8 event occurred on the fetch of this instruction.
12	IRSTAT12	IR Status Bit 12 This control bit indicates an Instruction Address Compare 7 event status for the IR. 0 No Instruction Address Compare 7 event occurred on the fetch of this instruction. 1 An Instruction Address Compare 7 event occurred on the fetch of this instruction.
13	IRSTAT11	IR Status Bit 11 This control bit indicates an Instruction Address Compare 6 event status for the IR. 0 No Instruction Address Compare 6 event occurred on the fetch of this instruction. 1 An Instruction Address Compare 6 event occurred on the fetch of this instruction.
14	IRSTAT10	IR Status Bit 10 This control bit indicates an Instruction Address Compare 5 event status for the IR. 0 No Instruction Address Compare 5 event occurred on the fetch of this instruction. 1 An Instruction Address Compare 5 event occurred on the fetch of this instruction.

Table 1847. CTL field descriptions (continued)

Bit	Name	Description
15	WAITING	WAITING State Status This bit indicates whether the CPU was in the waiting state prior to entering debug mode. If set, the CPU was in the waiting state. Upon exiting a debug session, the value of this bit in the restored CPUSCR will determine whether the CPU re-enters the waiting state on a go+exit. 0 CPU was not in the waiting state when debug mode was entered 1 CPU was in the waiting state when debug mode was entered
16:19	PCOFST	PC Offset Field This field indicates whether the value in the PC portion of the CPUSCR must be adjusted prior to exiting debug mode. Due to the pipelined nature of the CPU, the PC value must be backed up by emulation software in certain circumstances. The PCOFST field specifies the value to be subtracted from the original value of the PC. This adjusted PC value should be restored into the PC portion of the CPUSCR just prior to exiting debug mode with a go+exit. In the event the PCOFST is non-zero, the IR should be loaded with a nop instruction instead of the original IR value, other wise the original value of IR should be restored. (But see PCINV, which overrides this field). 0000 No correction required. 0001 Subtract 0x04 from PC. 0010 Subtract 0x08 from PC. 0011 Subtract 0x0C from PC. 0100 Subtract 0x10 from PC. 0101 Subtract 0x14 from PC. All other encodings are reserved
20	PCINV	PC and IR Invalid Status Bit This status bit indicates that the values in the IR and PC portions of the CPUSCR are invalid. Exiting debug mode with the saved values in the PC and IR will have unpredictable results. Debug firmware should initialize the PC and IR values in the CPUSCR with desired values prior to exiting debug mode if this bit was set when debug mode was initially entered. 0 No error condition exists. 1 Error condition exists. PC and IR are corrupted.
21	FFRA	Feed Forward RA Operand Bit This control bit causes the content of the WBBRlo to be used as the RA operand value (RS for logical, mtspr, mtdcr, cntlzw, and shift operations, RX for VLE se_ instructions, RT for e_{logical_op}2i type instructions, RB for evaddiw, evsubifw, and the value to use as the PC for calculating the LR update value for branch with link type instructions) of the first instruction to be executed following an update of the CPUSCR.For most LSP instructions using rA||rB as a 64-bit source operand, WBBRhi, lo is used to supply the 64-bit source value. This allows the debug firmware to update processor registers — initialize the WBBRlo with the desired value, set the FFRA bit, and execute a ori Rx,Rx,0 instruction to the desired register. Note: not all instructions support using the FFRA control. FFRA is mainly intended for use with the ori instruction to allow the debugger to write to a GPR. Support for other instructions is implementation-dependent. 0No action. 1 Content of $WBBR_{lo}$ or $WBBR_{hi,lo}$used as operand value.

Table 1847. CTL field descriptions (continued)

Bit	Name	Description
22	IRSTAT0	IR Status Bit 0 This control bit indicates a TEA status for the IR. 0 No TEA occurred on the fetch of this instruction. 1 TEA occurred on the fetch of this instruction.
23	IRSTAT1	IR Status Bit 1 This control bit is reserved.
24	IRSTAT2	IR Status Bit 2 This control bit indicates an Instruction Address Compare 1 event status for the IR. 0 No Instruction Address Compare event 1 occurred on the fetch of this instruction. 1 An Instruction Address Compare event 1 occurred on the fetch of this instruction.
25	IRSTAT3	IR Status Bit 3 This control bit indicates an Instruction Address Compare 2 event status for the IR. 0 No Instruction Address Compare 2 event occurred on the fetch of this instruction. 1 An Instruction Address Compare 2 event occurred on the fetch of this instruction.
26	IRSTAT4	IR Status Bit 4 This control bit indicates an Instruction Address Compare 3 event status for the IR. 0 No Instruction Address Compare event 3 occurred on the fetch of this instruction. 1 An Instruction Address Compare event 3 occurred on the fetch of this instruction.
27	IRSTAT5	IR Status Bit 5 This control bit indicates an Instruction Address Compare 4 event status for the IR. 0 No Instruction Address Compare 4 event occurred on the fetch of this instruction. 1 An Instruction Address Compare 4 event occurred on the fetch of this instruction.
28	IRSTAT6	IR Status Bit 6 This control bit indicates a Parity Error status for the IR. 0 No Parity Error occurred on the fetch of this instruction. 1 Parity Error occurred on the fetch of this instruction from the I-Cache or the IMEM.
29	IRSTAT7	IR Status Bit 7 This control bit indicates a Precise External Termination Error status for the IR. 0 No Precise External Termination Error occurred on the fetch of this instruction. 1 A Precise External Termination Error occurred on the fetch of this instruction.
30	IRSTAT8	IR Status Bit 8 This control bit indicates the PowerISA VLE status for the IR. 0 IR contains a BookE instruction. (unused encoding) 1 IR contains a PowerISA VLE instruction, aligned in the Most Significant Portion of IR if 16-bit. **Note:** This bit should not be modified by the debugger, otherwise the resulting operation is boundedly undefined. It should always indicate VLE (=1).
31	IRSTAT9	IR Status Bit 9 This control bit is reserved.

Emulation firmware should modify the content of the CTL, PC, and IR values in the CPUSCR during execution of debug related instructions as well as just prior to exiting debug with a go+exit command. During the debug session, the CTL register should be written with the FFRA bit set as appropriate, and all other bit set to ‘0’, and the IR set to the value of the

desired instruction to be executed. IRStat8 will be used to determine the type of instruction present in the IR.

Just prior to exiting debug mode with a go+exit, the PCINV status bit that was originally present when debug mode was first entered should be tested, and if set, the PC and IR initialized for performing whatever recovery sequence is appropriate for a faulted exception vector fetch. If the PCINV bit is cleared, then the PCOFST bits should be examined to determine whether the PC value must be adjusted. Due to the pipelined nature of the CPU, the PC value must be backed up by emulation software in certain circumstances. The PCOFST field specifies the value to be subtracted from the original value of the PC. This adjusted PC value should be restored in to the PC portion of the CPUSCR just prior to exiting debug mode with a go+exit. In the event the PCOFST is non-zero, the IR should be loaded with a nop instruction (such as **ori r0,r0,0**) instead of the original IR value, otherwise the original value of IR should be restored. Note that when a correction is made to the PC value, it will generally point to the last completed instruction, although that instruction will not be re-executed. The nop instruction is executed instead, and instruction fetch and execution will resume at location PC + 4. IRStat8 will be used to determine the type of instruction present in the IR, thus should be cleared in this case. Note that debug events that may occur on the nop (ICMP) will be generated if enabled.

For the CTL register, the internal state bits should be restored to their original value. The IRStatus bits should be set to ‘0’s if the PC was adjusted. If no PC adjustment was performed, emulation firmware should determine whether other IRStat flags should be set to ‘0’ to avoid re-entry into debug mode for an instruction breakpoint request. Upon exiting debug mode with go+exit, if one of these bits is set, debug mode will be reentered prior to any further instruction execution.

70.5.9.3 Program Counter (PC) register

The PC is a 32-bit register that stores the value of the program counter that was present when the chip entered the debug mode. It is affected by the operations performed during the debug mode and must be restored by the external command controller when the CPU returns to normal mode. PC normally points to the instruction contained in the IR portion of CPUSCR. If debug firmware wishes to redirect program flow to an arbitrary location, the PC and IR should be initialized to correspond to the first instruction to be executed upon resumption of normal processing. Alternatively, the IR may be set to a nop and the PC set to point to the location prior to the location at which it is desired to redirect flow to. On exiting debug mode, the nop will be executed, and instruction fetch and execution will resume at PC+4.

70.5.9.4 Write-Back Bus Register ($WBBR_{low}$, $WBBR_{high}$)

WBBR is used as a means of passing operand information between the CPU and the external command controller. Whenever the external command controller needs to read the contents of a register or memory location, it will force the chip to execute an instruction that brings that information to WBBR. $WBBR_{low}$ holds the 32-bit result of most instructions including load data returned for a load or load with update instruction. For LSP instructions that generate 64-bit results, $WBBR_{low}$ holds the low-order 32 bits of the result. $WBBR_{high}$ holds the updated effective address calculated by a load with update instruction. For LSP instructions that generate 64-bit results, $WBBR_{high}$ holds the high-order 32 bits of the result. It is undefined for other instructions.

As an example, to read the 32 bits of processor register **r1**, an **ori r1,r1,0** instruction is executed, and the result value of the instruction will be latched into $WBBR_{low}$. The contents

of $WBBR_{low}$ can then be delivered serially to the external command controller. To update a processor resource, this register is initialized with a data value to be written, and an **ori** instruction is executed, which uses this value as a substitute data value. The Control State register FFRA bit forces the value of the $WBBR_{low}$ to be substituted for the normal RS source value of the **ori** instruction, thus allowing updates to processor registers to be performed. (Refer to *Section 70.5.9.2: Control State Register (CTL)* for more detail on the CTL_{FFRA} bit.)

$WBBR_{low}$ and $WBBR_{high}$ are generally undefined on instructions that do not writeback a result, and due to control issues are not defined on **lmw** or branch instructions as well.

70.5.9.5 Machine State Register (MSR)

The MSR is a 32-bit register used to read/write the Machine State Register. Whenever the external command controller needs to save or modify the contents of the Machine State Register, this register is used.This register is affected by the operations performed during the debug mode and must be restored by the external command controller when returning to normal mode.

70.5.9.6 Exiting Debug mode and interrupt blocking

When exiting debug mode with a Go+Exit, “asynchronous” interrupts are blocked until the first instruction to be executed begins execution. This includes External and Critical input, NMI, and machine check interrupts. Asynchronous debug interrupts are not blocked however, and the CPU will reenter debug mode without executing an instruction following Go+Exit, although it may fetch an instruction and discard it. Exceptions due to an illegal instruction or error flags set within the CPUSCR CTL register are not blocked, since they apply to the instruction in the CPUSCR IR.

70.5.10 Reserved registers (Reserved)

The reserved registers are used to control various test control logic. These registers are not intended for customer use. To preclude device and/or system damage, these registers should not be accessed.

70.6 Watchpoint support

e200z425n3 supports the generation and signalling of watchpoints when operating in internal debug mode ($DBCR0_{IDM}$=1) or in external debug mode ($EDBCR0_{EDM}$=1). Watchpoints are indicated with a dedicated set of interface signals. The **jd_watchpt[0:31]** output signals are used to indicate that a watchpoint has occurred. Certain watchpoints however (DEVENT-based, DTC-based, and Performance Monitor watchpoints) are not qualified with $EDBCR0_{EDM}$ or $DBCR0_{IDM}$.

Each debug address compare function (IAC1–8, DAC1–4), as well as other event types are capable of triggering a watchpoint output. The DBCRx control fields are used to configure watchpoints, regardless of whether events are enabled in DBCR0. Watchpoints may occur whenever an associated event would have been posted in the Debug Status Register if enabled. No explicit enable bits are provided for watchpoints; they are always enabled by definition, although the data address compare watchpoints may be controlled for read and write accesses via configuration fields in DBCR4, 7, and 9. During a debug session, debug events (other than PMI, DEVT1 and DEVT2) with a corresponding DBSR bit are blocked from asserting a watchpoint. The Performance Monitor, DEVNT-based and DTC-based

watchpoints are not blocked during a debug session. If not desired, for address-based events the base address values for these events may be programmed to an unused system address. MSR_{DE} has no effect on watchpoint generation.

External logic may monitor the assertion of these signals for debugging or triggering purposes. Watchpoints are signaled in the clock cycle following the occurrence of the actual event. The Nexus3 module also monitors assertion of a portion of these output signals (**jd_watchpt[0:31])** for various development control purposes. (See the “Watchpoint Trace Messaging” section in the Core (e200z425n3) Nexus 3 Module chapter.)

Note that these signals are **m_clk** domain signals, not **j_tclk** domain signals.

Table 1848 shows the **jd_watchpt[0:31]** output signal assignments to various debug events.

Table 1848. Watchpoint output signal assignments

Signal Name	Type	Description
jd_watchpt[0]	IAC1	Instruction Address Compare 1 watchpoint Asserted whenever an IAC1 compare occurs regardless of being enabled to set DBSR status
jd_watchpt[1]	IAC2	Instruction Address Compare 2 watchpoint Asserted whenever an IAC2 compare occurs regardless of being enabled to set DBSR status
jd_watchpt[2]	IAC3	Instruction Address Compare 3 watchpoint Asserted whenever an IAC3 compare occurs regardless of being enabled to set DBSR status
jd_watchpt[3]	IAC4	Instruction Address Compare 4 watchpoint Asserted whenever an IAC4 compare occurs regardless of being enabled to set DBSR status
jd_watchpt[4]	DAC1	Data Address Compare 1 watchpoint Asserted whenever a DAC1 compare occurs regardless of being enabled to set DBSR status, based on $DBCR4_{DAC1CFG}$ settings
jd_watchpt[5]	DAC2	Data Address Compare 2 watchpoint Asserted whenever a DAC2 compare occurs regardless of being enabled to set DBSR status, based on $DBCR4_{DAC2CFG}$ settings
jd_watchpt[6]	DAC3	Data Address Compare 3 watchpoint Asserted whenever a DAC3 compare occurs regardless of being enabled to set DBSR status, based on $DBCR7_{DAC3CFG}$ settings
jd_watchpt[7]	DAC4	Data Address Compare 4 watchpoint Asserted whenever a DAC4 compare occurs regardless of being enabled to set DBSR status, based on $DBCR7_{DAC4CFG}$ settings
jd_watchpt[8]	IAC5	Instruction Address Compare 5 watchpoint Asserted whenever an IAC5 compare occurs regardless of being enabled to set DBSR status
jd_watchpt[9]	IAC6	Instruction Address Compare 6 watchpoint Asserted whenever an IAC6 compare occurs regardless of being enabled to set DBSR status

Table 1848. Watchpoint output signal assignments (continued)

Signal Name	Type	Description
jd_watchpt[10]	DEVT1	Debug Event Input 1 watchpoint Asserted whenever a DEVT1 debug event occurs regardless of being enabled to set DBSR status
jd_watchpt[11]	DEVT2	Debug Event Input 2 watchpoint Asserted whenever a DEVT2 debug event occurs regardless of being enabled to set DBSR status
jd_watchpt[12]	DEVNT0	Debug Event Output 0 watchpoint Asserted whenever a ‘1’ is written to the bit of the DEVNT field of the DEVENT debug register corresponding to jd_watchpt[12]
jd_watchpt[13]	DEVNT1	Debug Event Output 1 watchpoint Asserted whenever a ‘1’ is written to the bit of the DEVNT field of the DEVENT debug register corresponding to jd_watchpt[13]
jd_watchpt[14]	IAC7	Instruction Address Compare 7 watchpoint Asserted whenever an IAC7 compare occurs regardless of being enabled to set DBSR status
jd_watchpt[15]	IAC8	Instruction Address Compare 8 watchpoint Asserted whenever an IAC8 compare occurs regardless of being enabled to set DBSR status
jd_watchpt[16]	IRPT	Interrupt watchpoint Asserted whenever an IRPT debug event occurs regardless of being enabled to set DBSR status
jd_watchpt[17]	RET	Return watchpoint Asserted whenever a RET debug event occurs regardless of being enabled to set DBSR status
jd_watchpt[18]	CIRPT	Critical Interrupt watchpoint Asserted whenever a CIRPT debug event occurs regardless of being enabled to set DBSR status
jd_watchpt[19]	CRET	Critical Return watchpoint Asserted whenever a CRET debug event occurs regardless of being enabled to set DBSR status
jd_watchpt[20]	DEVNT2	Debug Event Output 2 watchpoint Asserted whenever a ‘1’ is written to the bit of the DEVNT field of the DEVENT debug register corresponding to jd_watchpt[20]
jd_watchpt[21]	DEVNT3	Debug Event Output 3 watchpoint Asserted whenever a ‘1’ is written to the bit of the DEVNT field of the DEVENT debug register corresponding to jd_watchpt[21]
jd_watchpt[22]	PMEVENT	Performance Monitor Event input watchpoint Asserted whenever **p_pm_event** transitions from a ‘0’ to a ‘1’ while **m_clk** is running
jd_watchpt[23]	PMC0	Performance Monitor Counter 0 watchpoint Asserted whenever PMC0 triggers an event based on $PMLCa0_{PMP}$
jd_watchpt[24]	PMC1	Performance Monitor Counter 1 watchpoint Asserted whenever PMC1 triggers an event based on $PMLCa1_{PMP}$

Table 1848. Watchpoint output signal assignments (continued)

Signal Name	Type	Description
jd_watchpt[25]	PMC2	Performance Monitor Counter 2 watchpoint Asserted whenever PMC2 triggers an event based on $PMLCa2_{PMP}$
jd_watchpt[26]	PMC3	Performance Monitor Counter 3 watchpoint Asserted whenever PMC3 triggers an event based on $PMLCa3_{PMP}$
jd_watchpt[27]	MPU	Memory Protection Unit watchpoint Asserted whenever a the MPU generates a debug event based on region descriptor matches with DEBUG control enabled.
jd_watchpt[28]	TRAP	TRAP watchpoint Asserted whenever an TRAP debug event occurs regardless of being enabled to set DBSR status
jd_watchpt[29]	DTC1	Data Trace Control Range 1 watchpoint Asserted whenever an access meets the conditions for DTC Range 1
jd_watchpt[30]	DTC2	Data Trace Control Range 2 watchpoint Asserted whenever an access meets the conditions for DTC Range 2
jd_watchpt[31]	DTC3	Data Trace Control Range 3 watchpoint Asserted whenever an access meets the conditions for DTC Range 3

70.7 MPU operation during debug

A debug session begins when the CPU initially enters debug mode, and ends when a OnCE command with GO+EXIT is executed, releasing the CPU for normal operation. If desired during a debug session, the debug firmware may substitute default values obtained from the OnCE Control Register for Access Attribute (I, G) bits. Refer to the bit definitions in the OCR (*Section 70.5.6.3: OnCE Control Register (OCR)* for more detail.

Normal operation of the MPU may be modified during a ‘debug session’ via the OnCE Control Register (OCR). A debug session begins when the CPU initially enters debug mode, and ends when a OnCE command with GO+EXIT is executed, releasing the CPU for normal operation. If desired during a debug session, the debug firmware may disable the MPU protection mechanism and may substitute default values for the Access Protection (UX, UR, UW, SX, SR, SW) bits, and values obtained from the OnCE Control Register for Memory Attributes (I, G) bits normally provided by a matching MPU entry.

When disabled during a debug session, no MPU-related storage interrupt conditions will occur. If the debugger desires to use the normal protection mechanism, the MPU may be left enabled in the OnCE OCR, and normal protections provided by the MPU (including the possibility of a storage interrupt) will remain in effect.

The OCR control bits are used when debug mode is entered. Refer to the bit definitions in the OCR (*Section 70.5.6.3: OnCE Control Register (OCR)* for more detail. When the MPU is disabled for instruction accesses (OCR_{I_DMDIS}) or for data accesses (OCR_{D_DMDIS}), substituted access attribute bits will control operation on respective accesses initiated during debug.

70.8 Cache array access during debug

The cache arrays may be read and written during debug mode via the CDACNTL and CDADATA debug registers.

70.9 Basic steps for enabling, using, and exiting External Debug mode

The following steps show one possible scenario for a debugger wishing to use the external debug facilities. *This simplified flow is intended to illustrate basic operations, but does not cover all potential methods in depth.*

Enabling External Debug mode and initializing debug registers:

- The debugger should ensure that the **jd_en_once** control signal is asserted in order to enable OnCE operation.
- Select the OCR and write a value to it in which OCR_{DR}, OCR_{WKUP}, are set to '1'. The tap controller must step through the proper states as outlined earlier. This step will place the CPU in a debug state in which it is halted and awaiting single-step commands or a release to normal mode
- Scan out the value of the OSR to determine that the CPU clock is running and the CPU has entered the Debug state. This can be done in conjunction with a Read of the CPUSCR. The OSR is shifted out during the Shift_IR state. The CPUSCR will be shifted out during the Shift_DR state. The debugger should save the scanned-out value of CPUSCR for later restoration.
- Select the DBCR0 register and update it with the $EDBCR0_{EDM}$ bit set.
- Clear the DBSR status bits.
- Write appropriate values to the DBCR0–8, IAC, and DAC registers. Note that the initial write to DBCR0 will only affect the EDM bit, so the remaining portion of the register must now be initialized, keeping the EDM bit set.

At this point the system is ready to commence debug operations. Depending on the desired operation, different steps must occur.

- Optionally, ensure that the values entered into the MSR portion of the CPUSCR during the following steps cause interrupt to be disabled (clearing MSR_{EE} and MSR_{CE}). This will ensure that external interrupt sources do not cause single-step errors.

To single-step the CPU:

- debugger scans in either a new or a previously saved value of the CPUSCR (with appropriate modification of the PC and IR as described in *Section 70.5.9.2: Control State Register (CTL)*), with a Go+Noexit OnCE Command value.
- The debugger scans out the OSR with "no-register selected", Go cleared, and determines that the PCU has reentered the Debug state and that no ERR condition occurred.

To return the CPU to normal operation (without disabling external debug mode)

- The OCR_{DR} control bit should be cleared, leaving the OCR_{WKUP} bit set.
- The debugger restores the CPUSCR with a previously saved value of the CPUSCR (with appropriate modification of the PC and IR as described in *Section 70.5.9.2: Control State Register (CTL)*), with a Go+Exit OnCE Command value.
- The OCR_{WKUP} bit may then be cleared.

To exit External Debug mode

- The debugger should place the CPU in the debug state via the OCR_{DR} with OCR_{WKUP} asserted, scanning out and saving the CPUSCR.
- The debugger should write the DBCR0–8 registers as needed, likely clearing every enable except the $EDBCR0_{EDM}$ bit.
- The debugger should write the DBSR to a cleared state.
- The debugger should rewrite the DBCR0 with all bits including EDM cleared.
- The debugger should clear the OCR_{DR} bit.
- The debugger restores the CPUSCR with the previously saved value of the CPUSCR (with appropriate modification of the PC and IR as described in *Section 70.5.9.2: Control State Register (CTL)*), with a Go+Exit OnCE Command value.
- The OCR_{WKUP} bit may then be cleared.

Note: These steps are meant by way of examples, and are not meant to be an exact template for debugger operation.

71 e200z420n3 Core Debug Support

This chapter describes the debug features of the e200z420n3 core.

71.1 Overview

Internal debug support in the e200z420n3 core allows for software and hardware debug by providing debug functions, such as instruction and data breakpoints and program trace modes. For software based debugging, debug facilities consisting of a set of software accessible debug registers and interrupt mechanisms are provided. These facilities are also available to a hardware-based debugger, which communicates using a modified IEEE 1149.1 Test Access Port (TAP) controller and pin interface. When hardware debug is enabled, the debug facilities controlled by hardware are protected from software modification.

Software debug facilities are defined as part of *PowerISA 2.06*. e200z420n3 supports a subset of these defined facilities. In addition to the facilities defined in *PowerISA 2.06*, e200z420n3 provides additional flexibility and functionality in the form of additional instruction breakpoints, linked instruction and data breakpoints, and extended range and value masking. These features are also available to a hardware-based debugger.

When not being used for debugging purposes, a portion of the debug facilities may be configured to provide a stack limit checking mechanism.

71.1.1 Software debug facilities

e200z420n3 provides debug facilities to enable hardware and software debug functions, such as instruction and data breakpoints and program single stepping. The debug facilities consist of a set of debug control registers (DBCR0–2,4–8, EDBRAC0), a set of address compare registers (IAC1–8, DAC1–4), a set of 64-bit data value compare registers (DVC1, DVC2), a Debug Status Register (DBSR) for enabling and recording various kinds of debug events, a Debug Data Effective Address Register (DDEAR), and a special Debug interrupt type built into the interrupt mechanism.

The debug facilities also provide a mechanism for software-controlled processor reset, and debug mode entry. In addition, the Performance Monitor may be configured to utilize the debug interrupt type. Also, the Memory Protection Unit (MPU) may be configured to generate debug events using region descriptors that are not utilized for performing a protection function.

Software debug facilities are enabled by setting the internal debug mode bit in Debug Control register 0 ($DBCR0_{IDM}$). When internal debug mode is enabled, debug events can occur, and can be enabled to record exceptions in the Debug Status register (DBSR). If enabled by MSR_{DE}, these recorded exceptions cause Debug interrupts to occur. When $DBCR0_{IDM}$ is cleared (and $EDBCR0_{EDM}$ is cleared as well), no debug events occur, and no status flags are set in DBSR unless already set. In addition, when $DBCR0_{IDM}$ is cleared (or is overridden by $EDBCR0_{EDM}$ being set and EDBRAC0 indicating no resource is “owned” by software) no Debug interrupts will occur, regardless of the contents of DBSR. A software Debug interrupt handler may access all system resources and perform necessary functions appropriate for system debug.

71.1.1.1 *PowerISA 2.06* compatibility

The e200z420n3 core implements a subset of the *PowerISA 2.06* internal debug features. The following restrictions on functionality are present:

- Instruction address compares do not support compare on physical (real) addresses.
- Data address compares do not support compare on physical (real) addresses.

71.1.2 Additional debug facilities

In addition to the debug functionality defined in *PowerISA 2.06*, e200z420n3 provides capability to link instruction and data breakpoints, provides additional instruction and data breakpoints, extended range and value masking, multiple watchpoint outputs, provides capability for the Performance Monitor to generate debug events, and allows for sharing of debug resources between software and a hardware debugger (Refer to *Section 71.1.4, Sharing debug resources by software/hardware*). A data trace port is also provided.

71.1.2.1 Data trace port

The data trace port interface is provided to assist in implementing extended debug watchpoint/breakpoint/trace capture capability with logic external to the e200z420n3 core. This port provides information corresponding to each read or write access completed without error by the CPU. In order to report data accesses, the Nexus 3 DTC register DI control bit must be set to trace data accesses, not instruction accesses (that is the normal setting). Refer to the “Data Trace Control Register (DTC)” section in the Core (e200z420n3) Nexus 3 Module chapter. The data trace port will report aligned accesses and misaligned accesses as one access, unless the two portions of a misaligned access are non-contiguous, such as when the second portion of the misaligned access is to address 0.

71.1.3 Hardware debug facilities

The e200z420n3 core contains facilities that allow for external test and debugging. A modified IEEE 1149.1 control interface is used to communicate with the core resources. This interface is implemented through a standard 1149.1 TAP (test access port) controller.

By using public instructions, the external debugger can freeze or halt the e200z420n3 core, read and write internal state and debug facilities, single-step instructions, and resume normal execution.

Hardware Debug is enabled by setting the External Debug mode enable bit in External Debug Control register 0 ($EDBCR0_{EDM}$), which is also aliased to $DBCR0_{EDM}$. Setting $EDBCR0_{EDM}$ overrides the Internal Debug mode enable bit $DBCR0_{IDM}$ unless resources are provided back to software via the settings in EDBRAC0. When the Hardware Debug facility is enabled, software is blocked from modifying the “hardware-owned” debug facilities. In addition, since resources are “owned” by the hardware debugger, inconsistent values may be present if software attempts to read “hardware-owned” debug-related resources.

When hardware debug is enabled by setting $EDBCR0_{EDM}$=1, the control registers and resources described in *Section 71.3, Debug registers* are reserved for use by the external debugger. The same events described in *Section 71.2, Software debug events and exceptions* are also used for external debugging, but exceptions are not generated to running software. Hardware-owned debug events enabled in the respective DBCR0–8 registers are recorded in the EDBSR0 register (not the DBSR) regardless of MSR_{DE}, and no debug interrupts are generated unless a) the resource is granted back to software via

EDBRAC0 settings, and b) debug mode entry is not masked by the corresponding event bit in EDBSRMSK0. Instead, the CPU will enter debug mode when an enabled event causes a EDBSR0 bit to become set. $EDBCR0_{EDM}$, EDBSR0, EDBSRMSK0, EDDEAR, and EDBRAC0 may only be written through the OnCE port.

Access to most debug resources (registers) requires that the CPU clock (**m_clk**) be running in order to perform write accesses from the external hardware debugger.

71.1.4 Sharing debug resources by software/hardware

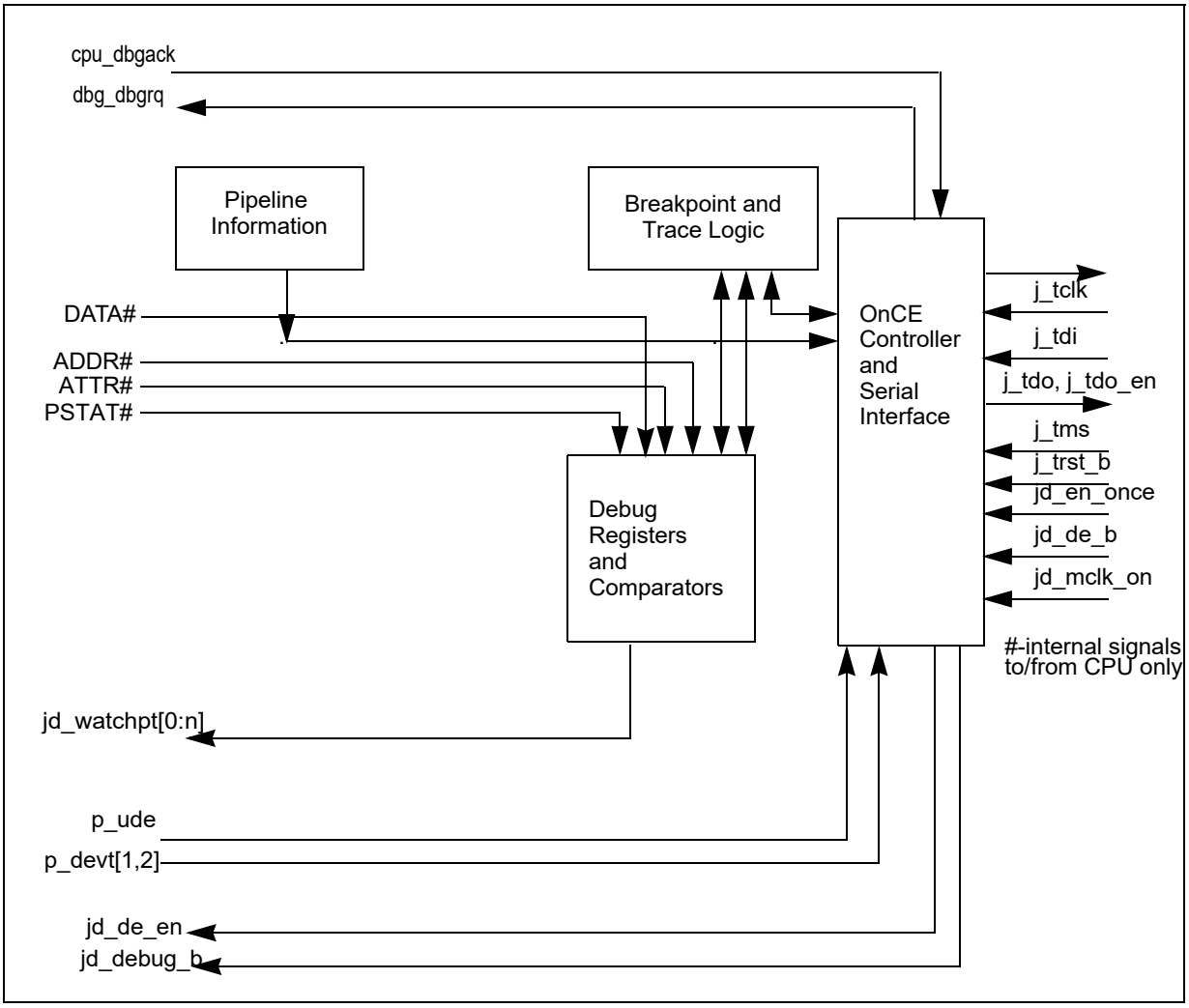
Debug resources may be shared by a hardware debugger and software debug based on the settings of debug control register EDBRAC0. When $EDBCR0_{EDM}$ is set, EDBRAC0 settings determine which debug resources are allocated to software and which resources remain under exclusive hardware control. Software-owned resources that set DBSR bits when $DBCR0_{IDM}$=1 will cause a debug interrupt to occur when enabled with MSR_{DE}. Hardware-owned resources that set EDBSR0 bits when $EDBCR0_{EDM}$=1 will cause an entry into debug mode if the event is not masked in EDBSRMSK0. EDBRAC0 is read-only by software. When resource sharing is enabled, ($EDBCR0_{EDM}$=1 and $EDBRAC0_{IDM}$=1), only software-owned resources may be modified by software. Hardware always has full access to all registers and all register fields through the OnCE register access mechanism, and it is up to the debug firmware to properly implement modifications to these registers with read-modify-write operations to implement any control sharing with software. Hardware-owned resources will set status bits in the EDBSR0 register instead of in DBSR, and will report the effective address of data breakpoints in EDDEAR instead of DDEAR. Settings in EDBRAC0 should be considered by the debug firmware in order to preserve software settings of control and status registers as appropriate when hardware modifications to the debug registers are performed.

71.1.4.1 Simultaneous hardware and software debug event handing

Since it is possible that a “hardware-owned” resource can produce a debug event in conjunction with a software-owned resource producing a different debug event simultaneously, a priority ordering mechanism is implemented that guarantees that the hardware event is handled as soon as possible, while preserving the recognition of the software event. The CPU will give highest priority to the software event initially in order to reach a recoverable boundary, and then will give highest priority to the hardware event in order to enter debug mode as near the point of event occurrence as possible. This is implemented by allowing software exception handing to begin internal to the CPU and to reach the point where the current program counter and MSR values have been saved into DSRR0/1, and the new PC pointing to the debug interrupt handler, along with the new MSR updates. At this point, hardware priority takes over, and the CPU enters debug mode.

Figure 1917 shows the e200z420n3 debug resources.

Figure 1917. e200z420n3 debug resources



71.2 Software debug events and exceptions

Software debug events and exceptions are available when internal debug mode is enabled ($DBCR0_{IDM}$=1) and not overridden by external debug mode ($EDBCR0_{EDM}$ must either be cleared or corresponding resources must be allocated to software debug by the settings in EDBRAC0). When enabled, debug events cause debug exceptions to be recorded in the Debug Status Register. Specific event types are enabled by the Debug Control Registers (DBCR0–8). The Unconditional Debug Event (UDE) is an exception to this rule; it is always enabled. Once a Debug Status Register (DBSR) bit is set by a debug resource that is “owned” by software (other than MRR, DAC_OFST, or VLES), if Debug interrupts are enabled by MSR_{DE}, a Debug interrupt will be generated. The debug interrupt handler is responsible for ensuring that multiple repeated debug interrupts do not occur by clearing the DBSR as appropriate.

Certain debug events are not allowed to occur when MSR_{DE}=0 and $DBCR0_{IDM}$=1. In such situations, no debug exception occurs and thus no DBSR bit is set. Other debug events may cause debug exceptions and set DBSR bits regardless of the state of MSR_{DE}. A Debug interrupt will be delayed until MSR_{DE} is later set to '1'.

When a Debug Status Register bit is set while MSR_{DE}=0, an Imprecise Debug Event flag ($DBSR_{IDE}$) will also be set to indicate that an exception bit in the Debug Status Register was set while Debug interrupts were disabled. Debug interrupt handler software can use this bit to determine whether the address recorded in Debug Save/Restore Register 0 is an address associated with the instruction causing the debug exception, or the address of the instruction that enabled a delayed Debug interrupt by setting the MSR_{DE} bit. A **mtmsr** or **mtdbcr0** that causes both MSR_{DE} and $DBCR0_{IDM}$ to become set, enabling precise debug mode, may cause an Imprecise (Delayed) Debug exception to be generated due to an earlier recorded event in the Debug Status register.

There are eight types of debug events defined by *PowerISA 2.06*:

1. Instruction Address Compare debug events
2. Data Address Compare debug events
3. Trap debug events
4. Branch Taken debug events
5. Instruction Complete debug events
6. Interrupt/Critical Interrupt Taken debug events
7. Return/Critical Return debug events
8. Unconditional debug events

In addition, e200z420n3 defines additional debug events:

- The External debug events DEVT1 and DEVT2, which are described in *Section 71.2.11, External debug event*.
- The Performance Monitor Interrupt event PMI, which is described in *Section 71.2.13, Performance Monitor Interrupt debug event*.

The e200z420n3 debug configuration supports most of these event types. Unsupported *PowerISA 2.06* defined functionality is as follows:

- Instruction Address Compare and Data Address Compare *Real address* mode is not supported.

A brief description of each of the event types follows.

71.2.1 Instruction Address Compare event

Instruction Address Compare debug events occur when enabled and execution is attempted of an instruction at an address that meets the criteria specified in the DBCR0, DBCR1, DBCR5, DBCR6, and IAC1–8 Registers. Instruction Address compares may specify user/supervisor mode, along with an effective address, masked effective address, or range of effective addresses for comparison. This event can occur and be recorded in DBSR regardless of the setting of MSR_{DE}. IAC events will not occur when an instruction would not have normally begun execution due to a higher priority exception at an instruction boundary.

IAC compares perform a 31-bit compare for VLE instruction storage accesses. Each halfword fetched by the instruction fetch unit will be marked with a set of bits indicating whether an Instruction Address Compare occurred on that halfword. Debug exceptions will occur if enabled and a 16-bit instruction, or the first halfword of a 32-bit instruction, is tagged with an IAC hit.

71.2.2 Data Address Compare event

Data Address Compare debug events occur when enabled and execution of a load or store class instruction or a cache maintenance instruction results in a data access that meets the criteria specified in the DBCR0, DBCR2, DBCR4, DBCR7–8, DAC1–4, DVC1, and DVC2 Registers. Data address compares may specify user/supervisor mode, along with an effective address, masked effective address, or range of effective addresses for comparison. This event can occur and be recorded in DBSR regardless of the setting of MSR_{DE}. Four address compare values (DAC1–4) are provided.

The effective address of the load, store, or cache control operation is recorded in the DDEAR register when a data address compare event is recorded in DBSR if the previous values of the $DBSR_{DAC\{R,W\}}$ bits are zero. Once a DDEAR update is performed, these bits must be cleared by software prior to another DDEAR hardware update occurring. A subsequent DAC event will not update the DDEAR register if either of the $DBSR_{DAC\{R,W\}}$ bits are set.

Note: In contrast to the PowerISA 2.06 definition, Data Address Compare events on e200z420n3 do not prevent the load or store class instruction from completing. If a load or store class instruction completes successfully without a Data Storage interrupt, Data Address Compare exceptions are reported at the completion of the instruction. If the exception results in a precise Debug interrupt, the address value saved in DSRR0 is the address of the instruction following the load or store class instruction. For DVC DAC events, the exception can be imprecisely reported even further past the load or store class instruction generating the event (without necessarily affecting $DBSR_{IDE}$) and the saved address value can point to a subsequent instruction past the next instruction. This occurrence is indicated in the $DBSR_{DAC_OFST}$ field.

If a load or store class instruction does not complete successfully due to a Data Storage exception or a machine check condition for the load or store, and a Data Address Compare debug exception also occurs, the result is an imprecise Debug interrupt, the address value saved in DSRR0 is the address of the load or store class instruction, and the $DBSR_{IDE}$ bit will be set. In addition to occurring when $DBCR0_{IDM}$=1, this circumstance can also occur when $EDBCR0_{EDM}$=1 and the event is hardware-owned, in which case $EDBSR0_{IDE}$ will be set.

DAC events will not be recorded if a load multiple word or store multiple word instruction is interrupted prior to completion by a critical input or external input interrupt.

DAC events will occur for cache control instructions without performing data compares, regardless of the settings of the DBCRx registers and DVC registers for data value comparison qualifications, that is no data comparisons are performed since these instructions do not have associated data values.

DAC events are not signaled on the second portion of a misaligned load or store that is broken up into two separate accesses.

*DAC events are not signaled on the **mpure** or **mpuwe** MPU instructions.*

*DAC[1,2] events are not signaled if DVC[1,2]M is non-zero and a DSI exception occurs on the load or store, since the load or store access is not performed. For a **lmw** or **stmw** transfer however, if a DVC successfully occurs on a transfer and a later transfer encounters a DSI exception, the DAC event will be reported, since a successful data value compare took place.*

*Due to internal pipeline status, for a reported DAC event on a **lmw** or **stmw** transfer, the value stored in the DDEAR/EDDEAR will be the effective address of the first transfer of the sequence, and not necessarily the precise transfer that caused the DAC event.*

71.2.2.1 Data Address Compare event status updates

Data Address Compare debug events with Data Value compares can be reported ambiguously in several circumstances involving issuing a sequence of load or store class instructions. Due to the CPU pipeline and the delay in performing the data value compare following completion of the access, if the first load or store class instruction generates a DVC DAC, a second and possibly third load or store class instruction may also generate a DAC or DVC DAC event, or may generate a DSI exception with or without a simultaneous DAC event.

Also, since non-load/store instructions may be dual-issued in combination with a load/store instruction, the actual number of additional instructions that are completed following a recognized DVC DAC on a load/store instruction may vary from 0 to 5. This value will be reported in the $DBSR_{DAC_OFST}$ field when the DVC DAC status is recorded.

Table 1849 outlines the settings of the DBSR, DSRR0 saved value, and potential updating of the ESR register for various exception cases on sequences of load/store class instructions. Not all exception combinations are covered in the table, such as IAC, ISI, or Alignment exceptions on subsequent instructions. In general these exceptions will cause further instruction issue to be halted, execution of the excepting instruction to be aborted, and reporting of these exceptions will be masked. The saved DSRR0 value will point to this excepting instruction, and the exception(s) may be regenerated after returning from the debug interrupt handler and attempting to re-execute the instruction pointed to by DSRR0. In addition, in the examples in *Table 1849*, the DAC_OFST and DSRR0 values assume no dual issue occurs. If dual-issue occurs with the first, second, or third column, then the DAC_OFST and DSRR0 values will point beyond the values shown.

Table 1849. DAC events and resultant updates

1st load/store class instruction	2nd instruction (load/store class unless otherwise specified)	3rd instruction (load/store class unless otherwise specified)	Result
DSI, no DAC	—	—	Take DSI exception, no DBSR update. Update ESR.
DSI, with DACx	—	—	Take Debug exception, DBSR update setting DACx and IDE, DAC_OFST not set. DSRR0 points to 1st load/store class instruction. No ESR update.
DACx	—	—	Take Debug exception, DBSR update setting DACx, DAC_OFST not set. DSRR0 points to 2nd load/store class instruction. No ESR update.
DVC DACx	No exceptions, any instruction	No exceptions, Non-ldst instruction	Take Debug exception, DBSR update setting DACx, DAC_OFST set to 3'b001. DSRR0 points to 3rd instruction. No ESR update.
DVC DACx	No exceptions	No exceptions, Ldst instruction	Take Debug exception, DBSR update setting DACx, DAC_OFST set to 3'b010. DSRR0 points to instruction after 3rd instruction. No ESR update.

Table 1849. DAC events and resultant updates (continued)

1st load/store class instruction	2nd instruction (load/store class unless otherwise specified)	3rd instruction (load/store class unless otherwise specified)	Result
DVC DACx	DSI, no DAC	—	Take Debug exception, DBSR update setting DACx, DAC_OFST not set. DSRR0 points to 2nd load/store class instruction. No ESR update. **Note:** in this case the 2nd ld/st exception is masked. This behavior is implementation dependent and may differ on other CPUs.
DVC DACx	DSI, with DACy	—	Take Debug exception, DBSR update setting DACx. DAC_OFST not set. DSRR0 points to 2nd load/store class instruction. No ESR update. **Note:** in this case the 2nd ld/st exception is masked. This behavior is implementation dependent and may differ on other CPUs.
DVC DACx	DACy	—	Take Debug exception, DBSR update setting DACx, DACy. DAC_OFST set to 3'b001. DSRR0 points to 3rd instruction. **Note:** in this case if x==y, then the resultant state of DBSR and DSRR0 may be indistinguishable from the "no DACy" case.
DVC DACx	DVC DACy, Normal Ldst	Non-Ldst instruction	Take Debug exception, DBSR update setting DACx, DACy. DAC_OFST set to 3'b001. DSRR0 points to the 3rd instruction. **Note:** in this case if x==y, then the resultant state of DBSR and DSRR0 may be indistinguishable from the "no DACy" case.
DVC DACx	DVC DACy, Normal Ldst	Ldst instruction, no exception	Take Debug exception, DBSR update setting DACx, DACy. DAC_OFST set to 3'b010. DSRR0 points to instruction after the 3rd load/store class instruction. **Note:** in this case if x==y, then the resultant state of DBSR and DSRR0 may be indistinguishable from the "no DACy" case.
DVC DACx	DVC DACy, Normal Ldst	DSI Error, with or without DAC	Take Debug exception, DBSR update setting DACx, DACy. DAC_OFST set to 3'b001. No ESR update. DSRR0 points to 3rd instruction. **Note:** in this case if x==y, then the resultant state of DBSR and DSRR0 may be indistinguishable from the "no DACy" case. **Note:** in this case the 3rd ld/st exception is masked. This behavior is implementation dependent and may differ on other CPUs.
DVC DACx	DVC DACy, Normal Ldst	DACy, or DVC DAC_y Normal Ldst or multiple word Ldst	Take Debug exception, DBSR update setting DACx, DACy. DAC_OFST set to 3'b010. DSRR0 points to instruction after the 3rd load/store class instruction. **Note:** in this case if x==y, then the resultant state of DBSR and DSRR0 may be indistinguishable from the "no DACy" case.
DVC DACx	DVC DACy, Ldst multiple (lmw, stmw)	Any instruction including ld/st	Take Debug exception, DBSR update setting DACx, DACy. DAC_OFST set to 3'b001. DSRR0 points to the 3rd instruction. **Note:** in this case if x==y, then the resultant state of DBSR and DSRR0 may be indistinguishable from the "no DACy" case.

Table 1849. DAC events and resultant updates (continued)

1st load/store class instruction	2nd instruction (load/store class unless otherwise specified)	3rd instruction (load/store class unless otherwise specified)	Result
DVC DACx	Any instruction (no exception)	DSI, with or without DAC, Normal Ldst or multiple word Ldst	Take Debug exception, DBSR update setting DACx. DAC_OFST set to 3'b001. DSRR0 points to the 3rd instruction. No ESR update. **Note:** in this case the 3rd ld/st exception is masked. This behavior is implementation dependent and may differ on other CPUs.
DVC DACx	Any instruction (no exception)	DACy, or DVC DAC_y Normal Ldst or multiple word Ldst	Take Debug exception, DBSR update setting DACx, DACy. DAC_OFST set to 3'b010. DSRR0 points to instruction after the 3rd class instruction. **Note:** in this case if x==y, then the resultant state of DBSR and DSRR0 may be indistinguishable from the "no DACy" case.

Table 1850 through *Table 1853* show some example updates for specific code sequences of dual issuing of load/store class instructions with non-load/store class instructions and the results of DAC and DVC events on selected ones of the load/store instructions.

Table 1850. DAC events and resultant updates, dual-issue case 1

Instruction sequence: The following pairs dual-issue: (1) load/store (2) alu (3) load/store (4) alu (5) load/store (6) alu	Event(s)	Result
	Instruction (1): DSI, no DAC	Take DSI exception, no DBSR update, Update ESR.
	Instruction (1): DSI, with DACx	Take Debug exception, DBSR update setting DACx and IDE, DAC_OFST set to 3'b000. DSRR0 points to instruction (1). No ESR update.
	Instruction (1): DACx	Take Debug exception, DBSR update setting DACx, DAC_OFST set to 3'b000. DSRR0 points to instruction (2). No ESR update.
	Instruction (1): DVC DACx No other exceptions	Take Debug exception, DBSR update setting DACx, DAC_OFST set to 3'b100. DSRR0 points to instruction (6). No ESR update. No debug event updates for instruction (6)

Table 1850. DAC events and resultant updates, dual-issue case 1 (continued)

Instruction sequence: The following pairs dual-issue: (1) load/store (2) alu (3) load/store (4) alu (5) load/store (6) alu	Event(s)	Result
	Instruction (1): DVC DACx Instruction (3): DSI, with or without DAC	Take Debug exception, DBSR update setting DACx, DAC_OFST set to 3'b001. DSRR0 points to instruction (3). No ESR update. No debug event updates for instructions (3)–(6). **Note:** in this case the 2nd ld/st exception is masked. This behavior is implementation dependent and may differ on other CPUs.
	Instruction (1): DVC DACx Instruction (3): DACy	Take Debug exception, DBSR update setting DACx, DACy. DAC_OFST set to 3'b010. DSRR0 points to instruction (4). No debug event updates for instructions (4)–(6). **Note:** in this case if x==y, then the resultant state of DBSR and DSRR0 may be indistinguishable from the "no DACy" case.
	Instruction (1): DVC DACx Instruction (3): DVC DACy	Take Debug exception, DBSR update setting DACx, DAC_OFST set to 3'b100. DSRR0 points to instruction (6). No ESR update. No debug event updates for instruction (6). **Note:** in this case if x==y, then the resultant state of DBSR and DSRR0 may be indistinguishable from the "no DACy" case.
	Instruction (1): DVC DACx Instruction (3): DVC DACy Instruction (5): DSI, with or without DAC	Take Debug exception, DBSR update setting DACx, DACy. DAC_OFST set to 3'b010. No ESR update. DSRR0 points to instruction (4). No debug event updates for instructions (4)–(6). **Note:** in this case if x==y, then the resultant state of DBSR and DSRR0 may be indistinguishable from the "no DACy" case. **Note:** in this case the 3rd ld/st exception is masked. This behavior is implementation dependent and may differ on other CPUs.
	Instruction (1): DVC DACx Instruction (3): DVC DACy Instruction (5): DACy or DVC DACy	Take Debug exception, DBSR update setting DACx, DACy. DAC_OFST set to 3'b100. No ESR update. DSRR0 points to instruction (6). **Note:** in this case if x==y, then the resultant state of DBSR and DSRR0 may be indistinguishable from the "no DACy" case.

Table 1851. DAC events and resultant updates, dual-issue case 2

Instruction Sequence: The following pairs dual-issue: (1) load/store (2) alu (3) load/store (4) alu (5) alu (6) load/store	**Event(s)**	**Result**
	Instruction (1): DSI, no DAC	Take DSI exception, no DBSR update, Update ESR.
	Instruction (1): DSI, with DACx	Take Debug exception, DBSR update setting DACx and IDE, DAC_OFST set to 3'b000. DSRR0 points to instruction (1). No ESR update.
	Instruction (1): DACx	Take Debug exception, DBSR update setting DACx, DAC_OFST set to 3'b000. DSRR0 points to instruction (2). No ESR update.
	Instruction (1): DVC DACx No other exceptions	Take Debug exception, DBSR update setting DACx, DAC_OFST set to 3'b101. DSRR0 points to instruction after instruction (6). No ESR update.
	Instruction (1): DVC DACx Instruction (3): DSI, with or without DAC	Take Debug exception, DBSR update setting DACx, DAC_OFST set to 3'b001. DSRR0 points to instruction (3). No ESR update. No debug event updates for instructions (3)–(6). **Note:** in this case the 2nd ld/st exception is masked. This behavior is implementation dependent and may differ on other CPUs.
	Instruction (1): DVC DACx Instruction (3): DACy	Take Debug exception, DBSR update setting DACx, DACy. DAC_OFST set to 3'b010. DSRR0 points to instruction (4). No debug event updates for instructions (4)–(6). **Note:** in this case if x==y, then the resultant state of DBSR and DSRR0 may be indistinguishable from the "no DACy" case.
	Instruction (1): DVC DACx Instruction (3): DVC DACy	Take Debug exception, DBSR update setting DACx, DAC_OFST set to 3'b101. DSRR0 points to instruction (7). No ESR update. **Note:** in this case if x==y, then the resultant state of DBSR and DSRR0 may be indistinguishable from the "no DACy" case.

Table 1851. DAC events and resultant updates, dual-issue case 2 (continued)

Instruction Sequence: The following pairs dual-issue: (1) load/store (2) alu (3) load/store (4) alu (5) alu (6) load/store	**Event(s)**	**Result**
	Instruction (1): DVC DACx Instruction (3): DVC DACy Instruction (6): DSI, with or without DAC	Take Debug exception, DBSR update setting DACx, DACy. DAC_OFST set to 3'b010. No ESR update. DSRR0 points to instruction (4). No debug event updates for instruction (4). **Note:** in this case if x==y, then the resultant state of DBSR and DSRR0 may be indistinguishable from the "no DACy" case. **Note:** in this case the 3rd ld/st exception is masked. This behavior is implementation dependent and may differ on other CPUs.
	Instruction (1): DVC DACx Instruction (3): DVC DACy Instruction (6): DACy or DVC DACy	Take Debug exception, DBSR update setting DACx, DACy. DAC_OFST set to 3'b101. No ESR update. DSRR0 points to instruction (7). No debug event updates for instruction (7). **Note:** In this case if x==y, then the resultant state of DBSR and DSRR0 may be indistinguishable from the "no DACy" case.

Table 1852. DAC events and resultant updates, dual-issue case 3

Instruction Sequence: The following pairs dual-issue: (1) load/store (2) alu (3) alu (4) alu (5) load/store (6) alu	**Event(s)**	**Result**
	Instruction (1): DSI, no DAC	Take DSI exception, no DBSR update, Update ESR.
	Instruction (1): DSI, with DACx	Take Debug exception, DBSR update setting DACx and IDE, DAC_OFST set to 3'b000. DSRR0 points to instruction (1). No ESR update.
	Instruction (1): DACx	Take Debug exception, DBSR update setting DACx, DAC_OFST set to 3'b000. DSRR0 points to instruction (2). No ESR update.

Table 1852. DAC events and resultant updates, dual-issue case 3 (continued)

Instruction Sequence: The following pairs dual-issue: (1) load/store (2) alu (3) alu (4) alu (5) load/store (6) alu	**Event(s)**	**Result**
	Instruction (1): DVC DACx No other exceptions	Take Debug exception, DBSR update setting DACx, DAC_OFST set to 3'b100. DSRR0 points to instruction (6). No ESR update. No debug event updates for instruction (6).
	Instruction (1): DVC DACx Instruction (5): DSI, with or without DAC	Take Debug exception, DBSR update setting DACx, DAC_OFST set to 3'b011. DSRR0 points to instruction (5). No ESR update. No debug event updates for instructions (5)–(6). **Note:** in this case the 2nd ld/st exception is masked. This behavior is implementation dependent and may differ on other CPUs.
	Instruction (1): DVC DACx Instruction (5): DACy	Take Debug exception, DBSR update setting DACx, DACy. DAC_OFST set to 3'b100. DSRR0 points to instruction (6). No debug event updates for instruction (6). **Note:** in this case if x==y, then the resultant state of DBSR and DSRR0 may be indistinguishable from the "no DACy" case.
	Instruction (1): DVC DACx Instruction (5): DVC DACy	Take Debug exception, DBSR update setting DACx, DAC_OFST set to 3'b100. DSRR0 points to instruction (6). No ESR update. No debug event updates for instruction (6) **Note:** in this case if x==y, then the resultant state of DBSR and DSRR0 may be indistinguishable from the "no DACy" case.

Table 1853. DAC events and resultant updates, dual-issue case 4

Instruction Sequence: The following pairs dual-issue: (1) load/store (2) alu (3) load/store (4) alu (5) alu (6) alu	**Event(s)**	**Result**
	Instruction (1): DSI, no DAC	Take DSI exception, no DBSR update, Update ESR.
	Instruction (1): DSI, with DACx	Take Debug exception, DBSR update setting DACx and IDE, DAC_OFST set to 3'b000. DSRR0 points to instruction (1). No ESR update.

Table 1853. DAC events and resultant updates, dual-issue case 4 (continued)

Instruction Sequence: The following pairs dual-issue: (1) load/store (2) alu (3) load/store (4) alu (5) alu (6) alu	Event(s)	Result
	Instruction (1): DACx	Take Debug exception, DBSR update setting DACx, DAC_OFST set to 3'b000. DSRR0 points to instruction (2). No ESR update.
	Instruction (1): DVC DACx No other exceptions	Take Debug exception, DBSR update setting DACx, DAC_OFST set to 3'b011. DSRR0 points to instruction (5). No ESR update. No debug event updates for instructions (5)–(6)
	Instruction (1): DVC DACx Instruction (3): DSI, with or without DAC	Take Debug exception, DBSR update setting DACx, DAC_OFST set to 3'b001. DSRR0 points to instruction (3). No ESR update. No debug event updates for instructions (3)–(6). **Note:** in this case the 2nd ld/st exception is masked. This behavior is implementation dependent and may differ on other CPUs.
	Instruction (1): DVC DACx Instruction (3): DACy	Take Debug exception, DBSR update setting DACx, DACy. DAC_OFST set to 3'b010. DSRR0 points to instruction (4). No debug event updates for instructions (4)–(6). **Note:** in this case if x==y, then the resultant state of DBSR and DSRR0 may be indistinguishable from the "no DACy" case.
	Instruction (1): DVC DACx Instruction (3): DVC DACy	Take Debug exception, DBSR update setting DACx, DAC_OFST set to 3'b011. DSRR0 points to instruction (5). No ESR update. No debug event updates for instructions (5)–(6). **Note:** in this case if x==y, then the resultant state of DBSR and DSRR0 may be indistinguishable from the "no DACy" case.

71.2.3 Linked Instruction Address Compare and Data Address Compare events

Data Address Compare 1, 2, 3, and 4 debug events may be 'linked' with an Instruction Address Compare event by setting the DAC[1–4]LNK control bits in DBCR2 and DBCR8 to further refine when a Data Address Compare debug event is generated. DAC1 may be linked with IAC1, DAC2 (when not used as a mask or range bounds register) may be linked with IAC3. DAC3 may be linked with IAC5, DAC4 (when not used as a mask or range bounds register) may be linked with IAC7. When linked, a DAC debug event occurs when the same instruction that generates the DAC 'hit' also generates a corresponding linked IAC 'hit'. When linked, the IAC event is not recorded in the Debug Status register, regardless of whether a corresponding linked DAC event occurs, or whether the IAC event enable is set.

When enabled and execution of a load or store class instruction results in a data access with an address that meets the criteria specified in the DBCRx, DACx, and DVCx Registers, and the instruction also meets the criteria for generating an Instruction Address Compare event, a Linked Data Address Compare debug event occurs. This event can occur and be

recorded in DBSR regardless of the setting of MSR_{DE}. The normal DAC1 status bit in the DBSR is used for recording these events. The IAC status bit is not set if the corresponding Instruction Address Compare register is linked.

Linking is enabled using control bits in DBCR2 and DBCR8. Note that linking is only available in EDM or IDM. Attempts to use linking otherwise are ignored.

Note: Linked DAC events will not be recorded if a load multiple word or store multiple word type instruction is interrupted prior to completion by a critical input or external input interrupt.

71.2.4 Trap debug event

A Trap debug event (TRAP) occurs if Trap debug events are enabled ($DBCR0_{TRAP}$=1), a Trap instruction (**tw**) is executed, and the conditions specified by the instruction for the trap are met. This event can occur and be recorded in DBSR regardless of the setting of MSR_{DE}. When a Trap debug event occurs, the $DBSR_{TRAP}$ bit is set to 1 to record the debug exception.

71.2.5 Branch Taken debug event

A Branch Taken debug event (BRT) occurs if Branch Taken debug events are enabled ($DBCR0_{BRT}$=1) and execution is attempted of a branch instruction that will be taken (either an unconditional branch, or a conditional branch whose branch condition is true), and MSR_{DE}=1. Branch Taken debug events are not recognized if MSR_{DE}=0 at the time of execution of the branch instruction and thus $DBSR_{IDE}$ cannot be set by a Branch Taken debug event. When a Branch Taken debug event is recognized, the $DBSR_{BRT}$ bit is set to 1 to record the debug exception, and the address of the branch instruction will be recorded in DSRR0.

71.2.6 Instruction Complete debug event

An Instruction Complete debug event (ICMP) occurs if Instruction Complete debug events are enabled ($DBCR0_{ICMP}$=1), execution of any instruction is completed, and MSR_{DE}=1. If execution of an instruction is suppressed due to the instruction causing some other exception that is enabled to generate an interrupt, then the attempted execution of that instruction does not cause an Instruction Complete debug event. The ***se_sc*** instruction does not fall into the category of an instruction whose execution is suppressed, since the instruction actually executes and then generates a System Call interrupt. In this case, the Instruction Complete debug exception will also be set. When an Instruction Complete debug event is recognized, $DBSR_{ICMP}$ is set to 1 to record the debug exception and the address of the next instruction to be executed will be recorded in DSRR0.

Instruction Complete debug events are not recognized if MSR_{DE}=0 at the time of execution of the instruction, thus $DBSR_{IDE}$ is not generally set by an ICMP debug event.

One circumstance may cause the $DBSR_{ICMP}$ and $DBSR_{IDE}$ bits to be set. This occurs when a EFPU Round exception occurs. Since the instruction is by definition completed (SRR0 points to the following instruction), this interrupt takes higher priority than the Debug interrupt so as not to be lost, and $DBSR_{IDE}$ is set to indicate the imprecise recognition of a Debug interrupt. In this case, the Debug interrupt will be taken with SRR0 pointing to the instruction following the instruction that generated the EFPU Round exception, and DSRR0 will point to the Round exception handler. In addition to occurring when $DBCR0_{IDM}$=1, this circumstance can also occur when $EDBCR0_{EDM}$=1 and the event is hardware-owned, in which case $EDBSR0_{IDE}$ will be set.

Note: Instruction complete debug events are not generated by the execution of an instruction that sets MSR_{DE} to '1' while $DBCR0_{ICMP}$=1, nor by the execution of an instruction that sets $DBCR0_{ICMP}$ to '1' while MSR_{DE}=1.

71.2.7 Interrupt Taken debug event

An Interrupt Taken debug event (IRPT) occurs if Interrupt Taken debug events are enabled ($DBCR0_{IRPT}$=1) and a base-class interrupt occurs. Only base-class interrupts (an interrupt using SRR0/1) cause an Interrupt Taken debug event. This event can occur and be recorded in DBSR regardless of the setting of MSR_{DE}. When an Interrupt Taken debug event occurs, the $DBSR_{IRPT}$ bit is set to 1 to record the debug exception. The value saved in DSRR0 will be the address of the non-critical interrupt handler.

71.2.8 Critical Interrupt Taken debug event

A Critical Interrupt Taken debug event (CIRPT) occurs if Critical Interrupt Taken debug events are enabled ($DBCR0_{CIRPT}$=1) and a critical interrupt occurs. Only critical class interrupts (an interrupt using CSRR0/1) cause a Critical Interrupt Taken debug event. This event can occur and be recorded in DBSR regardless of the setting of MSR_{DE}. When a Critical Interrupt Taken debug event occurs, the $DBSR_{CIRPT}$ bit is set to 1 to record the debug exception. The value saved in DSRR0 will be the address of the critical interrupt handler.

71.2.9 Return debug event

A Return debug event (RET) occurs if Return debug events are enabled ($DBCR0_{RET}$=1) and an attempt is made to execute an **se_rfi** instruction. This event can occur and be recorded in DBSR regardless of the setting of MSR_{DE}. When a Return debug event occurs, the $DBSR_{RET}$ bit is set to 1 to record the debug exception.

If MSR_{DE}=0 at the time of the execution of the **se_rfi** (that is before the MSR is updated by the **se_rfi**), then $DBSR_{IDE}$ is also set to 1 to record the imprecise debug event.

If MSR_{DE}=1 at the time of the execution of the **se_rfi**, a Debug interrupt will occur provided there exists no higher priority exception that is enabled to cause an interrupt. Debug Save/Restore Register 0 will be set to the address of the **se_rfi** instruction.

71.2.10 Critical Return Debug Event

A Critical Return debug event (CRET) occurs if Critical Return debug events are enabled ($DBCR0_{CRET}$=1) and an attempt is made to execute an **se_rfci** instruction. This event can occur and be recorded in DBSR regardless of the setting of MSR_{DE}. When a Critical Return debug event occurs, the $DBSR_{CRET}$ bit is set to 1 to record the debug exception.

If MSR_{DE}=0 at the time of the execution of the **se_rfci** (that is before the MSR is updated by the **se_rfci**), then $DBSR_{IDE}$ is also set to 1 to record the imprecise debug event.

If MSR_{DE}=1 at the time of the execution of the **se_rfci**, a Debug interrupt will occur provided there exists no higher priority exception that is enabled to cause an interrupt. Debug Save/Restore Register 0 will be set to the address of the **se_rfci** instruction.

71.2.11 External debug event

An External debug event (DEVT1, DEVT2) occurs if External debug events are enabled ($DBCR0_{DEVT1}$=1 or $DBCR0_{DEVT2}$=1), and the respective **p_devt1** or **p_devt2** input signal

transitions to the asserted state while the CPU is not in the Stopped state. This event can occur and be recorded in DBSR regardless of the setting of MSR_{DE}. When an External debug event occurs, $DBSR_{DEVT\{1,2\}}$ is set to '1' to record the debug exception. This debug event is an asynchronous event, but is only sampled when the CPU **m_clk** is active.

71.2.12 Unconditional debug event

An Unconditional debug event (UDE) occurs when the Unconditional Debug Event (**p_ude**) input transitions to the asserted state. The Unconditional debug event is the only debug event that does not have a corresponding enable bit for the event in DBCR0. This event can occur and be recorded in DBSR regardless of the setting of MSR_{DE}. When an Unconditional debug event occurs, the $DBSR_{UDE}$ bit is set to '1' to record the debug exception. This debug event is an asynchronous event.

71.2.13 Performance Monitor Interrupt debug event

A Performance Monitor Interrupt debug event (PMI) occurs if Performance Monitor Interrupt debug events are enabled ($PMGC0_{UDI}$=1), and a performance monitor interrupt event occurs. This event can occur and be recorded in DBSR regardless of the setting of MSR_{DE}. When a Performance Monitor Interrupt debug event occurs, $DBSR_{PMI}$ is set to '1' to record the debug exception. This debug event is an asynchronous event.

71.3 Debug registers

This section describes debug-related registers that are software accessible. These registers are intended for use by special debug tools and debug software, not by general application code.

Access to these registers (other than DBSR) by software is conditioned by the External Debug mode control bit ($EDBCR0_{EDM}$) and the settings of debug control register EDBRAC0, which can be set by the hardware debug port. If $EDBCR0_{EDM}$ is set and if the bit in EDBRAC0 corresponding to the resource is cleared, software is prevented from modifying debug register values other than in DBSR, since the resource is not "owned" by software. Software always has ownership of DBSR. Execution of a **mtspr** instruction targeting a debug register or register field not "owned" by software will not cause modifications to occur, and no exception will be signaled. In addition, since the external debugger hardware may be manipulating debug register values, the state of these registers or register fields not "owned" by software is not guaranteed to be consistent if accessed (read) by software with a **mfspr** instruction, other than the $DBCR0_{EDM}$ bit itself and the EDBRAC0 register. Hardware always has full access to all registers and all register fields through the OnCE register access mechanism, and it is up to the debug firmware to properly implement modifications to these registers with read-modify-write operations to implement any control sharing with software. Settings in EDBRAC0 should be considered by the debug firmware in order to preserve software settings of control registers as appropriate when hardware modifications to the debug registers is performed.

71.3.1 Debug Address and Value registers

Instruction Address Compare registers IAC1–8 are used to hold instruction addresses for address comparison purposes. In addition, IAC2 and IAC4 may hold mask information for IAC1 and IAC3 respectively and IAC6 and IAC8 may hold mask information for IAC5 and IAC7 respectively, when *Address Bit Match* compare modes are selected. Note that when

performing instruction address compares, the low order bit of the instruction address and the corresponding IAC register is ignored for VLE instructions.

Data Address Compare registers DAC1–4 are used to hold data access addresses for address comparison purposes. In addition, DAC2 and DAC4 may hold mask information for DAC1 and DAC3 respectively when *Address Bit Match* compare modes are selected.

Data Value Compare registers DVC1 and DVC2 are used to hold data values for data comparison purposes. DVC1 and DVC2 are 64-bit registers. Data value comparisons are used to qualify Data Address compare 1 and 2 debug events. Data value comparisons are not available for Data address compare 3–4 events. DVC1 is associated with DAC1, and DVC2 is associated with DAC2. The most significant byte of the DVC1(2) register (labeled B0 in *Figure 1918*) corresponds to the byte data value transferred to/from memory byte offset such as 0, 8, and the least significant byte of the register (labeled B7 in *Figure 1918*) corresponds to byte offset such as 7, F. When enabled for performing data value comparisons, each enabled byte in DVC1(2) is compared with the memory value transferred on the corresponding active byte lane of the data memory interface to determine if a match occurs. Inactive byte lanes do not participate in the comparison; they are implicitly masked. The Byte Strobe Assertion for Transfers table in the Core (e200z420n3) Core Complex Overview chapter shows active byte lanes for data transfers. Software must also program the DVC1(2) register byte positions based on the alignment of the access. Misaligned accesses are not fully supported, since the data address and data value comparisons are only performed on the initial access in the case of a misaligned access; thus, accesses that cross a 64-bit boundary cannot be fully matched. For address and size combinations that involve two transfers, only the initial transfer is used for data address and value matching.

DVC1 and DVC2 may be read or written using **mtspr** and **mfspr** instructions. The DVC1U and DVC2U SPR numbers (601,602) are used for accessing the upper 32-bit portion of the DVC register. The DVC1 and DVC2 SPR numbers (318,319) are used for accessing the lower 32-bit portion of the DVC register.

B0								B1								B2								B3							
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
B4								B5								B6								B7							
32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63

SPR 601 (DVC1U), 318 (DVC1), 602 (DVC2U) 319 (DVC2); Read/Write; Reset: Unaffected

Figure 1918. DVC1, DVC2 registers

71.3.2 Debug Control and Status registers

Debug Control Registers (DBCR0–8 and EDBRAC0) are used to enable debug events, reset the processor, and set the debug mode of the processor. The Debug Status register (DBSR) records debug exceptions while Internal Debug mode is enabled. The Debug Data Effective Address register (DDEAR) records the effective address of a Data Address Compare event while Internal Debug mode is enabled.

e200z420n3 requires that a context synchronizing instruction follow a **mtspr** DBCR0–8 or DBSR to ensure that any alterations enabling/disabling debug events are effective. The context synchronizing instruction may or may not be affected by the alteration. Typically, an **se_isync** instruction is used to create a synchronization boundary beyond which it can be guaranteed that the newly written control values are in effect.

For watchpoint generation, configuration settings contained in DBCR1–8 are used, even though the corresponding event(s) may be disabled (via DBCR0) from setting DBSR flags.

71.3.2.1 Debug Control Register 0 (DBCR0)

Debug Control Register 0 is used to enable debug modes and controls which debug events are allowed to set DBSR or EDBSR0 flags. e200z420n3 adds some implementation specific bits to this register, as seen in *Figure 1919*.

EDM	IDM	RST	ICMP	BRT	IRPT	TRAP	IAC1	IAC2	IAC3	IAC4	DAC1	DAC2	RET	IAC5	IAC6	IAC7	IAC8	DEVT1	DEVT2	0	CIRPT	CRET	0
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31

SPR - 308; Read/Write; Reset(1) - 0x0

Figure 1919. Debug Control Register 0 (DBCR0) register

1. $DBCR0_{EDM}$ is affected by **j_trst_b** or **m_por** assertion, and remains reset while in the Test_Logic_Reset state, but is not affected by **p_reset_b**. All other bits are reset by processor reset **p_reset_b** if $DBCR0_{EDM}$=0, as well as unconditionally by **m_por**. If $DBCR0_{EDM}$=1, EDBRAC0 masks off hardware-owned resources (other than RST) from reset by **p_reset_b,** and only software-owned resources indicated by EDBRAC0 and the $DBCR0_{RST}$ field will be reset by **p_reset_b**. The $DBCR0_{RST}$ field will always be reset by **p_reset_b** regardless of the value of $DBCR0_{EDM}$.

Table 1854 provides bit definitions for Debug Control Register 0.

Table 1854. DBCR0 field descriptions

Bit	Name	Description
0	EDM	External Debug mode. This bit is read-only by software. When external debug mode is enabled, hardware-owned resources in debug registers are not affected by processor reset **p_reset_b**. This allows the debugger to set up hardware debug events that remain active across a processor reset. 0 External debug mode disabled. Internal debug events not mapped into external debug events. 1 External debug mode enabled. Hardware-owned debug events will not cause the CPU to vector to interrupt code. Software is not permitted to write to debug registers {DBCR0–8, IAC1–8, DAC1–4, DVC1–2[U]} unless permitted by settings in EDBRAC0. Hardware-owned events will set status bits in EDBSR0.
		Programming Notes: It is recommended that debug status bits in the Debug Status Registers be cleared before disabling external debug mode to avoid any internal imprecise debug interrupts. Software may use this bit to determine if external debug has control over the debug registers. The hardware debugger must set the EDM bit to '1' before other bits in this register (and other debug registers) may be altered. On the initial setting of this bit to '1', all other bits are unchanged. This bit is only writable through the OnCE port.
1	IDM	Internal Debug mode 0 Debug exceptions are disabled. Debug events do not affect DBSR. 1 Debug exceptions are enabled. Enabled debug events owned by software will update the DBSR. If MSR_{DE}=1, the occurrence of a debug event, or the recording of an earlier debug event in the Debug Status Register when MSR_{DE} was cleared, will cause a Debug interrupt.

Table 1854. DBCR0 field descriptions (continued)

Bit	Name	Description
2:3	RST	Reset Control 00 No function 01 **p_dbrstc[1]** pin asserted by Debug Reset Control. Allows external device to initiate processor or system reset 10 **p_dbrstc[0]** pin asserted by Debug Reset Control. Allows external device to initiate processor or system reset. 11 Reserved
4	ICMP	Instruction Complete Debug Event Enable 0 ICMP debug events are disabled 1 ICMP debug events are enabled
5	BRT	Branch Taken Debug Event Enable 0 BRT debug events are disabled 1 BRT debug events are enabled
6	IRPT	Interrupt Taken Debug Event Enable 0 IRPT debug events are disabled 1 IRPT debug events are enabled
7	TRAP	Trap Taken Debug Event Enable 0 TRAP debug events are disabled 1 TRAP debug events are enabled
8	IAC1	Instruction Address Compare 1 Debug Event Enable 0 IAC1 debug events are disabled 1 IAC1 debug events are enabled
9	IAC2	Instruction Address Compare 2 Debug Event Enable 0 IAC2 debug events are disabled 1 IAC2 debug events are enabled
10	IAC3	Instruction Address Compare 3 Debug Event Enable 0 IAC3 debug events are disabled 1 IAC3 debug events are enabled
11	IAC4	Instruction Address Compare 4 Debug Event Enable 0 IAC4 debug events are disabled 1 IAC4 debug events are enabled
12:13	DAC1	Data Address Compare 1 Debug Event Enable 00 DAC1 debug events are disabled 01 DAC1 debug events are enabled only for store-type data storage accesses 10 DAC1 debug events are enabled only for load-type data storage accesses 11 DAC1 debug events are enabled for load-type or store-type data storage accesses
14:15	DAC2	Data Address Compare 2 Debug Event Enable 00 DAC2 debug events are disabled 01 DAC2 debug events are enabled only for store-type data storage accesses 10 DAC2 debug events are enabled only for load-type data storage accesses 11 DAC2 debug events are enabled for load-type or store-type data storage accesses
16	RET	Return Debug Event Enable 0 RET debug events are disabled 1 RET debug events are enabled

Table 1854. DBCR0 field descriptions (continued)

Bit	Name	Description
17	IAC5	Instruction Address Compare 5 Debug Event Enable 0 IAC5 debug events are disabled 1 IAC5 debug events are enabled
18	IAC6	Instruction Address Compare 6 Debug Event Enable 0 IAC6 debug events are disabled 1 IAC6 debug events are enabled
19	IAC7	Instruction Address Compare 7 Debug Event Enable 0 IAC7 debug events are disabled 1 IAC7 debug events are enabled
20	IAC8	Instruction Address Compare 8 Debug Event Enable 0 IAC8 debug events are disabled 1 IAC8 debug events are enabled
21	DEVT1	External Debug Event 1 Enable 0 DEVT1 debug events are disabled 1 DEVT1 debug events are enabled
22	DEVT2	External Debug Event 2 Enable 0 DEVT2 debug events are disabled 1 DEVT2 debug events are enabled
23:24	—	Reserved
25	CIRPT	Critical Interrupt Taken Debug Event Enable 0 CIRPT debug events are disabled 1 CIRPT debug events are enabled
26	CRET	Critical Return Debug Event Enable 0 CRET debug events are disabled 1 CRET debug events are enabled
27:31	—	Reserved

71.3.2.2 Debug Control Register 1 (DBCR1)

Debug Control Register 1 is used to configure Instruction Address Compare operation. The DBCR1 register is shown in *Figure 1920*.

IAC1US	IAC1ER	IAC2US	IAC2ER	IAC12M	0	IAC3US	IAC3ER	IAC4US	IAC4ER	IAC34M	0
0 1	2 3	4 5	6 7	8 9	10 11 12 13 14 15	16 17	18 19	20 21	22 23	24 25	26 27 28 29 30 31

SPR - 309; Read/Write; Reset(1) - 0x0

Figure 1920. DBCR1 register

1. Reset by processor reset **p_reset_b** if $EDBCR0_{EDM}$=0, as well as unconditionally by **m_por**. If $EDBCR0_{EDM}$=1, EDBRAC0 masks off hardware-owned resources from reset by **p_reset_b** and only software-owned resources indicated by EDBRAC0 will be reset by **p_reset_b.**

Table 1855 provides bit definitions for Debug Control Register 1.

Table 1855. DBCR1 field descriptions

Bit	Name	Description
0:1	IAC1US	Instruction Address Compare 1 User/Supervisor Mode 00 IAC1 debug events not affected by MSR_{PR} 01 Reserved 10 IAC1 debug events can only occur if MSR_{PR}=0 (Supervisor mode) 11 IAC1 debug events can only occur if MSR_{PR}=1. (User mode)
2:3	IAC1ER	Instruction Address Compare 1 Effective/Real Mode 00 IAC1 debug events are based on effective address 01 Unimplemented (Book E real address compare), no match can occur 10 IAC1 debug events are based on effective address and can only occur if MSR_{IS}=0 11 IAC1 debug events are based on effective address and can only occur if MSR_{IS}=1
4:5	IAC2US	Instruction Address Compare 2 User/Supervisor Mode 00 IAC2 debug events not affected by MSR_{PR} 01 Reserved 10 IAC2 debug events can only occur if MSR_{PR}=0 (Supervisor mode) 11 IAC2 debug events can only occur if MSR_{PR}=1. (User mode)
6:7	IAC2ER	Instruction Address Compare 2 Effective/Real Mode 00 IAC2 debug events are based on effective address 01 Unimplemented (Book E real address compare), no match can occur 10 IAC2 debug events are based on effective address and can only occur if MSR_{IS}=0 11 IAC2 debug events are based on effective address and can only occur if MSR_{IS}=1
8:9	IAC12M	Instruction Address Compare 1/2 Mode 00 Exact address compare. IAC1 debug events can only occur if the address of the instruction fetch is equal to the value specified in IAC1. IAC2 debug events can only occur if the address of the instruction fetch is equal to the value specified in IAC2. 01 Address bit match. IAC1 debug events can occur only if the address of the instruction fetch, ANDed with the contents of IAC2 are equal to the contents of IAC1, also ANDed with the contents of IAC2. IAC2 debug events do not occur. IAC1US and IAC1ER settings are used. 10 Inclusive address range compare. IAC1 debug events can occur only if the address of the instruction fetch is greater than or equal to the value specified in IAC1 and less than the value specified in IAC2. IAC2 debug events do not occur. IAC1US and IAC1ER settings are used. 11 Exclusive address range compare. IAC1 debug events can occur only if the address of the instruction fetch is less than the value specified in IAC1 or is greater than or equal to the value specified in IAC2. IAC2 debug events do not occur. IAC1US and IAC1ER settings are used.
10:15	—	Reserved
16:17	IAC3US	Instruction Address Compare 3 User/Supervisor Mode 00 IAC3 debug events not affected by MSR_{PR} 01 Reserved 10 IAC3 debug events can only occur if MSR_{PR}=0 (Supervisor mode) 11 IAC3 debug events can only occur if MSR_{PR}=1 (User mode)
18:19	IAC3ER	Instruction Address Compare 3 Effective/Real Mode 00 IAC3 debug events are based on effective address 01 Unimplemented (Book E real address compare), no match can occur 10 IAC3 debug events are based on effective address and can only occur if MSR_{IS}=0 11 IAC3 debug events are based on effective address and can only occur if MSR_{IS}=1

Table 1855. DBCR1 field descriptions (continued)

Bit	Name	Description
20:21	IAC4US	Instruction Address Compare 4 User/Supervisor Mode 00 IAC4 debug events not affected by MSR_{PR} 01 Reserved 10 IAC4 debug events can only occur if MSR_{PR}=0 (Supervisor mode). 11 IAC4 debug events can only occur if MSR_{PR}=1. (User mode)
22:23	IAC4ER	Instruction Address Compare 4 Effective/Real Mode 00 IAC4 debug events are based on effective address 01 Unimplemented (Book E real address compare), no match can occur 10 IAC4 debug events are based on effective address and can only occur if MSR_{IS}=0 11 IAC4 debug events are based on effective address and can only occur if MSR_{IS}=1
24:25	IAC34M	Instruction Address Compare 3/4 Mode 00 Exact address compare. IAC3 debug events can only occur if the address of the instruction fetch is equal to the value specified in IAC3. IAC4 debug events can only occur if the address of the instruction fetch is equal to the value specified in IAC4. 01 Address bit match. IAC3 debug events can occur only if the address of the instruction fetch, ANDed with the contents of IAC4 are equal to the contents of IAC3, also ANDed with the contents of IAC4. IAC4 debug events do not occur. IAC3US and IAC3ER settings are used. 10 Inclusive address range compare. IAC3 debug events can occur only if the address of the instruction fetch is greater than or equal to the value specified in IAC3 and less than the value specified in IAC4. IAC4 debug events do not occur. IAC3US and IAC3ER settings are used. 11 Exclusive address range compare. IAC3 debug events can occur only if the address of the instruction fetch is less than the value specified in IAC3 or is greater than or equal to the value specified in IAC4. IAC4 debug events do not occur. IAC3US and IAC3ER settings are used.
26:31	—	Reserved

71.3.2.3 Debug Control Register 2 (DBCR2)

Debug Control Register 2 is used to configure Data Address Compare and Data Value Compare operation.The DBCR2 register is shown in *Figure 1921*.

DAC1US	DAC1ER	DAC2US	DAC2ER	DAC12M	DAC1LNK	DAC2LNK	DVC1M	DVC2M	DVC1BE	DVC2BE
0 1	2 3	4 5	6 7	8 9	10	11	12 13	14 15	16 17 18 19 20 21 22 23	24 25 26 27 28 29 30 31

SPR - 310; Read/Write; Reset(1) - 0x0

Figure 1921. DBCR2 register

1. Reset by processor reset **p_reset_b** if $EDBCR0_{EDM}$=0, as well as unconditionally by **m_por**. If $EDBCR0_{EDM}$=1, EDBRAC0 masks off hardware-owned resources from reset by **p_reset_b** and only software-owned resources indicated by EDBRAC0 will be reset by **p_reset_b**.

Table 1856 provides bit definitions for Debug Control Register 2.

Table 1856. DBCR2 field descriptions

Bit	Name	Description
0:1	DAC1US	Data Address Compare 1 User/Supervisor Mode 00 DAC1 debug events not affected by MSR_{PR} 01 Reserved 10 DAC1 debug events can only occur if MSR_{PR}=0 (Supervisor mode) 11 DAC1 debug events can only occur if MSR_{PR}=1 (User mode)
2:3	DAC1ER	Data Address Compare 1 Effective/Real Mode 00 DAC1 debug events are based on effective address 01 Unimplemented (Book E real address compare), no match can occur 10 DAC1 debug events are based on effective address and can only occur if MSR_{DS}=0 11 DAC1 debug events are based on effective address and can only occur if MSR_{DS}=1
4:5	DAC2US	Data Address Compare 2 User/Supervisor Mode 00 DAC2 debug events not affected by MSR_{PR} 01 Reserved 10 DAC2 debug events can only occur if MSR_{PR}=0 (Supervisor mode) 11 DAC2 debug events can only occur if MSR_{PR}=1. (User mode)
6:7	DAC2ER	Data Address Compare 2 Effective/Real Mode 00 DAC2 debug events are based on effective address 01 Unimplemented (Book E real address compare), no match can occur 10 DAC2 debug events are based on effective address and can only occur if MSR_{DS}=0 11 DAC2 debug events are based on effective address and can only occur if MSR_{DS}=1
8:9	DAC12M	Data Address Compare 1/2 Mode 00 Exact address compare. DAC1 debug events can only occur if the address of the data access is equal to the value specified in DAC1. DAC2 debug events can only occur if the address of the data access is equal to the value specified in DAC2. 01 Address bit match. DAC1 debug events can occur only if the address of the data access ANDed with the contents of DAC2, are equal to the contents of DAC1 also ANDed with the contents of DAC2. DAC2 debug events do not occur. DAC1US and DAC1ER settings are used. 10 Inclusive address range compare. DAC1 debug events can occur only if the address of the data access is greater than or equal to the value specified in DAC1 and less than the value specified in DAC2. DAC2 debug events do not occur. DAC1US and DAC1ER settings are used. 11 Exclusive address range compare. DAC1 debug events can occur only if the address of the data access is less than the value specified in DAC1 or is greater than or equal to the value specified in DAC2. DAC2 debug events do not occur. DAC1US and DAC1ER settings are used.

Table 1856. DBCR2 field descriptions (continued)

Bit	Name	Description
10	DAC1LNK	Data Address Compare 1 Linked. When linked to IAC1, DAC1 debug events are conditioned based on whether the instruction also generated an IAC1 debug event. Note that linking is only available in EDM or IDM. 0 No effect 1 DAC1 debug events are linked to IAC1 debug events. IAC1 debug events do not affect DBSR
11	DAC2LNK	Data Address Compare 2 Linked. When linked to IAC3, DAC2 debug events are conditioned based on whether the instruction also generated an IAC3 debug event. DAC2 can only be linked if DAC12M specifies *Exact Address Compare* since DAC2 debug events are not generated in the other compare modes. Note that linking is only available in EDM or IDM. 0 No effect 1 DAC 2 debug events are linked to IAC3 debug events. IAC3 debug events do not affect DBSR.
12:13	DVC1M	Data Value Compare 1 Mode When $DBCR4_{DVC1C}$=0: 00 DAC1 debug events not affected by data value compares. 01 DAC1 debug events can only occur when all bytes specified in the DVC1BE field match the corresponding data byte values for active byte lanes of the memory access. 10 DAC1 debug events can only occur when any byte specified in the DVC1BE field matches the corresponding data byte value for active byte lanes of the memory access. 11 DAC1 debug events can only occur when all bytes specified in the DVC1BE field within at least one of the halfwords of the data value of the memory access matches the corresponding DVC1 value. **Note:** Inactive byte lanes of the memory access are automatically masked. When $DBCR4_{DVC1C}$=1: 00 Reserved 01 DAC1 debug events can only occur when any byte specified in the DVC1BE field does not match the corresponding data byte value for active byte lanes of the memory access. If all active bytes match, then no event will be generated. 10 DAC1 debug events can only occur when all bytes specified in the DVC1BE field do not match the corresponding data byte values for active byte lanes of the memory access. If any active byte match occurs, no event will be generated. 11 Reserved **Note:** Inactive byte lanes of the memory access are automatically masked.

Table 1856. DBCR2 field descriptions (continued)

Bit	Name	Description
14:15	DVC2M	Data Value Compare 2 Mode When $DBCR4_{DVC2C}$=0: 00 DAC2 debug events not affected by data value compares. 01 DAC2 debug events can only occur when all bytes specified in the DVC2BE field match the corresponding data byte values for active byte lanes of the memory access. 10 DAC2 debug events can only occur when any byte specified in the DVC2BE field matches the corresponding data byte value for active byte lanes of the memory access. 11 DAC2 debug events can only occur when all bytes specified in the DVC2BE field within at least one of the halfwords of the data value of the memory access matches the corresponding DVC2 value. **Note:** Inactive byte lanes of the memory access are automatically masked. When $DBCR4_{DVC2C}$=1: 00 Reserved 01 DAC2 debug events can only occur when any byte specified in the DVC2BE field does not match the corresponding data byte value for active byte lanes of the memory access. If all active bytes match, then no event will be generated. 10 DAC2 debug events can only occur when all bytes specified in the DVC2BE field do not match the corresponding data byte values for active byte lanes of the memory access. If any active byte match occurs, no event will be generated. 11 Reserved **Note:** Inactive byte lanes of the memory access are automatically masked.
16:23	DVC1BE	Data Value Compare 1 Byte Enables Specifies which bytes in the aligned doubleword value associated with the memory access are compared to the corresponding bytes in DVC1. Inactive byte lanes of a memory access smaller than 64 bits are automatically masked by hardware. If all bits in the DVC1BE field are clear, then a match will occur regardless of the data. Misaligned accesses that cross a doubleword boundary are not fully supported. 1*xxxxxxx* Byte lane 0 is enabled for comparison with the value in bits 0:7 of DVC1. *x*1*xxxxxx* Byte lane 1 is enabled for comparison with the value in bits 8:15 of DVC1. *xx*1*xxxxx* Byte lane 2 is enabled for comparison with the value in bits 16:23 of DVC1. *xxx*1*xxxx* Byte lane 3 is enabled for comparison with the value in bits 24:31 of DVC1. *xxxx*1*xxx* Byte lane 4 is enabled for comparison with the value in bits 32:39 of DVC1. *xxxxx*1*xx* Byte lane 5 is enabled for comparison with the value in bits 40:47 of DVC1. *xxxxxx*1*x* Byte lane 6 is enabled for comparison with the value in bits 48:55 of DVC1. *xxxxxxx*1 Byte lane 7 is enabled for comparison with the value in bits 56:63 of DVC1.
24:31	DVC2BE	Data Value Compare2 Byte Enables Specifies which bytes in the aligned doubleword value associated with the memory access are compared to the corresponding bytes in DVC2. Inactive byte lanes of a memory access smaller than 64 bits are automatically masked by hardware. If all bits in the DVC1BE field are clear, then a match will occur regardless of the data. Misaligned accesses that cross a doubleword boundary are not fully supported. 1*xxxxxxx* Byte lane 0 is enabled for comparison with the value in bits 0:7 of DVC2. *x*1*xxxxxx* Byte lane 1 is enabled for comparison with the value in bits 8:15 of DVC2. *xx*1*xxxxx* Byte lane 2 is enabled for comparison with the value in bits 16:23 of DVC2. *xxx*1*xxxx* Byte lane 3 is enabled for comparison with the value in bits 24:31 of DVC2. *xxxx*1*xxx* Byte lane 4 is enabled for comparison with the value in bits 32:39 of DVC2. *xxxxx*1*xx* Byte lane 5 is enabled for comparison with the value in bits 40:47 of DVC2. *xxxxxx*1*x* Byte lane 6 is enabled for comparison with the value in bits 48:55 of DVC2. *xxxxxxx*1 Byte lane 7 is enabled for comparison with the value in bits 56:63 of DVC2.

71.3.2.4 Debug Control Register 4 (DBCR4)

Debug Control Register 4 is used to extend data address and value compare matching functionality. DBCR4 is shown in *Figure 1922*.

0	DVC1C	0	DVC2C	0	0	DAC1XMH	0	DAC2XMH	DAC1XM	DAC2XM	DAC1CFG	DAC2CFG
0	1	2	3	4:5	6:11	12	13	14	15	16:19	20:23	24:27	28:29	30:31

SPR - 563; Read/Write; Reset(1) - 0x0

Figure 1922. DBCR4 register

1. DBCR4 is reset by processor reset **p_reset_b** if $EDBCR0_{EDM}$=0, as well as unconditionally by **m_por**. If $EDBCR0_{EDM}$=1, EDBRAC0 masks off hardware-owned resources from reset by **p_reset_b** and only software-owned resources indicated by EDBRAC0 will be reset by **p_reset_b**.

Table 1857 provides bit definitions for Debug Control Register 4.

Table 1857. DBCR4 field descriptions

Bit	Name	Description
0	—	Reserved
1	DVC1C	Data Value Compare 1 Control. DVC1C controls whether DVC1 data value comparisons utilize the normal compare operation, or an alternate “inverted compare” operation. In inverted polarity mode, data value compares perform a not-equal comparison. Refer to details in the DBCR2 register definition. 0 Normal DVC1 operation. 1 Inverted polarity DVC1 operation
2	—	Reserved
3	DVC2C	Data Value Compare 2 Control. DVC2C controls whether DVC2 data value comparisons utilize the normal compare operation, or an alternate “inverted compare” operation. In inverted polarity mode, data value compares perform a not-equal comparison. Refer to details in the DBCR2 register definition 0 Normal DVC2 operation. 1 Inverted polarity DVC2 operation
4:12	—	Reserved
13	DAC1XMH	Data Address Compare 1 Extended Mask Control High. DAC1XMH extends the range of the DAC1XM field 0 DAC1XM masks 0–15 low-order address bits 1 DAC1XM masks 16–31 low-order address bits
14	—	Reserved
15	DAC2XMH	Data Address Compare 2 Extended Mask Control High. DAC2XMH extends the range of the DAC2XM field. 0 DAC2XM masks 0–15 low-order address bits 1 DAC2XM masks 16–31 low-order address bits

Table 1857. DBCR4 field descriptions (continued)

Bit	Name	Description
16:19	DAC1XM	Data Address Compare 1 Extended Mask Control. DAC1XM allows for binary power of 2 address range compares for DAC1 without requiring the use of DAC2. Value of DAC1XMH || DAC1XM: 00000 No additional masking when DBCR2[DAC12M] = 00 00001–11111 Exact Match Bit Mask. One to 31 low-order bits are masked in DAC1 when comparing the storage address with the value in DAC1 for exact address compare (DBCR2[DAC12M] = 00). Address ranges of 2 bytes to 2GB are supported.
20:23	DAC2XM	Data Address Compare 2 Extended Mask Control Value of DAC2XMH || DAC2XM: 00000 No additional masking when DBCR2[DAC12M] = 00 00001–11111 Exact Match Bit Mask. One to 31 low-order bits are masked in DAC2 when comparing the storage address with the value in DAC2 for exact address compare (DBCR2[DAC12M] = 00). Address ranges of 2 bytes to 2GB are supported. DAC2XM allows for binary power of 2 address range compares for DAC2.

Table 1857. DBCR4 field descriptions (continued)

Bit	Name	Description
24:27	DAC1CFG	Data Address Compare 1 Configuration 0000 DAC1 debug watchpoints are enabled for load-type or store-type data storage accesses when $DBCR0_{DAC1}$=00 0001 DAC1 debug watchpoints are enabled only for store-type data storage accesses when $DBCR0_{DAC1}$=00 0010 DAC1 debug watchpoints are enabled only for load-type data storage accesses when $DBCR0_{DAC1}$=00 0011 Reserved 01*xx* Reserved 1000 DAC1 address comparisons are used for stack limit checking. DAC1 address comparisons are qualified with use of GPR R1 in <EA> calculation for most load or store instructions. No debug events occur for DAC1. When a qualified DAC1 match occurs, a machine check exception is generated for supervisor-mode accesses or a DSI is generated for user-mode accesses, and a DAC1 watchpoint is generated. $DBCR0_{DAC1}$ and $DBCR2_{DVC1M}$ settings are ignored. 1001 – 1111 Reserved DAC1CFG controls whether DAC1 data address comparisons utilize the normal PowerISA operation for generating both debug events and watchpoints, a watchpoint-only function, or a stack limit checking function (with watchpoint). **Note:** Unlike the DAC3–4CFG fields, debug event enabling for DAC1 is controlled by the DAC1 field in DBCR0. The DAC1CFG encodings 0000–0010 are used to control watchpoint generation when $DBCR0_{DAC1}$=0; when $DBCR0_{DAC1}$!=00, DAC1 watchpoints will fire whenever a DAC1 debug event occurs.
28:31	DAC2CFG	Data Address Compare 2 Configuration 0000 DAC2 debug watchpoints are enabled for load-type or store-type data storage accesses when $DBCR0_{DAC2}$=00 0001 DAC2 debug watchpoints are enabled only for store-type data storage accesses when $DBCR0_{DAC2}$=00 0010 DAC2 debug watchpoints are enabled only for load-type data storage accesses when $DBCR0_{DAC2}$=00 0011 Reserved 01*xx* Reserved 1*xxx* Reserved DAC2CFG controls whether DAC2 data address comparisons utilize the normal compare operation for generating both debug events and watchpoints, or a watchpoint-only function. **Note:** Unlike the DAC3–4CFG fields, debug event enabling for DAC2 is controlled by the DAC2 field in DBCR0. The DAC2CFG encodings 0000–0010 are used to control watchpoint generation when $DBCR0_{DAC2}$=00. When $DBCR0_{DAC2}$!=00, DAC2 watchpoints will fire whenever a DAC2 debug event occurs.

71.3.2.5 Debug Control Register 5 (DBCR5)

Debug Control Register 5 is used to configure Instruction Address Compare operation for IAC5–8. The DBCR5 register is shown in *Figure 1923*

IAC5US	IAC5ER	IAC6US	IAC6ER	IAC56M	0	IAC7US	IAC7ER	IAC8US	IAC8ER	IAC78M	0
0 1	2 3	4 5	6 7	8 9	10 11 12 13 14 15	16 17	18 19	20 21	22 23	24 25	26 27 28 29 30 31

SPR - 564; Read/Write; Reset(1) - 0x0

Figure 1923. DBCR5 register

1. Reset by processor reset **p_reset_b** if $EDBCR0_{EDM}$=0, as well as unconditionally by **m_por**. If $EDBCR0_{EDM}$=1, EDBRAC0 masks off hardware-owned resources from reset by **p_reset_b** and only software-owned resources indicated by EDBRAC0 will be reset by **p_reset_b**.

Table 1858 provides bit definitions for Debug Control Register 5.

Table 1858. DBCR5 field descriptions

Bit	Name	Description
0:1	IAC5US	Instruction Address Compare 5 User/Supervisor Mode 00 IAC5 debug events not affected by MSR_{PR} 01 Reserved 10 IAC5 debug events can only occur if MSR_{PR}=0 (Supervisor mode) 11 IAC5 debug events can only occur if MSR_{PR}=1. (User mode)
2:3	IAC5ER	Instruction Address Compare 5 Effective/Real Mode 00 IAC5 debug events are based on effective address 01 Unimplemented (Book E real address compare), no match can occur 10 IAC5 debug events are based on effective address and can only occur if MSR_{IS}=0 11 IAC5 debug events are based on effective address and can only occur if MSR_{IS}=1
4:5	IAC6US	Instruction Address Compare 6 User/Supervisor Mode 00 IAC6 debug events not affected by MSR_{PR} 01 Reserved 10 IAC6 debug events can only occur if MSR_{PR}=0 (Supervisor mode) 11 IAC6 debug events can only occur if MSR_{PR}=1. (User mode)
6:7	IAC6ER	Instruction Address Compare 6 Effective/Real Mode 00 IAC6 debug events are based on effective address 01 Unimplemented (Book E real address compare), no match can occur 10 IAC6 debug events are based on effective address and can only occur if MSR_{IS}=0 11 IAC6 debug events are based on effective address and can only occur if MSR_{IS}=1
8:9	IAC56M	Instruction Address Compare 5/6 Mode 00 Exact address compare. IAC5 debug events can only occur if the address of the instruction fetch is equal to the value specified in IAC5. IAC6 debug events can only occur if the address of the instruction fetch is equal to the value specified in IAC6. 01 Address bit match. IAC5 debug events can occur only if the address of the instruction fetch, ANDed with the contents of IAC6 are equal to the contents of IAC5, also ANDed with the contents of IAC6. IAC6 debug events do not occur. IAC5US and IAC5ER settings are used. 10 Inclusive address range compare. IAC5 debug events can occur only if the address of the instruction fetch is greater than or equal to the value specified in IAC5 and less than the value specified in IAC6. IAC6 debug events do not occur. IAC5US and IAC5ER settings are used. 11 Exclusive address range compare. IAC5 debug events can occur only if the address of the instruction fetch is less than the value specified in IAC5 or is greater than or equal to the value specified in IAC6. IAC6 debug events do not occur. IAC5US and IAC5ER settings are used.
10:15	—	Reserved

Table 1858. DBCR5 field descriptions (continued)

Bit	Name	Description
16:17	IAC7US	Instruction Address Compare 7 User/Supervisor Mode 00 IAC7 debug events not affected by MSR_{PR} 01 Reserved 10 IAC7 debug events can only occur if MSR_{PR}=0 (Supervisor mode) 11 IAC7 debug events can only occur if MSR_{PR}=1 (User mode)
18:19	IAC7ER	Instruction Address Compare 7 Effective/Real Mode 00 IAC7 debug events are based on effective address 01 Unimplemented (Book E real address compare), no match can occur 10 IAC7 debug events are based on effective address and can only occur if MSR_{IS}=0 11 IAC7 debug events are based on effective address and can only occur if MSR_{IS}=1
20:21	IAC8US	Instruction Address Compare 8 User/Supervisor Mode 00 IAC8 debug events not affected by MSR_{PR} 01 Reserved 10 IAC8 debug events can only occur if MSR_{PR}=0 (Supervisor mode). 11 IAC8 debug events can only occur if MSR_{PR}=1. (User mode)
22:23	IAC8ER	Instruction Address Compare 8 Effective/Real Mode 00 IAC8 debug events are based on effective address 01 Unimplemented (Book E real address compare), no match can occur 10 IAC8 debug events are based on effective address and can only occur if MSR_{IS}=0 11 IAC8 debug events are based on effective address and can only occur if MSR_{IS}=1
24:25	IAC78M	Instruction Address Compare 7/8 Mode 00 Exact address compare. IAC7 debug events can only occur if the address of the instruction fetch is equal to the value specified in IAC7. IAC8 debug events can only occur if the address of the instruction fetch is equal to the value specified in IAC8. 01 Address bit match. IAC7 debug events can occur only if the address of the instruction fetch, ANDed with the contents of IAC8 are equal to the contents of IAC7, also ANDed with the contents of IAC8. IAC8 debug events do not occur. IAC7US and IAC7ER settings are used. 10 Inclusive address range compare. IAC7 debug events can occur only if the address of the instruction fetch is greater than or equal to the value specified in IAC7 and less than the value specified in IAC8. IAC8 debug events do not occur. IAC7US and IAC7ER settings are used. 11 Exclusive address range compare. IAC7 debug events can occur only if the address of the instruction fetch is less than the value specified in IAC7 or is greater than or equal to the value specified in IAC8. IAC8 debug events do not occur. IAC7US and IAC7ER settings are used.
26:31	—	Reserved

71.3.2.6 Debug Control Register 6 (DBCR6)

Debug Control Register 6 is used to extend instruction address compare matching functionality. DBCR6 is shown in *Figure 1924*.

IAC1XM	IAC2XM	IAC3XM	IAC4XM	IAC5XM	IAC6XM	IAC7XM	IAC8XM

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

SPR - 603; Read/Write; Reset(1) - 0x0

Figure 1924. DBCR6 register

1. DBCR6 is reset by processor reset **p_reset_b** if $EDBCR0_{EDM}$=0, as well as unconditionally by **m_por**. If $EDBCR0_{EDM}$=1, EDBRAC0 masks off hardware-owned resources from reset by **p_reset_b** and only software-owned resources indicated by EDBRAC0 will be reset by **p_reset_b**.

Table 1859 provides bit definitions for Debug Control Register 6.

Table 1859. DBCR6 field descriptions

Bit	Name	Description
0:3	IAC1XM	Instruction Address Compare 1 Extended Mask Control. IAC1XM allows for binary power of 2 address range compares for IAC1 without requiring the use of IAC2. 0000 No additional masking when DBCR1[IAC12M] = 00 0001–1100 Exact Match Bit Mask. Number of low order bits masked in IAC1 when comparing the storage address with the value in IAC1 for exact address compare (DBCR1[IAC12M] = 00). Ranges up to 4 KB are supported. 1101–1111 = Reserved
4:7	IAC2XM	Instruction Address Compare 2 Extended Mask Control. IAC2XM allows for binary power of 2 address range compares for IAC2 without requiring the use of IAC1. 0000 No additional masking when DBCR1[IAC12M] = 00 0001–1100 Exact Match Bit Mask. Number of low order bits masked in IAC2 when comparing the storage address with the value in IAC2 for exact address compare (DBCR1[IAC12M] = 00). Ranges up to 4 KB are supported. 1101–1111 Reserved
8:11	IAC3XM	Instruction Address Compare 3 Extended Mask Control. IAC3XM allows for binary power of 2 address range compares for IAC1 without requiring the use of IAC2. 0000 No additional masking when DBCR1[IAC34M] = 00 0001–1100 Exact Match Bit Mask. Number of low order bits masked in IAC3 when comparing the storage address with the value in IAC3 for exact address compare (DBCR1[IAC34M] = 00). Ranges up to 4 KB are supported. 1101–1111 Reserved
12:15	IAC4XM	Instruction Address Compare 4 Extended Mask Control. IAC4XM allows for binary power of 2 address range compares for IAC4 without requiring the use of IAC3. 0000 No additional masking when DBCR1[IAC34M] = 00 0001–1100 Exact Match Bit Mask. Number of low order bits masked in IAC4 when comparing the storage address with the value in IAC4 for exact address compare (DBCR1[IAC34M] = 00). Ranges up to 4 KB are supported. 1101–1111 Reserved
16:19	IAC5XM	Instruction Address Compare 5 Extended Mask Control. IAC5XM allows for binary power of 2 address range compares for IAC5 without requiring the use of IAC6. 0000 No additional masking when DBCR5[IAC56M] = 00 0001–1100 Exact Match Bit Mask. Number of low order bits masked in IAC5 when comparing the storage address with the value in IAC5 for exact address compare (DBCR5[IAC56M] = 00). Ranges up to 4 KB are supported. 1101–1111 Reserved
20:23	IAC6XM	Instruction Address Compare 6 Extended Mask Control. IAC6XM allows for binary power of 2 address range compares for IAC6 without requiring the use of IAC5. 0000 No additional masking when DBCR5[IAC56M] = 00 0001–1100 Exact Match Bit Mask. Number of low order bits masked in IAC6 when comparing the storage address with the value in IAC6 for exact address compare (DBCR5[IAC56M] = 00). Ranges up to 4 KB are supported. 1101–1111 Reserved

Table 1859. DBCR6 field descriptions (continued)

Bit	Name	Description
24:27	IAC7XM	Instruction Address Compare 7 Extended Mask Control. IAC7XM allows for binary power of 2 address range compares for IAC7 without requiring the use of IAC8. 0000 No additional masking when DBCR5[IAC78M] = 00 0001–1100 Exact Match Bit Mask. Number of low order bits masked in IAC7 when comparing the storage address with the value in IAC7 for exact address compare (DBCR5[IAC78M] = 00). Ranges up to 4 KB are supported. 1101 – 1111 = Reserved
28:31	IAC8XM	Instruction Address Compare 8 Extended Mask Control. IAC8XM allows for binary power of 2 address range compares for IAC8 without requiring the use of IAC7. 0000 No additional masking when DBCR5[IAC78M] = 00 0001–1100 Exact Match Bit Mask. Number of low order bits masked in IAC8 when comparing the storage address with the value in IAC8 for exact address compare (DBCR5[IAC78M] = 00). Ranges up to 4 KB are supported. 1101–1111 Reserved

71.3.2.7 Debug Control Register 7 (DBCR7)

Debug Control Register 7 is used to enable and configure Data Address Compare 3 and 4 functionality. DBCR7 is shown in *Figure 1925*.

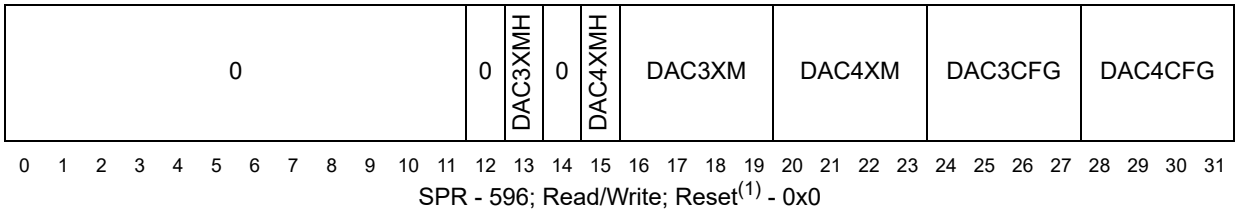


Figure 1925. DBCR7 register

1. DBCR7 is reset by processor reset **p_reset_b** if $EDBCR0_{EDM}$=0, as well as unconditionally by **m_por**. If $EDBCR0_{EDM}$=1, EDBRAC0 masks off hardware-owned resources from reset by **p_reset_b** and only software-owned resources indicated by EDBRAC0 will be reset by **p_reset_b**.

Table 1860 provides bit definitions for Debug Control Register 7.

Table 1860. DBCR7 field descriptions

Bit	Name	Description
0:12	—	Reserved
13	DAC3XMH	Data Address Compare 3 Extended Mask Control High. DAC3XMH extends the range of the DAC3XM field. 0 DAC3XM masks 0–15 low-order address bits 1 DAC3XM masks 16–31 low-order address bits
14	—	Reserved
15	DAC4XMH	Data Address Compare 4 Extended Mask Control High. DAC4XMH extends the range of the DAC4XM field. 0 DAC4XM masks 0–15 low-order address bits 1 DAC4XM masks 16–31 low-order address bits

Table 1860. DBCR7 field descriptions (continued)

Bit	Name	Description
16:19	DAC3XM	Data Address Compare 3 Extended Mask Control. DAC3XM allows for binary power of 2 address range compares for DAC3 without requiring the use of DAC4. Value of DAC3XMH || DAC3XM: 00000 No additional masking when DBCR8[DAC34M] = 00 00001–11111 Exact Match Bit Mask. One to 31 low-order bits are masked in DAC3 when comparing the storage address with the value in DAC3 for exact address compare (DBCR8[DAC34M] = 00). Address ranges of 2 bytes to 2GB are supported.
20:23	DAC4XM	Data Address Compare 4 Extended Mask Control. DAC4XM allows for binary power of 2 address range compares for DAC4 without requiring the use of DAC3. Value of DAC4XMH || DAC4XM: 00000 No additional masking when DBCR8[DAC34M] = 00 00001–11111 Exact Match Bit Mask. One to 31 low-order bits are masked in DAC4 when comparing the storage address with the value in DAC4 for exact address compare (DBCR8[DAC34M] = 00). Address ranges of 2 bytes to 2GB are supported.
24:27	DAC3CFG	Data Address Compare 3 Configuration. DAC3CFG controls whether DAC3 data address comparisons utilize the normal compare operation for generating both debug events and watchpoints, a watchpoint-only function, or a stack limit checking function (with watchpoint). 0000 DAC3 debug watchpoints are enabled for load-type or store-type data storage accesses 0001 DAC3 debug watchpoints are enabled only for store-type data storage accesses 0010 DAC3 debug watchpoints are enabled only for load-type data storage accesses 0011 Reserved 0100 Reserved 0101 DAC3 debug events and watchpoints are enabled only for store-type data storage accesses 0110 DAC3 debug events and watchpoints are enabled only for load-type data storage accesses 0111 DAC3 debug events and watchpoints are enabled for load-type or store-type data storage accesses 1000 DAC3 address comparisons are used for stack limit checking. DAC3 address comparisons are qualified with use of GPR R1 in <EA> calculation for most load or store instructions. No debug events occur for DAC3. When a qualified match occurs, a machine check exception is generated for supervisor-mode accesses or a DSI is generated for user-mode accesses, and a DAC3 watchpoints is generated. 1001–1111 Reserved
28:31	DAC4CFG	Data Address Compare 4 Configuration. DAC4CFG controls whether DAC4 data address comparisons utilize the normal compare operation for generating both debug events and watchpoints, or a watchpoint-only function. 0000 DAC4 debug watchpoints are enabled for load-type or store-type data storage accesses 0001 DAC4 debug watchpoints are enabled only for store-type data storage accesses 0010 DAC4 debug watchpoints are enabled only for load-type data storage accesses 0011 Reserved 0100 Reserved 0101 DAC4 debug events and watchpoints are enabled only for store-type data storage accesses 0110 DAC4 debug events and watchpoints are enabled only for load-type data storage accesses 0111 DAC4 debug events and watchpoints are enabled for load-type or store-type data storage accesses 1xxx Reserved

71.3.2.8 Debug Control Register 8 (DBCR8)

Debug Control Register 8 is used to configure Data Address Compare 3 and 4 operation. The DBCR8 register is shown in *Figure 1926*.



SPR - 597; Read/Write; Reset(1) - 0x0

Figure 1926. DBCR8 register

1. Reset by processor reset **p_reset_b** if $EDBCR0_{EDM}$=0, as well as unconditionally by **m_por**. If $EDBCR0_{EDM}$=1, EDBRAC0 masks off hardware-owned resources from reset by **p_reset_b** and only software-owned resources indicated by EDBRAC0 will be reset by **p_reset_b**.

Table 1861 provides bit definitions for Debug Control Register 8.

Table 1861. DBCR8 field descriptions

Bit	Name	Description
0:1	DAC3US	Data Address Compare 3 User/Supervisor Mode 00 DAC3 debug events not affected by MSR_{PR} 01 Reserved 10 DAC3 debug events can only occur if MSR_{PR}=0 (Supervisor mode) 11 DAC3 debug events can only occur if MSR_{PR}=1 (User mode)
2:3	DAC3ER	Data Address Compare 3 Effective/Real Mode 00 DAC3 debug events are based on effective address 01 Unimplemented (Book E real address compare), no match can occur 10 DAC3 debug events are based on effective address and can only occur if MSR_{DS}=0 11 DAC3 debug events are based on effective address and can only occur if MSR_{DS}=1
4:5	DAC4US	Data Address Compare 4 User/Supervisor Mode. 00 DAC4 debug events not affected by MSR_{PR} 01 Reserved 10 DAC4 debug events can only occur if MSR_{PR}=0 (Supervisor mode) 11 DAC4 debug events can only occur if MSR_{PR}=1 (User mode)
6:7	DAC4ER	Data Address Compare 4 Effective/Real Mode 00 DAC4 debug events are based on effective address 01 Unimplemented (Book E real address compare), no match can occur 10 DAC4 debug events are based on effective address and can only occur if MSR_{DS}=0 11 DAC4 debug events are based on effective address and can only occur if MSR_{DS}=1

Table 1861. DBCR8 field descriptions (continued)

Bit	Name	Description
8:9	DAC34M	Data Address Compare 3/4 Mode 00 Exact address compare. DAC3 debug events can only occur if the address of the data access is equal to the value specified in DAC3. DAC4 debug events can only occur if the address of the data access is equal to the value specified in DAC4. 01 Address bit match. DAC3 debug events can occur only if the address of the data access ANDed with the contents of DAC4, are equal to the contents of DAC3 also ANDed with the contents of DAC4. DAC4 debug events do not occur. DAC3US and DAC3ER settings are used. 10 Inclusive address range compare. DAC3 debug events can occur only if the address of the data access is greater than or equal to the value specified in DAC3 and less than the value specified in DAC4. DAC4 debug events do not occur. DAC3US and DAC3ER settings are used. 11 Exclusive address range compare. DAC3 debug events can occur only if the address of the data access is less than the value specified in DAC3 or is greater than or equal to the value specified in DAC4. DAC4 debug events do not occur. DAC3US and DAC3ER settings are used.
10	DAC3LNK	Data Address Compare 3 Linked 0 No effect 1 DAC3 debug events are linked to IAC5 debug events. IAC5 debug events do not affect DBSR When linked to IAC5, DAC3 debug events are conditioned based on whether the instruction also generated an IAC5 debug event. Note that linking is only available in EDM or IDM.
11	DAC4LNK	Data Address Compare 4 Linked 0 No effect 1 DAC4 debug events are linked to IAC7 debug events. IAC7 debug events do not affect DBSR When linked to IAC7, DAC4 debug events are conditioned based on whether the instruction also generated an IAC7 debug event. Note that linking is only available in EDM or IDM.
12:31	—	Reserved

71.3.2.9 Debug Status Register (DBSR)

The Debug Status Register (DBSR) contains status on debug events and the most recent processor reset. The Debug Status Register is set via hardware, and read and cleared via software. Bits in the Debug Status Register can be cleared using **mtspr** *DBSR,RS*. Clearing is done by writing to the Debug Status Register with a 1 in any bit position that is to be cleared and 0 in all other bit positions. The write data to the Debug Status Register is not direct data, but a mask. A '1' causes the bit to be cleared, and a '0' has no effect. Debug Status bits are set by Debug events only while Internal Debug mode is enabled ($DBCR0_{IDM}$=1). When debug interrupts are enabled (MSR_{DE}=1 $DBCR0_{IDM}$=1 and $EDBCR0_{EDM}$=0, or MSR_{DE}=1, $DBCR0_{IDM}$=1, $EDBCR0_{EDM}$=1 and software is allocated resource(s) via EDBRAC0), a set bit in DBSR other than MRR, DAC_OFST, or VLES will cause a debug interrupt to be generated. The debug interrupt handler is responsible for clearing DBSR bits prior to returning to normal execution. When resource sharing is enabled, ($EDBCR0_{EDM}$=1 and $EDBRAC0_{IDM}$=1), only software-owned resources may be modified by software, and status bits associated with hardware-owned resources will not be set by hardware in DBSR. The DBSR register is shown in *Figure 1927*.

IDE	UDE	MRR	ICMP	BRT	IRPT	TRAP	IAC	0	DACR	DACW	0	DNI	RET	0	DEVT1	DEVT2	PMI	MPU	CIRPT	CRET	VLES	DAC_OFST	0
0	1	2 3	4	5	6	7	8	9 10 11	12	13	14	15	16	17 18 19 20	21	22	23	24	25	26	27	28 29 30	31

SPR - 304; Read/Clear; Reset - 0x1000_0000

Figure 1927. Debug Status Register (DBSR)

Table 1862 provides bit definitions for the Debug Status Register.

Table 1862. DBSR field descriptions

Bit	Name	Description
0	IDE	Imprecise Debug Event Set to 1 if MSR_{DE}=0 and $DBCR0_{IDM}$=1 and a debug event causes its respective Debug Status Register bit to be set to 1. It may also be set to '1' if an imprecise debug event occurs due to a DAC event on a load or store that is terminated with error, or if an ICMP event occurs in conjunction with a EFPU FP round exception.
1	UDE	Unconditional Debug Event Set to 1 if an Unconditional debug event occurred.
2:3	MRR	Most Recent Reset. 00 No reset occurred since these bits were last cleared by software 01 A hard reset occurred since these bits were last cleared by software 10 Reserved 11 Reserved
4	ICMP	Instruction Complete Debug Event Set to 1 if an Instruction Complete debug event occurred.
5	BRT	Branch Taken Debug Event Set to 1 if an Branch Taken debug event occurred.
6	IRPT	Interrupt Taken Debug Event Set to 1 if an Interrupt Taken debug event occurred.
7	TRAP	Trap Taken Debug Event Set to 1 if a Trap Taken debug event occurred.
8	IAC	Instruction Address Compare Debug Event Set to 1 if an IAC debug event occurred.
9:11	—	Reserved
12	DACR	Data Address Compare Read Debug Event Set to 1 if a read-type DAC debug event occurred
13	DACW	Data Address Compare Write Debug Event Set to 1 if a write-type DAC debug event occurred
14	—	Reserved
15	DNI	DNI Debug Event Set to 1 if a DNI debug event occurred
16	RET	Return Debug Event Set to 1 if a Return debug event occurred

Table 1862. DBSR field descriptions (continued)

Bit	Name	Description
17:20	—	Reserved
21	DEVT1	External Debug Event 1 Debug Event Set to 1 if a DEVT1 debug event occurred
22	DEVT2	External Debug Event 2 Debug Event Set to 1 if a DEVT2 debug event occurred
23	PMI	Performance Monitor Interrupt Debug Event Set to 1 if a Performance Monitor Interrupt event occurred with $PMGC0_{UDI}$=1
24	MPU	Memory Protection Unit Debug Event Set to 1 if a MPU debug event occurred. For MPU debug events, the IAC, DACR, or DACW status bit will also be set, depending on the type of access that matched a region descriptor enabled to generate debug events, in order to further define the type of MPU debug event. Note that software MPU debug events on data accesses normally occur after the data access is performed and the instruction completes, thus act like a normal DAC debug event. The instruction may not complete if a DSI occurs. In this case, IDE will be set to indicate this occurrence.
25	CIRPT	Critical Interrupt Taken Debug Event Set to 1 if a Critical Interrupt Taken debug event occurred.
26	CRET	Critical Return Debug Event Set to 1 if a Critical Return debug event occurred
27	VLES	VLE Status Set to 1 if an ICMP, BRT, TRAP, RET, CRET, IAC, or DAC debug event occurred on a PowerISA VLE Instruction. Undefined for IRPT, CIRPT, DEVT[1,2], and UDE events
28:30	DAC_OFST	Data Address Compare Offset Indicates offset-1 of saved DSRR0 value from the address of the load or store instruction that took a DAC Debug exception, unless a simultaneous DSI error occurs, in which case this field is set to 3‘b000 and $DBSR_{IDE}$ is set to 1. Normally set to 3‘b000 by a non-DVC DAC. A DVC DAC may set this field to any value.
31	—	Reserved

71.3.2.10 Debug Data Effective Address Register (DDEAR)

The Debug Data Effective Address Register (DDEAR) contains address information for data address compare debug events (DAC or MPU DAC). DDEAR is updated by hardware with the effective address of the load, store, or cache control operation when a data address compare event is recorded in DBSR if the previous values of the $DBSR_{DAC\{R,W\}}$ bits are zero. Once a DDEAR update is performed, these bits must be cleared by software prior to another DDEAR hardware update occurring. A subsequent DAC or MPU DAC event will not update the DDEAR register if either of the $DBSR_{DAC\{R,W\}}$ bits are set, in order to capture the first event address.

The DDEAR register is shown in *Figure 1928*.

Data Effective Address

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

SPR - 600; Read/Write; Reset - unaffected

Figure 1928. Debug Data Effective Address Register (DDEAR)

71.3.3 External Debug Resource Allocation Control (EDBRAC0) register

The External Debug Resource Allocation Control Register (EDBRAC0) controls resource allocation when $EDBCR0_{EDM}$ is set to '1'. EDBRAC0 provides a mechanism for the hardware debugger to share certain debug resources with software. Individual resources are allocated based on the settings of EDBRAC0 when $EDBCR0_{EDM}$=1. EDBRAC0 settings are ignored when $EDBCR0_{EDM}$=0.

Hardware-owned resources that generate debug events update EDBSR0 instead of DBSR and cause entry into debug mode if the event is not masked in EDBSRMSK0, while software-owned resources that generate debug events if $DBCR0_{IDM}$=1 update DBSR, causing debug interrupts to occur if MSR_{DE}=1. EDBRAC0 is controlled via the OnCE port hardware, and is read-only to software.

The DBSR status register is always owned by software. Debug status bits in DBSR are set by software-owned debug events only while Internal Debug mode is enabled. When debug interrupts are enabled (MSR_{DE}=1 $DBCR0_{IDM}$=1 and $EDBCR0_{EDM}$=0, or MSR_{DE}=1, $DBCR0_{IDM}$=1 and $EDBCR0_{EDM}$=1 and software is allocated resource(s) via EDBRAC0), a set bit in DBSR by an event that is software-owned (other than MRR, DAC_OFST, or VLES) will cause a debug interrupt to be generated.

Debug status bits in EDBSR0 are set by hardware-owned debug events only while External Debug mode is enabled ($EDBCR0_{EDM}$=1). When $EDBCR0_{EDM}$=1, a set bit in EDBSR0 by an event that is hardware-owned (other than IDE, DAC_OFST, or VLES) will cause entry into debug mode unless entry is masked via EDBSRMSK0.

If $EDBCR0_{EDM}$=1, DBSR status bits corresponding to hardware-owned debug events are masked from being set by hardware.

Software-owned resources may be modified by software, but only the corresponding control bits in DBCR0–8 or MPU0CSR0 are affected by execution of a **mtspr**, thus only a portion of these registers may be affected, depending on the allocation settings in EDBRAC0. The debug interrupt handler is still responsible for clearing DBSR bits for software-owned resources prior to returning to normal execution. Hardware always has full access to all registers and register fields through the OnCE register access mechanism, and it is up to the debug firmware to properly implement modifications to these registers with read-modify-write operations to implement any control sharing with software. Settings in EDBRAC0 should be considered by the debug firmware in order to preserve software settings of control and status registers as appropriate when hardware modifications to the debug registers is performed.

The EDBRAC0 register is shown in *Figure 1929*.

0	IDM	RST	UDE	ICMP	BRT	IRPT	TRAP	IAC1	IAC2	IAC3	IAC4	DAC1	DAC34	DAC2	0	RET	IAC5	IAC6	IAC7	IAC8	DEVT1	DEVT2	PMI	MPU	CIRPT	CRET	DNI	DQM	0
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31

SPR - 638; Read-only by Software; Reset - Unaffected by **p_reset_b**, reset to 0x00000180 by **m_por** or while in the test-logic-reset OnCE controller state

Figure 1929. External Debug Resource Allocation Control (EDBRAC0) register

Table 1863 provides bit definitions for the Debug External Resource Control Register. Note that EDBRAC0 controls are disabled when EDBCR0EDM=0.

Table 1863. EDBRAC0 field descriptions

Bit	Name	Description
0	—	Reserved
1	IDM	Internal Debug Mode control. When $EDBRAC0_{IDM}$=1, software writes to hardware-owned bits in DBCR0–8 via **mtspr** are ignored. 0 Internal Debug mode may not be enabled by software. $DBCR0_{IDM}$ is owned exclusively by hardware. **mtspr** DBCR0–8 and other debug registers is always ignored. No resource sharing occurs, regardless of the settings of other fields in EDBRAC0. Hardware exclusively owns all resources. 1 Internal Debug mode may be enabled by software. $DBCR0_{IDM}$ is owned by software. $DBCR0_{IDM}$ is software readable/writable.
2	RST	Reset Field Control 0 $DBCR0_{RST}$ owned exclusively by hardware debug. No **mtspr** access by software to $DBCR0_{RST}$ field. 1 $DBCR0_{RST}$ accessible by software debug. $DBCR0_{RST}$ is software readable/writable.
3	UDE	Unconditional Debug Event 0 Event owned by hardware debug. 1 Event owned by software debug.
4	ICMP	Instruction Complete Debug Event 0 Event owned by hardware debug. No **mtspr** access by software to $DBCR0_{ICMP}$ field. 1 Event owned by software debug. $DBCR0_{ICMP}$ is software readable/writable.
5	BRT	Branch Taken Debug Event 0 Event owned by hardware debug. No **mtspr** access by software to $DBCR0_{BRT}$ field. 1 Event owned by software debug. $DBCR0_{BRT}$ is software readable/writable.
6	IRPT	Interrupt Taken Debug Event 0 Event owned by hardware debug. No **mtspr** access by software to $DBCR0_{IRPT}$ field. 1 Event owned by software debug. $DBCR0_{IRPT}$ is software readable/writable.
7	TRAP	Trap Taken Debug Event 0 Event owned by hardware debug. No **mtspr** access by software to $DBCR0_{TRAP}$ field. 1 Event owned by software debug. $DBCR0_{TRAP}$ is software readable/writable.
8	IAC1	Instruction Address Compare 1 Debug Event 0 Event owned by hardware debug. No **mtspr** access by software to IAC1 control and status fields. 1 Event owned by software debug. IAC1 control fields are software readable/writable.

Table 1863. EDBRAC0 field descriptions (continued)

Bit	Name	Description
9	IAC2	Instruction Address Compare 2 Debug Event 0 Event owned by hardware debug. No **mtspr** access by software to IAC2 control and status fields. 1 Event owned by software debug. IAC2 control fields are software readable/writable.
10	IAC3	Instruction Address Compare 3 Debug Event 0 Event owned by hardware debug. No **mtspr** access by software to IAC3 control and status fields. 1 Event owned by software debug. IAC3 control fields are software readable/writable.
11	IAC4	Instruction Address Compare 4 Debug Event 0 Event owned by hardware debug. No **mtspr** access by software to IAC4 control and status fields. 1 Event owned by software debug. IAC4 control fields are software readable/writable.
12	DAC1	Data Address Compare 1 Debug Event 0 Event owned by hardware debug. No **mtspr** access by software to DAC1 control and status fields. 1 Event owned by software debug. DAC1 control fields are software readable/writable.
13	DAC34	Data Address Compare 3 and 4 Debug Events 0 Events owned by hardware debug. No **mtspr** access by software to DAC3 and DAC4 control and status fields. 1 Event owned by software debug. DAC3 and DAC4 control fields are software readable/writable.
14	DAC2	Data Address Compare 2 Debug Event 0 Event owned by hardware debug. No **mtspr** access by software to DAC2 control and status fields. 1 Event owned by software debug. DAC2 control fields are software readable/writable.
15	—	Reserved
16	RET	Return Debug Event 0 Event owned by hardware debug. No **mtspr** access by software to $DBCR0_{RET}$ field. 1 Event owned by software debug. $DBCR0_{RET}$ is software readable/writable.
17	IAC5	Instruction Address Compare 5 Debug Event 0 Event owned by hardware debug. No **mtspr** access by software to IAC5 control and status fields. 1 Event owned by software debug. IAC5 control fields are software readable/writable.
18	IAC6	Instruction Address Compare 6 Debug Event 0 Event owned by hardware debug. No **mtspr** access by software to IAC6 control and status fields. 1 Event owned by software debug. IAC6 control fields are software readable/writable.
19	IAC7	Instruction Address Compare 7 Debug Event 0 Event owned by hardware debug. No **mtspr** access by software to IAC7 control and status fields. 1 Event owned by software debug. IAC7 control fields are software readable/writable.
20	IAC8	Instruction Address Compare 8 Debug Event 0 Event owned by hardware debug. No **mtspr** access by software to IAC8 control and status fields. 1 Event owned by software debug. IAC8 control are software readable/writable.

Table 1863. EDBRAC0 field descriptions (continued)

Bit	Name	Description
21	DEVT1	External Debug Event Input 1 Debug Event 0 Event owned by hardware debug. No **mtspr** access by software to $DBCR0_{DEVT1}$ field. 1 Event owned by software debug. $DBCR0_{DEVT1}$ is software readable/writable.
22	DEVT2	External Debug Event Input 2 Debug Event 0 Event owned by hardware debug. No **mtspr** access by software to $DBCR0_{DEVT2}$ field. 1 Event owned by software debug. $DBCR0_{DEVT2}$ is software readable/writable.
23	PMI	Performance Monitor Interrupt Debug Event 0 Event owned by hardware debug. No **mtpmr** access by software to the PMRs. Performance monitor interrupts set $EDBSR0_{PMI}$ regardless of the setting of $PMGC0_{UDI}$ 1 Event owned by software debug. PMRs are software readable/writable. **Note:** This bit is reset to '1'.
24	MPU	Memory Protection Unit Debug Event 0 Event owned by hardware debug. No **mpuwe** access by software to region descriptors that have the DEBUG control bit set to '1' or to the $MPU0CSR0_{DRDEN,DWDEN,IDEN}$ control bits, unless the CPU is in a debug session (**jd_debug_b** is asserted). MPU debug events set $EDBSR0_{MPU}$, and if not masked by $EDBSRMSK0_{MPU}$, one of $EDBSR0_{IAC}$, $EDBSR0_{DACR}$, or $EDBSR0_{DACW}$. MPU flash invalidates do not affect DEBUG=1 entries unless the CPU is in a debug session (**jd_debug_b** is asserted). 1 Event owned by software debug. All region descriptors and the $MPU0CSR0_{DRDEN,DWDEN,IDEN}$ control bits are software readable/writable. **Note:** This bit is reset to '1'.
25	CIRPT	Critical Interrupt Taken Debug Event 0 Event owned by hardware debug. No **mtspr** access by software to $DBCR0_{CIRPT}$ field. 1 Event owned by software debug. $DBCR0_{CIRPT}$ is software readable/writable.
26	CRET	Critical Return Debug Event 0 Event owned by hardware debug. No **mtspr** access by software to $DBCR0_{CRET}$ field. 1 Event owned by software debug. $DBCR0_{CRET}$ is software readable/writable.
27	DNI	DNI Instruction Debug Control 0 DNI resource owned by hardware debug. Execution of an **e_dni** or **se_dni** instruction results in entry into debug mode. 1 DNI resource owned by software debug. Execution of an **e_dni** or **se_dni** instruction results in either a debug interrupt ($DBCR0_{IDM}$=1 and MSR_{DE}=1) or a nop ($DBCR0_{IDM}$=0 or MSR_{DE}=0). **Note:** DNI events are not blocked during a debug session
28	DQM	Data Acquisition Messaging Registers 0 $DEVENT_{DQTAG}$ and DDAM register are exclusively owned by hardware debug. No **mtspr** access by software to $DEVENT_{DQTAG}$ field or DDAM register. Attempted access by software is ignored. 1 $DEVENT_{DQTAG}$ and DDAM register are owned by software. Software has read/write access to $DEVENT_{DQTAG}$ field and DDAM register.
29:31	—	Reserved

Table 1864 shows which resources are controlled by EDBRAC0 settings.

Table 1864. EDBRAC0 resource control

$EDBCR0_{EDM}$	$EDBRAC0_{IDM}$	$EDBRAC0_{RST}$	$EDBRAC0_{UDE}$	$EDBRAC0_{ICMP}$	$EDBRAC0_{BRT}$	$EDBRAC0_{IRPT}$	$EDBRAC0_{TRAP}$	$EDBRAC0_{IAC1}$	$EDBRAC0_{IAC2}$	$EDBRAC0_{IAC3}$	$EDBRAC0_{IAC4}$	$EDBRAC0_{IAC5}$	$EDBRAC0_{IAC6}$	$EDBRAC0_{IAC7}$	$EDBRAC0_{IAC8}$	$EDBRAC0_{DAC1}$	$EDBRAC0_{DAC34}$	$EDBRAC0_{DAC2}$	$EDBRAC0_{RET}$	$EDBRAC0_{DEVT1}$	$EDBRAC0_{DEVT2}$	$EDBRAC0_{PMI}$	$EDBRAC0_{MPU}$	$EDBRAC0_{CIRT}$	$EDBRAC0_{CRET}$	$EDBRAC0_{DNI}$	$EDBRAC0_{DQM}$	Software Accessible via mtspr, affected by p_reset_b
0	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	All debug registers
1	1	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	$DBCR0_{IDM}$
1	1	1	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	$DBCR0_{RST}$
1	1	—	1	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	$DBCR0_{UDE}$
1	1	—	—	1	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	$DBCR0_{ICMP}$
1	1	—	—	—	1	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	$DBCR0_{BRT}$
1	1	—	—	—	—	1	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	$DBCR0_{IRPT}$
1	1	—	—	—	—	—	1	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	$DBCR0_{TRAP}$
1	1	—	—	—	—	—	—	1	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	IAC1, $DBCR0_{IAC1}$, $DBCR1_{IAC1US,IAC1ER}$, $DBCR6_{IAC1XM}$
1	1	—	—	—	—	—	—	—	1	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	IAC2, $DBCR0_{IAC2}$, $DBCR1_{IAC2US,IAC2ER}$, $DBCR6_{IAC2XM}$
1	1	—	—	—	—	—	—	—	—	1	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	$DBCR1_{IAC12M}$

Table 1864. EDBRAC0 resource control (continued)

$EDBCR0_{EDM}$	$EDBRAC0_{IDM}$	$EDBRAC0_{RST}$	$EDBRAC0_{UDE}$	$EDBRAC0_{IGMP}$	$EDBRAC0_{BRT}$	$EDBRAC0_{IRPT}$	$EDBRAC0_{TRAP}$	$EDBRAC0_{IAC1}$	$EDBRAC0_{IAC2}$	$EDBRAC0_{IAC3}$	$EDBRAC0_{IAC4}$	$EDBRAC0_{IAC5}$	$EDBRAC0_{IAC6}$	$EDBRAC0_{IAC7}$	$EDBRAC0_{IAC8}$	$EDBRAC0_{DAC1}$	$EDBRAC0_{DAC34}$	$EDBRAC0_{DAC2}$	$EDBRAC0_{RET}$	$EDBRAC0_{DEVT1}$	$EDBRAC0_{DEVT2}$	$EDBRAC0_{PMI}$	$EDBRAC0_{MPU}$	$EDBRAC0_{CIRT}$	$EDBRAC0_{CRET}$	$EDBRAC0_{DNI}$	$EDBRAC0_{DCM}$	Software Accessible via mtspr, affected by p_reset_b
1	1	—	—	—	—	—	—	—	—	1	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	IAC3, $DBCR0_{IAC3}$, $DBCR1_{IAC3US,IAC3ER}$, $DBCR6_{IAC3XM}$
1	1	—	—	—	—	—	—	—	—	—	1	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	IAC4, $DBCR0_{IAC4}$, $DBCR1_{IAC4US,IAC4ER}$, $DBCR6_{IAC4XM}$
1	1	—	—	—	—	—	—	—	—	1	1	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	$DBCR1_{IAC34M}$
1	1	—	—	—	—	—	—	—	—	—	—	1	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	IAC5, $DBCR0_{IAC5}$, $DBCR5_{IAC5US,IAC5ER}$, $DBCR6_{IAC5XM}$
1	1	—	—	—	—	—	—	—	—	—	—	—	1	—	—	—	—	—	—	—	—	—	—	—	—	—	—	IAC6, $DBCR0_{IAC6}$, $DBCR5_{IAC6US,IAC6ER}$, $DBCR6_{IAC6XM}$
1	1	—	—	—	—	—	—	—	—	—	—	1	1	—	—	—	—	—	—	—	—	—	—	—	—	—	—	$DBCR5_{IAC56M}$
1	1	—	—	—	—	—	—	—	—	—	—	—	—	1	—	—	—	—	—	—	—	—	—	—	—	—	—	IAC7, $DBCR0_{IAC7}$, $DBCR5_{IAC7US,IAC7ER}$, $DBCR6_{IAC7XM}$
1	1	—	—	—	—	—	—	—	—	—	—	—	—	—	1	—	—	—	—	—	—	—	—	—	—	—	—	IAC8, $DBCR0_{IAC8}$, $DBCR5_{IAC8US,IAC8ER}$, $DBCR6_{IAC8XM}$
1	1	—	—	—	—	—	—	—	—	—	—	—	—	1	1	—	—	—	—	—	—	—	—	—	—	—	—	$DBCR5_{IAC78M}$

Table 1864. EDBRAC0 resource control (continued)

$EDBCR0_{EDM}$	$EDBRAC0_{IDM}$	$EDBRAC0_{RST}$	$EDBRAC0_{UDE}$	$EDBRAC0_{IGMP}$	$EDBRAC0_{BRT}$	$EDBRAC0_{IRPT}$	$EDBRAC0_{TRAP}$	$EDBRAC0_{IAC1}$	$EDBRAC0_{IAC2}$	$EDBRAC0_{IAC3}$	$EDBRAC0_{IAC4}$	$EDBRAC0_{IAC5}$	$EDBRAC0_{IAC6}$	$EDBRAC0_{IAC7}$	$EDBRAC0_{IAC8}$	$EDBRAC0_{DAC1}$	$EDBRAC0_{DAC34}$	$EDBRAC0_{DAC2}$	$EDBRAC0_{RET}$	$EDBRAC0_{DEVT1}$	$EDBRAC0_{DEVT2}$	$EDBRAC0_{PMI}$	$EDBRAC0_{MPU}$	$EDBRAC0_{CIRT}$	$EDBRAC0_{CRET}$	$EDBRAC0_{DNI}$	$EDBRAC0_{DCM}$	Software Accessible via mtspr, affected by p_reset_b
1	1	—	—	—	—	—	—	—	—	—	—	—	—	—	—	1	—	—	—	—	—	—	—	—	—	—	—	—	DAC1, DVC1, DVC1U $DBCR0_{DAC1}$, $DBCR2_{DAC1US,DAC1ER}$, $DBCR2_{DVC1M,DVC1BE}$ $DBCR4_{DVC1C,DAC1XM}$
1	1	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	1	—	—	—	—	—	—	—	—	—	—	—	DAC3, DAC4 $DBCR7_{DAC\{3,4\},DAC\{3,4\}CFG,DAC\{3,4\}XM,DAC\{3,4\}XMH}$ $DBCR8_{DAC\{3,4\}US,DAC\{3,4\}ER,DAC\{3,4\}M}$
1	1	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	1	—	—	—	—	—	—	—	—	—	—	DAC2, DVC2, DVC2U $DBCR0_{DAC2}$, $DBCR2_{DAC2US,DAC2ER}$, $DBCR2_{DVC2M,DVC2BE}$ $DBCR4_{DVC2C,DAC2XM}$
1	1	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	
1	1	—	—	—	—	—	—	—	—	—	—	—	—	—	—	1	—	1	—	—	—	—	—	—	—	—	—	—	$DBCR2_{DAC12M}$
1	1	—	—	—	—	—	—	1	—	—	—	—	—	—	—	1	—	—	—	—	—	—	—	—	—	—	—	—	$DBCR2_{DAC1LNK}$
1	1	—	—	—	—	—	—	—	—	1	—	—	—	—	—	—	—	1	—	—	—	—	—	—	—	—	—	—	$DBCR2_{DAC2LNK}$
1	1	—	—	—	—	—	—	—	—	—	—	1	—	—	—	—	1	—	—	—	—	—	—	—	—	—	—	—	$DBCR8_{DAC3LNK}$
1	1	—	—	—	—	—	—	—	—	—	—	—	1	—	—	—	1	—	—	—	—	—	—	—	—	—	—	—	$DBCR8_{DAC4LNK}$

Table 1864. EDBRAC0 resource control (continued)

$EDBCR0_{EDM}$	$EDBRAC0_{IDM}$	$EDBRAC0_{RST}$	$EDBRAC0_{UDE}$	$EDBRAC0_{IGMP}$	$EDBRAC0_{BRT}$	$EDBRAC0_{IRPT}$	$EDBRAC0_{TRAP}$	$EDBRAC0_{IAC1}$	$EDBRAC0_{IAC2}$	$EDBRAC0_{IAC3}$	$EDBRAC0_{IAC4}$	$EDBRAC0_{IAC5}$	$EDBRAC0_{IAC6}$	$EDBRAC0_{IAC7}$	$EDBRAC0_{IAC8}$	$EDBRAC0_{DAC1}$	$EDBRAC0_{DAC34}$	$EDBRAC0_{DAC2}$	$EDBRAC0_{RET}$	$EDBRAC0_{DEVT1}$	$EDBRAC0_{DEVT2}$	$EDBRAC0_{PMI}$	$EDBRAC0_{MPU}$	$EDBRAC0_{CIRT}$	$EDBRAC0_{CRET}$	$EDBRAC0_{DNI}$	$EDBRAC0_{DQM}$	Software Accessible via mtspr, affected by p_reset_b
1	1	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	1	—	—	—	—	—	—	—	—	—	$DBCR0_{RET}$
1	1	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	1	—	—	—	—	—	—	—	—	$DBCR0_{DEVT1}$
1	1	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	1	—	—	—	—	—	—	—	$DBCR0_{DEVT2}$
1	1	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	1	—	—	—	—	—	—	All PMRs
1	1	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	1	—	—	—	—	—	MPU entries with DEBUG bit set, $MPU0CSR0_{DRDEN,DWDEN,IDEN}$
1	1	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	1	—	—	—	—	$DBCR0_{CIRPT}$
1	1	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	1	—	—	—	$DBCR0_{CRET}$
1	1	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—		—	—	—	—	—	—	1	—	
1	— (1)	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	1	$DEVENT_{DQTAG}$, DDAM

1. IDM not required to be set to enable software access.

EDBRAC0 also controls which bits or fields in DBCR0–8, MPU0CSR0, and the PMRs are reset by assertion of **p_reset_b** when $EDBCR0_{EDM}$=1. Only software-owned bits or fields as shown in *Table 1864* are affected in this case, except that $DBCR0_{RST}$ and $DBSR_{MRR}$ are updated by assertion of **p_reset_b** regardless of the value of $EDBCR0_{EDM}$ or EDBRAC0.

71.3.4 Debug Event Select register (DEVENT) register

The Debug Event Select Register allows instrumented software to internally generate signals when a **mtspr** instruction is executed and this register is accessed. The values written to this register determine which of the **p_devnt_out[0:7]** processor output signals are asserted upon access. Writing a '1' to any of these bit positions will cause a one clock pulse to be generated on the corresponding output. For **p_devnt_out[0:3]**, a corresponding **jd_watchpt[x]** output is asserted as well to indicate a watchpoint has occurred. These signals may be used for internal core debug resources as well as for SoC level cross-triggering.

The $DEVENT_{DEVNT}$ register field value is undefined on a read; it may or may not remain set to the last value written. Since it is unconditionally shared by hardware debug and software, software should not rely on any value remaining.

The upper 8 bits of the DEVENT register also provide the DQTAG used to identify channels within Data Acquisition Messages. Refer to the "Data Acquisition ID Tag Field" section in the Core (e200z420n3) Nexus 3 Module chapter for more detail on the DQTAG.

The DEVENT register is shown in *Figure 1930*.

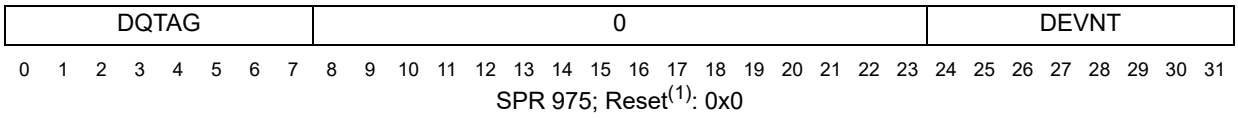


Figure 1930. Debug Event Select register (DEVENT) register

1. Reset by processor reset **p_reset_b** if $EDBCR0_{EDM}$=0, as well as unconditionally by **m_por**. If $EDBCR0_{EDM}$=1, EDBRAC0 masks off hardware-owned resources from reset by **p_reset_b** and only software-owned resources indicated by EDBRAC0 will be reset by **p_reset_b.** Note that DEVNT field is shared by hardware and software but is always reset by **p_reset_b**.

Table 1865 provides bit definitions for the Debug Event Register.

Table 1865. DEVENT field descriptions

Bit	Name	Description
0:7	DQTAG	Data Acquisition Message IDTAG channel identifier (supplied to Nexus 3)
8:23	—	Reserved, should be cleared.
24:31	DEVNT	Debug Event Signals 00000000 No signal is asserted *xxxxxxx*1 **p_devnt_out[0]** and **jd_watchpt[12]** are asserted for one clock *xxxxxx*1*x* **p_devnt_out[1]** and **jd_watchpt[13]** are asserted for one clock *xxxxx*1*xx* **p_devnt_out[2]** and **jd_watchpt[20]** are asserted for one clock *xxxx*1*xxx* **p_devnt_out[3]** and **jd_watchpt[21]** are asserted for one clock *xxx*1*xxxx* **p_devnt_out[4]** is asserted for one clock *xx*1*xxxxx* **p_devnt_out[5]** is asserted for one clock *x*1*xxxxxx* **p_devnt_out[6]** is asserted for one clock 1*xxxxxxx* **p_devnt_out[7]** is asserted for one clock

71.3.5 Debug Data Acquisition Message (DDAM) register

The Debug Data Acquisition Message Register allows instrumented software to generate real-time Data Acquisition Messages (as defined by Nexus 3) via a **mtspr** instruction to this

register. Refer to the “Data Acquisition Messaging” section in the Core (e200z420n3) Nexus 3 Module chapter for details.

The DDAM register is shown in *Figure 1931*.

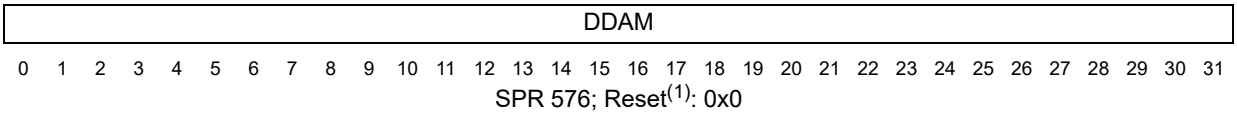


Figure 1931. Debug Data Acquisition Message (DDAM) register

1. Reset by processor reset **p_reset_b** if $EDBCR0_{EDM}$=0, as well as unconditionally by **m_por**. If $EDBCR0_{EDM}$=1, EDBRAC0 masks off hardware-owned resources from reset by **p_reset_b** and only software-owned resources indicated by EDBRAC0 will be reset by **p_reset_b**.

Table 1866 provides bit definitions for the Debug Data Acquisition Message Register.

Table 1866. DDAM field descriptions

Bit	Name	Description
0:31	DDAM	Value to be transmitted in a Data Acquisition Message (DQM) (supplied to Nexus 3 with strobe)

71.4 Using debug resources for stack limit checking

The DAC1,2 and DAC3,4 resources can be used for stack overflow/underflow detection when not being used as a hardware or software debug resource. Stack limit checking is available regardless of EDM or IDM mode, and when resources used for stack limit checking are owned by software, will utilize a DSI or machine check exception. Software-owned stack limit checking does not require IDM to be set. Hardware owned stack limit checking requires EDM to be set.

When stack limit checking is enabled, and DAC resources used for stack limit checking are owned by software, DAC events are not generated for resources configured to perform stack limit checking, and no DBSR DAC status flag will be set due to a detected stack limit violation. Instead, depending on the processor mode, a data storage interrupt or a machine check exception is signaled. When stack limit checking is enabled, and DAC resources used for stack limit checking are owned by hardware, DAC events will be generated for resources configured to perform stack limit checking, and the EDBSR0 DAC status flag will be set due to a detected stack limit violation, causing entry into debug halted mode in the same way as a DAC exception normally does. The only difference is that qualification of the access address is performed as discussed in the next paragraph.

Stack limit checking is implemented in the same way as range compares using DAC1,2 or 3,4, or extended masking using DAC1 or DAC3, but qualify a load or store access address with the use of GPR R1 as the base or index register used to compute an effective address when a load or store instruction is executed. No stack limit checking is performed for other instructions that may calculate an EA using GPR R1 such as cache, MMU/MPU management instructions, or instructions that indicate GPR R1 is used to hold a decoration value (for decorated load/store type instructions). When DAC resources configured to perform stack limit checking are not owned by hardware, if a stack limit violation occurs when performing the load or store, the access is aborted, and an error report machine check is generated, with MCSRR0 pointing to the address of the load or store access that generated the stack overflow/underflow. If DAC resources configured to perform stack limit

checking are owned by hardware, then a normal DAC event is generated (but qualified with use of GPR R1), and debug mode entry will occur in the same manner as for a non-stack limit DAC event.

Enabling of this functionality is described in more detail in *Table 1857*. Note that IDM is not required to be set for software-owned stack limit checking.

In contrast to the normal case of DAC address matching resulting in a debug event, the stack limit check address compare logic operates differently for stack limit checking. When using resources for stack limit checking, a DACn hit to a resource enabled for performing stack limit checking on a stack access indicates a stack limit violation.

When stack limit checking is enabled by the setting of the $DBCR4_{DAC1CFG}$ field to '1000' or the $DBCR8_{DAC3CFG}$ field to '1000' (or both), and the corresponding DACn resources are owned by software, DACn events are not generated and the DBSR DAC status flag will not be set due to a detected stack limit hit. Instead, if stack limit checking is enabled for supervisor mode stack accesses in DAC1 or DAC3, and a compare hit occurs for a supervisor mode stack access (A load or store using GPR R1 in the <EA> calculation), a machine check exception is signaled. If stack limit checking is enabled for user mode accesses, a DSI exception is signaled when a stack limit checking enabled DAC1 or DAC3 compare hit occurs for a user mode access. A watchpoint for DAC1 or DAC3 will be generated when the stack limit violation is detected, even though the instruction does not complete. Stack limit checking for supervisor mode stack accesses is considered enabled when either the $DBCR4_{DAC1CFG}$ field is set to '1000' with $DBCR2_{DAC1US}$ set to '00' or '10', or the $DBCR7_{DAC3CFG}$ field is set to '1000' with $DBCR8_{DAC3US}$ set to '00' or '10', or both. Stack limit checking for user mode stack accesses is considered enabled when either the $DBCR4_{DAC1CFG}$ field is set to '1000' with $DBCR2_{DAC1US}$ set to '00' or '01', or the $DBCR7_{DAC3CFG}$ field is set to '1000' with $DBCR8_{DAC3US}$ set to '00' or '01', or both. Note that unlike regular debug DAC events, both halves of a misaligned access are checked for limit violations.

When stack limit checking is enabled for a stack access, and DACn resources are owned by hardware, the EDBSR0 DAC status flag will be set due to a detected stack limit violation, to cause entry into debug halted mode or to generate a watchpoint, or both, in the same way as a DAC event normally does, i.e. after the access has completed. The only difference is that qualification of the access address is performed as discussed in the next paragraph. If the access is aborted due to a DSI or other exception such as machine check condition, the $EDBSR0_{IDE}$ status bit will also be set to indicate that the data access instruction was not completed.

Stack limit checking is implemented in the same way as address compares using DACn, but qualify a load or store access address with the use of GPR R1 as the base or index register used to compute an effective address when a load or store instruction is executed. No stack limit checking is performed for other instructions that may calculate an EA using GPR R1 such as cache, MMU/MPU management instructions, or instructions that indicate GPR R1 is used to hold a decoration value (for decorated load/store type instructions). When DACn resources are not owned by hardware, if a stack limit violation occurs (a hit to a stack limit enabled DAC is detected) when performing the load or store, the access request is aborted and the access is not performed. Instead, for supervisor mode accesses, an error report machine check exception is generated, with MCSRR0 pointing to the address of the load or store instruction that generated the stack overflow/underflow, and for user mode accesses, a DSI exception is generated, with SRR0 pointing to the address of the load or store instruction that generated the stack overflow/underflow. If all stack limit check enabled DACn resources are owned by hardware, then a DAC event is generated (but qualified with use of GPR R1), and debug mode entry will occur in the same manner as for a non-stack

limit DAC event. In this case, the instruction and access will normally have completed, in the same manner as a normal DAC event.

Independent limit checks for supervisor and user accesses may be implemented by allocating independent DACn resources to each, or a single limit may be applied using a single DACn resource. Typically, a DAC1,2 pair operating in exclusive address range mode is utilized for stack limit checking for a single shared limit. For separate limits, DAC3,4 may also be utilized. If more than one DACn resource is utilized, a DAC hit on any resource utilized for stack limit checking will cause the corresponding stack limit exception action to occur. If both a hardware-owned and a software-owned resource generate a stack limit exception for a given load or store, the software resource will have priority, since it is detected prior to completion of the access, and the access is aborted, thus the hardware event will not occur.

Enabling of this functionality is described in more detail in the description of the DACnCFG fields in *Table 1857* and *Table 1860* in *Section 71.3.2, Debug Control and Status registers*. Note that for DAC1 and DAC2, access type (read, write) control is part of DBCR0.

71.5 External debug support

External debug support is supplied through the OnCE controller serial interface, which allows access to internal CPU registers and other system state while the CPU is halted in debug mode. All debug resources including DBCR0–8, DBSR, IAC1–8, DAC1–4, and DVC1–2 are accessible through the serial OnCE interface in external debug mode. Setting the $EDBCR0_{EDM}$/$DBCR0_{EDM}$ bit to '1' through the OnCE interface enables external debug mode, and unless otherwise permitted by the settings in EDBRAC0, disables software updates to the debug control registers. When $EDBCR0_{EDM}$ is set, debug events enabled to set respective status bits will also cause the CPU to enter Debug mode if the event is not masked in EDBSRMSK0, as opposed to generating Debug Interrupts, unless the specific events are allocated to software via the settings in EDBRAC0. In Debug mode, the CPU is halted at a recoverable boundary, and an external Debug Control Module may control CPU operation through the On-Chip Emulation logic (OnCE).

Note that the descriptions of events in the subsections of *Section 71.2, Software debug events and exceptions* refer to setting DBSR status bits, however, when resources are owned by hardware, the events for those resources set the respective status bits in EDBSR0 instead of DBSR.

Note: On the initial setting of $EDBCR0_{EDM}$ to '1', other bits in DBCR0 will remain unchanged. After $EDBCR0_{EDM}$ has been set, all debug register resources may be subsequently controlled through the OnCE interface. The CPU should be placed into debug mode via the OCR_{DR} control bit prior to writing EDM to '1'. This gives the debugger the opportunity to cleanly write to the DBCRx registers and the DBSR to clear out any residual state / control information that could cause unintended operation.

*It is intended for the CPU to remain in external debug mode ($EDBCR0_{EDM}$=1) in order to single step or perform other debug mode entry/ reentry via the OCR_{DR}, by performing go+noexit commands, or by assertion of the **jd_de_b** signal.*

*$EDBCR0_{EDM}$ operation will be blocked if OnCE operation is disabled (**jd_en_once** negated) regardless of whether it is set or cleared. This means that if $EDBCR0_{EDM}$ was previously set, and then **jd_en_once** is negated (this should not occur), entry into debug mode will be blocked, and all hardware debug events are blocked. Watchpoints are not blocked.*

Due to clock domain design, the CPU clock (**m_clk**) must be active in order to perform writes to debug registers other than the OnCE Command register (OCMD), the OnCE Control register (OCR), External Debug Control Register 0 (EDBCR0), External Debug Status register 0 (EDBSR0), External Debug Status Register Mask 0 (EDBSRMSK0), or the $EDBCR0_{EDM}$ bit. Register read data is synchronized back to the **j_tclk** clock domain. The OnCE Control register provides the capability of signaling the system level clock controller that the CPU clock should be activated if not already active.

Updates to the DBCRx, DBSR, and most other debug registers via the OnCE interface should be performed with the CPU in debug mode to guarantee proper operation. Due to the various points in the CPU pipeline where control is sampled and event handshaking is performed, it is possible that modifications to these registers while the CPU is running may result in early or late entry into debug mode, and may have incorrect status posted in the DBSR register.

If resource sharing is enabled via EDBRAC0, updates to the EDBRAC0, DBCRx, and DBSR registers must be performed with the CPU in debug mode, since simultaneous updates of register portions could otherwise be attempted, and such updates are not guaranteed to properly occur. The results of such an attempt are undefined.

71.5.1 External debug registers

The external debug registers are used for controlling several debug aspects of the core and reporting status while in External Debug mode.

71.5.1.1 External Debug Control Register 0 (EDBCR0)

EDBCR0 is a control register accessible to an external debugger through the OnCE/Jtag port. An external development tool can write to this register in order to enable external debug mode or to enable Debugger Notify Halt instructions (**e_dnh**, **se_dnh**), as well as to control aspects of Debugger Notify Interrupt instructions (**e_dni**, **se_dni**).

EDBCR0 is not accessible by software, However, the state of $EDBCR0_{EDM}$ is reflected as a read-only bit in $DBCR0_{EDM}$ to software. There is only one physical EDM bit implemented; it is reflected in both the DBCR0 and EDBCR0 registers, and may be written and read using either register by the hardware debugger. For future compatibility, EDBCR0 updates are preferred.

EDBCR0 is shown in *Figure 1932*.

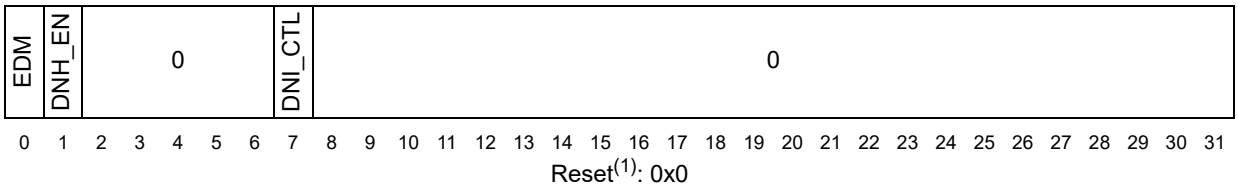


Figure 1932. External Debug Control Register 0 (EDBCR0)

1. EDBCR0 is affected (reset) by **j_trst_b** or **m_por** assertion, and remains reset while in the Test_Logic_Reset state, but is not affected by **p_reset_b**.

Table 1867 provides bit definitions for External Debug Control Register 0.

Table 1867. EDBCR0 field descriptions

Bit	Name	Description
0	EDM	External Debug Mode. This bit is also reflected in DBCR0. When external debug mode is enabled, hardware-owned resources in debug registers are not affected by processor reset **p_reset_b**. This allows the debugger to set up hardware debug events that remain active across a processor reset. 0 External debug mode disabled. Internal debug events not mapped into external debug events. 1 External debug mode enabled. Hardware-owned events will not cause the CPU to vector to interrupt code. Software is not permitted to write to debug registers {DBCRx, IAC1–8, DAC1–4, DVC1–2[U]} unless permitted by settings in EDBRAC0.
1	DNH_EN	**dnh** Instruction Enable 0 Execution of **e_dnh** and **se_dnh** instructions cause illegal instruction exceptions to occur. 1 Execution of **e_dnh** and **se_dnh** instructions cause entry into debug mode and a debug halt occurs, regardless of the value of EDM.
2:6	—	Reserved
7	DNI_CTL	**dni** Instruction Control 0 When the **dni** resource is owned by hardware, the MSR_{DE} bit is cared, and when MSR_{DE}=0, execution of **e_dni** and **se_dni** instructions are nop'ed and no entry into debug mode occurs. 1 When the **dni** resource is owned by hardware, the MSR_{DE} bit is don't-cared, and execution of **e_dni** and **se_dni** instructions cause entry into debug mode and a debug halt occurs, regardless of the value of MSR_{DE}. **Note:** This control bit is only used when the dni resource is owned by hardware via control in $EDBRAC0_{DNI}$, and thus also only when EDM=1.
8:31	—	Reserved

71.5.1.2 External Debug Status Register 0 (EDBSR0)

The External Debug Status Register 0 (EDBSR0) contains status on debug events owned by hardware. The External Debug Status Register 0 is set via hardware, and read and cleared via OnCE access by the debugger. Clearing is done by writing to the External Debug Status Register via the OnCE port, with a '1' in any bit position that is to be cleared and '0' in all other bit positions. The write data to EDBSR0 is not direct data, but a mask. A '1' causes the bit to be cleared, and a '0' has no effect. The EDBSR0 register is shown in *Figure 1933*.

IDE	UDE	DNH	0	ICMP	BRT	IRPT	TRAP	IAC	0	DACR	DACW	0	DNI	RET	0	DEVT1	DEVT2	PMI	MPU	CIRPT	CRET	VLES	DAC_OFST	0
0	1	2	3	4	5	6	7	8	9 10 11	12	13	14	15	16	17 18 19 20	21	22	23	24	25	26	27	28 29 30	31

Read/Write; Reset(1) - 0x0000_0000

Figure 1933. External Debug Status Register 0 (EDBSR0)

1. Reset by **j_trst_b** or **m_por** assertion, and remains reset while in the Test_Logic_Reset state or while $EDBCR0_{EDM}$=0.

Table 1868 provides bit definitions for External Debug Status Register 0.

Table 1868. EDBSR0 field descriptions

Bit	Name	Description
0	IDE	Imprecise Debug Event Set to 1 if $EDBCR0_{EDM}$=1 and an imprecise debug event occurs for a hardware-owned DAC event due to a load or store that is terminated with error, or if a hardware-owned ICMP event occurs in conjunction with a EFPU round exception. This bit will not be set for imprecise debug events that are masked via settings in EDBSRMSK0.
1	UDE	Unconditional Debug Event Set to 1 if a hardware-owned Unconditional debug event occurred.
2	DNH	Debugger Notify Halt Event Set to 1 if a debugger notify halt instruction was executed and caused a debug halt.
3	—	Reserved
4	ICMP	Instruction Complete Debug Event Set to 1 if a hardware-owned Instruction Complete debug event occurred.
5	BRT	Branch Taken Debug Event Set to 1 if a hardware-owned Branch Taken debug event occurred.
6	IRPT	Interrupt Taken Debug Event Set to 1 if a hardware-owned Interrupt Taken debug event occurred.
7	TRAP	Trap Taken Debug Event Set to 1 if a hardware-owned Trap Taken debug event occurred.
8	IAC	Instruction Address Compare 1 Debug Event Set to 1 if a hardware-owned IAC debug event occurred.
9:11	—	Reserved
12	DACR	Data Address Compare Read Debug Event Set to 1 if a hardware-owned read-type DAC debug event occurred
13	DACW	Data Address Compare Write Debug Event Set to 1 if a hardware-owned write-type DAC1 debug event occurred
14	—	Reserved
15	DNI	DNI Debug Event Set to 1 if a hardware-owned DNI debug event occurred
16	RET	Return Debug Event Set to 1 if a hardware-owned Return debug event occurred

Table 1868. EDBSR0 field descriptions (continued)

Bit	Name	Description
17:20	—	Reserved
21	DEVT1	External Debug Event 1 Debug Event Set to 1 if a hardware-owned DEVT1 debug event occurred
22	DEVT2	External Debug Event 2 Debug Event Set to 1 if a hardware-owned DEVT2 debug event occurred
23	PMI	Performance Monitor Interrupt Debug Event Set to 1 if a Performance Monitor Interrupt event occurred with $EDBRAC0_{PMI}$=0 regardless of the setting of $PMGC0_{UDI}$
24	MPU	Memory Protection Unit Debug Event Set to 1 if a hardware-owned MPU debug event occurred. For MPU debug events, if $EDBSRMSK0_{MPU}$ is cleared, the IAC, DACR, or DACW status bit will also be set, depending on the type of access that matched a region descriptor enabled to generate debug events, in order to further define the type of MPU debug event. If $EDBSRMSK0_{MPU}$ is set at the time a MPU debug event occurs, the IAC, DACR, and DACW status bits will not be set by MPU debug events, in order to properly mask the MPU debug event. Note that hardware MPU debug events on data accesses normally occur after the data access is performed and the instruction completes, thus act like a normal DAC debug event. The instruction may not complete if a DSI occurs. In this case, IDE will be set to indicate this occurrence.
25	CIRPT	Critical Interrupt Taken Debug Event Set to 1 if a hardware-owned Critical Interrupt Taken debug event occurred.
26	CRET	Critical Return Debug Event Set to 1 if a hardware-owned Critical Return debug event occurred
27	VLES	VLE Status Set to 1 if a hardware-owned ICMP, BRT, TRAP, RET, CRET, IAC, or DAC debug event occurred on a PowerISA VLE Instruction. Also set for execution of an e_dnh or se_dnh instruction when enabled by $EDBCR0_{DNH_EN}$. Undefined for IRPT, CIRPT, DEVT[1,2], and UDE events
28:30	DAC_OFST	Data Address Compare Offset Indicates offset-1 of saved DSRR0 value from the address of the load or store instruction that took a hardware-owned DAC Debug exception, unless a simultaneous DSI error occurs, in which case this field is set to 3‘b000 and $EDBSR0_{IDE}$ is set to 1. Normally set to 3‘b000 by a non-DVC DAC. A DVC DAC may set this field to any value.
31	—	Reserved

71.5.1.3 External Debug Status Register Mask 0 (EDBSRMSK0)

The External Debug Status Register Mask 0 (EDBSRMSK0) is used to mask debug events set in EDBSR0 from causing entry into debug halted mode. A ‘1’ stored in any mask bit prevents debug mode entry caused by the corresponding bit being set in EDBSR0. The mask has no effect on DBSR actions or on the setting of EDBSR0 status bits by hardware-owned events, except that the IDE bit will not be set by imprecise hardware-owned debug events that are masked. EDBSRMSK0 may be used to allow debug events owned by hardware to be configured for watchpoint generation purposes without causing debug mode entry when the watchpoint occurs. EDBSRMSK0 is read and written via OnCE access by

the debugger. No software access is provided. The EDBSRMSK0 register is shown in *Figure 1934*.

0	UDE	DNH	0	ICMP	BRT	IRPT	TRAP	IAC	0	DACR	DACW	0	DNI	RET	0	DEVT1	DEVT2	PMI	MPU	CIRPT	CRET	0
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31

Read/Write; Reset(1) - 0x0000_0000

Figure 1934. External Debug Status Register Mask 0 (EDBSRMSK0)

1. Reset by **j_trst_b** or **m_por** assertion, and remains reset while in the Test_Logic_Reset state or while $EDBCR0_{EDM}$=0.

Table 1869 provides bit definitions for External Debug Status Register Mask 0.

Table 1869. EDBSRMSK0 field descriptions

Bit	Name	Description
0	—	Reserved
1	UDE	Unconditional Debug Event Set to 1 to mask debug mode entry by $EDBSR0_{UDE}$
2	DNH	Debugger Notify Halt Event Set to 1 to mask debug mode entry by $EDBSR0_{DNH}$
3	—	Reserved
4	ICMP	Instruction Complete Debug Event Set to 1 to mask debug mode entry by $EDBSR0_{ICMP}$
5	BRT	Branch Taken Debug Event Set to 1 to mask debug mode entry by $EDBSR0_{BRT}$
6	IRPT	Interrupt Taken Debug Event Set to 1 to mask debug mode entry by $EDBSR0_{IRPT}$
7	TRAP	Trap Taken Debug Event Set to 1 to mask debug mode entry by $EDBSR0_{TRAP}$
8	IAC	Instruction Address Compare Debug Event Set to 1 to mask debug mode entry by $EDBSR0_{IAC}$
9:11	—	Reserved
12	DACR	Data Address Compare 1 Read Debug Event Set to 1 to mask debug mode entry by $EDBSR0_{DACR}$
13	DACW	Data Address Compare 1 Write Debug Event Set to 1 to mask debug mode entry by $EDBSR0_{DACW}$
14	—	Reserved
15	DNI	DNI Debug Event Set to 1 to mask debug mode entry by $EDBSR0_{DNI}$
16	RET	Return Debug Event Set to 1 to mask debug mode entry by $EDBSR0_{RET}$
17:20	—	Reserved

Table 1869. EDBSRMSK0 field descriptions (continued)

Bit	Name	Description
21	DEVT1	External Debug Event 1 Debug Event Set to 1 to mask debug mode entry by $EDBSR0_{DEVT1}$
22	DEVT2	External Debug Event 2 Debug Event Set to 1 to mask debug mode entry by $EDBSR0_{DEVT2}$
23	PMI	Performance Monitor Interrupt Debug Event Set to 1 to mask debug mode entry by $EDBSR0_{PMI}$
24	MPU	Memory Protection Unit Debug Event Set to 1 to mask debug mode entry by $EDBSR0_{MPU}$
25	CIRPT	Critical Interrupt Taken Debug Event Set to 1 to mask debug mode entry by $EDBSR0_{CIRPT}$
26	CRET	Critical Return Debug Event Set to 1 to mask debug mode entry by $EDBSR0_{CRET}$
22:31	—	Reserved

71.5.1.4 External Debug Data Effective Address Register (EDDEAR)

The External Debug Data Effective Address Register (EDDEAR) contains address information for hardware-owned data address compare debug events, including MPU DAC events. EDDEAR is update by hardware with the effective address of the load, store, or cache control operation when an unmasked data address compare event is recorded in EDBSR0 if the previous values of $EDBSR0_{DAC\{R,W\}}$ bits that are unmasked in EDBSRMSK0 are zero. Once an EDDEAR update is performed, these unmasked bits must be cleared by the hardware debugger prior to another EDDEAR hardware update occurring. A subsequent hardware-owned unmasked DAC event will not update the EDDEAR register if either of the $EDBSR0_{DAC\{R,W\}}$ bits are set and are not masked by EDBSRMSK0, in order to capture the first unmasked event address. EDDEAR is read and written via OnCE access by the debugger. No software access is provided.

The EDDEAR register is shown in *Figure 1935*.

Data Effective Address

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

Read/Write; Reset: unaffected

Figure 1935. External Debug Data Effective Address Register (EDDEAR)

71.5.2 OnCE introduction

The e200z420n3 on-chip emulation circuitry (OnCE™/Nexus Class 1 interface) provides a means of interacting with the e200z420n3 core and integrated system so that a user may examine registers, memory, or on-chip peripherals facilitating hardware/software development. OnCE operation is controlled via an industry standard IEEE 1149.1 TAP controller. By using public instructions, the external hardware debugger can freeze or halt the CPU, read and write internal state, and resume normal execution. The core does not contain IEEE 1149.1 standard boundary cells on its interface, as it is a building block for further integration. It does not support the JTAG related boundary scan instruction

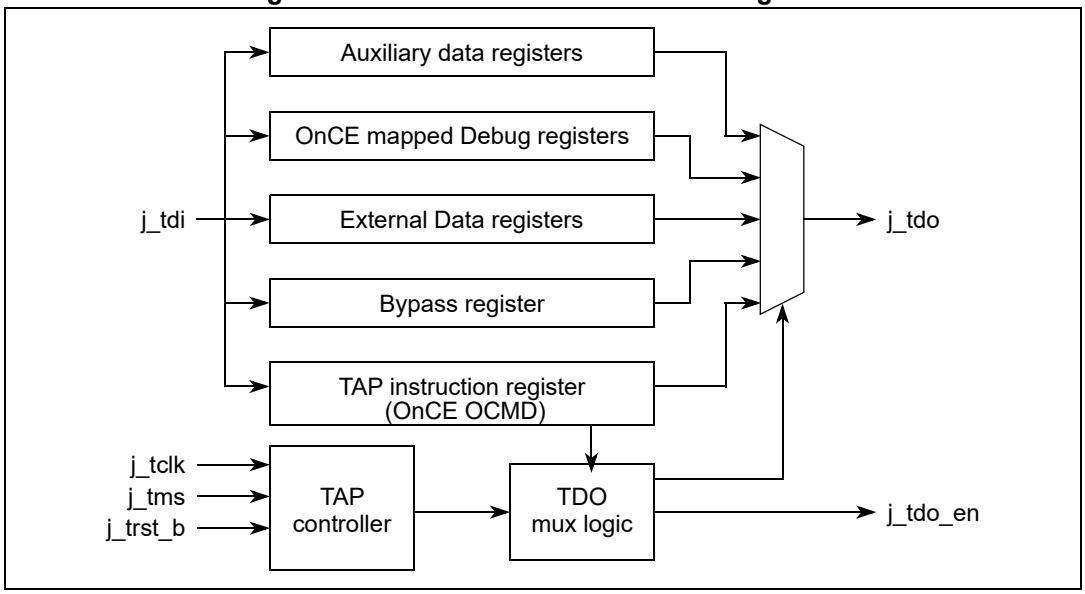
functionality, although JTAG public instructions may be decoded and signaled to external logic.

The OnCE logic provides for Nexus Class 1 static debug capability (utilizing the same set of resources available to software while in internal debug mode).

In order to enable full OnCE operation, the **jd_enable_once** input signal must be asserted. In some system integrations, this is automatic, since the input will be tied asserted. Other integrations may require the execution of the Enable OnCE command via the TAP and appropriate entry of serial data. Exact requirements will be documented by the integrated product specification. The **jd_enable_once** input signal should not change state during a debug session, or undefined activity may occur.

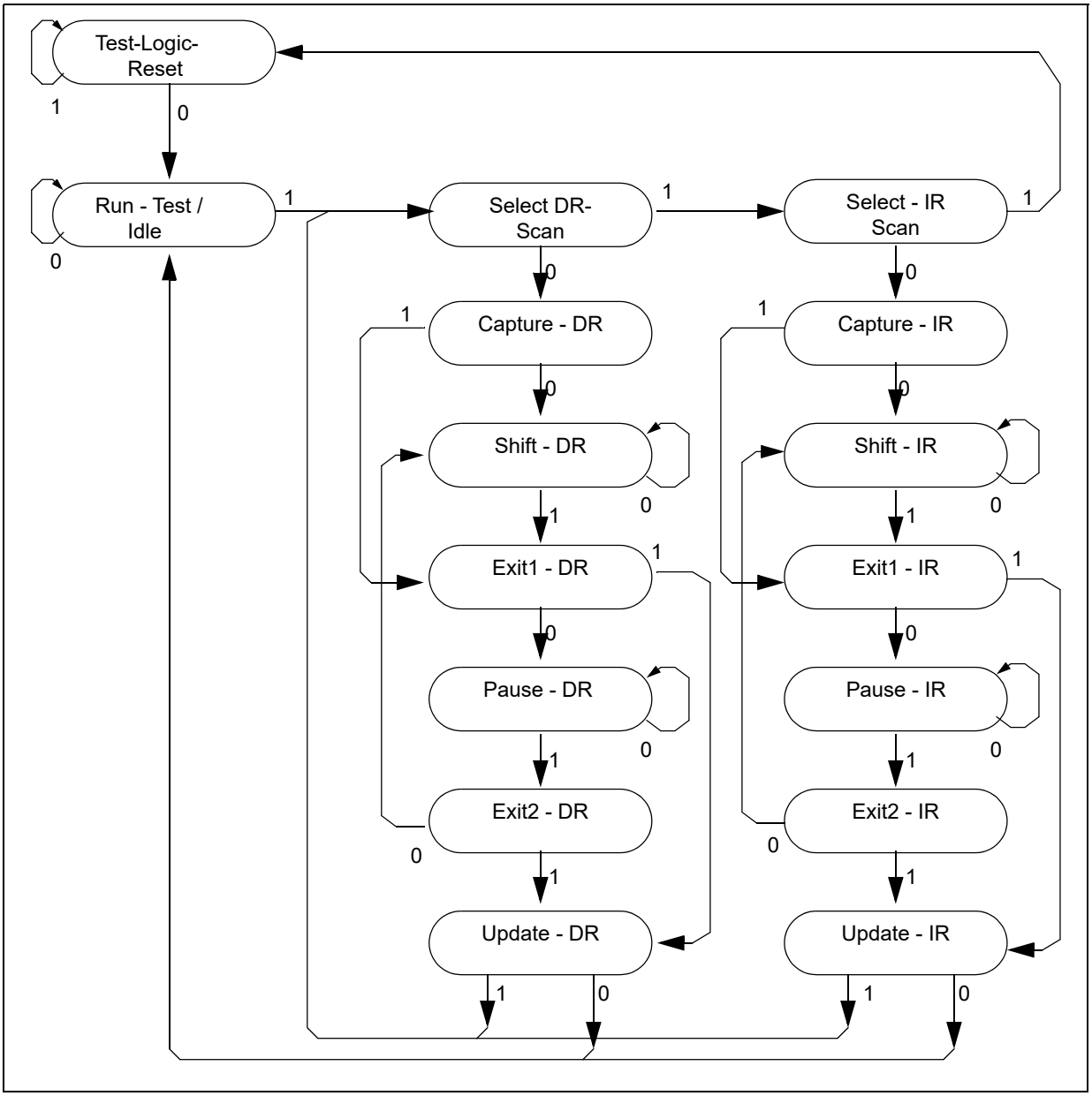
The following figures show the TAP controller state model and the TAP registers implemented by the OnCE logic.

Figure 1936. OnCE TAP controller and registers



The OnCE controller is implemented as a 16-state FSM, with a one-to-one correspondence to the states defined for the JTAG TAP controller.

Figure 1937. OnCE controller implementation



Access to processor registers and the contents of memory locations are performed by enabling external debug mode (setting $EDBCR0_{EDM}$ to '1'), placing the processor into debug mode, followed by scanning instructions and data into and out of the CPU Scan Chain (CPUSCR); execution of scanned instructions by the CPU is used as the method to access required data. Memory locations may be read by scanning a load instruction into the CPU core, which will reference the desired memory location, executing the load instruction, and then scanning out the result of the load. Other resources are accessed in a similar manner.

The initial entry by the CPU into the debug state (or mode) from normal, Waiting, Stopped, or Halted states (all indicated via the OnCE Status Register (OSR), *Section 71.5.6.1, OnCE Status register*) by assertion of one or more debug requests, begins a *debug session*. The **jd_debug_b** output signal indicates that a debug session is in progress, and the OSR will

indicate the CPU is in the debug state. Instructions may the be single-stepped by scanning new values into the CPUSCR, and performing a OnCE go+noexit command (refer to *Section 71.5.6.2, OnCE Command (OCMD) register*). The CPU will then temporarily exit the debug state (but not the debug session) to execute the instruction, and will then return to the debug state (again indicated via the OnCE Status Register (OSR)). The debug session remains in force until the final OnCE go+exit command is executed, at which time the CPU will return to the previous state it was in (unless a new debug request is pending). A scan into the CPUSCR is required prior to executing each go+exit or go+noexit OnCE command.

71.5.3 JTAG/OnCE pins

The JTAG/OnCE pin interface is used to transfer OnCE instructions and data to the OnCE control block. Depending on the particular resource being accessed, the CPU may need to be placed in the Debug mode. For resources outside of the CPU block and contained in the OnCE block, the processor is not disturbed, and may continue execution. If a processor resource is required, an internal debug request (**dbg_dbgrq**) may be asserted to the CPU by the OnCE controller, and causes the CPU to finish the current instruction being executed, save the instruction pipeline information, enter Debug mode, and wait for further commands. Asserting **dbg_dbgrq** will cause the chip to temporarily exit the Waiting, Stopped or Halted power management states.

Table 1870 details the primary JTAG/OnCE interface signals.

Table 1870. JTAG/OnCE primary interface signals

Signal name	Type	Description
j_trst_b	I	JTAG test reset
j_tclk	I	JTAG test clock
j_tms	I	JTAG test mode select
j_tdi	I	JTAG test data input
j_tdo	O	Test data out to master controller or pad
j_tdo_en(1)	O	Enables TDO output buffer

1. j_tdo_en is asserted when the TAP controller is in the shift_DR or shift_IR state.

A full description of JTAG pins is provided in the JTAG Support Signals section of the Core (e200z420n3) Core Complex Overview.

71.5.4 OnCE internal interface signals

The following paragraphs describe the OnCE interface signals to other internal blocks associated with the OnCE controller.

71.5.4.1 CPU Debug Request (dbg_dbgrq)

The **dbg_dbgrq** signal is asserted by the OnCE control logic to request the CPU to enter the debug state. It may be asserted for a number of different conditions, and causes the CPU to finish the current instruction being executed, save the instruction pipeline information, enter the debug mode, and wait for further commands.

71.5.4.2 CPU Debug Acknowledge (cpu_dbgack)

The **cpu_dbgack** signal is asserted by the CPU upon entering the debug state. This signal is used as part of the handshake mechanism between the OnCE control logic and the rest of the CPU. The CPU core may enter debug mode either through a software or hardware event.

71.5.4.3 CPU address and attributes

The CPU address and attribute information are used by the Nexus3 unit with information for real-time address trace information.

71.5.4.4 CPU data

The CPU data buses are used to supply the Nexus3 debug unit with information for real-time data trace capability.

71.5.5 OnCE interface signals

The following paragraphs describe additional OnCE interface signals to other external blocks such as a Nexus controller and external blocks that may need information pertaining to debug operation.

71.5.5.1 OnCE Enable (jd_en_once)

The OnCE enable signal **jd_en_once** is used to enable the OnCE controller to allow certain instructions and operations to be executed. Assertion of this signal will enable the full OnCE command set, as well as operation of control signals and OnCE Control register functions. When this signal is disabled, only the Bypass, ID and Enable_OnCE commands are executed by the OnCE unit, and all other commands default to a “Bypass” command. The OnCE Status register (OSR) is not visible when OnCE operation is disabled. In addition, OnCE Control register (OCR) functions are disabled, as is the operation of the **jd_de_b** input. Secure systems may choose to leave the **jd_en_once** signal negated until a security check has been performed. Other systems should tie this signal asserted to enable full OnCE operation. The **j_en_once_regsel** output signal is provided to assist external logic performing security checks.

The **jd_en_once** input must only change state during the Test-Logic-Reset, Run-Test/Idle, or Update_DR TAP states. A new value will take effect after one additional **j_tclk** cycle of synchronization. In addition, **jd_enable_once** input signal must not change state during a debug session, or undefined activity may occur.

71.5.5.2 OnCE Debug Request/Event (jd_de_b, jd_de_en)

If implemented at the SoC level, a system level bidirectional open drain debug event pin **DE_b** (not part of the e200z420n3 interface) provides a fast means of entering the Debug mode of operation from an external command controller (when input) as well as a fast means of acknowledging the entering of the Debug mode of operation to an external command controller (when output). The assertion of this pin by a command controller causes the CPU core to finish the current instruction being executed, save the instruction pipeline information, enter Debug mode, and wait for commands to be entered. If **DE_b** was used to enter the Debug mode then **DE_b** must be negated after the OnCE controller responds with an acknowledge and before sending the first OnCE command. The assertion of this pin by the CPU Core acknowledges that it has entered the Debug mode and is waiting for commands to be entered.

To support operation of this system pin, the OnCE logic supplies the **jd_de_en** output and samples the **jd_de_b** input when OnCE is enabled (**jd_en_once** asserted). Assertion of **jd_de_b** will cause the OnCE logic to place the CPU into Debug mode. Once Debug mode has been entered, the **jd_de_en** output will be asserted for three **j_tclk** periods to signal an acknowledge. **jd_de_en** can be used to enable the open-drain pulldown of the system level **DE_b** pin.

For systems that do not implement a system level bidirectional open drain debug event pin **DE_b**, the **jd_de_en** and **jd_de_b** signals may still be used to handshake debug entry.

71.5.5.3 OnCE Debug output (jd_debug_b)

The OnCE Debug output **jd_debug_b** is used to indicate to on-chip resources that a debug session is in progress. Peripherals and other units may use this signal to modify normal operation for the duration of a debug session, which may involve the CPU executing a sequence of instructions solely for the purpose of visibility/system control that are not part of the normal instruction stream the CPU would have executed had it not been placed in debug mode. This signal is asserted the first time the CPU enters the debug state, and remains asserted until the CPU is released by a write to the OnCE Command Register with the GO and EX bits set, and a register specified as either “No Register Selected” or the CPUSCR. This signal will remain asserted even though the CPU may enter and exit the debug state for each instruction executed under control of the OnCE controller. Refer to *Section 71.5.6.2, OnCE Command (OCMD) register* for more information on the function of the GO and EX bits. This signal is not normally used by the CPU.

71.5.5.4 CPU Clock On input (jd_mclk_on)

The CPU Clock On input **jd_mclk_on** is used to indicate that the CPU’s **m_clk** input is active. This input signal is expected to be driven by system logic external to the e200z420n3 core, is synchronized to the **j_tclk** (scan clock) clock domain, and is presented as a status flag on the **j_tdo** output during the Shift_IR state. External firmware may use this signal to ensure proper scan sequences will occur to access debug resources in the **m_clk** clock domain.

71.5.5.5 Watchpoint events (jd_watchpt[0:31])

The **jd_watchpt[0:31]** signals may be asserted by the OnCE control logic to signal that a watchpoint condition has occurred. Watchpoints do not directly cause the CPU to be affected. They are provided to allow external visibility only or for triggering purposes. Watchpoint events are conditioned by the settings in the DBCRxx registers, as well as by the DEVENT register, the DTC/DTSA/DTEA registers, the Performance Monitor control register settings, and generation of MPU debug events. Refer to *Table 1877* for details of the signal assignments. Note that assertion of most watchpoint outputs is conditioned on being in EDM or IDM. The Performance monitor, DEVENT, and DTC watchpoints are not conditioned however, and may assert regardless of the state of EDM or IDM. In addition, DAC1 or DAC3 watchpoints may be generated for stack limit violation occurrences when those resources are configured to perform stack limit checking, regardless of the state of EDM or IDM.

71.5.5.6 Update DR w/go+exit (j_ocmd_go_exit)

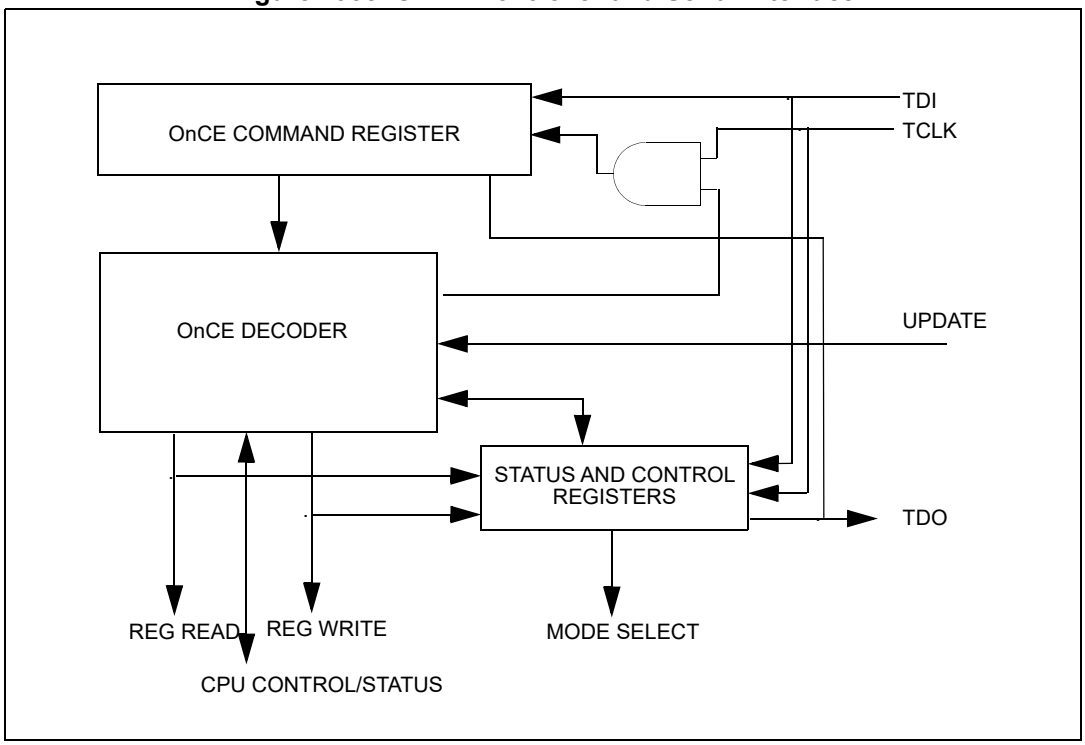
This signal indicates the TAP controller is in the Update_DR state and that the go and exit bits in the OnCE Command register are high, and RS indicates “no register selected”. This signal will assert regardless of whether the CPU is currently in debug mode. It may be

monitored by external logic to cause a synchronous exit from debug mode of other modules (but not other CPUs) in the system. A debugger can place all of the CPU tap controllers in parallel, scan in a go+exit command with “no register selected”, and sequence through the Update_DR state to cause any CPU currently in debug mode to exit. CPUs that are not in debug mode will ignore the command.

71.5.6 OnCE controller and serial interface

The OnCE Controller contains the OnCE command register, the OnCE decoder, and the status/control register. *Figure 1938* is a block diagram of the OnCE controller. In operation, the OnCE Command register acts as the IR for the TAP controller, and all other OnCE resources are treated as data registers (DR) by the TAP controller. The Command register is loaded by serially shifting in commands during the TAP controller Shift-IR state, and is loaded during the Update-IR state. The Command register selects a resource to be accessed as a data register (DR) during the TAP controller Capture-DR, Shift-DR, and Update-DR states.

Figure 1938. OnCE Controller and Serial Interface



71.5.6.1 OnCE Status register

Status information regarding the state of the CPU is latched into the OnCE Status register when the OnCE controller state machine enters the Capture-IR state. When OnCE operation is enabled, this information is provided on the **j_tdo** output in serial fashion when the Shift_IR state is entered following a Capture-IR. Information is shifted out least significant bit first.

MCLK	ERR	0	RESET	HALT	STOP	DEBUG	WAIT	0	1
0	1	2	3	4	5	6	7	8	9

Figure 1939. OnCE Status register

Table 1871 provides bit definitions for the OnCE Status register.

Table 1871. OnCE Status register field descriptions

Bit	Name	Description
0	MCLK	MCLK **m_clk** Status Bit 0 Inactive state 1 Active state This status bit reflects the logic level on the **jd_mclk_on** input signal after capture by **j_tclk**.
1	ERR	ERROR This bit is used to indicate that an error condition occurred during attempted execution of the last single-stepped instruction (GO+NoExit with CPUSCR or No Register Selected in OCMD), and that the instruction may not have been properly executed. This could occur if an Interrupt (all classes including such as External, Critical, machine check, Storage, Alignment, Program) occurred while attempting to perform the instruction single step. In this case, the CPUSCR will contain information related to the first instruction of the Interrupt handler, and no portion of the handler will have been executed.
2	—	Reserved, set to zero
3	RESET	RESET Mode This bit reflects the inverted logic level on the CPU **p_reset_b** input after capture by **j_tclk**.
4	HALT	HALT Mode This bit reflects the logic level on the CPU **p_halted** output after capture by **j_tclk**.
5	STOP	STOP Mode This bit reflects the logic level on the CPU **p_stopped** output after capture by **j_tclk**.
6	DEBUG	Debug Mode This bit is asserted once the CPU is in debug mode. It is negated once the CPU exits debug mode (even during a debug session)
7	WAIT	Waiting Mode This bit reflects the logic level on the CPU **p_waiting** output after capture by **j_tclk**.
8	0	Reserved, set to 0 for 1149.1 compliance
9	1	Reserved, set to 1 for 1149.1 compliance

71.5.6.2 OnCE Command (OCMD) register

The OnCE Command (OCMD) register is a 10-bit shift register that receives its serial data from the TDI pin and serves as the instruction register (IR). It holds the 10-bit commands to be used as input for the OnCE Decoder. The Command Register is shown in *Figure 1940*. The OCMD is updated when the TAP controller enters the Update-IR state. It contains fields for controlling access to a resource, as well as controlling single-step operation and exit from OnCE mode.

Although the OCMD is updated during the Update-IR TAP controller state, the corresponding resource is accessed in the DR scan sequence of the TAP controller, and as

such, the Update-DR state must be transitioned through in order for an access to occur. In addition, the Update-DR state must also be transitioned through in order for the single-step or exit functionality, or both, to be performed, even though the command appears to have no data resource requirement associated with it.

R/W	GO	EX	RS[0:6]
0	1	2	3 4 5 6 7 8 9

Reset - 10'b1000000010 on assertion of **j_trst_b** or **m_por**, or while in the Test_Logic_Reset state

Figure 1940. OnCE Command (OCMD) register

Table 1872 provides bit definitions for the OCMD.

Table 1872. OCMD field descriptions

Bit	Name	Description
0	R/W	Read/Write Command Bit The R/W bit specifies the direction of data transfer. The table below describes the options defined by the R/W bit. **Note:** The R/W bit generally ignored for read-only or write-only registers, although the PC FIFO pointer is only guaranteed to be update when R/W=1. In addition, it is ignored for all bypass operations. When performing writes, most registers are sampled in the Capture-DR state into a 32-bit shift register, and subsequently shifted out on **j_tdo** during the first 32 clocks of Shift-DR. 0 Write the data associated with the command into the register specified by RS[0:6] 1 Read the data contained in the register specified by RS[0:6]
1	GO	Go Command Bit If the GO bit is set, and the CPU is currently in debug mode, the CPU will execute the instruction that resides in the IR register in the CPUSCR. To execute the instruction, the processor leaves the debug mode, executes the instruction, and if the EX bit is cleared, returns to the debug mode immediately after executing the instruction. The processor goes on to normal operation if the EX bit is set, and no other debug request source is asserted. The GO command is executed only if the operation is a read/write to CPUSCR or a read/write to "No Register Selected". Otherwise the GO bit is ignored.The processor will leave the debug mode after the TAP controller Update-DR state is entered. On a GO+NoExit operation, returning to debug mode is treated as a debug event, thus exceptions such as machine checks and interrupts may take priority and prevent execution of the intended instruction. Debug firmware should mask these exceptions as appropriate. The OSR_{ERR} bit indicates such an occurrence. If the CPU is not currently in debug mode, then the GO command will be ignored. **Note:** Asynchronous interrupts are blocked on a GO+Exit operation until the first instruction to be executed begins execution. Refer to *Section 71.5.9.6, Exiting Debug mode and interrupt blocking*. 0 Inactive (no action taken) 1 Execute instruction in IR

Table 1872. OCMD field descriptions (continued)

Bit	Name	Description
2	EX	Exit Command Bit 0 Remain in debug mode 1 Leave debug mode If the EX bit is set, the processor will leave the debug mode and resume normal operation until another debug request is generated. The Exit command is executed only if the Go command is issued, and the operation is a read/write to CPUSCR or a read/write to “No Register Selected”. Otherwise the EX bit is ignored. The processor will leave the debug mode after the TAP controller Update-DR state is entered. **Note:** If the DR bit in the OnCE control register is set or remains set, or if a bit in EDBSR0 is set and $EDBCR0_{EDM}$=1 (external debug mode is enabled), or if another debug request source is asserted, then the processor may return to the debug mode *without* execution of an instruction, even though the EX bit was set. **Note:** Asynchronous interrupts are blocked on a GO+Exit operation until the first instruction to be executed begins execution. Refer to *Section 71.5.9.6, Exiting Debug mode and interrupt blocking*.
3:9	RS	Register Select The Register Select bits define which register is source (destination) for the read (write) operation. *Table 1873* indicates the OnCE register addresses. Attempted writes to read-only registers are ignored.

Table 1873 indicates the OnCE register addresses.

Table 1873. OnCE register addressing

RS[0:6]	Register Selected
000 0000	Reserved
000 0001	Reserved
000 0010	JTAG ID (read-only)
000 0011–000 1111	Reserved
001 0000	CPU Scan Register (CPUSCR)
001 0001	No Register Selected (Bypass)
001 0010	OnCE Control Register (OCR)
001 0011	Reserved
001 0100–001 0111	Reserved
001 1000	Data Address Compare 3 (DAC3)
001 1001	Data Address Compare 4 (DAC4)
001 1010–001 1111	Reserved
010 0000	Instruction Address Compare 1 (IAC1)
010 0001	Instruction Address Compare 2 (IAC2)
010 0010	Instruction Address Compare 3 (IAC3)
010 0011	Instruction Address Compare 4 (IAC4)

Table 1873. OnCE register addressing (continued)

RS[0:6]	Register Selected
010 0100	Data Address Compare 1 (DAC1)
010 0101	Data Address Compare 2 (DAC2)
010 0110	Data Value Compare 1 (DVC1) — *all 64 bits of the DVC register are accessed
010 0111	Data Value Compare 2 (DVC2) — *all 64 bits of the DVC register are accessed
010 1000	Instruction Address Compare 5 (IAC5)
010 1001	Instruction Address Compare 6 (IAC6)
010 1010	Instruction Address Compare 7 (IAC7)
010 1011	Instruction Address Compare 8 (IAC8)
010 1100	Debug Data Effective Address (DDEAR)
010 1101	External Debug Data Effective Address (EDDEAR)
010 1110	External Debug Control Register 0 (EDBCR0)
010 1111	External Debug Status Register 0 (EDBSR0)
011 0000	Debug Status Register (DBSR)
011 0001	Debug Control Register 0 (DBCR0)
011 0010	Debug Control Register 1 (DBCR1)
011 0011	Debug Control Register 2 (DBCR2)
011 0100	Reserved (do not access)
011 0101	Debug Control Register 4 (DBCR4)
011 0110	Debug Control Register 5 (DBCR5)
011 0111	Debug Control Register 6 (DBCR6)
011 1000	Debug Control Register 7 (DBCR7)
011 1001	Debug Control Register 8 (DBCR8)
011 1010	Reserved (do not access)
011 1011	Reserved (do not access)
011 1100	External Debug Status Register MASK 0 (EDBSRMSK0)
011 1101	Debug Data Acquisition Message Register (DDAM)
011 1110	Debug Event Control (DEVENT)
011 1111	External Debug Resource Allocation Control 0 (EDBRAC0)
100 0000	Debug L1 Cache Control/Status 0 (DBL1CCSR0)
100 0001–110 1100	Reserved (do not access)
110 1101	MPU0 Control/Status 0 (MPU0CSR0)
110 1110	Performance Monitor Register Access
110 1111	Reserved for Shared Nexus Control Register Select
111 0000–111 1001	General Purpose register selects [0:9]
111 1010	Cache Debug Access Control Register (CDACNTL)

Table 1873. OnCE register addressing (continued)

RS[0:6]	Register Selected
111 1011	Cache Debug Access Data Register (CDADATA)
111 1100	Nexus3 Access
111 1101	LSRL Select (refer to Test Specification)
111 1110	Enable_OnCE(1)
111 1111	Bypass

1. Causes assertion of the j_en_once_regsel output.

The OnCE Decoder receives as input the 10-bit command from the OCMD, and status signals from the processor, and generates all the strobes required for reading and writing the selected OnCE registers.

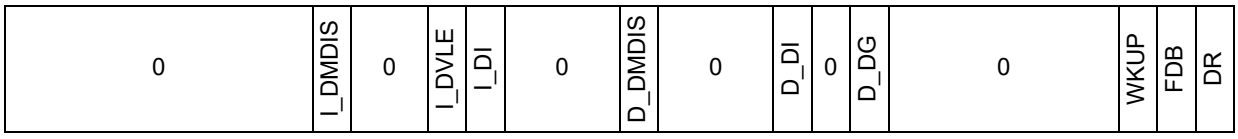
Single-stepping of instructions is performed by placing the CPU in debug mode, scanning in appropriate information into the CPUSCR, and setting the Go bit (with the EX bit cleared) with the RS field indicating either the CPUSCR or No Register Selected. After executing a single instruction, the CPU will re-enter debug mode and await further commands. During single-stepping, exception conditions may occur if not properly masked by debug firmware (such as interrupts, machine checks, bus error conditions) and may prevent the desired instruction from being successfully executed. The OSR_{ERR} bit is set to indicate this condition. In these cases, values in the CPUSCR will correspond to the first instruction of the exception handler.

Additionally, the $EDBCR0_{EDM}$ bit is forced to '1' internally while single-stepping to prevent Debug events from generating Debug interrupts. Also, during a debug session, the DBSR register is frozen from updates due to debug events other than execution of a DNI-type instruction, regardless of $EDBCR0_{EDM}$. DBSR may still be modified during a debug session via a single-stepped **mtspr** instruction, or via OnCE access.

If the CPU is not currently in debug mode, go+exit and go+noexit commands are ignored, although for a go+exit command with "No Register Selected", the j_ocmd_go_exit output will be asserted.

71.5.6.3 OnCE Control Register (OCR)

The OnCE Control Register is a 32-bit register used to force the e200z420n3 core into debug mode and to enable / disable sections of the OnCE control logic. It also provides control over the MPU during a debug session (refer to *Section 71.7, MPU operation during debug*). The control bits are read/write. These bits are only effective while OnCE is enabled (**jd_en_once** asserted). The OCR is shown in *Figure 1941*.



Reset - 0xo000_0000 on **m_por**, **j_trst_b**, or entering Test_logic_Reset state

Figure 1941. OnCE Control Register (OCR)

Table 1874 provides bit definitions for the OnCE Control Register.

Table 1874. OCR field descriptions

Bit	Name	Description
0:7	—	Reserved
8	I_DMDIS	Instruction Side Debug MPU Disable Control Bit This bit may be used to control whether the MPU is enabled normally, or whether the MPU is disabled during a debug session for Instruction Accesses. When enabled, the MPU functions normally. When disabled, for Instruction Accesses, the I bit is taken from the OCR bit I_DI. The SX and UX access permission control bits are set to‘1’ to allow full access. When disabled, no MPU exceptions are generated for Instruction accesses. External access errors can still occur. MPU entries with DEBUG=1 are not affected by this control bit. 0 MPU not disabled for debug sessions 1 MPU disabled for debug sessions
9:10	—	Reserved
11	I_DVLE	Instruction Side Debug ‘VLE’ Attribute Bit This bit is used to provide the ‘VLE’ attribute bit to be used during a debug session. **Note:** This bit is ignored, since VLE is always enabled.
12	I_DI	Instruction Side Debug ‘I’ Attribute Bit This bit is used to provide the ‘I’ attribute bit to be used for Instruction accesses for Instruction accesses during a debug session.
13:15	—	Reserved
16	D_DMDIS	Data Side Debug MPU Disable Control Bit This bit may be used to control whether the MPU is enabled normally, or whether the MPU is disabled during a debug session for Data Accesses. When enabled, the MPU functions normally. When disabled, for Data Accesses, the MPU I and G bits are taken from the OCR bits D_DI and D_DG. The SR, SW, UR, and UW access permission control bits are set to‘1’ to allow full access. When disabled, no MPU exceptions are generated for Data accesses. External access errors can still occur. MPU entries with DEBUG=1 are not affected by this control bit. 0 MPU not disabled for debug sessions 1 MPU disabled for debug sessions
17:19	—	Reserved
20	D_DI	Data Side Debug ‘I’ Attribute Bit This bit is used to provide the ‘I’ attribute bit to be used for Data accesses during a debug session.
21	—	Reserved
22	D_DG	Data Side Debug ‘G’ Attribute Bit This bit is used to provide the ‘G’ attribute bit to be used for Data accesses during a debug session.
23:28	—	Reserved
29	WKUP	Wakeup Request Bit This control bit may be used to force the **p_wakeup** output signal to be asserted. This control function may be used by debug firmware to request that the chip-level clock controller restore the **m_clk** input to normal operation regardless of whether the CPU is in a low power state to ensure that debug resources may be properly accessed by external hardware through scan sequences.

Table 1874. OCR field descriptions (continued)

Bit	Name	Description
30	FDB	Force Breakpoint Debug Mode Bit This control bit is used to determine whether the processor is operating in breakpoint debug enable mode or not. The processor may be placed in breakpoint debug enable mode by setting this bit. In breakpoint debug enable mode, execution of the '**bkpt**' pseudo-instruction will cause the processor to enter debug mode, as if the **jd_de_b** input had been asserted. This bit is qualified with $EDBCR0_{EDM}$, which must be set for FDB to take effect. **Note:** This bit has no effect on **e_dnh** or **se_dnh** instruction operation.
31	DR	CPU Debug Request Control Bit This control bit is used to unconditionally request the CPU to enter Debug mode. The CPU will indicate that Debug mode has been entered via the data scanned out in the shift-IR state. 0 No Debug mode request 1 Unconditional Debug mode request When the DR bit is set, the processor will enter Debug mode at the next instruction boundary.

71.5.7 Access to debug resources

Resources contained in the OnCE Module that do not require the processor core to be halted for access may be accessed while the e200z420n3 core is running, and will not interfere with processor execution. Accesses to other resources such as the CPUSCR require the e200z420n3 core to be placed in debug mode to avoid synchronization hazards. Debug firmware may ensure that it is safe to access these resources by determining the state of the e200z420n3 core prior to access. Note that a scan operation to update the CPUSCR is required prior to exiting debug mode if debug mode has been entered.

Some cases of write accesses other than accesses to the OnCE Command and Control registers, or the EDM bit of DBCR0 require **m_clk** to be running for proper operation. The OnCE control register provides a means of signaling this need to a system level clock control module via the OCR_{WKUP} control bit.

In addition, since the CPU may cause multiple bits of certain registers to change state, reads of certain registers while the CPU is running (such as DBSR) may not have consistent bit settings unless read twice with the same value indicated. In order to guarantee that the contents are consistent, the CPU should be placed into debug mode, or multiple reads should be performed until consistent values have been obtained on consecutive reads.

Table 1875 provides a list of access requirements for OnCE registers.

Table 1875. OnCE register access requirements

Register name	Access requirements					Notes
	Requires jd_en_once to be asserted	Requires EDBCR0 $_{EDM}$ = 1 for write access	Requires m_clk active for write access	Requires CPU to be halted for read access	Requires CPU to be halted for write access	
Enable_OnCE	N	N	N	N	—	
Bypass	N	N	N	N	N	
CPUSCR	Y	Y	Y	Y	Y	
DAC1–4	Y	Y	Y	N	*(1)	
DBCR0	Y	Y	Y	N	*(1)	*EDBCR0$_{EDM}$ access only requires **jd_en_once** asserted
DBCR1–8	Y	Y	Y	N	*(1)	
DEVENT	Y	Y	Y	N	*(1)	
EDBRAC0 (DBERC0)	Y	N	Y	N	*(1)	
DBSR	Y	Y	Y	N(2)	*(1)	
EDBCR0	Y	N	N	N	N	
EDBSR0	Y	N	Y	N	N	
EDBSRMSK0	Y	N	N	N	N	
IAC1–8	Y	Y	Y	N	*(1)	
JTAG ID	N	N	—	N	—	Read-only
MPU0CSR0	Y	N	Y	N	N	
OCR	Y	N	N	N	N	
OSR	Y	N	—	N	—	Read-only, accessed by scanning out IR while **jd_en_once** is asserted
Cache Debug Access Control (CDACNTL)	Y	N	Y	N	N	CPU gives lowest priority to a Cache access request when it is not debug halted
Cache Debug Access Data (CDADATA)	Y	N	Y	N	N	CPU gives lowest priority to a Cache access request when it is not debug halted
Nexus3-Access	Y	N	N	N	N	
PMR-Access	Y	N	N	N	N	
PMR Registers	Y	Y	Y	N	N	

Table 1875. OnCE register access requirements (continued)

Register name	Access requirements					Notes
	Requires jd_en_once to be asserted	Requires EDBCR0 EDM = 1 for write access	Requires m_clk active for write access	Requires CPU to be halted for read access	Requires CPU to be halted for write access	
External GPRs	Y	N	N	N	N	
LSRL Select	Y	N	—	—	—	System Test logic implementation determines LSRL functionality

1. Writes to these registers while the CPU is running may have unpredictable results due to the pipelined nature of operation, and the fact that updates are not synchronized to a particular clock, instruction, or bus cycle boundary, therefore it is strongly recommended to ensure the processor is first placed into debug mode before updates to these registers are performed.
2. Reads of these registers while the CPU is running may not give data that is self-consistent due to synchronization across clock domains.

71.5.8 Methods of entering Debug mode

The OnCE Status Register indicates that the CPU has entered the debug mode via the DEBUG status bit. The following sections describe how Debug mode is entered assuming the OnCE circuitry has been enabled. OnCE operation is enabled by the assertion of the **jd_en_once** input. Refer to *Section 71.5.5.1, OnCE Enable (jd_en_once)* for details.

71.5.8.1 External debug request during RESET

Holding the **jd_de_b** signal asserted during the assertion of **p_reset_b**, and continuing to hold it asserted following the negation of **p_reset_b** causes the e200z420n3 core to enter Debug mode. After receiving an acknowledge via the OnCE Status Register DEBUG bit, the external command controller should negate the **jd_de_b** signal before sending the first command. Note that in this case the e200z420n3 core does not execute an instruction before entering Debug mode, although the first instruction to be executed will be fetched prior to entering Debug mode in order to properly initialize the CPUSCR.

71.5.8.2 Debug request during RESET

Asserting a debug request by setting the DR bit in the OCR during the assertion of **p_reset_b** and then negating **p_reset_b** causes the chip to enter debug mode. In this case the CPU will fetch the first instruction of the reset exception handler in order to properly initialize the CPUSCR, but does not execute an instruction before entering debug mode.

71.5.8.3 Debug request during normal activity

Asserting a debug request by setting the DR bit in the OCR during normal chip activity causes the chip to finish the execution of the current instruction and then enter the debug mode. Note that in this case the chip completes the execution of the current instruction and stops after the newly fetched instruction enters the CPU instruction register. This process is the same for any newly fetched instruction including instructions fetched by the interrupt processing, or those that will be aborted by the interrupt processing.

71.5.8.4 Debug request during Waiting, Halted, or Stopped state

Asserting a debug request by setting the DR bit in the OCR when the chip is in the Waiting state (**p_waiting** asserted), Halted state (**p_halted** asserted) or Stopped state (**p_stopped** asserted) causes the CPU to exit the state and enter the debug mode once the CPU clock **m_clk** has been restored. Note that in this case, the CPU will negate the **p_waiting, p_halted** and **p_stopped** outputs. Once the debug session has ended, the CPU will return to the state it was in prior to entering debug mode.

To signal the chip-level clock generator to re-enable **m_clk**, the **p_wakeup** output will be asserted whenever the debug block is asserting a debug request to the CPU due to OCR_{DR} being set, or **jd_de_b** assertion, and will remain set from then until the debug session ends (**jd_debug_b** goes from asserted to negated). In addition, the status of the **jd_mclk_on** input (after synchronization to the **j_tclk** clock domain) may be sampled along with other status bits from the **j_tdo** output during the Shift_IR TAP controller state. This status may be used if necessary by external debug firmware to ensure proper scan sequences occur to registers in the **m_clk** clock domain.

71.5.8.5 Software request during normal activity

Upon executing a '**bkpt'** pseudo-instruction (for e200z420n3, defined to be an all 0's instruction opcode) when the OCR register's (FDB) bit is set (debug mode enable control bit is true), and $EDBCR0_{EDM}$=1, the CPU enters the debug mode after the instruction following the '**bkpt'** pseudo-instruction has entered the instruction register.

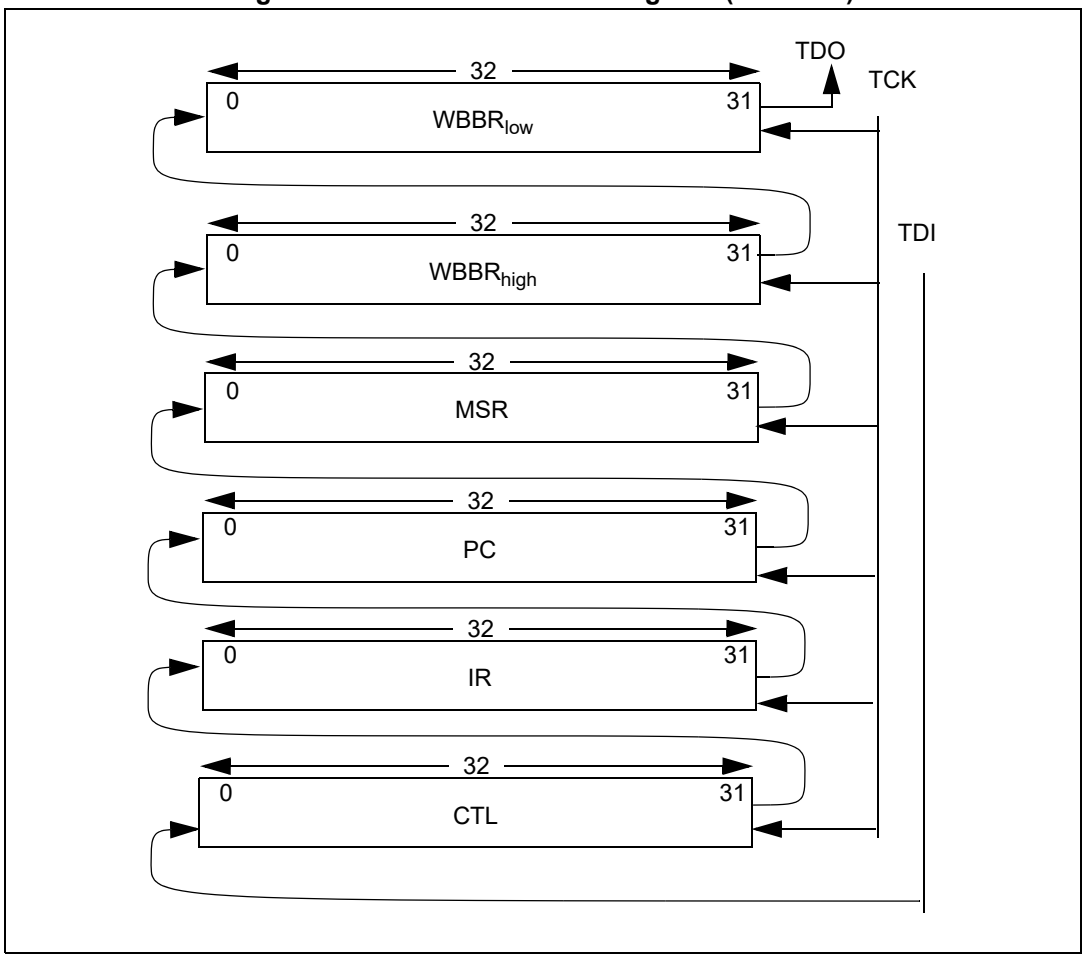
71.5.8.6 Debug notify halt instructions

The **e_dnh** and **se_dnh** instructions allow software to transition the core from a running state to a debug halted state if enabled by $EDBCR0_{DNH_EN}$, and provide the external debugger with bits reserved in the instruction itself to pass additional information. Entry into debug mode is ***not*** conditioned on $EDBCR0_{EDM}$, allowing for debug of software debug handlers as well as other software. For e200z420n3, when the CPU enters a debug halted state due to a **e_dnh** or **se_dnh** instruction, the instruction will be stored in the CPUSCR[IR] portion, and the CPUSCR[PC] value will point to the instruction. The external debugger should update the CPUSCR prior to exiting the debug halted state to point past the **e_dnh** or **se_dnh** instruction.

71.5.9 CPU Status and Control Scan Chain Register (CPUSCR)

A number of on-chip registers store the CPU pipeline status and are configured in a single scan chain for access by the OnCE controller. The CPUSCR register contains these processor resources, which are used to restore the pipeline and resume normal chip activity upon return from the debug mode, as well as a mechanism for the emulator software to access processor and memory contents. *Figure 1942* shows the block diagram of the pipeline information registers contained in the CPUSCR. Once debug mode has been entered, it is required to scan in and update this register prior to exiting debug mode.

Figure 1942. CPU Scan Chain Register (CPUSCR)



71.5.9.1 Instruction Register (IR)

The Instruction Register (IR) provides a mechanism for controlling the debug session by serving as a means for forcing in selected instructions, and then causing them to be executed in a controlled manner by the debug control block. The opcode of the next instruction to be executed when entering debug mode is contained in this register when the scan-out of this chain begins. This value should be saved for later restoration if continuation of the normal instruction stream is desired.

On scan-in, in preparation for exiting debug mode, this register is filled with an instruction opcode selected by debug control software. By selecting appropriate instructions and controlling the execution of those instructions, the results of execution may be used to examine or change memory locations and processor registers. The debug control module external to the processor core will control execution by providing a single-step capability. Once the debug session is complete and normal processing is to be resumed, this register may be loaded with the value originally scanned out.

71.5.9.2 Control State Register (CTL)

The Control State Register (CTL) is a 32-bit register that stores the value of certain internal CPU state variables before the debug mode is entered. This register is affected by the operations performed during the debug session and should normally be restored by the

external command controller when returning to normal mode. In addition to saved internal state variables, two of the bits are used by emulation firmware to control the debug process. In certain circumstances, emulation firmware must modify the content of this register as well as the PC and IR values in the CPUSCR before exiting debug mode. These cases are described below.

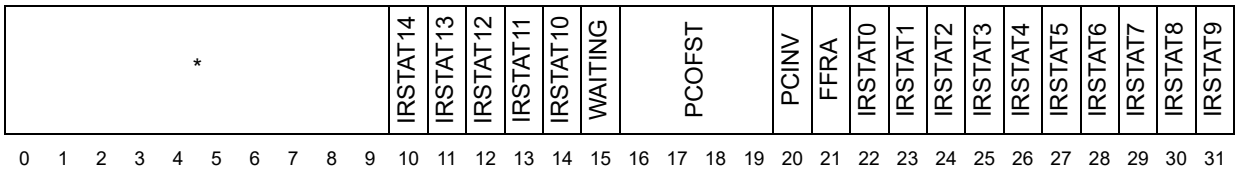


Figure 1943. Control State Register (CTL)

Table 1876. CTL field descriptions

Bit	Name	Description
0:9	*	Internal State Bits These control bits represent internal processor state and should be restored to their original value after a debug session is completed, i.e when a OnCE command is issued with the GO and EX bits set and not ignored. When performing instruction execution during a debug session. Refer to *Section 71.5.5.3, OnCE Debug output (jd_debug_b)*, which is not part of the normal program execution flow, these bits should be set to a 0.
10	IRSTAT14	IR Status Bit 14 This control bit indicates an MPU Instruction Address Compare event status for the IR. 0 No MPU Instruction Address Compare event occurred on the fetch of this instruction. 1 An MPU Instruction Address Compare event occurred on the fetch of this instruction.
11	IRSTAT13	IR Status Bit 13 This control bit indicates an Instruction Address Compare 8 event status for the IR. 0 No Instruction Address Compare 8 event occurred on the fetch of this instruction. 1 An Instruction Address Compare 8 event occurred on the fetch of this instruction.
12	IRSTAT12	IR Status Bit 12 This control bit indicates an Instruction Address Compare 7 event status for the IR. 0 No Instruction Address Compare 7 event occurred on the fetch of this instruction. 1 An Instruction Address Compare 7 event occurred on the fetch of this instruction.
13	IRSTAT11	IR Status Bit 11 This control bit indicates an Instruction Address Compare 6 event status for the IR. 0 No Instruction Address Compare 6 event occurred on the fetch of this instruction. 1 An Instruction Address Compare 6 event occurred on the fetch of this instruction.
14	IRSTAT10	IR Status Bit 10 This control bit indicates an Instruction Address Compare 5 event status for the IR. 0 No Instruction Address Compare 5 event occurred on the fetch of this instruction. 1 An Instruction Address Compare 5 event occurred on the fetch of this instruction.
15	WAITING	WAITING State Status This bit indicates whether the CPU was in the waiting state prior to entering debug mode. If set, the CPU was in the waiting state. Upon exiting a debug session, the value of this bit in the restored CPUSCR will determine whether the CPU re-enters the waiting state on a go+exit. 0 CPU was not in the waiting state when debug mode was entered 1 CPU was in the waiting state when debug mode was entered

Table 1876. CTL field descriptions (continued)

Bit	Name	Description
16:19	PCOFST	PC Offset Field This field indicates whether the value in the PC portion of the CPUSCR must be adjusted prior to exiting debug mode. Due to the pipelined nature of the CPU, the PC value must be backed up by emulation software in certain circumstances. The PCOFST field specifies the value to be subtracted from the original value of the PC. This adjusted PC value should be restored into the PC portion of the CPUSCR just prior to exiting debug mode with a go+exit. In the event the PCOFST is non-zero, the IR should be loaded with a nop instruction instead of the original IR value, other wise the original value of IR should be restored. (Refer to PCINV, which overrides this field). 0000 No correction required. 0001 Subtract 0x04 from PC. 0010 Subtract 0x08 from PC. 0011 Subtract 0x0C from PC. 0100 Subtract 0x10 from PC. 0101 Subtract 0x14 from PC. All other encodings are reserved
20	PCINV	PC and IR Invalid Status Bit This status bit indicates that the values in the IR and PC portions of the CPUSCR are invalid. Exiting debug mode with the saved values in the PC and IR will have unpredictable results. Debug firmware should initialize the PC and IR values in the CPUSCR with desired values prior to exiting debug mode if this bit was set when debug mode was initially entered. 0 No error condition exists. 1 Error condition exists. PC and IR are corrupted.
21	FFRA	Feed Forward RA Operand Bit This control bit causes the content of the WBBRlo to be used as the RA operand value (RS for logical, mtspr, mtdcr, cntlzw, and shift operations, RX for VLE se_ instructions, RT for e_{logical_op}2i type instructions, RB for evaddiw, evsubifw, and the value to use as the PC for calculating the LR update value for branch with link type instructions) of the first instruction to be executed following an update of the CPUSCR.For most LSP instructions using rA||rB as a 64-bit source operand, WBBRhi, lo is used to supply the 64-bit source value. This allows the debug firmware to update processor registers — initialize the WBBRlo with the desired value, set the FFRA bit, and execute a ori Rx,Rx,0 instruction to the desired register. Note: not all instructions support using the FFRA control. FFRA is mainly intended for use with the ori instruction to allow the debugger to write to a GPR. Support for other instructions is implementation-dependent. 0 No action. 1 Content of $WBBR_{lo}$ used as operand value.
22	IRSTAT0	IR Status Bit 0 This control bit indicates a TEA status for the IR. 0 No TEA occurred on the fetch of this instruction. 1 TEA occurred on the fetch of this instruction.
23	IRSTAT1	IR Status Bit 1 This control bit is reserved.

Table 1876. CTL field descriptions (continued)

Bit	Name	Description
24	IRSTAT2	IR Status Bit 2 This control bit indicates an Instruction Address Compare 1 event status for the IR. 0 No Instruction Address Compare event 1 occurred on the fetch of this instruction. 1 An Instruction Address Compare event 1 occurred on the fetch of this instruction.
25	IRSTAT3	IR Status Bit 3 This control bit indicates an Instruction Address Compare 2 event status for the IR. 0 No Instruction Address Compare 2 event occurred on the fetch of this instruction. 1 An Instruction Address Compare 2 event occurred on the fetch of this instruction.
26	IRSTAT4	IR Status Bit 4 This control bit indicates an Instruction Address Compare 3 event status for the IR. 0 No Instruction Address Compare event 3 occurred on the fetch of this instruction. 1 An Instruction Address Compare event 3 occurred on the fetch of this instruction.
27	IRSTAT5	IR Status Bit 5 This control bit indicates an Instruction Address Compare 4 event status for the IR. 0 No Instruction Address Compare 4 event occurred on the fetch of this instruction. 1 An Instruction Address Compare 4 event occurred on the fetch of this instruction.
28	IRSTAT6	IR Status Bit 6 This control bit indicates a Parity Error status for the IR. 0 No Parity Error occurred on the fetch of this instruction. 1 Parity Error occurred on the fetch of this instruction from the I-Cache or the IMEM.
29	IRSTAT7	IR Status Bit 7 This control bit indicates a Precise External Termination Error status for the IR. 0 No Precise External Termination Error occurred on the fetch of this instruction. 1 A Precise External Termination Error occurred on the fetch of this instruction.
30	IRSTAT8	IR Status Bit 8 This control bit indicates the PowerISA VLE status for the IR. 0 IR contains a BookE instruction. (unused encoding) 1 IR contains a PowerISA VLE instruction, aligned in the Most Significant Portion of IR if 16-bit. **Note:** This bit should not be modified by the debugger, otherwise the resulting operation is boundedly undefined. It should always indicate VLE (=1).
31	IRSTAT9	IR Status Bit 9 This control bit is reserved.

Emulation firmware should modify the content of the CTL, PC, and IR values in the CPUSCR during execution of debug related instructions as well as just prior to exiting debug with a go+exit command. During the debug session, the CTL register should be written with the FFRA bit set as appropriate, and all other bit set to ‘0’, and the IR set to the value of the desired instruction to be executed. IRStat8 will be used to determine the type of instruction present in the IR.

Just prior to exiting debug mode with a go+exit, the PCINV status bit that was originally present when debug mode was first entered should be tested, and if set, the PC and IR initialized for performing whatever recovery sequence is appropriate for a faulted exception vector fetch. If the PCINV bit is cleared, then the PCOFST bits should be examined to determine whether the PC value must be adjusted. Due to the pipelined nature of the CPU,

the PC value must be backed up by emulation software in certain circumstances. The PCOFST field specifies the value to be subtracted from the original value of the PC. This adjusted PC value should be restored in to the PC portion of the CPUSCR just prior to exiting debug mode with a go+exit. In the event the PCOFST is non-zero, the IR should be loaded with a nop instruction (such as **ori r0,r0,0**) instead of the original IR value, otherwise the original value of IR should be restored. Note that when a correction is made to the PC value, it will generally point to the last completed instruction, although that instruction will not be re-executed. The nop instruction is executed instead, and instruction fetch and execution will resume at location PC + 4. IRStat8 will be used to determine the type of instruction present in the IR, thus should be cleared in this case. Note that debug events that may occur on the nop (ICMP) will be generated if enabled.

For the CTL register, the internal state bits should be restored to their original value. The IRStatus bits should be set to '0's if the PC was adjusted. If no PC adjustment was performed, emulation firmware should determine whether other IRStat flags should be set to '0' to avoid re-entry into debug mode for an instruction breakpoint request. Upon exiting debug mode with go+exit, if one of these bits is set, debug mode will be reentered prior to any further instruction execution.

71.5.9.3 Program Counter (PC) register

The PC is a 32-bit register that stores the value of the program counter that was present when the chip entered the debug mode. It is affected by the operations performed during the debug mode and must be restored by the external command controller when the CPU returns to normal mode. PC normally points to the instruction contained in the IR portion of CPUSCR. If debug firmware wishes to redirect program flow to an arbitrary location, the PC and IR should be initialized to correspond to the first instruction to be executed upon resumption of normal processing. Alternatively, the IR may be set to a nop and the PC set to point to the location prior to the location at which it is desired to redirect flow to. On exiting debug mode, the nop will be executed, and instruction fetch and execution will resume at PC+4.

71.5.9.4 Write-Back Bus Register ($WBBR_{low}$, $WBBR_{high}$)

WBBR is used as a means of passing operand information between the CPU and the external command controller. Whenever the external command controller needs to read the contents of a register or memory location, it will force the chip to execute an instruction that brings that information to WBBR. $WBBR_{low}$ holds the 32-bit result of most instructions including load data returned for a load or load with update instruction. $WBBR_{high}$ holds the updated effective address calculated by a load with update instruction. It is undefined for other instructions.

As an example, to read the 32 bits of processor register **r1**, an **ori r1,r1,0** instruction is executed, and the result value of the instruction will be latched into $WBBR_{low}$. The contents of $WBBR_{low}$ can then be delivered serially to the external command controller. To update a processor resource, this register is initialized with a data value to be written, and an **ori** instruction is executed, which uses this value as a substitute data value. The Control State register FFRA bit forces the value of the $WBBR_{low}$ to be substituted for the normal RS source value of the **ori** instruction, thus allowing updates to processor registers to be performed. (Refer to *Section 71.5.9.2, Control State Register (CTL)* for more detail on the CTL_{FFRA} bit.)

$WBBR_{low}$ and $WBBR_{high}$ are generally undefined on instructions that do not writeback a result, and due to control issues are not defined on **lmw** or branch instructions as well.

71.5.9.5 Machine State Register (MSR)

The MSR is a 32-bit register used to read/write the Machine State Register. Whenever the external command controller needs to save or modify the contents of the Machine State Register, this register is used.This register is affected by the operations performed during the debug mode and must be restored by the external command controller when returning to normal mode.

71.5.9.6 Exiting Debug mode and interrupt blocking

When exiting debug mode with a Go+Exit, “asynchronous” interrupts are blocked until the first instruction to be executed begins execution. This includes External and Critical input, NMI, and machine check interrupts. Asynchronous debug interrupts are not blocked however, and the CPU will reenter debug mode without executing an instruction following Go+Exit, although it may fetch an instruction and discard it. Exceptions due to an illegal instruction or error flags set within the CPUSCR CTL register are not blocked, since they apply to the instruction in the CPUSCR IR.

71.5.10 Reserved registers (Reserved)

The reserved registers are used to control various test control logic. These registers are not intended for customer use. To preclude device or system damage, or both, these registers should not be accessed.

71.6 Watchpoint support

e200z420n3 supports the generation and signalling of watchpoints when operating in internal debug mode ($DBCR0_{IDM}$=1) or in external debug mode ($EDBCR0_{EDM}$=1). Watchpoints are indicated with a dedicated set of interface signals. The **jd_watchpt[0:31]** output signals are used to indicate that a watchpoint has occurred. Certain watchpoints however (DEVENT-based, DTC-based, and Performance Monitor watchpoints) are not qualified with $EDBCR0_{EDM}$ or $DBCR0_{IDM}$.

Each debug address compare function (IAC1–8, DAC1–4), as well as other event types are capable of triggering a watchpoint output. The DBCRx control fields are used to configure watchpoints, regardless of whether events are enabled in DBCR0. Watchpoints may occur whenever an associated event would have been posted in the Debug Status Register if enabled. No explicit enable bits are provided for watchpoints; they are always enabled by definition, although the data address compare watchpoints may be controlled for read and write accesses via configuration fields in DBCR4, 7, and 9. During a debug session, debug events (other than PMI, DEVT1 and DEVT2) with a corresponding DBSR bit are blocked from asserting a watchpoint. The Performance Monitor, DEVNT-based and DTC-based watchpoints are not blocked during a debug session. If not desired, for address-based events the base address values for these events may be programmed to an unused system address. MSR_{DE} has no effect on watchpoint generation.

External logic may monitor the assertion of these signals for debugging or triggering purposes. Watchpoints are signaled in the clock cycle following the occurrence of the actual event. The Nexus3 module also monitors assertion of a portion of these output signals (**jd_watchpt[0:31])** for various development control purposes (refer to the “Watchpoint Trace Messaging” section in the Core (e200z420n3) Nexus 3 Module chapter).

Note that these signals are **m_clk** domain signals, not **j_tclk** domain signals.

Table 1877 shows the **jd_watchpt[0:31]** output signal assignments to various debug events.

Table 1877. Watchpoint output signal assignments

Signal Name	Type	Description
jd_watchpt[0]	IAC1	Instruction Address Compare 1 watchpoint Asserted whenever an IAC1 compare occurs regardless of being enabled to set DBSR status
jd_watchpt[1]	IAC2	Instruction Address Compare 2 watchpoint Asserted whenever an IAC2 compare occurs regardless of being enabled to set DBSR status
jd_watchpt[2]	IAC3	Instruction Address Compare 3 watchpoint Asserted whenever an IAC3 compare occurs regardless of being enabled to set DBSR status
jd_watchpt[3]	IAC4	Instruction Address Compare 4 watchpoint Asserted whenever an IAC4 compare occurs regardless of being enabled to set DBSR status
jd_watchpt[4]	DAC1	Data Address Compare 1 watchpoint Asserted whenever a DAC1 compare occurs regardless of being enabled to set DBSR status, based on $DBCR4_{DAC1CFG}$ settings
jd_watchpt[5]	DAC2	Data Address Compare 2 watchpoint Asserted whenever a DAC2 compare occurs regardless of being enabled to set DBSR status, based on $DBCR4_{DAC2CFG}$ settings
jd_watchpt[6]	DAC3	Data Address Compare 3 watchpoint Asserted whenever a DAC3 compare occurs regardless of being enabled to set DBSR status, based on $DBCR7_{DAC3CFG}$ settings
jd_watchpt[7]	DAC4	Data Address Compare 4 watchpoint Asserted whenever a DAC4 compare occurs regardless of being enabled to set DBSR status, based on $DBCR7_{DAC4CFG}$ settings
jd_watchpt[8]	IAC5	Instruction Address Compare 5 watchpoint Asserted whenever an IAC5 compare occurs regardless of being enabled to set DBSR status
jd_watchpt[9]	IAC6	Instruction Address Compare 6 watchpoint Asserted whenever an IAC6 compare occurs regardless of being enabled to set DBSR status
jd_watchpt[10]	DEVT1	Debug Event Input 1 watchpoint Asserted whenever a DEVT1 debug event occurs regardless of being enabled to set DBSR status
jd_watchpt[11]	DEVT2	Debug Event Input 2 watchpoint Asserted whenever a DEVT2 debug event occurs regardless of being enabled to set DBSR status
jd_watchpt[12]	DEVNT0	Debug Event Output 0 watchpoint Asserted whenever a ‘1’ is written to the bit of the DEVNT field of the DEVENT debug register corresponding to jd_watchpt[12]

Table 1877. Watchpoint output signal assignments (continued)

Signal Name	Type	Description
jd_watchpt[13]	DEVNT1	Debug Event Output 1 watchpoint Asserted whenever a '1' is written to the bit of the DEVNT field of the DEVENT debug register corresponding to jd_watchpt[13]
jd_watchpt[14]	IAC7	Instruction Address Compare 7 watchpoint Asserted whenever an IAC7 compare occurs regardless of being enabled to set DBSR status
jd_watchpt[15]	IAC8	Instruction Address Compare 8 watchpoint Asserted whenever an IAC8 compare occurs regardless of being enabled to set DBSR status
jd_watchpt[16]	IRPT	Interrupt watchpoint Asserted whenever an IRPT debug event occurs regardless of being enabled to set DBSR status
jd_watchpt[17]	RET	Return watchpoint Asserted whenever a RET debug event occurs regardless of being enabled to set DBSR status
jd_watchpt[18]	CIRPT	Critical Interrupt watchpoint Asserted whenever a CIRPT debug event occurs regardless of being enabled to set DBSR status
jd_watchpt[19]	CRET	Critical Return watchpoint Asserted whenever a CRET debug event occurs regardless of being enabled to set DBSR status
jd_watchpt[20]	DEVNT2	Debug Event Output 2 watchpoint Asserted whenever a '1' is written to the bit of the DEVNT field of the DEVENT debug register corresponding to jd_watchpt[20]
jd_watchpt[21]	DEVNT3	Debug Event Output 3 watchpoint Asserted whenever a '1' is written to the bit of the DEVNT field of the DEVENT debug register corresponding to jd_watchpt[21]
jd_watchpt[22]	PMEVENT	Performance Monitor Event input watchpoint Asserted whenever **p_pm_event** transitions from a '0' to a '1' while **m_clk** is running
jd_watchpt[23]	PMC0	Performance Monitor Counter 0 watchpoint Asserted whenever PMC0 triggers an event based on $PMLCa0_{PMP}$
jd_watchpt[24]	PMC1	Performance Monitor Counter 1 watchpoint Asserted whenever PMC1 triggers an event based on $PMLCa1_{PMP}$
jd_watchpt[25]	PMC2	Performance Monitor Counter 2 watchpoint Asserted whenever PMC2 triggers an event based on $PMLCa2_{PMP}$
jd_watchpt[26]	PMC3	Performance Monitor Counter 3 watchpoint Asserted whenever PMC3 triggers an event based on $PMLCa3_{PMP}$
jd_watchpt[27]	MPU	Memory Protection Unit watchpoint Asserted whenever a the MPU generates a debug event based on region descriptor matches with DEBUG control enabled.

Table 1877. Watchpoint output signal assignments (continued)

Signal Name	Type	Description
jd_watchpt[28]	TRAP	TRAP watchpoint Asserted whenever an TRAP debug event occurs regardless of being enabled to set DBSR status
jd_watchpt[29]	DTC1	Data Trace Control Range 1 watchpoint Asserted whenever an access meets the conditions for DTC Range 1
jd_watchpt[30]	DTC2	Data Trace Control Range 2 watchpoint Asserted whenever an access meets the conditions for DTC Range 2
jd_watchpt[31]	DTC3	Data Trace Control Range 3 watchpoint Asserted whenever an access meets the conditions for DTC Range 3

71.7 MPU operation during debug

A debug session begins when the CPU initially enters debug mode, and ends when a OnCE command with GO+EXIT is executed, releasing the CPU for normal operation. If desired during a debug session, the debug firmware may substitute default values obtained from the OnCE Control Register for Access Attribute (I, G) bits. Refer to the bit definitions in the OCR *Section 71.5.6.3, OnCE Control Register (OCR)* for more detail.

Normal operation of the MPU may be modified during a ‘debug session’ via the OnCE Control Register (OCR). A debug session begins when the CPU initially enters debug mode, and ends when a OnCE command with GO+EXIT is executed, releasing the CPU for normal operation. If desired during a debug session, the debug firmware may disable the MPU protection mechanism and may substitute default values for the Access Protection (UX, UR, UW, SX, SR, SW) bits, and values obtained from the OnCE Control Register for Memory Attributes (I, G) bits normally provided by a matching MPU entry.

When disabled during a debug session, no MPU-related storage interrupt conditions will occur. If the debugger desires to use the normal protection mechanism, the MPU may be left enabled in the OnCE OCR, and normal protections provided by the MPU (including the possibility of a storage interrupt) will remain in effect.

The OCR control bits are used when debug mode is entered. Refer to the bit definitions in the OCR (*Section 71.5.6.3, OnCE Control Register (OCR)* for more detail. When the MPU is disabled for instruction accesses (OCR_{I_DMDIS}) or for data accesses (OCR_{D_DMDIS}), substituted access attribute bits will control operation on respective accesses initiated during debug.

71.8 Cache array access during debug

The cache arrays may be read and written during debug mode via the CDACNTL and CDADATA debug registers.

71.9 Basic steps for enabling, using, and exiting External Debug mode

The following steps show one possible scenario for a debugger wishing to use the external debug facilities. *This simplified flow is intended to illustrate basic operations, but does not cover all potential methods in depth.*

Enabling External Debug mode and initializing debug registers:

- The debugger should ensure that the **jd_en_once** control signal is asserted in order to enable OnCE operation.
- Select the OCR and write a value to it in which OCR_{DR}, OCR_{WKUP}, are set to '1'. The tap controller must step through the proper states as outlined earlier. This step will place the CPU in a debug state in which it is halted and awaiting single-step commands or a release to normal mode
- Scan out the value of the OSR to determine that the CPU clock is running and the CPU has entered the Debug state. This can be done in conjunction with a Read of the CPUSCR. The OSR is shifted out during the Shift_IR state. The CPUSCR will be shifted out during the Shift_DR state. The debugger should save the scanned-out value of CPUSCR for later restoration.
- Select the DBCR0 register and update it with the $EDBCR0_{EDM}$ bit set.
- Clear the DBSR status bits.
- Write appropriate values to the DBCR0–8, IAC, and DAC registers. Note that the initial write to DBCR0 will only affect the EDM bit, so the remaining portion of the register must now be initialized, keeping the EDM bit set.

At this point the system is ready to commence debug operations. Depending on the desired operation, different steps must occur.

- Optionally, ensure that the values entered into the MSR portion of the CPUSCR during the following steps cause interrupt to be disabled (clearing MSR_{EE} and MSR_{CE}). This will ensure that external interrupt sources do not cause single-step errors.

To single-step the CPU:

- debugger scans in either a new or a previously saved value of the CPUSCR (with appropriate modification of the PC and IR as described in *Section 71.5.9.2, Control State Register (CTL)*), with a Go+Noexit OnCE Command value.
- The debugger scans out the OSR with "no-register selected", Go cleared, and determines that the PCU has reentered the Debug state and that no ERR condition occurred.

To return the CPU to normal operation (without disabling external debug mode)

- The OCR_{DR} control bit should be cleared, leaving the OCR_{WKUP} bit set.
- The debugger restores the CPUSCR with a previously saved value of the CPUSCR (with appropriate modification of the PC and IR as described in *Section 71.5.9.2, Control State Register (CTL)*), with a Go+Exit OnCE Command value.
- The OCR_{WKUP} bit may then be cleared.

To exit External Debug mode

- The debugger should place the CPU in the debug state via the OCR_{DR} with OCR_{WKUP} asserted, scanning out and saving the CPUSCR.
- The debugger should write the DBCR0–8 registers as needed, likely clearing every enable except the $EDBCR0_{EDM}$ bit.
- The debugger should write the DBSR to a cleared state.
- The debugger should rewrite the DBCR0 with all bits including EDM cleared.
- The debugger should clear the OCR_{DR} bit.
- The debugger restores the CPUSCR with the previously saved value of the CPUSCR (with appropriate modification of the PC and IR as described in *Section 71.5.9.2, Control State Register (CTL)*), with a Go+Exit OnCE Command value.
- The OCR_{WKUP} bit may then be cleared.

Note: These steps are meant by way of examples, and are not meant to be an exact template for debugger operation.

72 Debug and Calibration Interface (DCI)

72.1 Introduction

The Debug and Calibration Interface (DCI) module provides debug and calibration features for the MCU. It includes a standard 1149.1/1149.7 compatible JTAG interface which is used to connect with an external JTAG tool using a low frequency clock interface.

To provide high-speed calibration capability, the DCI also provides a mechanism to use 5-pin LFAST debug tool which shares its pins with the standard JTAG interface.

It also features a software based debug mode which can be controlled by the MCU without using an external tool. Additionally, the DCI provides features for debug control using break-points and synchronous restart of the cores. This chapter describes the DCI features for the Production device (PD) only.

72.1.1 Features

The DCI features are the following:

- Debug mode enable control for connected tool or software
- Port sharing logic to allow the 5-pin debug port to be shared between the JTAG and the LFAST
- 1149.1 and 1149.7 controllers
- Debug break and cross-triggering control
- Synchronous restart control for all CPUs when exiting debug mode
- Tool hot plug capability
- Security access control
- Debug reset control

72.1.2 Overview

The DCI can take the following different inputs for its JTAG signals:

- P.JTAG—JTAG signals coming from the Production device JTAG pads. This is the standard 5-pin JTAG interface and is the default operating mode.
- M.JTAG—JTAG signals generated by JTAGM module using LFAST or software debug mode.
  - When using LFAST with a connected calibration/debug tool, JTAG packets are received from the tool via LFAST and output JTAG packets are transmitted back to the tool. The LFAST's 5-pin interface consists of an LVDS pair for transmit signals, an LVDS pair of receive signals and a reference clock output signal.
  - In software debug mode, there is no tool, and debug software running on the MCU generates the JTAG packets and writes them to JTAGM using special debug registers in the JTAGM module.

By default the DCI operates in standard 5-pin JTAG mode (IEEE 1149.1). It can be configured in 3-pin reduced pin JTAG mode (IEEE 1149.7) after starting from standard 5-pin JTAG mode. The LFAST-based debug interface shares its five pins with the standard JTAG pins.

When reduced pin JTAG mode (IEEE 1149.7) is configured TMSC pin is configured in bidirectional mode during operation.

As the LFAST-based debug interface shares its five pins with JTAG pins, the DCI provides a safe switching logic to switch to LFAST-based debug mode.

The DCI also provides a centralized break control for system IPs and cores, which can be controlled by an external tool as well as the internal debug peripherals such as GTM/SPU. The DCI also provides the Event Out ($\overline{\text{EVTO}}$) signals to the debug tool in case of the occurrence of any internal debug event.

Figure 1944 shows the DCI block diagram.

Figure 1944. DCI block diagram

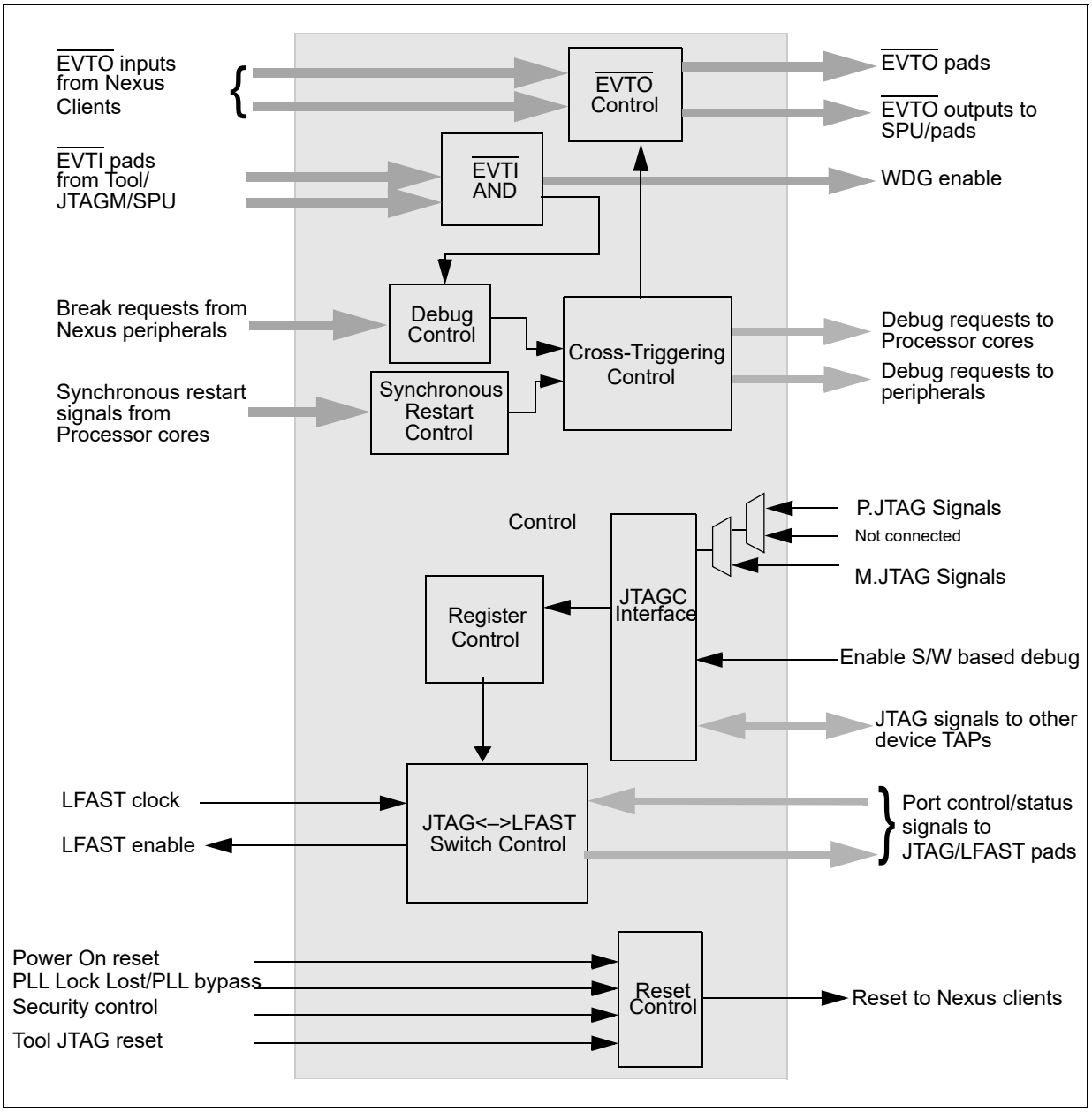
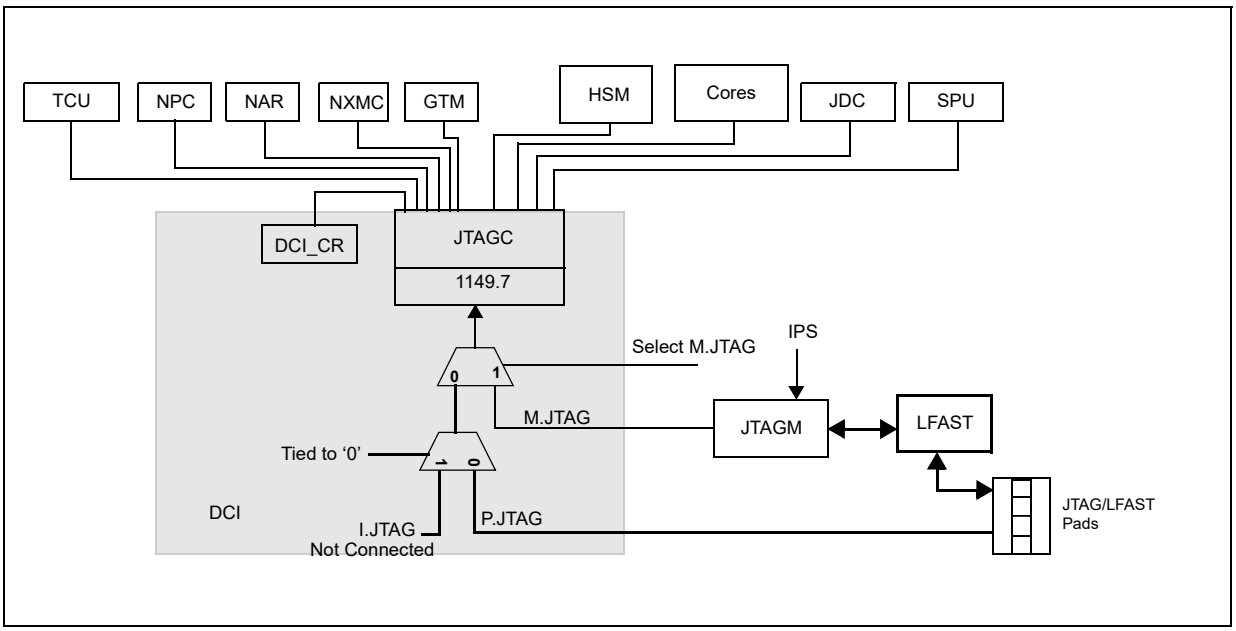


Figure 1945 shows the DCI configuration for the standard production package.

Figure 1945. DCI in a standard configuration



72.1.3 Operating modes

Selection between various JTAG sources is done using a JTAG register bit, DCI_CR[EN_LFAST], and a JTAGM register bit, JTAGM_MCR[DTM]. The JTAG bit, DCI_CR[EN_LFAST], can only be programmed using JTAG instructions and is used to select standard JTAG or LFAST debug tool. The JTAGM bit, JTAGM_MCR[DTM], is programmable via memory-mapped registers. JTAGM is selected as a JTAG source when either DCI_CR[EN_LFAST] or JTAGM_MCR[DTM] is set.

Table 1878 shows a summary of the configuration options.

Table 1878. DCI JTAG source selection

DCI_CR[EN_LFAST]	JTAGM_MCR[DTM]	DCI source	JCOMP
0	0	P.JTAG (Standard JTAG)	Asserted
1	x	M.JTAG (LFAST)	Asserted
x	1	M.JTAG (Software)	Negated

The reset signal to the DCI block consists of the following:

- Power-on reset, Low/High voltage detection on IO/Core supplies
- JCOMP reset

The DCI puts the JTAGC TAP in reset when either power-on/low-voltage-detect reset is asserted or JCOMP is negated.

The M.JTAG provides an alternate interface to the debug logic.

When enabled, it allows high speed debug through the LFAST and JTAGM modules. The high speed link uses LFAST commands that are translated to JTAG commands in the

JTAGM module. The selection between LFAST based debug or software based debug is managed in the JTAGM. However, it is important to note that the dynamic switching between tool-based debug and software debug is not allowed.Software based debug is done only in those cases when an external tool is not present.

An LFAST based tool starts in JTAG mode and then switches to LFAST mode. This switching is managed through a control set of operations which is described in *Section 72.1.3.1: Port sharing between JTAG and LFAST modes*. During the switch from JTAG to LFAST and back, the DCI uses a latched value of JCOMP to prevent all debug configuration reset.

The LFAST uses LVDS ports for LFAST communication and the JTAG ports uses CMOS level GPIOs. These signals are double-bonded to share the same pins at the tool interface. *Table 1879* shows the port mapping.

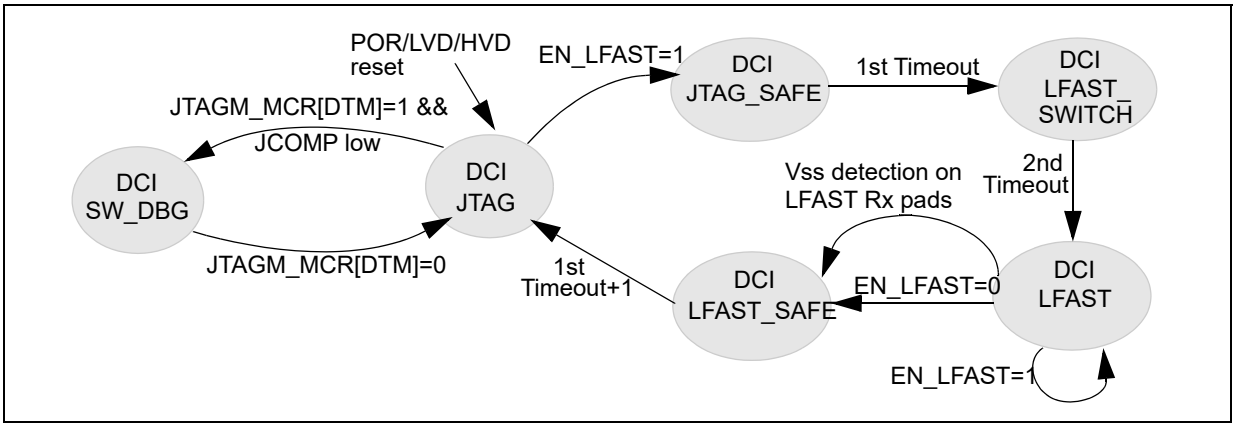
When no debug tool is connected, device power-on reset puts the DCI in standard 5-pin JTAG mode. This configures the TDO pin in output mode with the reset value (1'b0). There is no change in port values if no debug tool is connected.

Table 1879. DCI LFAST/JTAG port sharing

JTAG pad	Type in JTAG mode	LFAST pad	Type in LFAST mode
TDI	Input	LVDS TxN	Output
TMS	Input	LVDS TxP	Output
TDO	Output	LVDS RxP	Input
JCOMP	Input	LVDS RxN	Input
TCK	Input	DRCLK	Output

Figure 1946 shows the DCI mode switching operation.

Figure 1946. DCI operating modes state diagram



Note: JTAGM_MCR[jtagm_JCOMP] should not be asserted in LFAST mode to avoid any undesired behavior.

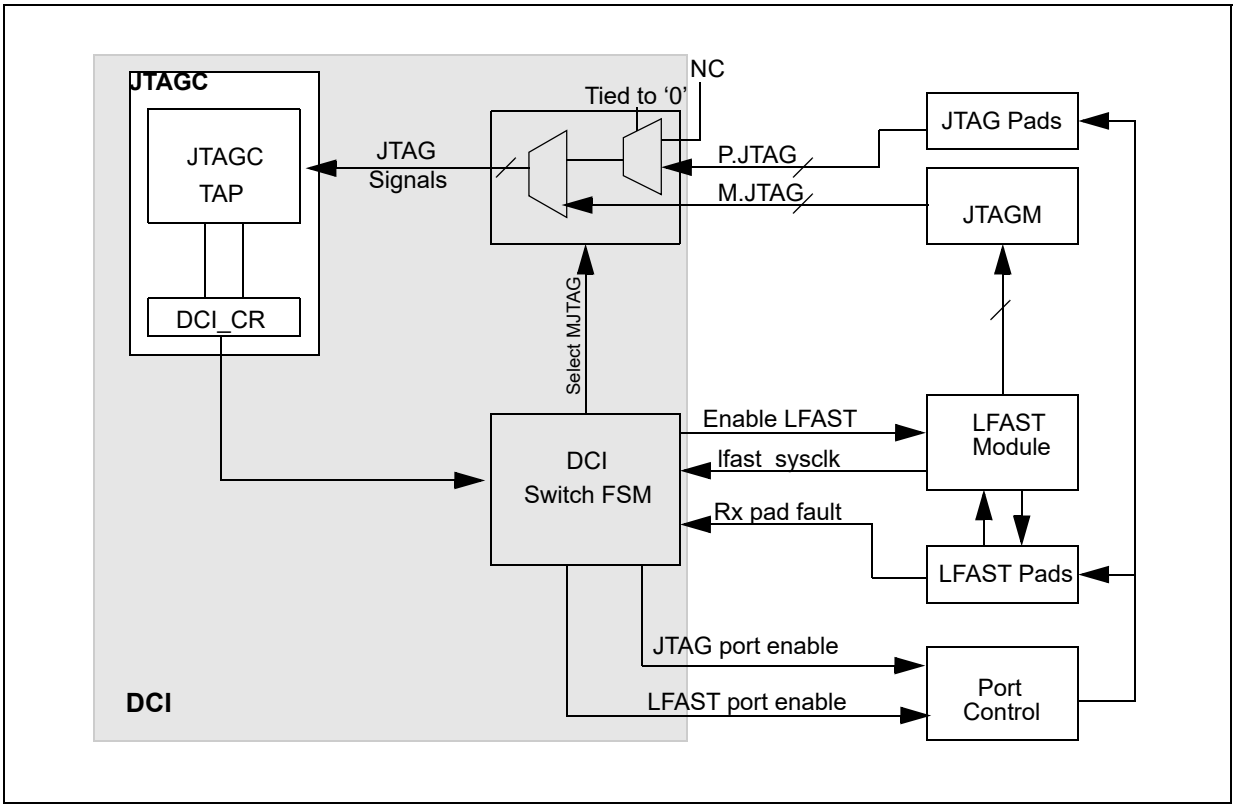
Table 1880 contains the operating mode descriptions.

Table 1880. DCI operating modes

Mode	Description
DCI_JTAG	JTAG mode. This is the default state after power-on reset. In this mode the DCI is in JTAG mode (controlled with P.JTAG) and is controlled by an external JTAG tool.
DCI_SW_DBG	Software Debug State. This state is achieved when no tool is present and software sets the JTAGM_MCR[DTM] bit. The DCI is then controlled using M.JTAG signals.
DCI_JTAG_SAFE	SAFE_JTAG state. When an external tool sets the DCI_CR[EN_LFAST] bit, this indicates that the external tool is going to switch to the LFAST mode. The external tool puts the DCI in Run-Test-Idle state to start the switching operation. To enable safe switching operation, the DCI enters the JTAG_SAFE state where it performs the following operations: – JCOMP is latched inside to control until switching gets completed – The DCI releases control of the JTAG ports and uses high on TMS inside, thus keeping the DCI in the Run-Test-Idle state – An internal counter starts to count to a tool specified value as programmed in DCI_CR[SWITCH_CNTR1].
DCI_LFAST_SWITCH	LFAST port switching state. In this state, the DCI asserts the enable signal to indicate to the LFAST that it is now ready for LFAST port switching. In this mode, the LFAST Rx pads are enabled. The switch counter starts another count operation to count to the DCI_CR[SWITCH_CNTR2] value.
DCI_LFAST	LFAST mode with escape mode enabled. This mode indicates that the switching is now completed and the debug tool is operating in the LFAST mode. However, there is another feature called Escape Mode, which enables the LFAST pads to detect any disconnection fault on its Rx pads. When these pads are working in Escape Mode, a sensing logic is enabled in the pad which detects and flags any out of common mode voltage range for the Rx pads. If this flag is asserted, it is used to force the DCI to go to LFAST_SAFE mode to ensure safe switching to the JTAG state.
DCI_LFAST_SAFE	LFAST_SAFE mode. While the tool is switching from LFAST mode to JTAG mode, it resets the DCI_CR[EN_LFAST] bit. The external tool puts the DCI in the Run-Test-Idle state to start the switching operation. To account for the switching time for the pads, the DCI enters in a safe LFAST state. It then restarts a counter to count to a tool specified value in DCI_CR[SWITCH_CNTR1], and disables both the LFAST function on the pads and the LFAST module.

Figure 1947 shows a block diagram of the JTAG/LFAST connections.

Figure 1947. DCI JTAG/LFAST switch integration



72.1.3.1 Port sharing between JTAG and LFAST modes

By default the DCI is configured in JTAG mode using the standard JTAG port (P.JTAG). Therefore, the initial configuration is done using low speed JTAG signals. In order to switch from JTAG mode to LFAST mode, the tool accesses the following fields in the DCI control register (DCI_CR):

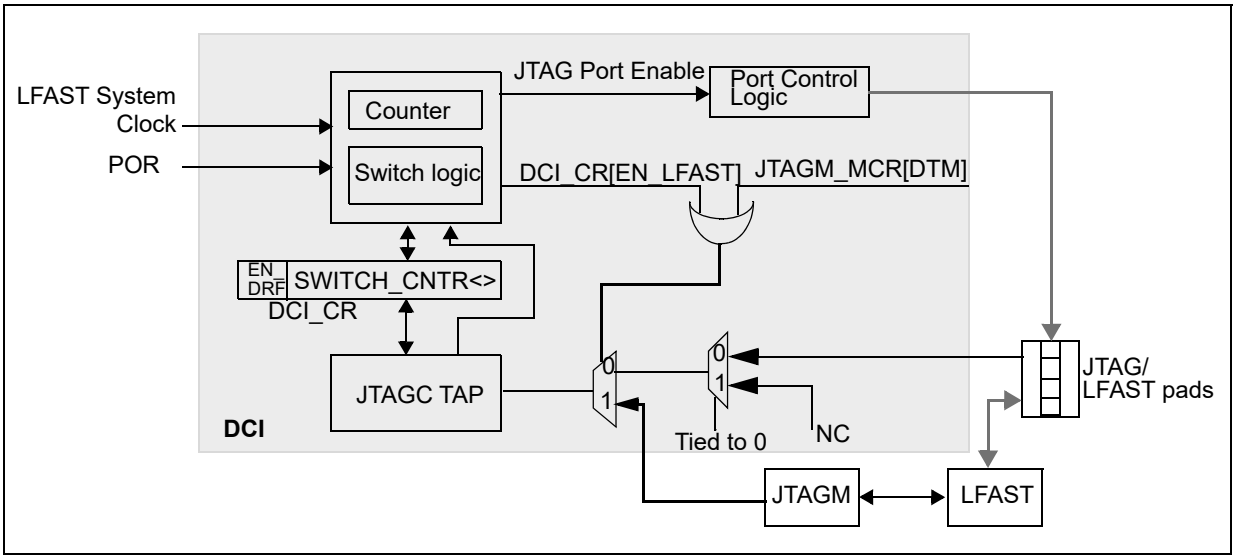
- EN_LFAST – Bit to indicate enable/disable LFAST mode
- SWITCH_CNTR1 and SWITCH_CNTR2 – Bit fields to program a count value that corresponds to the delay for switching the pin functions when changing modes.

As the JTAG to LFAST mode switching reflects change in port input/output behavior, it is done through a controlled procedure.The DCI_CR is only accessible through JTAG sequences. While switching from JTAG to LFAST mode, the tool programs this register via P.JTAG to enable switching to LFAST mode. As the pins are shared between JTAG and LFAST functions, a programmable delay based on switch counters is required which allows a safe switching for the MCU as well as for the tool to change its mode. In LFAST mode, the tool can write this register to switch back from high speed debug to low speed debug. Again, the similar switching delay mechanism is used. The switching schematic is shown in *Figure 1948*.

A programmable switching delay and tri-stating port pins during switching allow a connected tool to safely tune the device turn-around of pin functions from JTAG mode to LFAST mode and vice versa. These delays are programmed according to the time required by the tool to change its operation from JTAG mode to LFAST and vice versa. This reduces the possibility of device damage during switching.

The programmable counter uses the LFAST system clock as the clock source.

Figure 1948. DCI switching between JTAG and LFAST modes



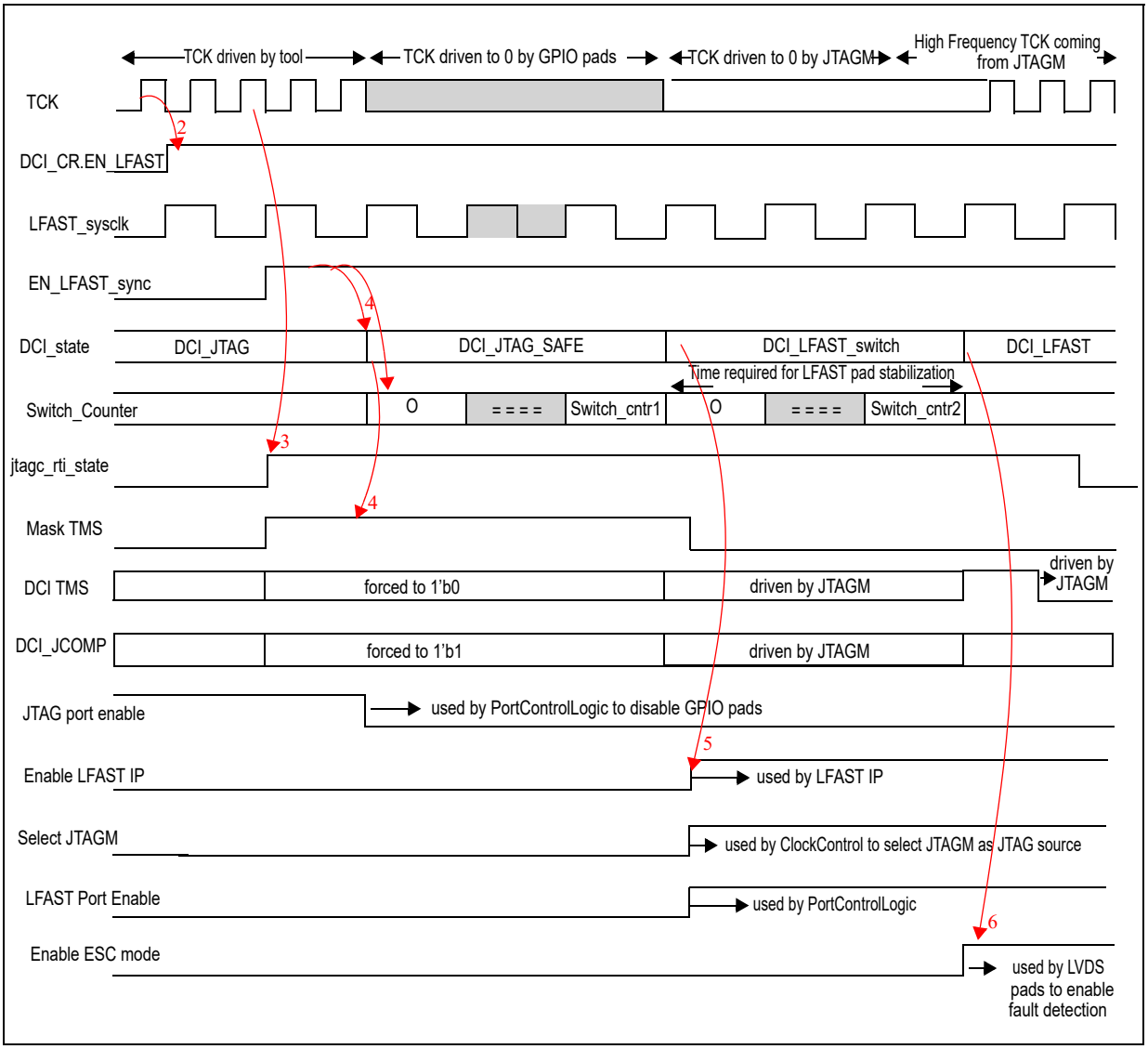
72.1.3.1.1 Switching from JTAG mode to LFAST mode

The steps to switch operation from JTAG to LFAST mode are as follows:

1. System starts in port JTAG mode after power-on-reset.
2. External tool programs the following bits in the DCI_CR:
   - EN_LFAST
   - SWITCH_CNTR1
   - SWITCH_CNTR2
3. Tool puts JTAG TAP in the Run-Test-Idle mode.
4. An internal finite state machine (FSM) starts the switching operation using the LFAST reference clock. The switching process takes four LFAST reference clocks to start, and an external tool should not change the JTAG signals during this period. The DCI forces TMS to 0 and JCOMP to 1 internally so as not to change JTAG TAP status during switching. The DCI also triggers a switching counter to allow for some time for the tool to change its mode from JTAG to LFAST.
5. After the counter counts to the DCI_CR[SWITCH_CNTR1] value, the LFAST IP and LFAST pads are enabled. Then the counter is reset. Also, internally JTAGM is selected as the JTAG source for the DCI.
6. Counter counts to the DCI_CR[SWITCH_CNTR2] value to enable the Escape mode feature for the LFAST Rx pads. This time allows the tool to settle down in LFAST mode.
7. Tool starts LFAST transmission when it detects pulses on the DRCLK pin. Tool now can use the LFAST to program the JTAGM to generate JTAG sequences.

Figure 1949 shows the timing sequence.

Figure 1949. DCI JTAG to LFAST switching diagram



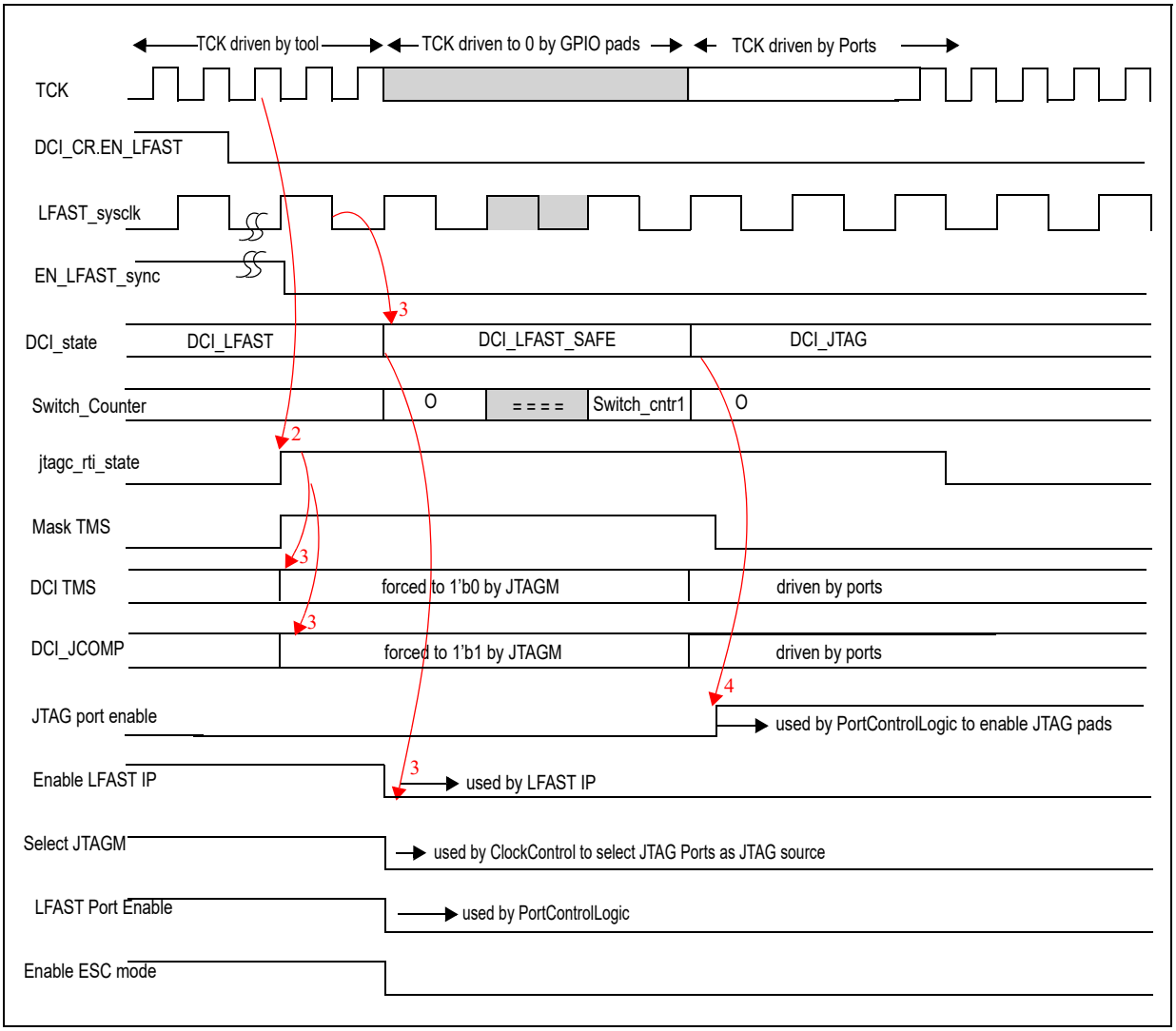
72.1.3.1.2 Switching from LFAST mode to JTAG mode

The steps to switch operation from LFAST to JTAG mode are as follows:

1. External tool programs the DCI_CR to reset the EN_LFAST bit and SWITCH_CNTR1 for the switching delay.
2. Tool puts JTAG TAP in the Run-Test-Idle mode.
3. An internal FSM starts switching operation. It forces TMS to 0 and JCOMP to 1 internally so as not to change JTAG TAP status during switching. It also triggers a switching counter to count to DCI_CR[SWITCH_CNTR1] to allow for the tool to change its mode from LFAST to JTAG.
4. After the counter overflows, the JTAG pads are enabled and JTAGM releases its control for the DCI JTAG source.

Table 1884 and *Table 1886* summarize the MCU and tool behaviors. *Figure 1950* shows the timing sequence.

Figure 1950. DCI LFAST to JTAG switching diagram



72.1.3.2 VSS detection

When the LFAST pads are operating in Escape mode, a sensing logic is enabled in the pad which detects and flags any disconnection of the LFAST Rx pads. This flag, if asserted, is used to force the DCI to go to JTAG mode to ensure safe operation. This flag, when asserted, also resets the JTAG TAP to restart in JTAG mode and the EN_LFAST bit is reset accordingly. This flag resets only JTAG TAPs and does not affect any Nexus blocks.

This feature is enabled only when DCI_CR[EN_ESC_MODE] is set. By default this feature is not enabled. To tune the tool behavior it is possible to emulate the pad fault using the DCI_CR[FLVDS_PAD_FLT] bit.

72.1.3.3 Software enabled debug and calibration mode

Apart from the JTAG sources, such as a JTAG-based tool or LFAST-based tool, it is possible to generate JTAG sequences through software programming using JTAGM IP. The JTAGM, which generates JTAG sequences using LFAST protocol, is also used to generate JTAG sequences using software programming.

To support a software debug session via a CAN or SPI link, the software can enable debug and calibration mode via an memory-mapped register. It is done first by setting a control bit in the JTAGM (JTAGM_MCR[DTM]). For more details, refer to the JTAGM documentation. When enabled, the DCI is configured to receive M.JTAG signals which are driven by JTAGM registers accessible through software. Software can control all DCI features like any other JTAG source.

After a system reset, the JTAGM_MCR[DTM] bit is reset and the JCOMP pad is pulled-low, thus keeping the DCI in the reset state. Therefore, software is required to first change to a non-secured mode. Then it can program the JTAGM_MCR[DTM] bit, to select M.JTAG as the source inside the DCI, and then set the JTAGM_MCR[jtagm_JCOMP] bit present in the JTAGM, which takes all debug modules out of reset.

When MCR[jtagm_JCOMP] is programmed to 0, the JTAGC block reset is asserted, thus putting the debug and calibration logic in reset state. This bit is used during software based debug to initialize the debug and calibration logic.

Software based debug is not possible when a tool is connected. The presence of the tool is determined using the rising edge of the JCOMP pad to set an internal flag which, when set, disables the selection of JTAGM as a JTAG source.

Note: It is recommended to utilize either hardware based debug (JTAG or LFAST) or software based debug. If only one of the debug method is used, all three connection methods (connect and reset target, connect and break application, connect and keep application running) are supported.

If both debug methods are used simultaneously, then an external tool may arbitrarily connect when the SoC has already been halted using the software debug mode. In this case, two consecutive assertions of external JCOMP are necessary to allow the external tool to take control of debug session.

72.1.3.4 Nexus reset control

The JCOMP input that is used as the primary reset signal for the DCI is also used by the DCI to generate a single-bit reset signal for other Nexus blocks. It is used to reset all debug functions without affecting any system level functions. The DCI provides the reset signal for all Nexus blocks. This reset is controlled using the following signals:

- Power-on reset/Low voltage detect
- JCOMP
- SoC Debug_Enable signal
- System PLL is out of lock state and not bypassed

Asserting power-on reset, negating JCOMP, or negating the Debug_Enable signal, results in asynchronous entry into reset state. Following negation of power-on reset, the Nexus reset signal remains asserted until the system clock achieves lock or in bypass mode, as indicated by the system clock lock status signal, and Debug_Enable is asserted and JCOMP is asserted. Any subsequent loss of PLL lock does not generate a Nexus reset. This reset signal is unaffected by other sources of reset.If Debug_Enable is de-asserted by the SoC security modules, then the DCI holds the debug and trace logic in reset. The Debug_Enable signal is generated by security logic after functional or destructive reset based on the device censored mode, the PASS_LOCK3[DBL] control, and the device life cycle. If the Debug_Enable signal is asserted by the security logic, it is possible to trace through functional resets

The following table shows how the Nexus reset is asserted/negated based on various signals.

Table 1881. Nexus reset truth table

POR	JCOMP	Debug_Enable	PLL lock	Nexus reset
Assert	X	X	X	Assert
X	Negate	X	X	Assert
X	X	Disable	X	Assert
X	X	X	Unlocked	Assert
Negate	Assert	Enable	Locked	Negate

72.1.3.5 System watchdog control

he DCI is also used to enable or disable the Software Watchdog Timer (SWT). The DCI provides a single bit SWT enable signal which is used to control all SWTs in the MCU. For this, the $\overline{\text{EVTI}}$[0] pin is used. If the $\overline{\text{EVTI}}$[0] pin remains high when the tool asserts JCOMP, then the SWT is enabled. But if the tool drives the $\overline{\text{EVTI}}$[0] pin low before asserting JCOMP pin, then the SWT is disabled. By default the SWT is enabled and a tool has an option to enable/disable the SWT using the $\overline{\text{EVTI}}$[0] pin.

Note: Debugger should check the SWT counter to confirm if SWT has actually stopped or not, otherwise repeat the sequence mentioned above to disable the SWT.

72.1.3.6 Break control

The DCI provides central break control among the CPUs and non-CPU debug modules such as GTM, DMA, Watchdog and other peripherals.

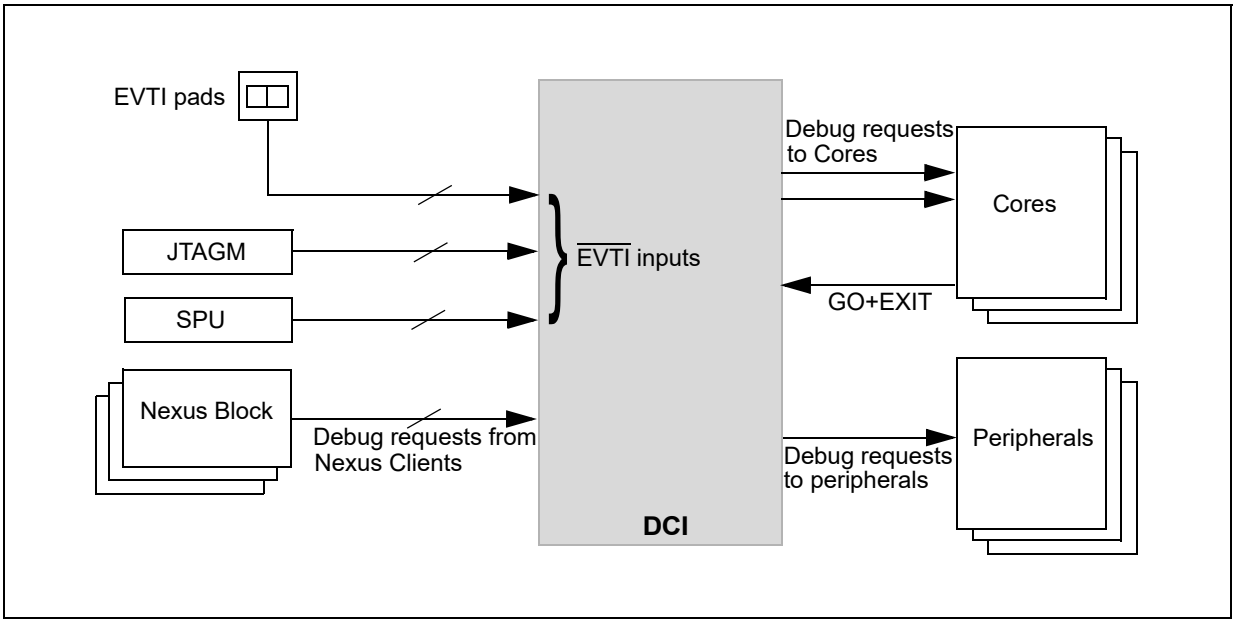
There are various different input sources that provide debug requests:

- $\overline{\text{EVTI}}$[1:0] signals from EVTI pads
- $\overline{\text{EVTI}}$[1:0] signals from JTAGM
- $\overline{\text{EVTI}}$[1:0] signals from the Sequence Processing Unit (SPU)

The DCI generates following debug request signals:

- Debug Request signal to be used for all non-CPU peripherals
- External Debug Request to be used to request external debug event to processor cores
- OnCE Debug Request to be used to request OnCE debug event to processor cores
- All break control requests are issued simultaneously to CPUs and peripherals, then synchronized to each clock domain

Figure 1951. DCI debug control mechanism



The various sources of $\overline{\text{EVTI}}$ inputs are synchronized and logically ANDed in the DCI to generate the following:

- debug requests for each processor core
- external debug requests for each processor core
- debug request for all non-CPU peripherals

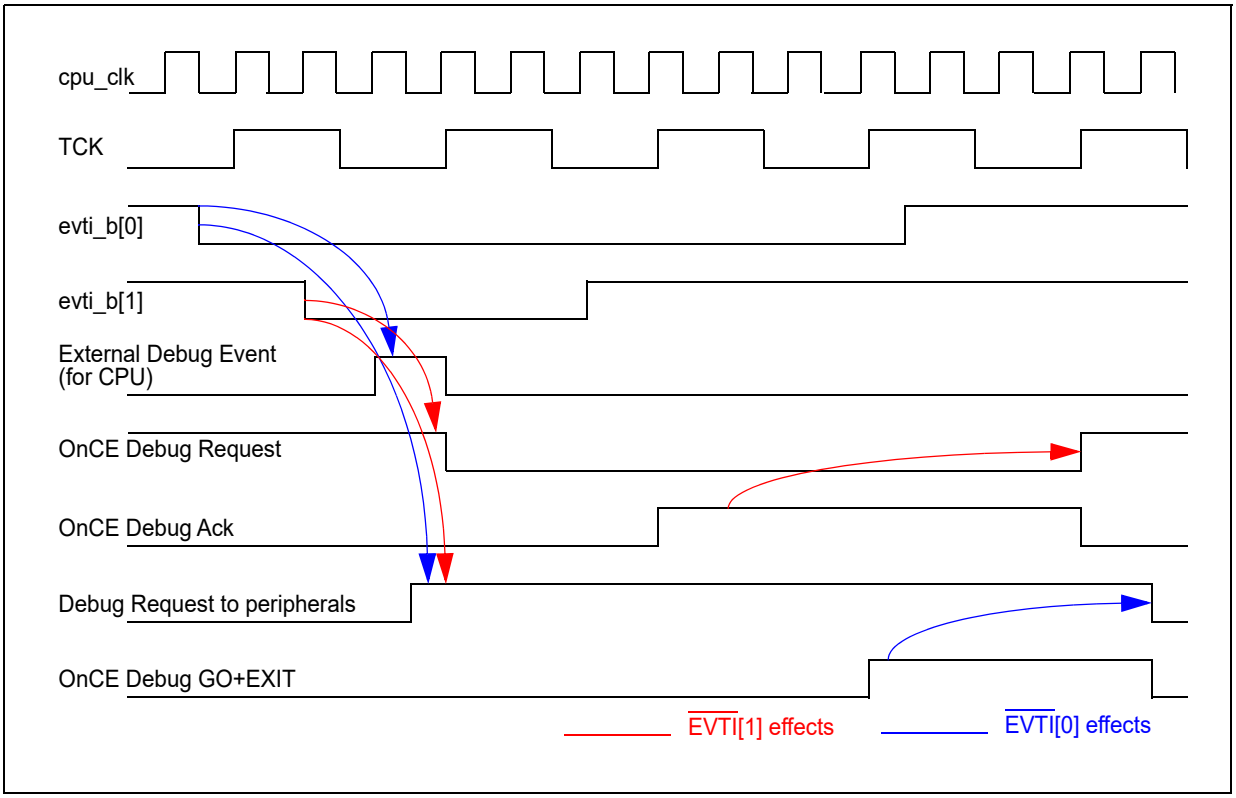
The $\overline{\text{EVTI}}$[0] from the EVTI pads, SPU and JTAGM are used to generate debug requests for non-CPU peripherals and for processor cores through the external debug request signal. The $\overline{\text{EVTI}}$[1] from the SPU and JTAGM, is also used to generate debug requests for non-CPU peripherals and it generates debug requests to processor cores through the OnCE debug request signal.

The external debug request to processor core is asserted as a single cycle pulse and is synchronized on the respective core clock. The OnCE debug request is negated when the acknowledge signal from the Cores is asserted.The CPU(s) remain in the debug state until GO+EX command is executed for that CPU. Exiting a CPU out of debug state by executing GO+EX command does not take other CPUs out of debug state. The peripheral debug request is negated when the processor core(s) execute GO+EXIT command.The $\overline{\text{EVTI}}$[1:0] signals coming from the pads are enabled for debug control using DCI_CR[5:4] bits.

In order to selectively enable/disable debug requests to system peripherals, the respective debug enable bit inside the peripheral should be used.

Figure 1952 shows the timing sequence.

Figure 1952. DCI debug signal generation



72.1.3.7 Cross-triggering control

Many of the blocks in the device, including the Nexus trace blocks and more sophisticated peripherals, have debug features that allow them to request device-wide breaks to enter debug mode. Many blocks can be programmed to halt operation upon debug mode entry or ignore it and continue normal operation. All debug mode entry requests are input to the DCI, ORed together and the result is used to produce the debug requests to the CPUs and non-CPU peripherals.

The cross-triggering feature is implemented through selectively enabling debug control bits in the respective modules. Processor cores (enabled using DBCR0[DEVT1] bit) and non-core peripherals can be programmed to halt operation through their respective control bits. Cross-triggering provides a selective feature to put modules in debug mode upon receiving halt request signals from Nexus modules.

72.1.3.8 Synchronous restart control

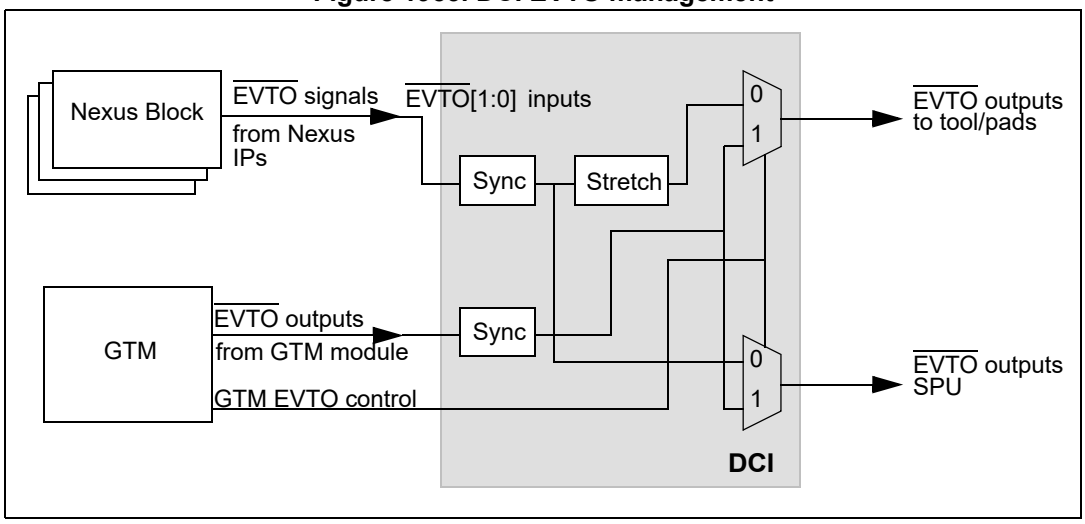
The DCI implements a synchronous restart control feature to synchronously restart all debug peripherals (CPU and non-CPU peripherals which are put in debug state). This is accomplished with the following steps:

1. The tool sequentially scans into CPUSCR for each processor core to restore the proper processor state.
2. The tool scans into the Production device tap controller the AUX_PARALLEL instruction, which then selects all processor core tap controllers in parallel.
3. To synchronously exit any combination of processor cores from debug mode, the tool simultaneously scans into all processor core tap instruction registers (OCMD) the 10-bit sequence 0x180 (that is set GO and EX, clear RS[0:6]).
4. If a processor core is out of debug mode, it sets a flag OnCE GO+EXIT.
5. When the OR of the processor core OnCE GO+EXIT signals is asserted, then all other selected non-processor cores and peripherals in debug would also be synchronously exited from debug mode. It is done by negating peripheral debug request signal.

72.1.3.9 EVTO management

The EVTO management block diagram is presented in *Figure 1953*.

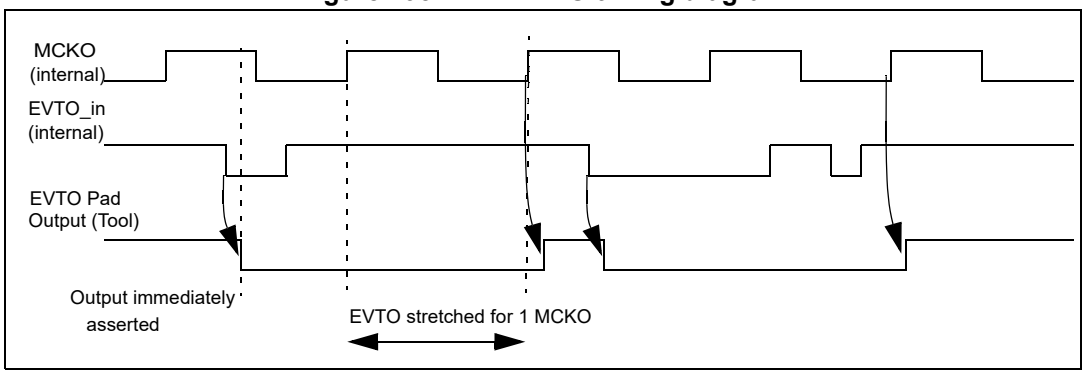
Figure 1953. DCI EVTO management



The DCI module controls distribution of the EVTO outputs from all Nexus Clients including the cores that produce EVTO[1:0] signals. The DCI uses a high frequency clock to sample EVTO signals coming from all modules which might generate EVTO outputs at their own clock frequency. After receiving a single clock period of asserted EVTO[1:0] signals from any Nexus client, the DCI immediately asserts EVTO[1:0] outputs and stretches it for a minimum of one output clock period for the output to the tool (*Figure 1954*). The DCI also generates EVTO outputs in other device clock domains so that other Nexus clients can receive EVTO inputs without the need of any synchronization. EVTO function is active as long as the DCI is not in reset.

The DCI also allows the GTM to take control of the EVTO outputs by controlling the EOS0/EOS1 fields of the GTMDI_DC register. If these fields are programmed to HALT/TIM/TOM/ATOMEVTO value, the GTM EVTO outputs are directly available on the ports bypassing the stretch logic.

Figure 1954. DCI EVTO timing diagram



72.1.3.10 Reset control

The DCI includes control bits to allow a connected tool to force a reset of the connected device. The description is as follows:

- Force functional reset—This reset is generated by programming the DCI_CR[FPD_FUNC] bit. When set, it forces functional reset. This bit has to be reset by the tool to negate the reset signal.
- Force POR or destructive reset—This reset is generated by programming DCI_CR[FPD_DEST] bit. When set it forces destructive reset. This bit has to be reset by the tool to negate the reset signal.

These reset signals are directly connected to the respective DCI_CR bits. The DCI does not reset the DCI_CR bits after generation of the respective reset signals. The tool must specifically reset the appropriate DCI_CR bits to remove the reset signals. These reset control bits can be set/reset through JTAG programming, and only reset through JCOMP assertion or Power-on-reset.

72.1.3.11 Security

The SoC security logic provides a Debug_Enable signal to the DCI. The DCI uses the Debug_Enable signal as a control input to generate debug module reset. When debug disable is requested, the DCI keeps all debug blocks in the reset state. This is a protection mode used to disable debug accesses to the peripheral when the device is in a secured state.

72.2 External signal description

The JTAGC module used inside DCI module defines the interface with external signals. It consists of five signals (TMS/TDI/TDO/TCK/JCOMP) that connect to off chip development tools and allow access to test support functions. For more details for JTAG signals, refer to JTAGC documentation.

72.3 Register description

The DCI module contains two registers, DCI_CR and DCI_PINCR, which are present inside the JTAGC module. These registers are only controlled using the JTAGC TAP and no memory-mapped registers are present.

72.3.1 DCI control register (DCI_CR)

The DCI_CR is a 32-bit data register. This register is programmed using JTAG shift with the corresponding enable instruction described in the JTAGC chapter. *Figure 1955* shows the format of the DCI_CR.

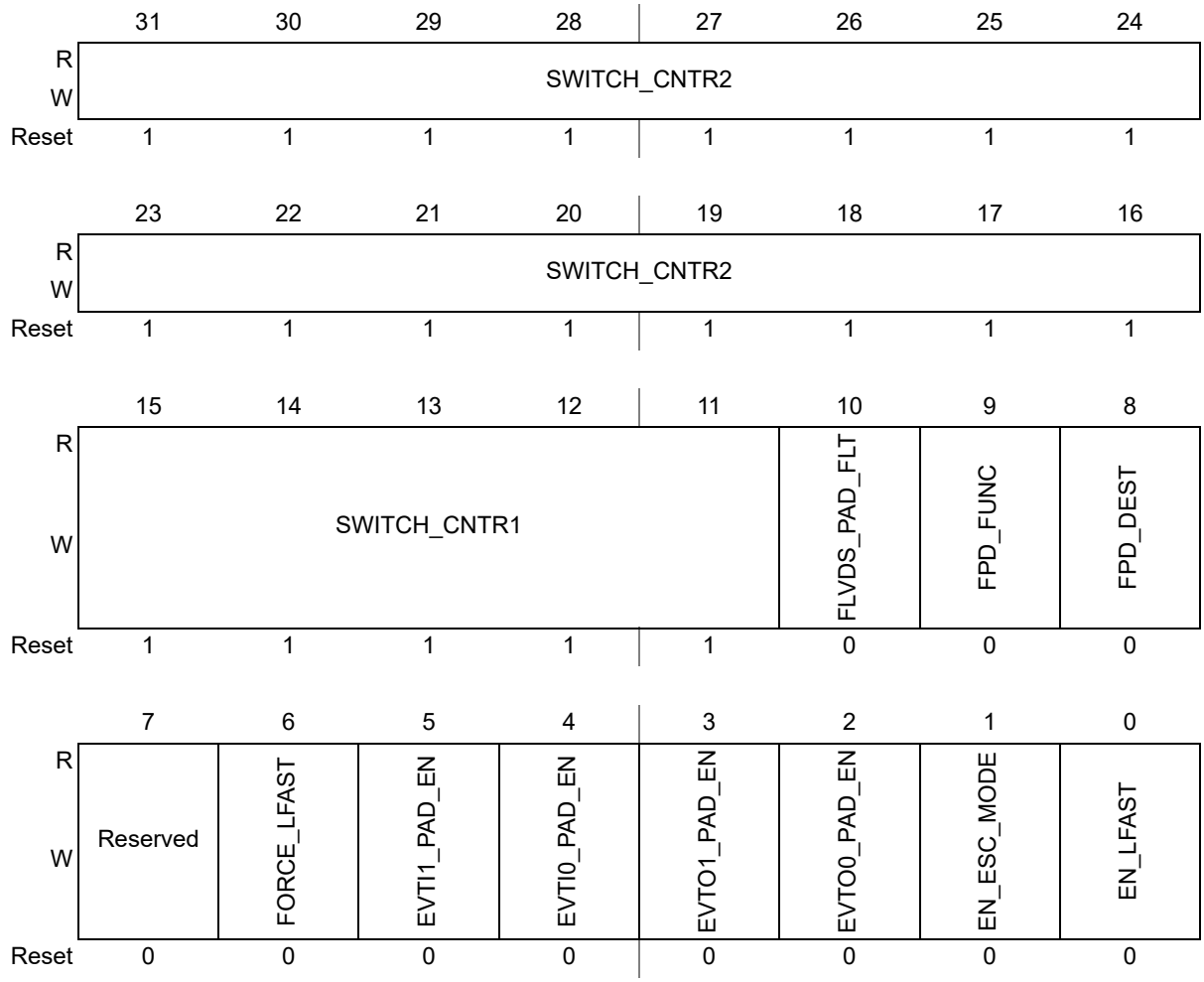


Figure 1955. DCI control register (DCI_CR)

The DCI_CR is described in *Table 1882*.

Table 1882. DCI_CR field descriptions

Field	Description
31:16 SWITCH_CNTR2	Width of Switch_CNTR2 Decides the second timeout value while switching from JTAG to LFAST mode. The maximum timeout duration = 32,768×T_{lfast_sysclk}
15:11 SWITCH_CNTR1	Width of Switch_CNTR1 Decides the first timeout value while switching from JTAG to LFAST mode, and the timeout value while switching from LFAST mode to JTAG mode. The maximum timeout duration = 32×T_{lfast_sysclk}

Table 1882. DCI_CR field descriptions (continued)

Field	Description
10 FLVDS_PAD_FLT	Force LVDS Pad Fault Forces the pad fault condition to tune the tool without actually having the pad fault on LFAST Rx pads. This bit is effective only when DCI_CR[EN_ESC_MODE] is set. 0 Do not force Rx LVDS Pad Fault 1 Force Rx LVDS Pad Fault if DCI_CR[EN_ESC_MODE] is set
9 FPD_FUNC	Force Functional reset 0 Do not force functional reset 1 Force functional reset
8 FPD_DEST	Force Destructive reset 0 Do not force destructive reset 1 Force destructive reset
6 FORCE_LFAST	Force enable for LFAST 0 LFAST not forced to enable and is controlled using DCI_CR[EN_LFAST] bit 1 LFAST forcefully enabled (without using DCI_CR[EN_LFAST] bit)
5 EVTI1_PAD_EN	$\overline{\text{EVTI}}$[1] feature enable 0 $\overline{\text{EVTI}}$[1] disabled (default) 1 $\overline{\text{EVTI}}$[1] enabled
4 EVTI0_PAD_EN(1)	$\overline{\text{EVTI}}$[0] feature enable 0 $\overline{\text{EVTI}}$[0] disabled (default) 1 $\overline{\text{EVTI}}$[0] enabled
3 EVTO1_PAD_EN	Output enable for $\overline{\text{EVTO}}$[1] pad 0 $\overline{\text{EVTO}}$[1] pad output not enabled (default) 1 $\overline{\text{EVTO}}$[1] pad output enabled
2 EVTO0_PAD_EN	Output enable for $\overline{\text{EVTO}}$[0] pad 0 $\overline{\text{EVTO}}$[0] pad output not enabled (default) 1 $\overline{\text{EVTO}}$[0] pad output enabled
1 EN_ESC_MODE	Enable Escape Mode Fault detect feature enable for LFAST Rx pads. 0 Fault detect feature not present for LFAST Rx pads (default) 1 Fault detect feature present for LFAST Rx pads and are enabled after the successful switching to LFAST mode
0 EN_LFAST(1)	Enable for LFAST-based tool selection This bit can be reset either by programming by tool, or pad fault condition in which JTAGC block is reset. 0 LFAST tool not enabled. DCI operates in JTAG mode (default) 1 LFAST tool enabled

1. DCI_CR is a JTAG data register and is not accessible for software read. However, the DCI_CR[EN_LFAST] bit is mapped to JTAGM register bit JTAGM_SR[1] to allow for software read access and a mechanism for software to be aware of an active debug tool interface.

72.3.2 DCI EVTx pin multiplexing control register (DCI_PINCR)

The DCI_PINCR is a 32-bit data register. This register is programmed using JTAG shift with the corresponding enable instruction described in the JTAGC chapter, ENABLE_DATA2. For

a description of the DCI_PINCR functionality, refer to Calibration and Debug configuration chapter (or the Debug and Trace chapter) which describes how the modules are configured.

72.4 Functional description

72.4.1 DCI mode transition

After power-on reset, the DCI starts in JTAG mode. To enable LFAST mode, the external tool has to first operate in JTAG mode, configure DCI and then switch to LFAST mode. This requires certain careful steps so as to avoid any contention on double-bonded pads for JTAG/LFAST. *Table 1883* and *Table 1884* show the steps and expected states for the MCU and tool for JTAG to LFAST switching. *Table 1885* and *Table 1886* contain the different steps, and the MCU and tool expected states for the LFAST to JTAG switching.

Table 1883. MCU operation (JTAG to LFAST switching)

MCU pin	JCOMP low	JCOMP high (JTAG mode)	DCI_CR [EN_LFAST]=1	Run-Test-Idle state	1st Timeout	2nd Timeout	LFAST ICLC enable command received
TDI	Logic Input	Logic Input	Logic Input	Hi-Z	Hi-Z	Hi-Z	LVDS TxN
TMS	Logic Input	Logic Input	Logic Input	Hi-Z	Hi-Z	Hi-Z	LVDS TxP
TDO	Hi-Z	Logic Output	Logic Output	Hi-Z	LVDS Rx	LVDS Rx	LVDS RxP
JCOMP	Logic Input	Logic Input	Logic Input Latch	Hi-Z	LVDS Rx	LVDS Rx	LVDS RxN
TCK	Logic Input	Logic Input	Logic Input	Hi-Z	Logic Output	Logic Output	Logic Output
Mode Escape	Inhibit	Inhibit	Inhibit	Inhibit	Inhibit	Enable	Enable
LFAST_ SYSCL K _EN	Inhibit	Inhibit	Inhibit	Inhibit	Enable	Enable	Enable

Table 1884. Tool operation (JTAG to LFAST switching)

Tool	JCOMP low	JCOMP high (JTAG mode)(1)	DCI_CR [EN_ LFAST]=1	Run-Test-Idle state	TCK detected	Internal timer elapse
TDI	Don't Care	Logic Output	Hi-Z	Hi-Z	LVDS Rx	LVDS TxN
TMS	Don't Care	Logic Output	Hi-Z	Hi-Z	LVDS Rx	LVDS TxP
TDO	Hi-Z	Logic Input	Hi-Z	Hi-Z	LVDS Tx	LVDS RxP
JCOMP	Hi-Z	Logic Output (High)	Hi-Z	Hi-Z	LVDS Tx	LVDS RxN
TCK	Don't Care	Logic Output	Logic Input	Logic Input	Logic Input	Logic Input

1. This state is used to set DCI_CR[EN_LFAST] bit.

Table 1885. MCU operation (LFAST to JTAG switching)

MCU pin	LFAST mode	DCI_CR[EN_LFAST]=0	First timeout
TDI	LVDS Tx	Logic Input	Logic Input
TMS	LVDS Tx	Logic Input	Logic Input
TDO	LVDS Rx	Hi-Z	Logic Output
JCOMP	LVDS Rx	Logic Input Latch	Logic Input (sample JCOMP pin)
TCK	Logic Output	Logic Input	Logic Input
Mode Escape	Enable	Inhibit	Inhibit
LFAST_SYSCLK_EN	Enable	Inhibit	Inhibit

Table 1886. Tool operation (LFAST to JTAG switching)

Tool	Send command to clear DCI_CR[EN_LFAST]	LFAST transmits idle after command sent	TCK input clock stops or LVDS Rx both low	Internal timeout: start sending JTAG messages
TDI	LVDS Rx	LVDS Rx	Logic Output	Logic Output
TMS	LVDS Rx	LVDS Rx	Logic Output	Logic Output
TDO	LVDS Tx	Hi-Z	Logic Input	Logic Input
JCOMP	LVDS Tx	Hi-Z	Logic Output (High)	Logic Output (High)
TCK	Logic Input	Logic Input	Logic Output	Logic Output

72.4.2 LFAST LVDS pad fault

It is possible that when the tool is operating in LFAST mode, the tool may get disconnected due to external disturbances, which could put the MCU in an undetermined state. It is imperative to keep the MCU in a safe state. Thus the LFAST Rx pads have a sensing logic which detects if the Rx pad voltages are not equal to the common mode voltage when the DCI is in LFAST mode. If it is found that the LFAST Rx pads are not corresponding to the common mode voltage, the pad logic sends a signal to the DCI which then puts the DCI into JTAG mode. *Table 1887* and *Table 1888* contain the different steps and expected states for the MCU and tool for LFAST to JTAGRST switching.

Table 1887. MCU operation (LFAST to JTAGRST switching)

MCU pin	LFAST mode	JCOMP detected low (mode escape)
TDI	LVDS TxN	Logic Input
TMS	LVDS TxP	Logic Input
TDO	LVDS RxP	Hi-Z
JCOMP	LVDS RxN	Logic Input
TCK	Logic Output	Logic Input
Mode Escape	Enable	Inhibit

Table 1887. MCU operation (LFAST to JTAGRST switching) (continued)

MCU pin	LFAST mode	JCOMP detected low (mode escape)
LFAST_SYSCLK_EN	Enable	Inhibit
Internal TRST	Negated	Asserted

Table 1888. Tool operation (LFAST to JTAGRST switching)

Tool	LFAST mode	Tool forces JCOMP low
TDI	LVDS TxN	Don’t Care
TMS	LVDS TxP	Don’t Care
TDO	LVDS RxP	Hi-Z
JCOMP	LVDS RxN	Logic Output (Low)
TCK	Logic Input	Don’t Care

73 JTAG Controller (JTAGC)

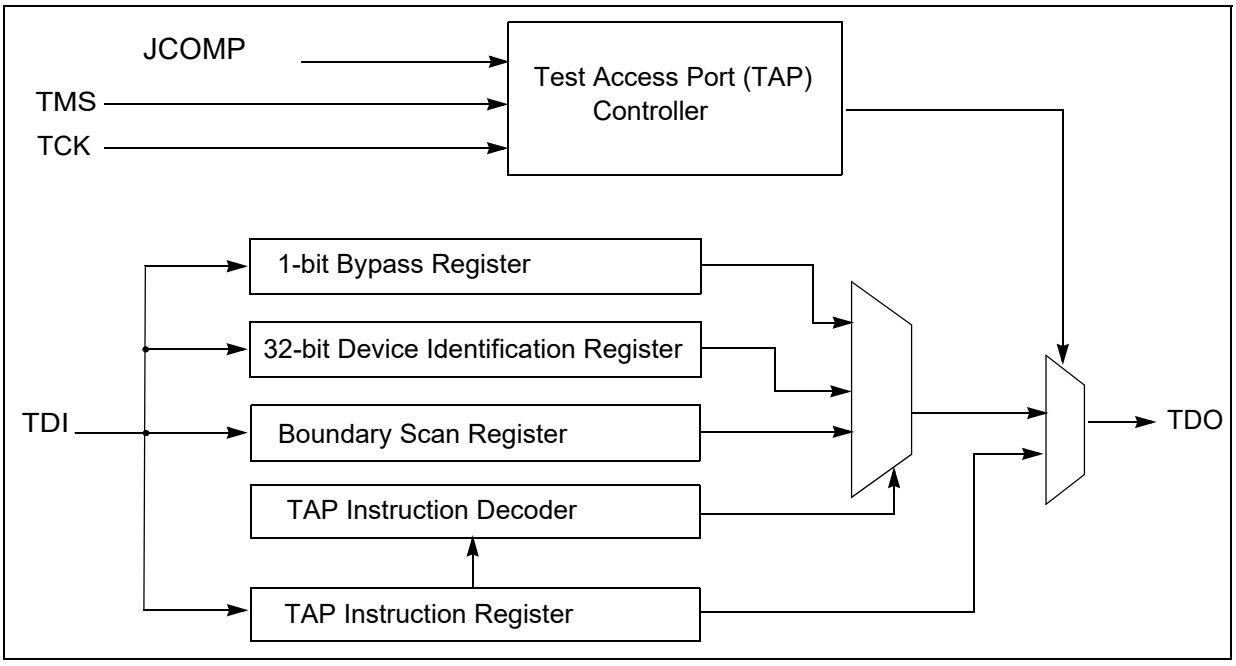
73.1 Introduction

The JTAGC block provides the means to test chip functionality and connectivity while remaining transparent to system logic when not in test mode. Testing is performed via a boundary scan technique, as defined in the IEEE 1149.1-2001 standard. All data input to and output from the JTAGC block is communicated in serial format.

73.1.1 Block Diagram

The following is a block diagram of the JTAG Controller (JTAGC) block.

Figure 1956. JTAG (IEEE 1149.1) block diagram



73.1.2 Features

The JTAGC block is compliant with the IEEE 1149.1-2001 standard, and supports the following features:

- IEEE 1149.1-2001 Test Access Port (TAP) interface
  - 4 pins (TDI, TMS, TCK, and TDO)
- JCOMP input that provides reset control and the ability to share the TAP.
- Instruction register that supports several IEEE 1149.1-2001 defined instructions as well as several public and private device-specific instructions. Refer to *Table 1892* for a list of supported instructions.
- Sharing of the TAP with other TAP controllers via ACCESS_AUX_x instructions
- JTAG_PASSWORD register
- Data registers, bypass register, boundary scan register, and device identification register.
- TAP controller state machine that controls the operation of the data registers, instruction register and associated circuitry.

73.1.3 Modes of operation

The JTAGC block uses JCOMP and a power-on reset indication as its primary reset signals. Several IEEE 1149.1-2001 defined test modes are supported, as well as a bypass mode.

73.1.3.1 Reset

The JTAGC block is placed in reset when either power-on reset is asserted, JCOMP is negated, or the TMS input is held high for enough consecutive rising edges of TCK to sequence the TAP controller state machine into the Test-Logic-Reset state. Holding TMS high for five consecutive rising edges of TCK guarantees entry into the Test-Logic-Reset state regardless of the current TAP controller state. Asserting power-on reset or setting JCOMP to a value other than the value required to enable the JTAGC block results in asynchronous entry into the reset state. While in reset, the following actions occur:

- The TAP controller is forced into the Test-Logic-Reset state, thereby disabling the test logic and allowing normal operation of the on-chip system logic to continue unhindered.
- The instruction register is loaded with the IDCODE instruction.

73.1.3.2 IEEE 1149.1-2001 defined test modes

The JTAGC block supports several IEEE 1149.1-2001 defined test modes. A test mode is selected by loading the appropriate instruction into the instruction register while the JTAGC is enabled. Supported test instructions include EXTEST, HIGHZ, CLAMP, SAMPLE and SAMPLE/PRELOAD. Each instruction defines the set of data register(s) that may operate and interact with the on-chip system logic while the instruction is current. Only one test data register path is enabled to shift data between TDI and TDO for each instruction.

The boundary scan register is enabled for serial access between TDI and TDO when the EXTEST, SAMPLE or SAMPLE/PRELOAD instructions are active. The single-bit bypass register shift stage is enabled for serial access between TDI and TDO when the BYPASS, HIGHZ, CLAMP or reserved instructions are active. The functionality of each test mode is explained in more detail in *Section 73.4.4*.

73.1.3.3 Bypass mode

When no test operation is required, the BYPASS instruction can be loaded to place the JTAGC block into bypass mode. While in bypass mode, the single-bit bypass shift register is used to provide a minimum-length serial path to shift data between TDI and TDO.

73.2 External signal description

The JTAGC consists of five signals that connect to off chip development tools and allow access to test support functions. The JTAGC signals are outlined in the following table and described in the following sections.

Table 1889. JTAG signal properties

Name	I/O	Function	Reset State	Pull
TCK	Input	Test Clock	—	Down
TDI	Input	Test Data In	—	Up
TDO	Output	Test Data Out	High Z(1)	—
TMS	Input	Test Mode Select	—	Up
JCOMP	Input	JTAG Compliancy	—	Down

1. TDO output buffer enable is negated when the JTAGC is not in the Shift-IR or Shift-DR states. A weak pull may be implemented at the TDO pad for use when JTAGC is inactive.

73.2.1 TCK—Test clock input

Test Clock Input (TCK) is an input pin used to synchronize the test logic and control register access through the TAP.

73.2.2 TDI—Test data input

Test Data Input (TDI) is an input pin that receives serial test instructions and data. TDI is sampled on the rising edge of TCK.

73.2.3 TDO—Test data output

Test Data Output (TDO) is an output pin that transmits serial output for test instructions and data. TDO is three-stateable and is actively driven only in the Shift-IR and Shift-DR states of the TAP controller state machine, which is described in *Section 73.4.3*.

73.2.4 TMS—Test mode select

Test Mode Select (TMS) is an input pin used to sequence the IEEE 1149.1-2001 test control state machine. TMS is sampled on the rising edge of TCK.

73.2.5 JCOMP—JTAG compliancy

The JCOMP signal provides IEEE 1149.1-2001 compatibility and provides the ability to share the TAP. The JTAGC TAP controller is enabled when JCOMP is set to the JTAGC enable encoding, otherwise the JTAGC TAP controller remains in reset.

73.3 Register description

This section provides a detailed description of the JTAGC block registers accessible through the TAP interface, including data registers and the instruction register. Individual bit-level descriptions and reset states of each register are included. These registers are not memory-mapped and can only be accessed through the TAP.

73.3.1 Instruction register

The JTAGC block uses a 6-bit instruction register as shown in the following figure. The instruction register allows instructions to be loaded into the block to select the test to be performed or the test data register to be accessed or both. Instructions are shifted in through TDI while the TAP controller is in the Shift-IR state, and latched on the falling edge of TCK in the Update-IR state. The latched instruction value can only be changed in the Update-IR and Test-Logic-Reset TAP controller states. Synchronous entry into the Test-Logic-Reset state results in the IDCODE instruction being loaded on the falling edge of TCK. Asynchronous entry into the Test-Logic-Reset state results in asynchronous loading of the IDCODE instruction. During the Capture-IR TAP controller state, the instruction shift register is loaded with the value 000001b, making this value the register's read value when the TAP controller is sequenced into the Shift-IR state.

	0	1	2	3	4	5
R	0	0	0	0	0	1
W	Instruction Code					
Reset:	0	0	0	0	0	1

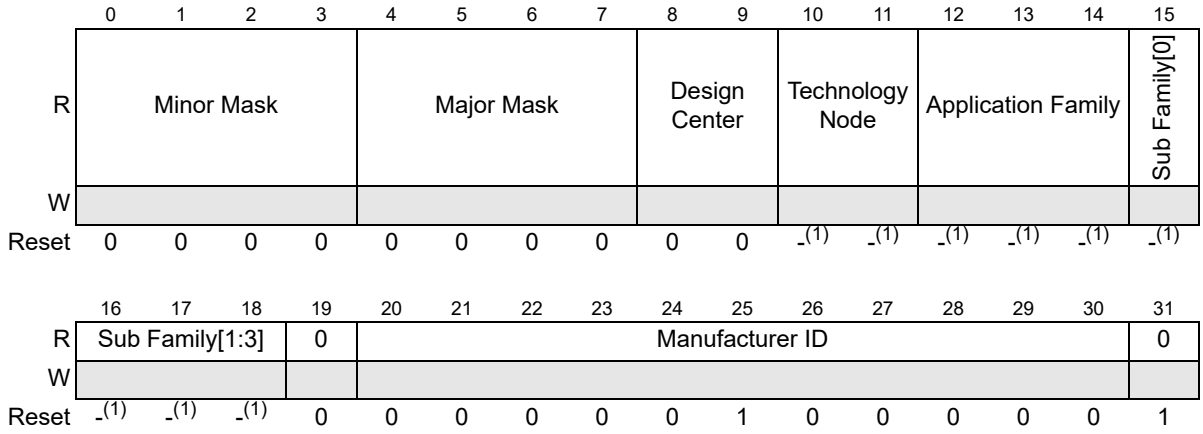
Figure 1957. 6-bit instruction register

73.3.2 Bypass register

The bypass register is a single-bit shift register path selected for serial data transfer between TDI and TDO when the BYPASS, CLAMP, HIGHZ or reserve instructions are active. After entry into the Capture-DR state, the single-bit shift register is set to a logic 0. Therefore, the first bit shifted out after selecting the bypass register is always a logic 0.

73.3.3 Device identification register

The device identification (JTAG ID) register, shown in the following figure, allows the revision number, part number, manufacturer, and design center responsible for the design of the part to be determined through the TAP. The device identification register is selected for serial data transfer between TDI and TDO when the IDCODE instruction is active. Entry into the Capture-DR state while the device identification register is selected loads the IDCODE into the shift register to be shifted out on TDO in the Shift-DR state. No action occurs in the Update-DR state.



1. The value is shown in *Table 1890*

Figure 1958. Device identification register

The following table describes the device identification register functions.

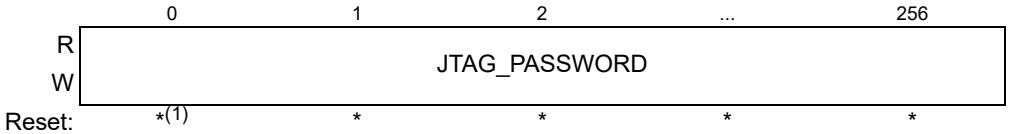
Table 1890. Device identification register field descriptions

Field	Description
0:3 Minor Mask	Minor Mask These bits contain the minor mask of the part
4:7 Major Mask	Major Mask These bits contain the major mask of the part
8:9 Design Center	Design Center These bits indicate the device design center
10:11 Technology Node	Technology Node These bits indicate the device technology node 01 40nm
12:14 Application Family	Application Family These bits indicate the device application family 000 SPC58N - Powertrain 40nm
15:18 Sub Family	Sub Family These bits indicate the device sub family 1000 SPC58NE84C3 , SPC58NE84E7
20:30 Manufacturer ID	Manufacturer Identity Code These bits contain the reduced Joint Electron Device Engineering Council (JEDEC) ID 00000100000: STMicroelectronics

73.3.4 JTAG_PASSWORD register

The JTAG_PASSWORD register is a 256-bit shift register path from TDI to TDO selected when the ENABLE_JTAG_PASSWORD instruction is active. The default reset value of the JTAG_PASSWORD register is 256'b0. The JTAG_PASSWORD register transfers its value to a parallel hold register on the rising edge of TCK when the TAP controller state machine

is in the Update-DR state. Once the ENABLE_JTAG_PASSWORD instruction is executed, the register value remains valid until a JTAG reset occurs. The operation of this register is described in the security documentation.



1. The reset value of JTAG_PASSWORD is 256'b0.

Figure 1959. JTAG_PASSWORD register

The following table describes the JTAG_PASSWORD register functions.

Table 1891. JTAG_PASSWORD register field descriptions

Field	Description
JTAG_PASSWORD	JTAG Password. The JTAG_PASSWORD bits are used to provide the JTAG password for security.

73.3.5 Boundary scan register

The boundary scan register is connected between TDI and TDO when the EXTEST, SAMPLE or SAMPLE/PRELOAD instructions are active. It is used to capture input pin data, force fixed values on output pins, and select a logic value and direction for bidirectional pins. Each bit of the boundary scan register represents a separate boundary scan register cell, as described in the IEEE 1149.1-2001 standard and discussed in *Section 73.4.5*. The size of the boundary scan register and bit ordering is device-dependent and can be found in the device BSDL file.

73.4 Functional description

This section explains the JTAGC functional description.

73.4.1 JTAGC reset configuration

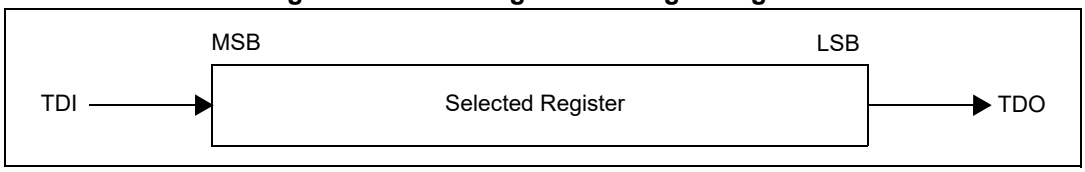
While in reset, the TAP controller is forced into the Test-Logic-Reset state, thus disabling the test logic and allowing normal operation of the on-chip system logic. In addition, the instruction register is loaded with the IDCODE instruction.

73.4.2 IEEE 1149.1-2001 (JTAG) Test Access Port

The JTAGC block uses the IEEE 1149.1-2001 TAP for accessing registers. This port can be shared with other TAP controllers on the MCU. Ownership of the port is determined by the value of the currently loaded instruction. For more detail on TAP sharing via JTAGC instructions refer to *Section 73.4.4.8: ACCESS_AUX_x instructions*.

Data is shifted between TDI and TDO through the selected register starting with the least significant bit, as illustrated in the following figure. This applies for the instruction register, test data registers, and the bypass register.

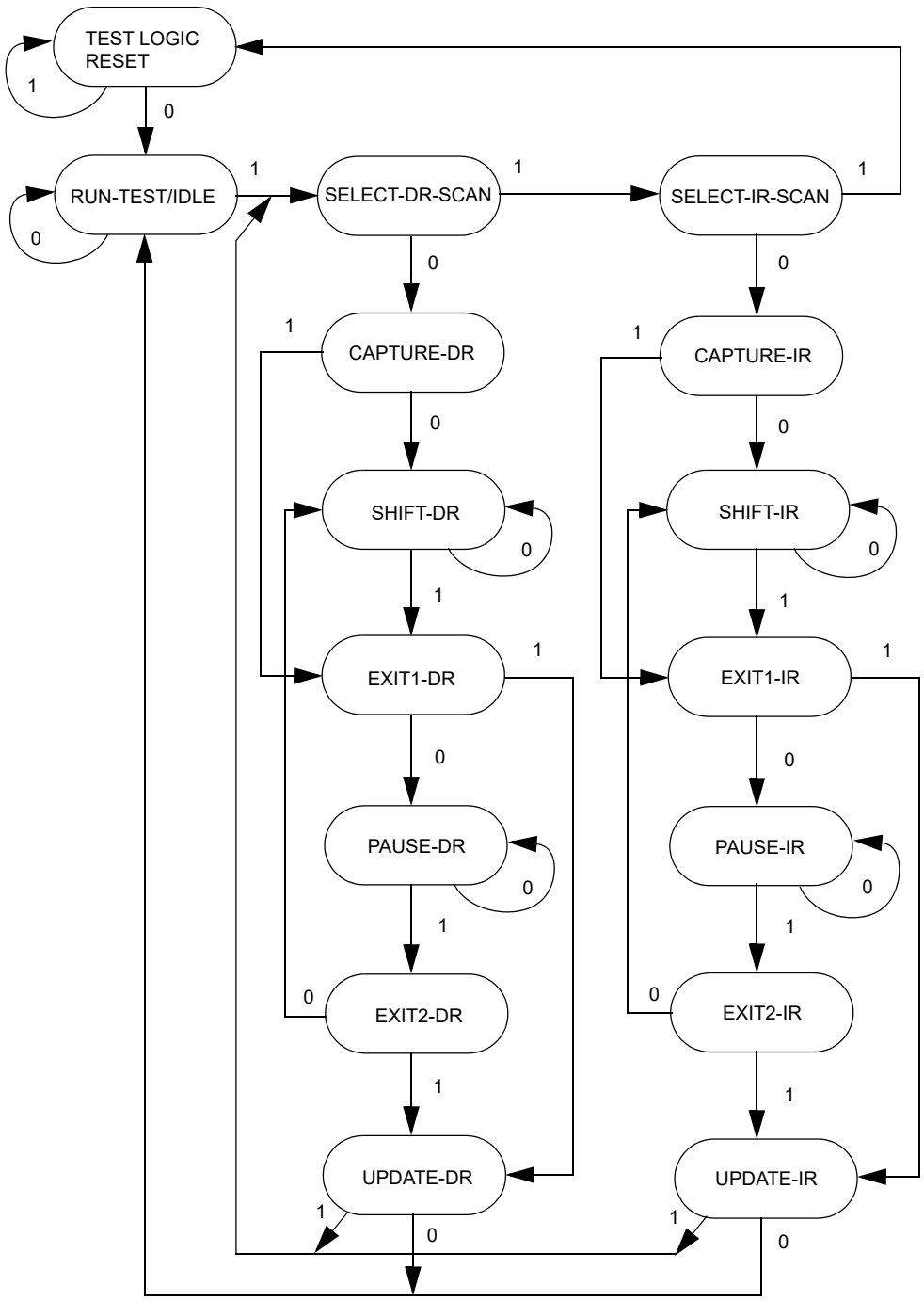
Figure 1960. Shifting data through a register



73.4.3 TAP controller state machine

The TAP controller is a synchronous state machine that interprets the sequence of logical values on the TMS pin. The following figure shows the machine's states. The value shown next to each state is the value of the TMS signal sampled on the rising edge of the TCK signal. As the following figure shows, holding TMS at logic 1 while clocking TCK through a sufficient number of rising edges also causes the state machine to enter the Test-Logic-Reset state.

Figure 1961. IEEE 1149.1-2001 TAP controller finite state machine



Note:

1. The value shown adjacent to each state transition represents the value of TMS at the time of a rising edge of TCK.

73.4.3.1 Enabling the TAP controller

The JTAGC TAP controller is enabled by setting JCOMP to a logic 1 value.

73.4.3.2 Selecting an IEEE 1149.1-2001 register

Access to the JTAGC data registers is achieved by loading the instruction register with any of the JTAGC block instructions while the JTAGC is enabled. Instructions are shifted in via the Select-IR-Scan path and loaded in the Update-IR state. At this point, all data register access is performed via the Select-DR-Scan path.

The Select-DR-Scan path is used to read or write the register data by shifting in the data (LSB first) during the Shift-DR state. When reading a register, the register value is loaded into the IEEE 1149.1-2001 shifter during the Capture-DR state. When writing a register, the value is loaded from the IEEE 1149.1-2001 shifter to the register during the Update-DR state. When reading a register, there is no requirement to shift out the entire register contents. Shifting may be terminated once the required number of bits have been acquired.

73.4.4 JTAGC block instructions

The JTAGC block implements the IEEE 1149.1-2001 defined instructions listed in the following table. This section gives an overview of each instruction; refer to the IEEE 1149.1-2001 standard for more details. All undefined opcodes are reserved.

Table 1892. 6-bit JTAG instructions

Instruction	Code[5:0]	Instruction Summary
IDCODE	000001	Selects device identification register for shift
SAMPLE/PRELOAD	000010	Selects boundary scan register for shifting, sampling, and preloading without disturbing functional operation
SAMPLE	000011	Selects boundary scan register for shifting and sampling without disturbing functional operation
EXTEST	000100	Selects boundary scan register while applying preloaded values to output pins and asserting functional reset
Reserved	000101	Reserved
Reserved	000110	Reserved
ENABLE_JTAG_PASSWORD	000111	Selects JTAG_PASSWORD register
HIGHZ	001001	Selects bypass register while three-stating all output pins and asserting functional reset
Reserved	001010	Reserved
CLAMP	001100	Selects bypass register while applying preloaded values to output pins and asserting functional reset
ENABLE_DCI_CR	001110	Enables access to DCI CR register
ENABLE_DCI_PINCR	001111	Enables access to DCI PINCR register
ACCESS_AUX_x	100000-111110	Grants one of the auxiliary TAP controllers ownership of the TAP as shown in the cells below.
Reserved	100000	Reserved
ACCESS_AUX_NR_PD	100001	Enables access to the NAR TAP controller on production device

Table 1892. 6-bit JTAG instructions (continued)

Instruction	Code[5:0]	Instruction Summary
Reserved	100010	Reserved
Reserved	100011	Reserved
Reserved	100100	Reserved
Reserved	100101	Reserved
ACCESS_AUX_JDC	100110	Enables access to the JDC TAP controller, to access the HSM challenge and response registers
ACCESS_AUX_HSM	100111	Enables access to the HSM core TAP controller
ACCESS_AUX_CORE_0	101000	Enabled access to E200 Core 0
ACCESS_AUX_CORE_1	101001	Enables access to E200 Core 1
ACCESS_AUX_CORE_2	101010	Enables access to E200 Core 2
Reserved	101011	Reserved
Reserved	101100	Reserved
Reserved	101101	Reserved
Reserved	101110	Reserved
Reserved	101111	Reserved
Reserved	110000	Reserved
Reserved	110001	Reserved
Reserved	110010	Reserved
ACCESS_AUX_GTM	110011	Enables access to GTM TAP controller
ACCESS_AUX_BUS_MON_0	110100	Enables access to NXMC Client 0 TAP controller
ACCESS_AUX_BUS_MON_1	110101	Enables access to NXMC Client 1 TAP controller
ACCESS_AUX_BUS_MON_2	110110	Enables access to NXMC Client 2 TAP controller
ACCESS_AUX_BUS_MON_3	110111	Enables access to NXMC Client 3 TAP controller
ACCESS_AUX_BUS_MON_4	111000	Enables access to NXMC Client 4 TAP controller
Reserved	111001	Reserved
ACCESS_AUX_SPU_A	111010	Enables access to SPU Client TAP controller
Reserved	111011	Reserved
ACCESS_AUX_PARALLEL(1)	111100	Enables access to all cores in parallel
Reserved	111101	Reserved
Reserved	111110	Reserved
BYPASS	111111	Selects bypass register for data operations
Reserved(2)	All other opcodes	Decoded to select bypass register

1. The use of this instruction is limited to the synchronous exit of debug mode via the execution of the GO+EXIT OnCE command. The execution of other instructions or commands is not supported
2. The manufacturer reserves the right to change the decoding of reserved instruction codes in the future

73.4.4.1 IDCODE instruction

IDCODE selects the 32-bit device identification register as the shift path between TDI and TDO. This instruction allows interrogation of the MCU to determine its version number and other part identification data. IDCODE is the instruction placed into the instruction register when the JTAGC block is reset.

73.4.4.2 SAMPLE/PRELOAD instruction

The SAMPLE/PRELOAD instruction has two functions:

- First, the SAMPLE portion of the instruction obtains a sample of the system data and control signals present at the MCU input pins and just before the boundary scan register cells at the output pins. This sampling occurs on the rising edge of TCK in the Capture-DR state when the SAMPLE/PRELOAD instruction is active. The sampled data is viewed by shifting it through the boundary scan register to the TDO output during the Shift-DR state. Both the data capture and the shift operation are transparent to system operation.
- Secondly, the PRELOAD portion of the instruction initializes the boundary scan register cells before selecting the EXTEST or CLAMP instructions to perform boundary scan tests. This is achieved by shifting in initialization data to the boundary scan register during the Shift-DR state. The initialization data is transferred to the parallel outputs of the boundary scan register cells on the falling edge of TCK in the Update-DR state. The data is applied to the external output pins by the EXTEST or CLAMP instruction. System operation is not affected.

73.4.4.3 SAMPLE instruction

The SAMPLE instruction obtains a sample of the system data and control signals present at the MCU input pins and just before the boundary scan register cells at the output pins. This sampling occurs on the rising edge of TCK in the Capture-DR state when the SAMPLE instruction is active. The sampled data is viewed by shifting it through the boundary scan register to the TDO output during the Shift-DR state. There is no defined action in the Update-DR state. Both the data capture and the shift operation are transparent to system operation.

73.4.4.4 EXTEST—External test instruction

EXTEST selects the boundary scan register as the shift path between TDI and TDO. It allows testing of off-chip circuitry and board-level interconnections by driving preloaded data contained in the boundary scan register onto the system output pins. Typically, the preloaded data is loaded into the boundary scan register using the SAMPLE/PRELOAD instruction before the selection of EXTEST. EXTEST asserts the internal system reset for the MCU to force a predictable internal state while performing external boundary scan operations.

73.4.4.5 ENABLE_JTAG_PASSWORD instruction

The ENABLE_JTAG_PASSWORD instruction selects the JTAG_PASSWORD register for connection as the shift path between TDI and TDO.

73.4.4.6 HIGHZ instruction

HIGHZ selects the bypass register as the shift path between TDI and TDO. While HIGHZ is active all output drivers are placed in an inactive drive state (for example high impedance).

HIGHZ also asserts the internal system reset for the MCU to force a predictable internal state.

73.4.4.7 CLAMP instruction

CLAMP allows the state of signals driven from MCU pins to be determined from the boundary scan register while the bypass register is selected as the serial path between TDI and TDO. CLAMP enhances test efficiency by reducing the overall shift path to a single bit (the bypass register) while conducting an EXTEST type of instruction through the boundary scan register. CLAMP also asserts the internal system reset for the MCU to force a predictable internal state.

73.4.4.8 ACCESS_AUX_x instructions

The JTAGC is configurable to allow other TAP controllers on the device to share the port with it. This is done by providing ACCESS_AUX_x instructions for each of these TAP controllers. When this instruction is loaded, control of the JTAG pins are transferred to the selected TAP controller. Any data input via TDI and TMS is passed to the selected TAP controller, and any TDO output from the selected TAP controller is sent back to the JTAGC to be output on the pins. The JTAGC regains control of the JTAG port during the UPDATE-DR state if the PAUSE-DR state was entered. Auxiliary TAP controllers are held in RUN-TEST/IDLE while they are inactive. Instructions not used to access an auxiliary TAP controller on a device are treated like the BYPASS instruction.

73.4.4.9 BYPASS instruction

BYPASS selects the bypass register, creating a single-bit shift register path between TDI and TDO. BYPASS enhances test efficiency by reducing the overall shift path when no test operation of the MCU is required. This allows more rapid movement of test data to and from other components on a board that are required to perform test functions. While the BYPASS instruction is active the system logic operates normally.

73.4.5 Boundary scan

The boundary scan technique allows signals at component boundaries to be controlled and observed through the shift-register stage associated with each pad. Each stage is part of a larger boundary scan register cell, and cells for each pad are interconnected serially to form a shift-register chain around the border of the design. The boundary scan register consists of this shift-register chain, and is connected between TDI and TDO when the EXTEST, SAMPLE, or SAMPLE/PRELOAD instructions are loaded. The shift-register chain contains a serial input and serial output, as well as clock and control signals.

73.5 Initialization/Application information

The test logic is a static logic design, and TCK can be stopped in either a high or low state without loss of data. However, the system clock is not synchronized to TCK internally. Any mixed operation using both the test logic and the system functional logic requires external synchronization.

To initialize the JTAGC block and enable access to registers, must be loaded the appropriate instruction for the test or action to be performed.

74 IEEE 1149.7 Compact JTAG Test Access Port Controller (CJTAG)

74.1 References

- IEEE Std 1149.1-2001 (R2008), Standard Test Access Port and Boundary-Scan Architecture
- IEEE Std 1149.7, Standard for Reduced-Pin and Enhanced-Functionality Test Access Port and Boundary-Scan Architecture

74.2 Introduction

The CJTAG module implements parts of the IEEE 1149.7 standard for test and debug capabilities. IEEE 1149.7 defines enhanced test and debug capabilities which are backward compatible with IEEE P1149.1. This standard is also known as “Compact JTAG” or “CJTAG”.

The module implements the following parts of the IEEE 1149.7 standard:

- IEEE 1149.1 compliant behavior while allowing multiple on-chip TAP controllers
- Ability to assert test reset
- Improved system scan performance with one-bit IR and DR chip bypass paths
- Physical connection of a Debug and Test System (external tool) to a running system without corrupting its operation
- Operation with both 4-pin and 2-pin scan topologies

74.2.1 Types of operation

The 1149.7 standard provides a scalable and flexible solution to meet the needs of varied component and system complexities. There are three types of operation:

- Compliant
  - Compliant with the IEEE 1149.1 standard
  - Four-pin operation using the “Standard Protocol” – IEEE 1149.1 signaling where TCK, TMS, TDI and TDO signals transfer control/data information
  - More than one TAPC may be associated with the TAP (once made visible by private instructions)
- Extended
  - Compatible with the IEEE 1149.1 standard
  - Four-pin operation using the “Standard Protocol” – IEEE 1149.1 signaling where TCK, TMS, TDI(C) and TDO(C) signals transfer control/data information
  - Additional test/debug functions
- Advanced
  - Two-pin operation using the “Advanced Protocol” – signaling where TCK(C) and TMS(C) signals transfer control/data information
  - The TMS(C) signal is operated in a bidirectional manner with the TMS, TDI and TDO TAP information serialized and transferred via this signal

74.2.2 Deployment by class

The TAP.7 capabilities are deployed using five capability classes (T0–T4), each matching the needs of certain scan topologies and mixes of 1149.1 TAPs and 1149.7 TAPs. The classes provide the following capability:

- **Compliant Operation** – provides behavior compliant with the IEEE 1149.1-2001 standard:
  - **Class T0** – 1149.1 compliance when multiple TAPs are used on the same chip.
- **Extended Operation** – extends the capabilities of the IEEE 1149.1-2001 standard:
  - **Class T1** – T0 + Test access port power management, functional reset, and test reset.
  - **Class T2** – T1 + Chip bypass (1-bit) for IR/DR Scans + hot-connection with no errors.
  - **Class T3** – T2 + Star operation (TAPs are paralleled) with series equivalent scans for test.
- **Advanced Operation** – adds capabilities addressing test and debug needs in complex environments:
  - **Class T4** – T3 + Scan with two pins with the Advanced Protocol. A number of protocol optimizations maximize scan performance. TDI(C)/TDO(C) pins are optional and may have programmable function when implemented.

74.2.3 1149.7 TAP signals

The TAP signals for the T0–T4 TAP.7 classes are listed in *Table 1893*. Signal names that end in “C” are associated with expanded functionality (above that provided by the IEEE 1149.1 standard).

The notation used in the table is as follows:

- M = Mandatory signal
- MC = Mandatory signal with expanded functionality
- O = Optional signal
- OC = Optional signal with expanded functionality

The CJTAG module provides T4 capability and implements the TCKC, TMSC, TDIC, TDOC, and nTRST pins.

Table 1893. TAP.7 signal list

Signal Name	Description	Class				
		T4	T3	T2	T1	T0
TCK/TCKC	Test clock	MC	M	M	M	M
TMS/TMSC	Test mode select	MC	M	M	M	M
TDI/TDIC	Test data input	OC	MC	M	M	M
TDO/TDOC	Test data output	OC	MC	M	M	M
nTRST	Test reset	O	O	O	O	O
Mandatory signal count (minimum)		2	4	4	4	4

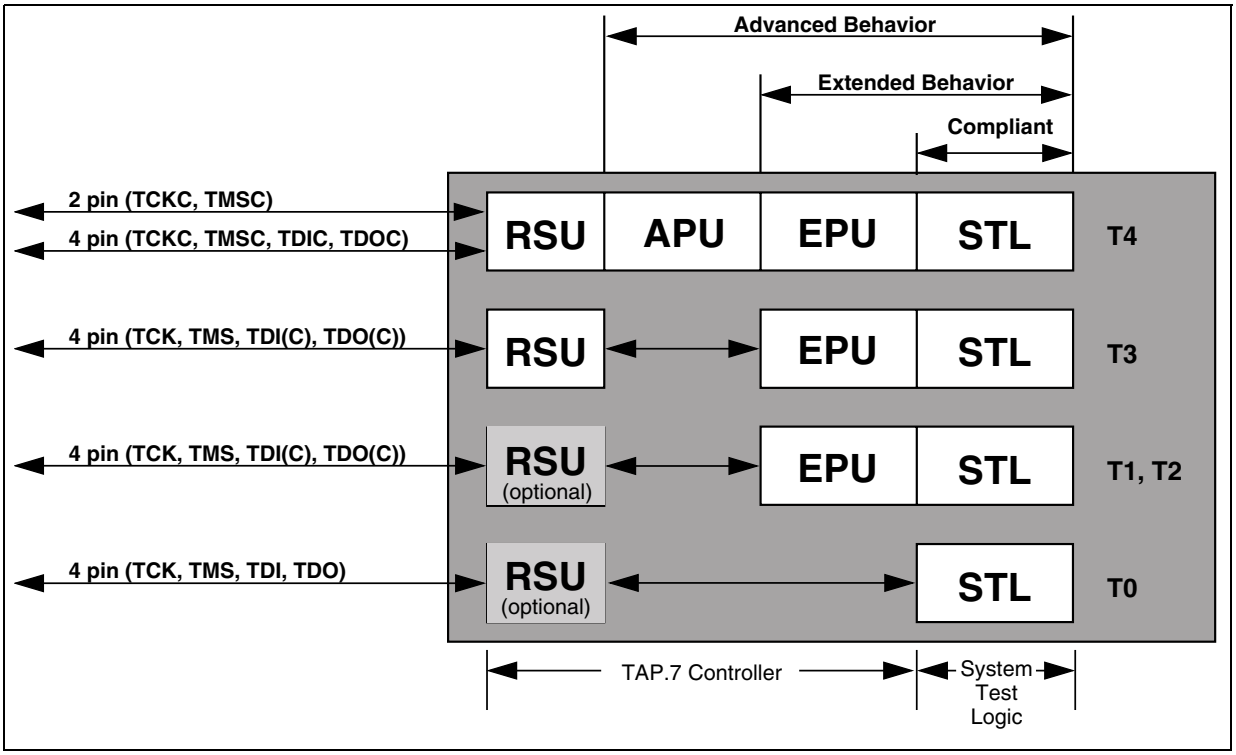
74.2.4 TAP.7 architecture

The TAP.7 hardware architecture is shown in *Figure 1962*. It is described with the hardware layers listed below:

- **STL System Test Logic** – Logic with 1149.1 compliant behavior and underlying TAP hierarchy (T0 capabilities). Provides an IEEE 1149.1 interface for the T0 TAP.7. This logic is outside the CJTAG module.
- **RSU – Reset and Selection Unit** – A hardware layer that is placed between the APU, EPU, or STL and the TAP.7 signals (Added as an option to support the use of the Control Protocol) Provides reset and TAP.7 Controller selection services.
- **EPU – Extended Protocol Unit** – A hardware layer that is placed between the STL and the TAP.7 signals. (Added for T1, T2, and T3 capabilities). Provides an IEEE 1149.1 interface for the T1–T3 TAP.7s.
- **APU – Advanced Protocol Unit** – A hardware layer that is placed between the STL/EPU and the TAP.7 signals (added for T4 capabilities). Provides a T4 TAP.7 interface that is either narrow or wide, with the wide version providing an IEEE 1149.1 interface.

The CJTAG module supports T4 functionality that includes STL, RSU, EPU, and APU hardware layers.

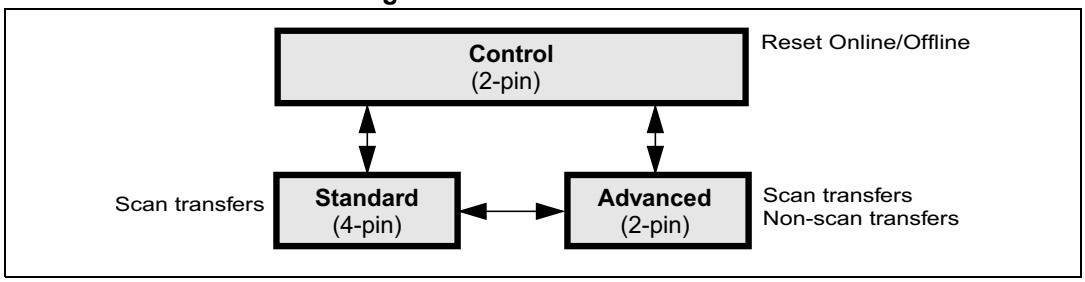
Figure 1962. TAP.7 Controller Architecture



74.2.5 Protocols

The TAP.7 architecture utilizes the three protocols shown in *Figure 1963*.

Figure 1963. TAP.7 Behaviors



The Standard Protocol transfers data with the TMS(C) signal value being sampled with a TCK(C) edge. The Advanced and Control Protocols use both data transferred with the TMS(C) signal value being sampled with a TCK(C) signal edge and escape sequences. The Standard Protocol may be used to switch to the use of the Advanced Protocol and vice versa. The Control Protocol can also be used to switch between the use of the Standard and the Advanced Protocol. The mandatory and optional deployment of these protocols with the TAP.7 classes is shown in *Table 1894*.

Table 1894. Class/protocol relationship

Class	Protocol		
	Standard	Control	Advanced
T0 – T2	Mandatory	Optional	N/A
T3	Mandatory	Mandatory	N/A
T4	Mandatory	Mandatory	Mandatory

The CJTAG module implements T4 functionality and supports the Standard, Advanced, and Control Protocols.

74.2.5.1 Standard Protocol

The Standard Protocol is the signaling defined by the IEEE 1149.1 standard. This protocol is also used to implement TAP.7 commands using only the TCK(C) and TMS(C) signals in a manner that does not require either TAP.7 controller instruction or data registers. The TAP.7 controller does not add bits to the scan paths of underlying technology. The Standard Protocol can be used to manage the TAP.7 controller functionality independent of the function associated with the availability of the TDI and TDO signals. The Standard Protocol supports STL data transfers only when these signals are present and provide the 1149.1 functionality defined by the IEEE 1149.1 standard.

74.2.5.2 Advanced Protocol

The Advanced Protocol supports both STL data transfers and management of the TAP.7 controller functionality with both the 2-signal (narrow) and 4-signal (wide) configurations. The Advanced Protocol serializes the TMS, TDI, and TDO information used with the Standard Protocol using only the TCK(C) and TMS(C) signals. A number of additional scan formats provide bit sequences optimized for specific use cases. Each of these additional scan formats affects the TMSC signaling sequences.

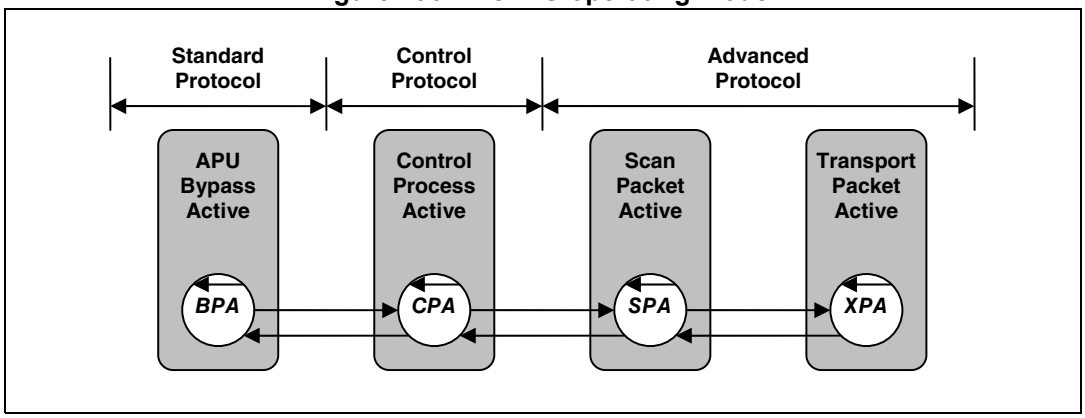
74.2.5.3 Control Protocol

The Control Protocol uses Escape sequences to reset the TAP.7 controller, place a TAP.7 controller offline or online, and indicate End of Transmission (EOT) with the Advanced Protocol. The characteristics of this protocol provide Hot-Connect-Protection by requiring a preamble of alternating ones and zeroes preceding a selection escape when a TAP controller is started up offline.

74.3 Operating models

Figure 1964 illustrates a simplified view of the CJTAG operating model.

Figure 1964. CJTAG operating model



74.4 CJTAG implementation summary

The CJTAG module is a T4 TAP.7 controller that implements the following functions.

74.4.1 T0 Functions

- IDCODE. - implemented in chip level TAPC (JTAGC module)
- 1149.1 Compliance at start-up
- Isolating / Excluding of embedded TAPs – implemented in chip level TAPC (JTAGC module)
- Operation with other technologies and scan topologies
- Multiple embedded TAPs

74.4.2 T1 functions

- EPU Command processing
- EPU Class-Specific Registers
- Asserting a test reset to the STL (TRESET Register)

74.4.3 T2 functions

- “Super Bypass” function – JScan0, JScan1, JScan2 scan formats. Allows the STL to be coupled or decoupled.

74.4.4 T3 functions

- Generation of a TAP.7 controller reset with a Reset Escape Sequence
- JScan3 Scan Format

74.4.5 T4 functions

- TMSC sampling on rising edge or falling edge of TCKC
- MScan (For test and debug)
- OScan 0-7
- TDIC and TDOC / Alternate pin function multiplexing (wide 4 only)

74.5 Ancillary services

74.5.1 Overview

Ancillary services are pertinent to any TAP.7 controller implemented with an RSU, such as the CJTAG module.

Ancillary services include:

- Resets
- Start-up options
- Escape sequences
- TAPC state machine

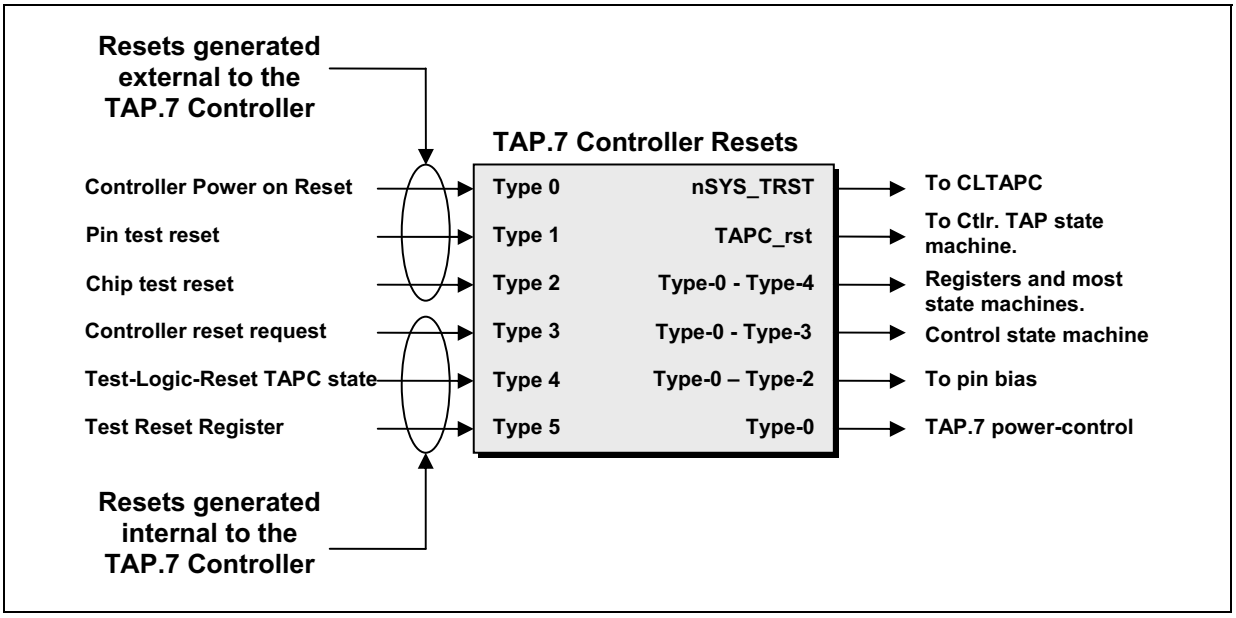
74.5.2 Resets

The TAP.7 controller reset function expands the reset function provided by the IEEE 1149.1 standard to provide additional reset types. It supports six reset types, Type-0 – Type-5, listed below. It can be implemented with as few as two reset types (Type-2 and Type-4), or as many as six reset types (all types) of reset inputs: This is determined by the class and options implemented.

The CJTAG module utilizes Type-0, Type-2, Type-3, Type-4, and Type-5 resets.

- **Type-0** – Reset of the TAP.7 by power management logic (power-on reset)
- **Type-1** – Chip level generated Test Reset (equivalent to Type-2)
- **Type-2** – Externally generated Test Reset (nTRST/nTRST_PD pin)
- **Type-3** – TAP.7 controller generated reset
- **Type-4** – *Test-Logic-Reset* state
- **Type-5** – TAP.7 Controller TReset Register reset of the CLTAPC

Figure 1965. Conceptual view of the EPU reset logic and state machines



74.5.2.1 Type 5 reset

A Type-5 reset affects only the STL via the nSYS_TRST pin. It is asserted when the TRESET Register value is a logic 1. This register is set to its inactive state by reset Types 0-4.

74.5.2.2 Type 4 reset

A Type-4 reset is created by the *Test-Logic-Reset* state. It does not affect the TAP.7 TAPC state machine state as this state is the source of this reset, nor does it cause the assertion of the nSYS_TRST signal. It does however affect the coupling and decoupling of the STL. The Type-4 reset initializes the TAP.7 registers and most controller state machines. It affects but does not initialize the control state machine managing TAP.7 controller selection. It also establishes the default operating conditions.

74.5.2.3 Type 3 reset

A Type-3 reset is created by reset requests generated by the TAP.7 controller. A Type-3 reset performs the functions associated with a Type-4 reset and creates the *Test-Logic-Reset* state.

A Type-3 reset is generated in response to reset requests generated by EPU and APU logic

- A reset escape sequence
- Reset protocol sequences (delay packet directive, check packet directive)

A Type-3 reset request is generated with a reset escape sequence. This function is similar to but not the same as the reset function generated by a test reset pin. The Advanced Protocol includes certain bit patterns that initiate Type-3 resets. These bit patterns assist in the initialization of the TAP.7 controller when the TCKC signal is sourced by the TS and the external tool/TS connection is broken. The external tool may also generate these sequences, if desired, while the external tool/TS connection remains intact. These reset requests cause a Type-3 reset that is one clock wide.

With this being the case, consideration may be given to integrating the CJTAG module without the use of the Test Reset signal (nTRST/nTRST_PD pin).

74.5.2.4 Type 2 reset

A Type-2 reset is created by chip level logic to initialize the TAP.7 controller when TAP.7 power control is not implemented. It is unimplemented otherwise. It performs all of the functions of the Type-3 reset in addition to initializing the escape sequence detection logic.

74.5.2.5 Type 1 reset

A Type-1 reset is created with either the nTRST or nTRST_PD pin. It is not created when neither of these pins is implemented. It performs all of the functions of the Type-2 reset. It initiates TAP.7 controller power down provided the TAP.7 power control mode permits power down.

74.5.2.6 Type 0 reset

A Type-0 reset is implemented when TAP.7 controller power down is implemented. It performs all of the functions of a Type-1 reset and also initializes the TAP.7 power control logic.

74.5.2.7 Reset State Machine

The Reset State Machine assists in the translation of the reset inputs to the reset outputs. It handles Type-3 reset controller reset requests. The state machine state assures Type-0, Type-1, and Type-2 resets are forwarded to the STL without modifying their duration.

This state machine allows a Type-3 reset only when its state is “T3 Reset” (two TCK(C) falling edges have occurred in the absence of a Type-0, Type-1, and Type-2 reset).

Figure 1966. Reset state machine

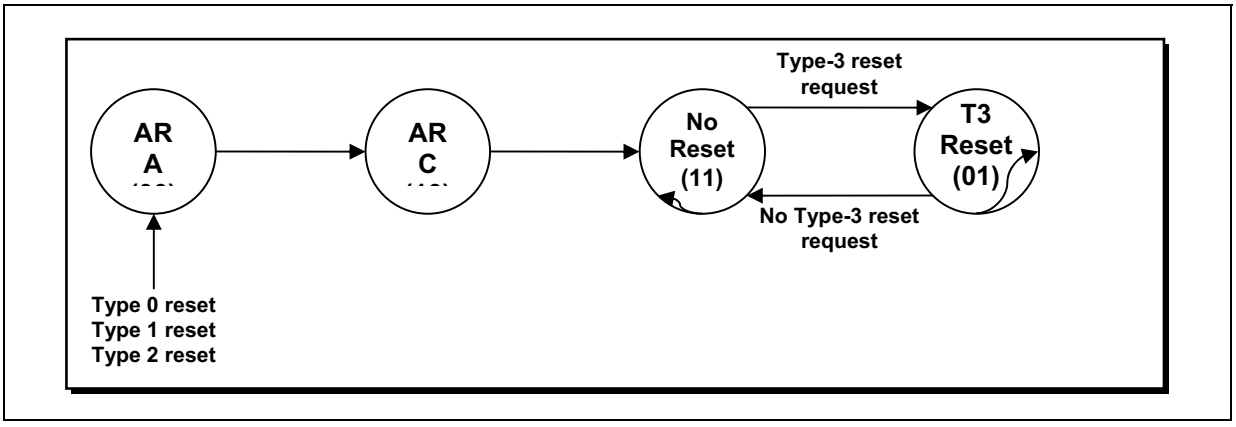


Table 1895. Reset state descriptions

Reset state		Description
ARA	Async. Reset Asserted	An async. reset has occurred or is occurring
ARC	Async. Reset Completed	An async. reset has occurred and has been released
No Reset	No Reset	No Reset
T3 Reset	Type-3 Reset	A Type-3 reset request has been processed.

Type-0, Type-1, and Type-2 resets are fully asynchronous. These resets are passed directly to nSYS_TRST without modification. The Type-3, Type-4, and Type-5 resets are fully synchronous.

The effects of the TAP.7 controller reset types are shown in *Table 1896*.

Table 1896. Reset effects

Reset	CLTAPC	Pin-Hi-Z	State Machines other than CSM	Control State Machine (CSM)	TAPC SM State	Escape Sequence Detection	Power Control State
Type-5	Yes	–	–	–	–	–	–
Type-4	–	Yes	Yes	–	–	–	–
Type-3	Yes	Yes	Yes	Yes	Yes	–	–
Type-2	Yes	Yes	Yes	Yes	Yes	Yes	–
Type-1	Yes	Yes	Yes	Yes	Yes	Yes	–
Type-0	Yes	Yes	Yes	Yes	Yes	Yes	Yes

74.5.3 Start-up Options

74.5.3.1 Overview

The TAP.7 start-up options are shown in *Table 1897*. The only startup option currently supported by the CJTAG module is 1149.1 compliant.

Table 1897. Start-up Options

Start-up Option	Description
1149.1 Compliant	TAP.1 signal functionality, the CLTAPC is coupled
1149.1 Compatible	TAP.7 signal functionality, the CLTAPC is decoupled
1149.1 Protocol Compatible	No, or alternate TDIC/TDOC signal functionality, the CLTAPC is decoupled.
Offline-at-Start-up (Dormant)	The TAP.7 controller is offline and awaits synchronization to external tool operation so as to be placed online, the CLTAPC is decoupled after its state is moved to *Run-Test/Idle*

74.5.3.2 1149.1 Compliant start-up

Start-up behavior with the 1149.1 Compliant behavior start-up option:

- 1149.1-specific behavior / Standard protocol
- The TAPC finite state machine operated in lock step with the CLTAPC finite state machine
- The TDIC and TDOC pins are present and provide the TDI and TDO function
- The CLTAPC is coupled at start-up

With this start-up option, the TAP.7 controller may be programmed and the scan topology may be interrogated. If the scan topology is known to be the Series Scan Topology at power-up, the TAP.7 can be operated as though it is a TAP.1.

Table 1898. Reset values for TAP.7 start-up options

Start-up Option	Type-0 – Type-3 Reset?	Default TAP.7 and TAP.7 controller characteristics				
		TAP.7 Controller online	CLTAPC Coupled	SGC Reg. == 1	SCNFMT	Initial TMSC pin bias
1149.1 Compliant	x	Yes	Yes	Yes	JScan0	PU

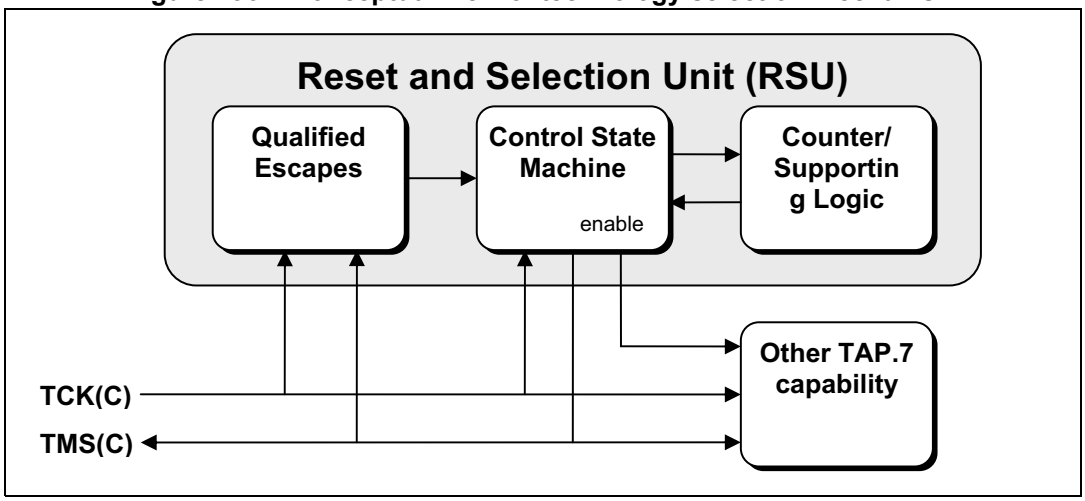
74.5.4 RSU operation

The RSU uses TCKC and TMSC with the control protocol for configuration control.

74.5.4.1 General Operation

From a very high level, the RSU enables the remainder of the TAP.7 functionality when the TAP.7 controller is Online and disables the remainder of the TAP.7 functionality otherwise. The RSU either consumes or discards the TMS(C) information while the TAP.7 controller is Offline. The relationship of the escape sequence detection, technology selection logic, and the underlying technology is shown in *Figure 1967*.

Figure 1967. Conceptual view of technology selection mechanism



74.5.4.2 Common Signaling Across Technologies

With the TAP.7 architecture, the signaling conventions chosen for technology selection is multiple data transitions of the TMS(C) signal while the TCK(C) signal is a logic 1. The common signaling space is available when the external tool sources the TCK(C) signal as it is capable of creating these signaling conventions. It is unavailable when the TS sources the TCK(C) signal as the external tool cannot stop the TCKC signal at a logic 1 to create these signaling conventions.

74.5.4.3 Escape Sequence Detection

TAP.7 TAPC interprets the count of the number of TMS(C) edges while TCK(C) is a logic 1 as one of four escape sequences. Each escape sequence has a different function and TMSC edge count. These escape sequences, their edge counts, and their functions are described below:

- **Custom** (2 or 3 edges) – Ends scan and transport transfers with a TAP.7 controller, may be used for other purposes with another technology.
- **Reset** (> 7 edges) – Resets all technologies (generates a Type-3 TAP.7 TAPC reset).

The external tool creates an escape sequence by generating one or more TMS(C) edge pairs while the TCK(C) signal is a logic 1 value. Even and odd TMS(C) edge counts beginning with two are given the same meaning.

An escape sequence:

- Can be detected only when a Type-0, Type-1 or Type-2 reset is **not** asserted.
- Begins and ends while the TCK(C) is a logic 1.
- Uses TMS(C) as a clock while the TCK(C) is a logic 1.
- Overlays additional control information onto the normal information that is transferred with the TCK(C) and the TMS(C) without changing the normal information.
- Must be detected when TCK(C) is a logic 1 regardless of the TCK(C) and TMS(C) drive histories.

74.5.4.3.1 Custom escape sequence

An online technology can use the custom escape sequence in any manner it chooses. It is used as the End of Transmission (EOT) of certain data types (formats OScan4-7) with T4 and above TAP.7s.

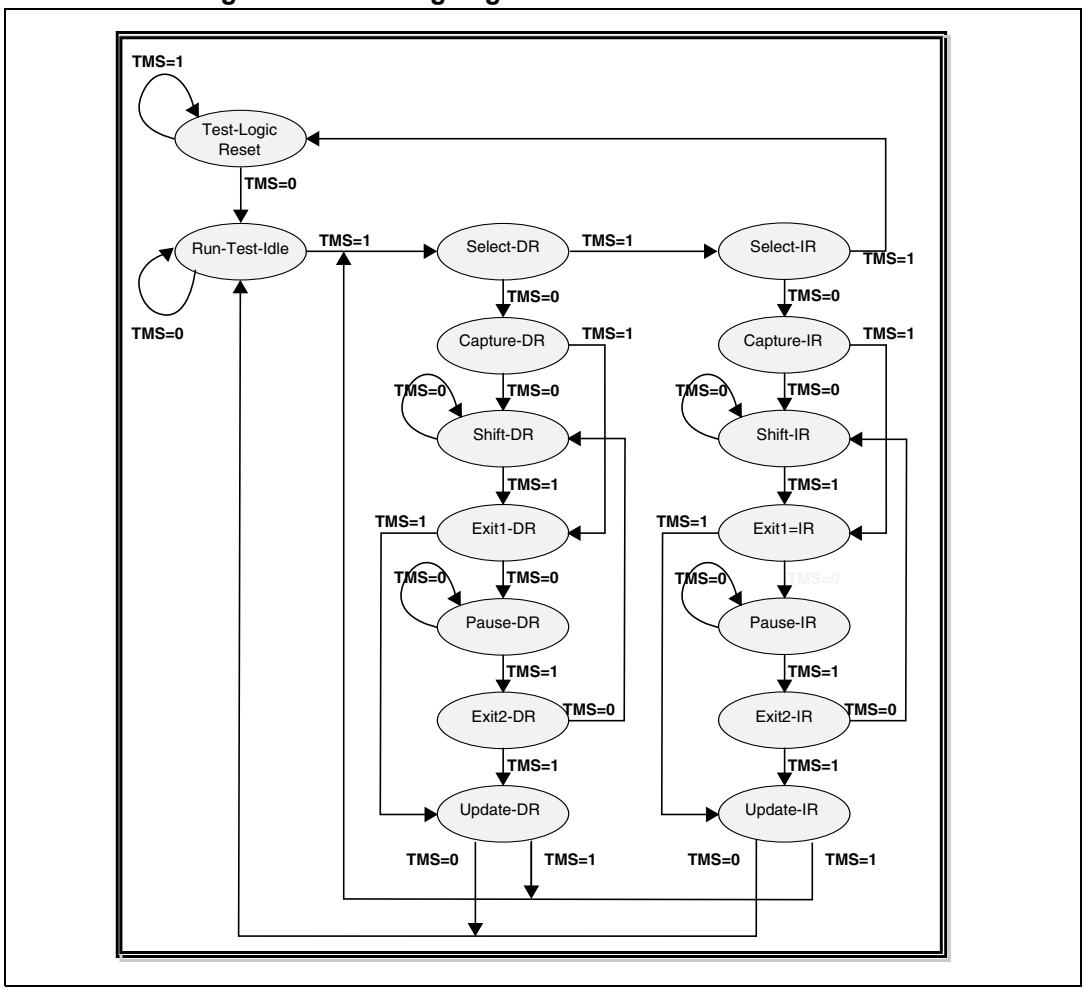
74.5.4.3.2 Reset escape sequences

A reset event initializes all technologies sharing the TCK(C) and TMS(C) connectivity. A reset escape sequence resets the technology. It is recommended that this reset occurs with the falling edge of TCKC. With a TAP.7 controller a reset escape sequence generates a Type-3 reset beginning with the falling edge of TCK(C) following the asynchronous detection of the event.

74.5.5 TAPC State Machine

T1 and above TAP.7s require knowledge of the TAPC state machine state. With T2 and above TAPs this knowledge must be developed independently of the CLTAPC. This is generally accomplished by implementing a TAPC state machine within the EPU. This TAPC state machine provides the function of the state machine within a standard 1149.1 TAPC. It is recommended that this state machine is implemented to change state on the falling edge of TCK(C) as this has numerous advantages to a TAP.7 controller. The EPU does not contain 1149.1 Instruction or Data Registers. The APU controls the state changes of this TAPC state machine with T4 and above TAP.7s.

Figure 1968. Falling edge TAP controller state machine



74.6 EPU (Extended Protocol Unit) Operation

74.6.1 EPU Operation

The EPU adds functionality to the BYPASS and IDCODE instructions.

74.6.1.1 Zero-bit DR-Scans (ZBS)

A zero-bit DR-Scan (ZBS) is a TAPC-state sequence that begins with the *Select-DR-Scan* state and ends with an exit from the *Update-DR* state without an intervening *Shift-DR* state. The external tool controls the EPU using ZBSs.

The use of ZBSs can begin immediately following the *Test-Logic-Reset* state since this state initializes the Instruction Registers of TAPs with either the *BYPASS* or *IDCODE* instructions.

74.6.1.2 Utilizing ZBSs for TAP.7 Controller Functionality

Beginning with a ZBS count of zero, the ZBS count is incremented with each consecutive occurrence of a ZBS without encountering a *Shift-DR* state. This count cannot be

incremented past seven. *Run-Test/Idle* is the only state that may occur between consecutive occurrences of ZBSs without terminating the count of consecutive ZBSs.

74.6.1.3 Locking the ZBS Count

When a DR-Scan containing a *Shift-DR* state occurs, and the ZBS count is greater than zero, the ZBS count is not incremented when the *Update-DR* state is reached. It is instead locked at its current value. ZBSs occurring after locking the ZBS count do not affect the locked ZBS count. Locking the ZBS count is the equivalent of storing the count for subsequent use.

74.6.1.4 Control levels

Locking the ZBS count activates a control level that is equal to the locked ZBS count (1-7) when ZBSs are being used for TAP.7 functionality and merely locks the ZBS count otherwise.

1. ZBSs may be used by the STL or another function
2. TAP.7 command generation
3. Reserved control level
4. Accesses to optional TAP.7 scan paths
5. Accesses to optional TAP.7 scan paths
6. Force offline operation (optional)
7. External tool uses

Control level one may be used by the STL since there is no TAP.7 controller functionality associated with this control level.

Control level two enables DR-Scans to create TAP.7 commands. A portfolio of these commands supports the deployment of TAP.7 controllers in the Series or Star Scan Topologies. These commands manage TAP.7 controller registers and perform other functions.

Control level three is reserved for use with a subsequent specification revision.

Control levels four and five provide access to custom (private) scan paths when access to these paths is permitted by a TAP.7 register bit used for this purpose. The EPU provides a 1-bit scan path otherwise. No other EPU functionality is associated with these control levels. The CJTAG module does not support the use of this control level.

Control level six may be optionally used to force offline operation with a TAP.7 controller that supports this capability. The EPU provides a 1-bit scan path for this control level. The CJTAG module does not support the use of this control level.

Control level seven is reserved for external tool use. With this control level, external tool capabilities may be managed using the same infrastructure used to manage TAP.7 controller capability. The EPU provides a 1-bit scan path for this control level.

74.6.1.5 Exiting a Control Level

A control level is exited when one of the following occurs:

- The *Select-IR-Scan* TAP.7 controller state
- The *Test-Logic-Reset* state
- Certain TAP.7 controller commands (Exit Control Level – ECL register bit)
- An event that synchronizes the operation of TAP.7 controllers

74.6.1.6 Zeroing the ZBS Count

Events causing an exit from a control level also zero the ZBS count. Scan Selection Directives (SSDs) associated with T3 and above TAP.7s zero the ZBS count when the control level has not been locked.

74.6.1.7 Utilizing ZBSs within the STL

After the *Test-Logic-Reset* state, ZBSs are used for basic EPU control. ZBSs that are intended for TAP.7 use are henceforth called "***TAP.7 ZBSs***". ZBSs that are intended for other uses are henceforth called "***Other ZBSs***". Two TAP.7 registers enable the STL's use of ZBSs. Storing these registers causes a control level exit, zeroes the ZBS count, and enables the use of Other ZBSs. Storing the Suspend Register (**SUSPEND**) suspends the use of control levels. Storing the ZBS Inhibit Register (**ZBSINH**) inhibits ZBS detection.

When either of these methods is used, the *Test-Logic-Reset* state enables the use of TAP.7 ZBSs in addition to initializing the TAP.7 controller.

The operation of the TAP.7 controller changes with the use of these two paths.

When the System Path is used:

- A count of eight consecutive ZBSs is used to create a logic 0 SUSPEND Register value.
- The CLTAPC supplies the TAP.7 scan path and controls the drive of TDO(C), provided it is not bypasses.
- The EPU supplies the TAP.7 scan path (a one bit bypass) when the CLTAPC is bypassed.
- Functions supporting Boundary Scan in a Star Scan Topology are enabled.

When the EPU Path is used:

- When the control level has not been locked, the TDO(C) signal remains high-impedance.
- The command control level may be created.
- When commands are used, the TDO(C) signal remains high-impedance except during *Shift-DR* states within CR-Scans, and remains high impedance at other times.
- Auxiliary scan paths may be accessed with control levels four and five.

74.6.2 EPU Registers

74.6.2.1 Global

A global register is a register whose value is stored at the same time and with the same value as registers with the same name in all TAP.7 TAPCs sharing an external tool connection. They are stored using TAP.7 TAPC commands dedicated for this purpose.

These commands store a single global register. Global registers manage TAP.7 TAPC functions that affect the synchronized operation of a TAP.7 TAPC. An example of a global register is the Scan Format (SCN_FMT) Register used to define the TAPC signal protocol.

A second means of storing all global registers is available. The selection of the TAP.7 TAPC with the technology selection function stores either the Scan Format register value (short form) or all global registers (long form). Storing all global registers provides a simple means to synchronize the values of all global registers when Technology Selection causes a non-operating TAP.7 TAPC to become operational while other TAPCs are operational.

74.6.2.2 Local

Local registers manage TAP.7 TAPC functions that do not affect the synchronized operation of TAP.7 TAPCs. Local registers are conditionally stored using TAP.7 controller commands dedicated for this purpose. These commands provide for changing the value of these registers one a time with a unique value in a Series Scan Topology based on the TAPC's position on the scan chain. Local registers are stored with a unique value in a Star Scan Topology based on a TAPC address assigned to the TAPC.

Registers programmed with commands are described in this section. The CJTAG module implements all mandatory T4 registers as well as the optional APFC, TRESET, and RDBACK0 registers.There are 16 mandatory and 10 optional registers that can be grouped in classes as shown in Read-only – Configuration and register read back.

- **Control** – TAP.7 controller behaviors and characteristics.
- **Options** – Control of optional functions.
- **Select** – Controls the selection of a TAP.7 for Scan and execution of conditional commands.
- **Read-only** – Configuration and register read back.

Table 1899. TAP.7 controller register list (managed with commands)

Register type	Global/ local	Width	Register mnemonic	Name
Control	Global	1	ECL	Exit Control Level
		1	SUSPEND	Suspend
		1	ZBSINH	ZBS Detect Inhibit
		5	SCNFMT	Scan Format
		1	SSDE	SSD Enable
		2	DLYC	Delay control
		2	RDYC	Ready Control
		1	TRESET	Test Reset
		2	APFC	Aux. Pin Func. Cntl.
Options Select	Local Local	1	CGM	Cond. Group Member
		1	SGC	Scan Group Candidate
		1	SEDGE	Sampling Edge

Table 1899. TAP.7 controller register list (managed with commands) (continued)

Register type	Global/ local	Width	Register mnemonic	Name
		32	RDBACK0	Read-back 0
		32	CNFG0	Configuration Register 0
Read-only	Local	M = Mandatory register O = Optional register		

74.6.2.3 Registers

74.6.2.3.1 Exit Control Level (ECL)

Storing this register zeroes the ZBS count and unlocks the control level.

This register is set using the STMC command. It remains logic 1 only momentarily before immediately returning to logic 0.

This register is cleared by a global register load.

74.6.2.3.2 Suspend (SUSPEND)

Setting this register suspends the use of control levels. When this register is a logic 1, the “System Path” (STL) is used.

- **1** – Suspend use of control levels. Enables “System Path.”
- **0** – Enable use of control levels.

SUSPEND is set with the STMC command.

This register is cleared when 8 consecutive ZBSs are detected without the locking of the ZBS count.

This register is cleared by a global register load.

74.6.2.3.3 ZBS Inhibit (ZBSINH)

Storing to ZBSINH inhibits ZBS detection.

- **1** – Inhibit ZBS detection.
- **0** – Enable ZBS detection.

ZBSINH is set with the STMC command.

This register is cleared by a global register load.

74.6.2.3.4 Scan Format (SCNFMT[4:0])

Set using Store Scan Format (STFMT) command.

Read using Scan String (SCNS) command.

Table 1900. Scan Format Register Values

TAPC.7 Class	SCNFMT	Description
T2–T4	00000	JScan0: – 1149.1 Compliant Operation. – Requests the coupling of the CLTAPC when the Run-Test/Idle state is reached.
	00001	JScan1: – 1149.1 Compatible Operation – Requests the decoupling of the CLTAPC when the Run-Test/Idle state is reached.
	00010	JScan2: – 1149.1 Compatible Operation. – Requests the coupling or decoupling of the CLTAPC when the Run-Test/Idle state is reached based on the value of the CACT register
T3–T4	00011	JScan3: – Non-compatible 1149.1 Operation – Allows the detection of a T2 TAP.7 controller that is deployed in a Star-4 topology. – Enables TDOC signal behavior that prevents drive conflicts in a Star-4 Scan Topology. – Enables the use of SSDs
T4	01000	OScan0 – Data rate dependent components – Single TAP.7 communication – Debug and Test applications
	01001	OScan1 – TAP.1 components – Single/Multi-TAP.7 communication – Debug and Test applications
	01010	OScan2 – TAP.1 components – No TDI or TDO in non-shift states – Debug applications
	01011	OScan3 – TAP.1 components – No TDI in non-shift states – No TDO in any state

Table 1900. Scan Format Register Values (continued)

TAPC.7 Class	SCNFMT	Description
T4	01100	OScan4 – Data rate dependent components – Single TAP.7 communication – Debug and Test applications
	01101	OScan5 – TAP.1 components – Single/Multi-TAP.7 communication – Debug and Test applications
	01110	OScan6 – TAP.1 components – No TDI or TDO in non-shift states – Debug applications
	01111	OScan7 – TAP.1 components – No TDI in non-shift states – No TDO in any state
	10000	MScan – Data rate dependent components – Multi TAP.7 communication – Directed CID assignment – Virtually any IP with compliant or non-compliant 1149.1 behavior
	10001 – 11111	Reserved

The use of a Scan Format value other than 00000b – 10000 shall not change the value of the SCNFMT Register.

74.6.2.3.5 Scan Selection Directive Enable (SSDE)

This register specifies whether the Scan Selection State may be managed using the scan selection directives.

- **0** – Scan selection directives are not enabled.
- **1** – Scan selection directives are enabled.

This register is set using the STMC command.

This register is read using SCNS (Scan String), RDBACK0.

74.6.2.3.6 Delay Control (DLYC[1:0])

The Delay Control register provides a means for the external tool to delay the completion of an SP. This delay is essentially an external tool stall. This delay may be zero, one, two, or n TCKC periods. This delay is especially useful when the TCKC is sourced by the TS and a Standard-to-Advanced Protocol adapter is added to an external tool supporting only the Standard Protocol. It allows an external tool design additional time to receive TDO, advance

the external tool TAPC state, receive TMS and TDI information, and begin generation of a new SP payload. The DLYC register should be programmed before the use of the Advanced Protocol begins after the use of the Standard Protocol.

Set using the STMC command.

Read using SCNS, RDBACK0.

Bits 18-17 of the “Global Register.”

74.6.2.3.7 Ready Control (RDYC[1:0])

The Ready Control register defines behavior exhibited in the SP payload output bit frames when the STL stall opportunities (RDY bits) are included as control information. This register defines the number of additional bits inserted in these bit frames. These bits provide more time for a high-performance external tool to ascertain whether the TAP.7 controller is ready to complete the SP payload. The use of these bits makes a high-performance external tool design easier as input and output buffer delays play less of a role in limiting the TAP.7 controller performance. The RDYC register should be programmed before the use of the Advanced Protocol begins.

Set using the STMC command.

Read using SCNS, RDBACK0.

Bits 18-17 of the Global register state.

74.6.2.3.8 Test Reset (TRESET)

Asserts a test reset to the STL.

- **0** – The nSYS_TRST and SYS_TMS signals presented to the STL is not influenced by this bit.
- **1** – The nSYS_TRST presented to the STL is asserted and the SYS_TMS signal is a logic 1.

Set using the STC1 command.

The effects of the TRESET Register may be modified by a private TAP.7 controller register. When this private register is unimplemented or has a value of zero, the TRESET register causes the maximum initialization of the STL with nSYS_TRST and the SYS_TMS logic 1 value. Otherwise, the value of the private register may modify the function of the TRESET Register bit.

When the TRESET Register is a logic 1, the STL scan path appears broken as the state of CLTAPC remains *Test-Logic-Reset* while the EPUTAPC state progression continues. When the operation of the EPUTAPC and CLTAPC controllers is coupled, as with a T1 TAP.7, the state of these two TAPCs may be resynchronized to the *Run-Test/Idle* state by:

- An STC1 Command that sets the TRESET Register value to a logic 0
- A stay in the Run-Test/Idle state of two or more SYS_TCK periods immediately following the Update-DR state of this command

A failure to follow the guidelines is considered a programming error and will result in erroneous system operation. The EPUTAPC and CLTAPC states will not be synchronized as they progress through the state diagram.

74.6.2.3.9 Auxiliary Pin Function Control (APFC[1:0])

This register enables TDI and TDO alternate functions when 2-pin TAP.7 operation is selected.

- **0x** – TDI and TDO have JTAG function.
- **1x** – TDI and TDO have alternate function.

Set using STC2 command.

Read using SCNS, RDBACK0.

74.6.2.3.10 Conditional Group Member (CGM)

Used by STC1, STC2, STTESTM, and EXC3 – EXC0 commands as the “condition” for updating registers.

Set and cleared using MCM and SCNB commands.

Read using SCNS, RDBACK0.

74.6.2.3.11 Scan Group Candidate (SGC)

This register determines whether the STL is coupled when the *Run-Test/Idle* state is reached when scan formats other than JScan0 and JScan1 are used.

- **0** – Decoupling action.
- **1** – Coupling action.

Set and cleared using STMC, MSC, and SCNB commands.

Read using SCNS, RDBACK0.

74.6.2.3.12 Sampling Edge (SEDGE)

The Sampling Edge register defines the TCKC edge used to sample the TMSC input.

- **0** – Sample TMSC on the TCKC falling edge.
- **1** – Sample TMSC on the TCKC rising edge.

Set and cleared using STC1 command.

Read using SCNS, RDBACK0.

74.6.2.3.13 Read Back 0 (RDBACK0)

Non-transport related register information.

Read using SCNS command.

Unimplemented registers shall read back as “0”.

Table 1901. RDBACK register format

Bit	Width	Register Mnemonic	Name
4:0	5	SCNFMT	Scan Format
6:5	2	PWRMODE	Power-control Modes
7	1	FRESET	Functional Reset
8	1	TRESET	Test Reset

Table 1901. RDBACK register format (continued)

Bit	Width	Register Mnemonic	Name
9	1	SGC	Scan Group Candidate
		CGM	Conditional Group Member
11	1	SSDE	Scan Selection Directive Enable
13:12	2	TOPOL	Topology
14	1	SREDGE	Sampling Edge
16:15	2	DLYC	Delay Control
18:17	2	RDYC	Ready Control
20:19	2	APFC	Auxiliary Pin Functional Control
22:21	2	STCKDC	SYS_TCK Duty Cycle
31:23	11	Reserved	Read as a logic 0

74.6.2.3.14 Configuration Register 0 (CNFG0)

Table 1902. Configuration Register 0 Format

Field	Bit #	Width	Name	Description
CNFG (Class)	3:0	4	CLASS[3:0]	TAP.7 Class
				0000 – TAP.1 or TAP.7 Class T0
				0001 – Class T1
				0010 – Class T2
				0011 – Class T3
				0100 – Class T4
				0101 – Class T5
CNFG (Revision)	7:4	4	REV[3:0]	TAP.7 specification revision. Initial revision = all zeroes (0s)
				0000 – Reserved
				0001 – IEEE Std 1149.7-2008
				0010 – 1111 - Reserved
CNFG (Format)	11:8	4	CNFGFMT[3:0]	Configuration Register Format
				xxx1 – CNFG0[31:12] are implemented
				xx1x – CNFG1 implemented
				x1xx – CNFG2 implemented
				1xxx – CNFG3 implemented

Table 1902. Configuration Register 0 Format (continued)

Field	Bit #	Width	Name	Description
T1 Options	12	1	RDBKS	RDBACK supported
				0 – Unsupported
				1 – Supported
	13	1	TRESETS	TRESET supported
				0 – Unsupported
				1 – Supported
	14	1	FRESETS	TRESET supported
				0 – Unsupported
				1 – Supported
	17:15	3	PWRMODES[2:0]	PWRMODES[x] – Power Control Mode x supported where x = 0,1,2
				0 – Unsupported
				1 – Supported
	18	1	PCMR	PCMR – PM default POWRMODE value
				0 – No
				1 – Yes
T2 Options	19	1	DCAS	DCAS – Decouple at start-up
				0 – No
				1 - Yes

Table 1902. Configuration Register 0 Format (continued)

Field	Bit #	Width	Name	Description
T4 Options	21:20	2	SSCANS[1:0]	x0 – SScan1:0 Unsupported
				x1 – SScan1:0 Supported
				0x – SScan3:2 Unsupported
				1x – SScan3:2 Supported
	24:22	3	OSCANS[2:0]	xx0 – OScan3:2 Unsupported
				xx1 – OScan3:2 Supported
				x0x – OScan5:4 Unsupported
				x1x – OScan5:4 Supported
				0xx – OScan7:6 Unsupported
				1xx – OScan7:6 Supported
	25	1	APFCS	Auxiliary Pin Function Control supported
				0 – Unsupported
				1 – Supported
	26	1	TAPWIDS	TAP Width Default
				0 – 2-pin supported
				1 – 4-pin supported
	27	1	TAPWLCK	TAP Width Locked
				0 – Not locked
				1 – Locked
	28	1	STCKDCS	SYS_TCK Duty Cycle supported
				0 – Unsupported
				1 – Supported
Reserved	31:29	3	Reserved	Reserved, read as zero

74.6.2.3.15 Register reset values

Table 1903. Reset values of TAP.7 controller registers managed with commands

Register type	Width	Register mnemonic	Name	Values for reset type			
				0	1 – 3	4	5
Control	1	ECL	Exit Control Level	0	0	0	NC
	1	SUSPEND	Suspend	0	0	0	
	1	ZBSINH	ZBS Detect Inhibit	0	0	0	
	5	SCNFMT	Scan Format	DF	DF	DF	
	1	SSDE	SSD Enable	0	0	0	
	2	DLYC	Delay control	00	00	00	
	2	RDYC	Ready Control	00	00	00	
Options	1	TRESET	Test Reset	0	0	0	NC
	2	APFC	Aux. Pin Functional Cntl.	DP	DP	NC	
Select	1	CGM	Conditional Group Member	0	0	0	NC
	1	SGC	Scan Group Candidate	DSGM	DSGM	DSGM	
	1	SREDGE	Sampling Rising Edge	0	0	0	
Read-only	32	RDBACK0	Read-back 0	N/A	N/A	N/A	N/A
	32	CNFG0	Configuration Register 0				

NC = No Change

DF = Default Scan Format

DM = Default Power Control Mode

DCA = Default Coupling Action

DT = Default Topology

DP = Default Pin Function

74.6.3 EPU Commands

There are two command types:

- Two-Part Commands
- Three-Part Commands

Any number of two-part and three-part commands may be issued in any combination before the command control level is exited. If the command control level is exited before a two- or three-part command is completed, the command is aborted and becomes a NOP.

74.6.3.1 Two-Part Commands

TAP.7 controller commands are 10-bit values. The 10-bit commands are created by two consecutive DR-Scans when the control level is locked at the command control level (two).

The first DR-Scan provides a five-bit command opcode. The second DR-Scan provides a five-bit operand. These two scans are called the command part one (CP1) and command part two (CP2), respectively. A command created entirely with these two scans is called a two-part command.

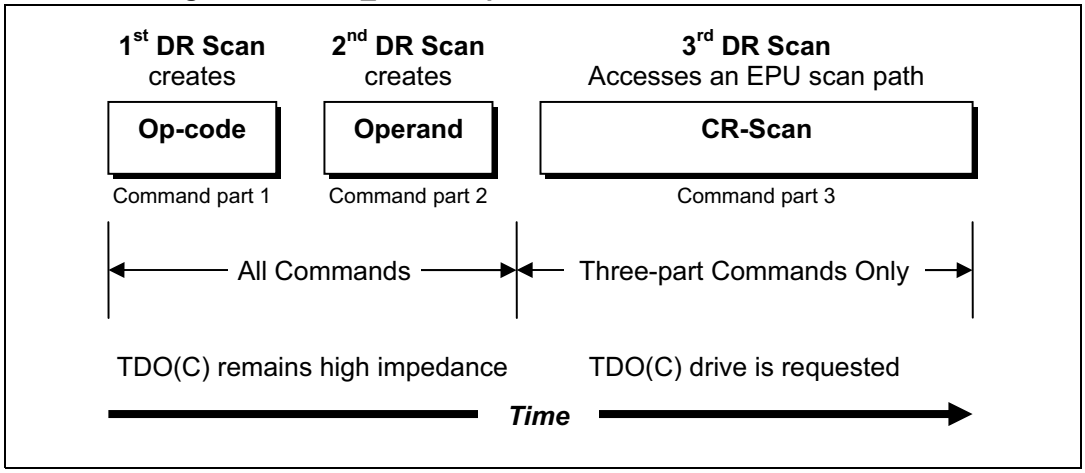
74.6.3.2 Three-Part Commands

Some commands are used to send or receive, or both, data values other than values embedded in the command's operand. These commands are called three-part commands. With these commands, CP1 and CP2 are followed by an additional DR-Scan to transport a data value to or from an EPU scan path. When used for this purpose, the DR-Scan is called a Control Register Scan (CR-Scan). The CR-Scan is a minimum of zero bits in length with there being no maximum length. The CR-Scan path is described in SECTION.

74.6.3.3 Command Sequence

The two 5-bit values concatenated to create the 10-bit command represent the number of *Shift-DR* states between the *Capture-DR* and *Update-DR* states of CP1 and CP2. The count within CP1 creates the MSBs of the command while the count within CP2 creates the LSBs of the command. The command is decoded when the *Update-DR* state of the second DR-Scan is reached. A determination as to whether the command is a two- or three-part command is made at this point. The command format is shown in *Figure 1969*.

Figure 1969. DR_Scan sequence used for command creation



The 10-bit command created by CP1 and CP2 determines the command's function and whether the command is a two- or three-part command. If the command is a two-part command, the function specified by the command is performed when command part 2 completes. If the command is a three-part command, the DR-Scan following CP2 performs a TAP.7 controller function designated by the 10-bit controller command. This DR-Scan is called a ***Control-Register Scan*** (CR-Scan) and can be any length. A CR-Scan has many of the attributes of the DR-Scan as it moves data between the external tool and TAP.7 controller and accesses various EPU scan paths. There are many two-part commands, but only 3 three-part commands, each having a special purpose.

74.6.3.4 Commands

There are 11 mandatory and 5 optional commands that can be grouped in classes as shown in the following list:

- Store – The operand is stored in a register or causes an action.
- Select – Controls participation of Scan and conditional command execution.
- Scan – Inputs and outputs with a CR-Scan.
- Enumerate – Controller ID allocation and de-allocation.
- Private – Commands available for chip-specific definition.

The definition of the commands with their operands is shown in *Table 1904*.

Table 1904. TAP.7 controller command list

Command class	Op-code count	Mnemonic	Name	TAP.7 class			
				1	2	3	4
Store	0x00	STMC	Store Miscellaneous Control	M	M	M	M
	0x01	STC1	Store Conditional 1-bit	M	M	M	M
	0x02	STC2	Store Conditional 2-bit	M	M	M	M
	0x03	STFMT	Store Format	-	M	M	M
	0x04	STDCST	Store Data Channel State	-	-	-	-
Select	0x06	MCM	Make Cond. Group Member	M	M	M	M
	0x07	MSC	Make Scan Group Candidate	-	M	M	M
Scan	0x08	SCNB	Scan Bit	M	M	M	M
	0x09	SCNS	Scan String	M	M	M	M
Enumerate(1)	0x0A	CIDA	Allocate Controller ID	-	-	M	M
	0x0B	CIDD	De-allocate Controller ID	-	-	M	M
Reserved	0x0C–0x1F	Reserved	Reserved	-	-	-	-

1. Enumerate commands are not currently supported by the CJTAG module.

74.6.3.4.1 Store Commands

Store commands are two-part commands that are available in both Star and Series Scan Topologies. The Store Miscellaneous Control (STMC) Command performs control functions that affect all TAP.7 controllers sharing a external tool connection. It stores values in one-bit and two-bit registers.

The Store Conditional 1 bit (STC1) and Store Conditional 2 bit (STC2) Commands store one-bit and two-bit values in registers, respectively. These commands operate two ways:

1. Unconditionally storing registers in all TAP.7 controllers or
2. Storing registers in only the TAP.7 controllers that have a logic 1 Conditional Group Member (CGM) Register value.

The Store Format (STFMT) Command unconditionally stores a five-bit Scan Format value.

The Store Data Channel State (STDCST) Command stores a five-bit value identifying the states supporting transport.

74.6.3.4.2 Select Commands

Select commands are two-part commands that are available in both Star and Series Scan Topologies. Their primary use is to select TAP.7 controllers of interest in a Star Scan Topology. These commands define the TAP.7 controller(s) participating in scans and the TAP.7 controllers that conditionally execute STC1 and STC2 Commands.

The Make Conditional Group Member (MCM) Command specifies which TAP.7 controllers execute store conditional commands as it manages the Command and Path Enable (CAPE) Register. This register also affects the execution of private commands and is utilized in EPU scan path selection.

The Make Scan Group Candidate (MSC) command manages the SGC register. This register determines whether the STL is coupled when the *Run-Test/Idle* state is reached when scan formats other than JScan0 and JScan1 are used. Select commands execute only when the CIDI Register is a logic 0, indicating the TAP.7 controller's CID is valid, and are treated as no operations otherwise.

74.6.3.4.3 Scan Commands

Scan commands are three-part commands that are available in both Star and Series Scan Topologies. The CR-Scan portion of Scan Bit (SCNB) and Scan String (SCNS) Commands is used to set and test bits (SCNB) or transfer strings of bits (SCNS). The commands may be used to access both public and private registers.

74.6.3.5 Command Definitions

74.6.3.5.1 Store Commands

74.6.3.5.1.1 Store Miscellaneous Control Command (STMC)

Table 1905. Store Miscellaneous Control

STMC		Store Miscellaneous Control				
Opcode	Operand	bbb		x	y	Description
00000	bbb x y	000	StateCtl	0	0	NOP
				0	1	ExitCmdLev(ECL)
				1	0	Exit/Suspend (SUSPEND) = 1
				1	1	ZBS Inhibit (ZBSINH) = 1
		001	ScanCtl	0	—	SGC = y
				1	—	CGM = y
		010	RdyCtl	-	—	RDYC = xy
		011	DlyCtl	-	—	DLYC = xy
		100	ScnFunc	0	—	SSDE = y
				1	0	Star-4 Topology Test
				1	1	Reserved
		101	Reserved	—	—	Reserved
		110		—	—	
		111		—	—	

74.6.3.5.1.2 Store Conditional 1-bit (STC1)

Table 1906. Store Conditional 1 bit (STC1)

STC1		Store Conditional 1 bit			
Opcode	Operand	c	bbb	CAPE	Description
00001	c bbb v	0	000	—	SREDGE = v
			001	—	Reserved
			010	—	TRESET = v
			011 – 111	—	Reserved
		1	—	0	No change
			000	1	SREDGE = v
			001	1	Reserved
			010	1	TRESET = v
			011 – 111	1	Reserved

74.6.3.5.1.3 Store Conditional 2-bit (STC2)

Table 1907. Store Conditional 2 bit (STC2)

STC2		Store Conditional 2 bit			
Opcode	Operand	c	bb	CAPE	Description
00010	c bb vv	0	00	—	Reserved
			01	—	STCKDC = vv
			10	—	APFC = vv
			11	—	Reserved
		1	—	0	No change
			00	1	Reserved
			01	1	STCKDC = vv
			10	1	APFC = vv
			11	1	Reserved

74.6.3.5.1.4 Store Scan Format (STFMT)

Table 1908. Store Scan Format (STFMT)

STFMT		Store Scan Format
Opcode	Operand	
00011	nnnnn	SCNFMT = nnnnn

74.6.3.5.2 Select Commands

74.6.3.5.2.1 Make Conditional Group Member (MCM)

This command operates as a no operation when the CIDI Register is a logic 1 (The Controllers CID is invalid). Otherwise it sets the CGM bit in a TAP.7 controller that has a valid CID and the CID matches the operand CID field (iiii). The CGM Register is also stored with the STMC and SCNB commands.

Table 1909. Make Conditional Group Member (MCM)

MCM		Conditional Group Member	
Opcode	**Operand**	**m**	**Description**
00110	m iiii	0	CGM bit of the non-targeted controller is cleared
		1	CGM bit of the targeted controller is set. CGM bit of a non-targeted controller is unaffected.

74.6.3.5.2.2 Make Scan Group Candidate (MSC)

This command operates as a no operation when the CIDI Register is a logic 1 (The Controllers CID is invalid). Otherwise it sets the SGC bit in a TAP.7 controller that has a valid CID and the CID matches the operand CID field (iiii). The SGC Register is also stored with the STMC and SCNB commands.

Table 1910. Make Scan Group Candidate (MSC)

MSC		Make Scan Group Candidate	
Opcode	**Operand**	**m**	**Description**
00111	m iiii	0	SGC bit of the non-targeted controller is cleared
		1	SGC bit of the targeted controller is set. SGC bit of a non-targeted controller is unaffected.

74.6.3.5.3 Scan Commands

74.6.3.5.3.1 Scan Bit (SCNB)

This is a 3-part command. The operand chooses the bit to be read or written by the CR-Scan.

Table 1911. Scan Bit (SCNB)

SCNB			Scan Bit		
Opcode	**Operand**	**CR-Scan**	**yyyyy**	**Description**	**R/W**
01000	yyyyy	-	0	Scan Group Candidate (SGC)	Write only
		-	1	Conditional Group Member (CGM)	Write only
		-	2	CNFG[0]	Read only
		-	3	CNFG[1] (not implemented)	Read only
		-	4	CNFG[2] (not implemented)	Read only
		-	5	CNFG[3] (not implemented)	Read only
		-	6	Functional reset requested (FRESETR)	Read only
		-	7	Series topology detect	Write only
		-	15 – 8	Reserved	Read only

74.6.3.5.3.2 Scan String (SCNS)

This is a 3-part command. The operand chooses the string to be read or written by the CR-Scan.

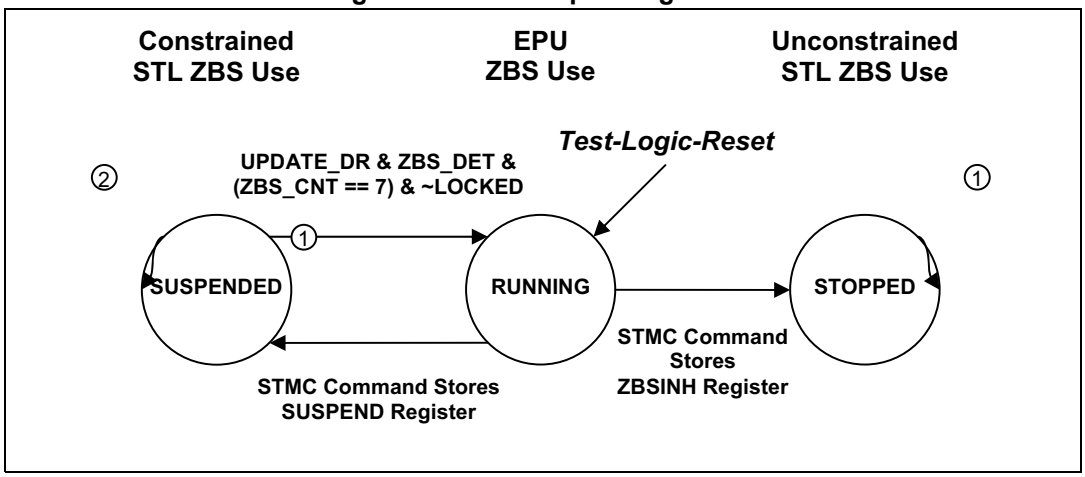
Table 1912. Scan String (SCNS)

SCNS			Scan String		
Opcode	**Operand**	**CR-Scan**	**yyyyy**	**Description**	**R/W**
01001	yyyyy	-	0	RDBACK[0] Register [31:0]	Read only
		-	1	RDBACK[1] Register [31:0] (not implemented)	Read only
		-	2	CNFG[0] Register [31:0]	Read only
		-	3	CNFG[1] Register [31:0] (not implemented)	Read only
		-	4	CNFG[2] Register [31:0] (not implemented)	Read only
		-	5	CNFG[3] Register [31:0] (not implemented)	Read only
		-	6 - 7	Reserved (read logic 0)	Read only
		-	31 - 8	Private – available for customization	

74.6.4 EPU operating states

The TAP.7 controller and STL share the use of ZBSs. The EPU has three operating states as shown in *Figure 1970*.

Figure 1970. EPU operating states



The values of the SUSPEND and ZBSINH Registers determine the type of operation as shown in *Table 1913*. The Test-Logic-Reset state sets the SUSPEND and ZBSINH Register values to a logic 0.

Table 1913. ZBS use state characteristics with both TAP.7 and other ZBS use

ZBSINH	SUSPEND	EPU operating state	ZBS use by:	Until
0	0	RUNNING	EPU	Command allocates to STL
0	1	SUSPENDED	STL	ZBS sequence allocates to EPU
1	0	STOPPED	STL	Test-Logic-Reset state allocates to EPU
1	1	Not possible		

74.6.5 System and EPU Paths

The System path is utilized when the external tool requires access to system (STL) resources. This path is used when either the SUSPEND Register is a logic 1 or the ZBS count is less than or equal to one. The EPU path is used otherwise.

The EPU scan path is constructed from five types of EPU scan paths:

- **Bypass** – A 1-bit scan path.
- **Bit** – Supports set and test bit operations (a 1-bit shift register).
- **String** – Support 32-bit reads or 32-bit writes.
- **Enumerate** – A 36-bit scan path that transports a TAPC address with T3 and above TAP.7s.
- **Auxiliary** – An n-bit scan path that transports 1 – n bits using a conventional IEEE 1149.1 style DR-Scan path using Control Levels 4 and 5.

74.7 APU (Advanced Protocol Unit) Operation

74.7.1 Overview

The Advanced Protocol Unit (APU) is placed between the EPU and the RSU to create the T4 TAP.7 controller. The APU delivers advanced capabilities with a number of operating modes (new scan formats) that utilize the Standard Protocol and the Advanced Protocol in both the narrow and wide T4 TAP.7 versions.

The APU enables scan transfers with two signals by serializing the information related to a TAPC state machine state. It provides a number of serialization formats to balance scan flexibility and scan performance. It multiplexes the use of the TMSC signal for T4 TAP.7 controller functions. The serialization is bypassed to provide T3 TAP.7 capability.

With a wide TAP.7, the chip architect may choose to implement the TDIC and TDOC signal functions as fixed functionality (the T3 TDIC and TDOC functions) or as programmable between the T3 TDIC and TDOC signal functions and an alternate function using a TAP.7 controller register. In this case, the default signal function may be either of the programmable choices. Programmable TDIC and TDOC functionality provides a means to fully utilize these signals in any scan topology as the T3 TAP.7 functionality is fully supported.

The entire EPU infrastructure (that is, ZBS detection, control-level management, and command generation) is utilized when either the Standard Protocol or the Advanced Protocol is used, as these TAP.7 controller attributes are controlled entirely from the TAPC state machine state progression. The EPU is unaware of the APU's existence. All T4 features are managed outside of the EPU.

The TAP.7 serialization schemes are specified with scan formats that complement the JScan 0-3 Scan Formats inherited from the T3 TAP.7. The scan formats are called advanced scan formats as they support two-signal operation.

Three groups of scan formats (MScan, OScan0-7, and SScan0-3) provide a number of options for the serialization of scan information for use in a Star-2 scan topology. The OScan and SScan Scan Formats are broken into two groups with the same function, one group has better performance but may only be used with an external tool sourced TCKC. The external tool chooses a scan format that matches the constraints imposed by the application, chip components, and possibly the external tool itself.

The MScan and OScan formats address Test and Debug use cases. The mandatory MScan and OScan0-1 formats emphasize flexibility over performance. The remaining optional Oscan2-7 formats emphasize performance over flexibility by not transmitting unneeded information such as TDI and TDO in non-shift TAPC state machine states.

The optional capabilities added with T4 and above TAP.7s change the bit sequences seen at the TAP.7 signals. A TAP.7 controller comprehends only the bit sequences generated by the use of the features it supports. It is never exposed to bit sequences it does not comprehend. A check for supporting an enabled feature is made before the feature is used (alters the bit sequences). When a T4 TAP.7 controller detects the use of an unsupported scan format, it halts its operation and places itself in a state where its operation may be resumed. This state is henceforth called "offline". A TAP.7 controller that is not offline is henceforth called "online". While offline the TAP.7 controller awaits the detection of a special signaling sequence called the Online Activation Protocol. This protocol sequence places an offline TAP.7 controller online and synchronizes the operation of all online TAP.7s.

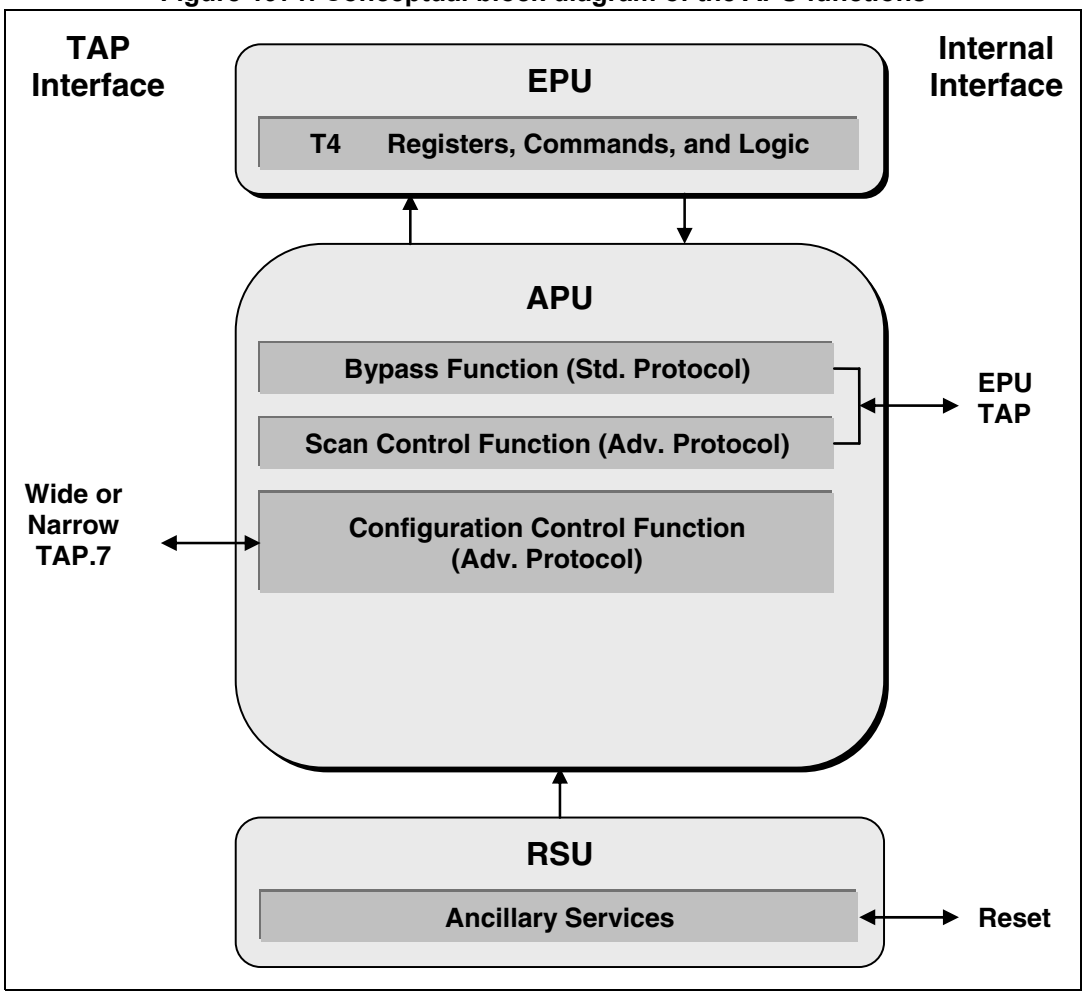
The Advanced Protocol Unit supports the stall of a transfer by both the external tool and STL at any TAPC state.

The APU provides the following functions:

- Bypass (for operation with the Standard Protocol)
- Scan control (for SP use)
- Configuration Control (for CP use)

The Scan Control, Configuration Control, and Transport Control functions are idled when the Standard Protocol is used as the Bypass Function connects the TCKC and TMSC pins to the EPU.

Figure 1971. Conceptual block diagram of the APU functions



The APU's internal interface includes a connection to EPU registers, the EPU's TAP interface, connections to chip-level data channels (when they are implemented), and reset signaling with the EPU. The APU's TAP interface provides mutually exclusive use of the TMSC pin for:

- Bypass Function
- Scan Control Function
- Configuration Control Function

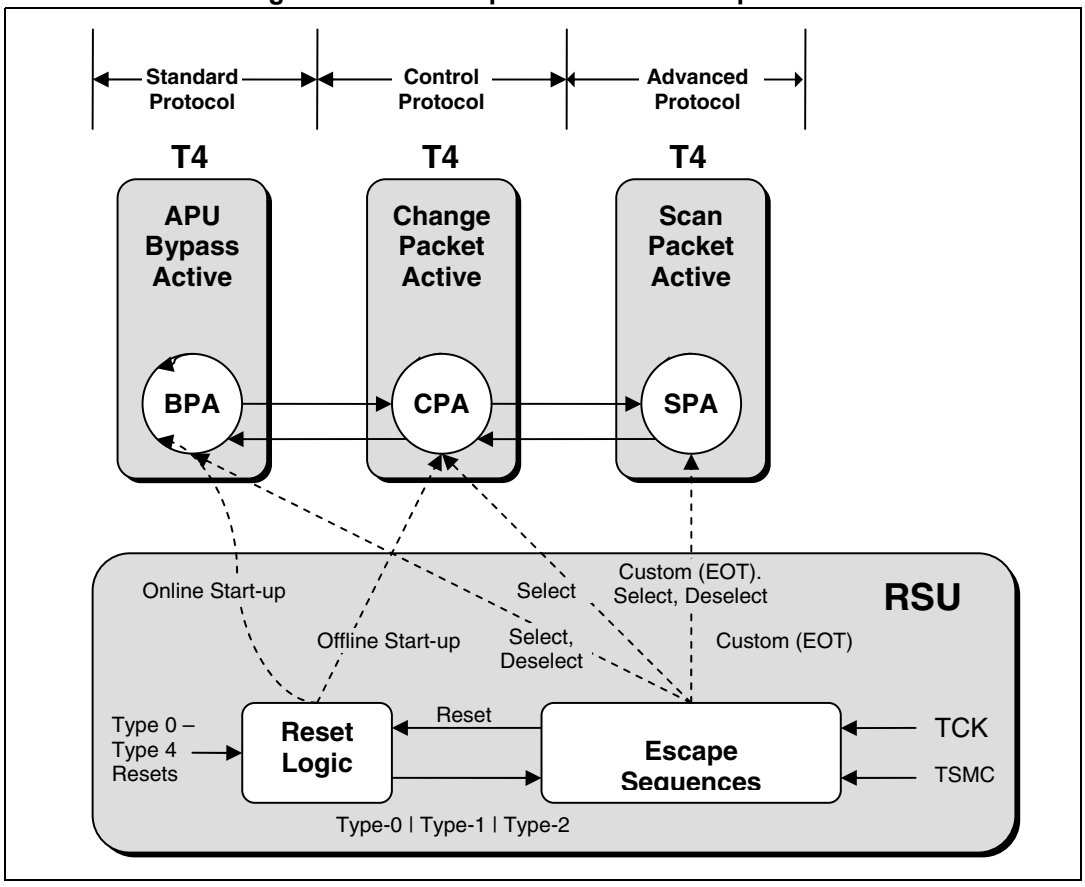
When the Standard Protocol is used, the APU's Bypass function connects the EPU directly to the TAP.7 pins. When the TDIC and TDOC pin functions are not available (narrow TAP.7 or TDIC and TDOC pin functions are an alternate function), the APU sources a logic 1 to the EPU inputs associated with these pins.

When the Advanced Protocol is used, the APU multiplexes the use of the TMSC pin between the other APU functions. Scan Control acts as a serial to parallel converter, translating the parallel information at the EPU's TAP interface into bidirectional serial transfers. Since the Scan Control is literally a protocol adaptor, the entire EPU infrastructure (ZBS detection, control-level management, and command generation) is utilized when either the Standard or Advanced Protocol is used, as these TAP.7 controller attributes are controlled entirely from the TAPC state progression.

74.7.2 Operation

The conceptual view of the APU operation is shown in *Figure 1972*.

Figure 1972. Conceptual view of APU operation



The APU operating states managing the use of the TAP pins and the ancillary (RSU) services use of the TMSC pin.

APU operating states:

- ***BPA*** – **B**y**P**ass **A**ctive
- ***CPA*** – **C**hange **P**acket **A**ctive
- ***SPA*** – **S**can **P**acket **A**ctive

74.7.2.1 BPA

The *BPA* state allocates the use of the TMSC pin to the Standard Protocol. This state is entered two ways:

- As a result of a TAP.7 controller reset when the TAP.7 controller starts up online.
- From the CPA state when a TAP.7 controller register write specifies the use of the Standard Protocol.

74.7.2.2 CPA

The *CPA* state allocates the use of the TMSC pin to a CP. This state is entered in one of three ways:

- As a result of a TAP.7 controller reset when the TAP.7 controller starts up offline.
- A value specifying the use of the Advanced Protocol is written to the SCNFMT Register when using the Standard Protocol.
- A TAP.7 controller register write occurs when using the Advanced Protocol.

74.7.2.3 SPA

The SPA state allocates the use of the TMSC pin to an SP. This state is entered when the SCNFMT Register specifies the use of the Advance Protocol when a CP ends.

74.7.3 Escape sequences

Escape-sequence detection provides services to the Reset Logic, Scan, Configuration Control, and Transport Control functions as outlined below:

- **Scan and Transport Control** – End of Transfer (EOT) escape sequence detected.
- **Configuration Control** – Synchronize Advanced Protocol (SAP) escape sequence.
- **Reset Logic** – Reset (RES) escape sequence detected.

An EOT escape sequence is used to increase the efficiency of OScan Scan formats.

An SAP escape sequence is used to place TAP.7 controllers online.

A RES escape sequence asynchronously causes generation of a Type-3 Reset. Following a RES escape sequence, the drive of the TMSC pin is inhibited so it is not driven when TCKC becomes a logic 0 immediately following the RES escape sequence.

74.7.4 Signal behaviors

The TCKC signal:

- The test clock

The TMSC signal:

- An input when the Standard Protocol is used
- Bidirectional when the Advanced Protocol is used

The TDI and TDO signals:

- Deleted with a narrow TAP.7
- One of two functions with a wide TAP.7
  - Fixed – as the TAP.7 TDIC and TDOC pin functions, or
  - Programmable – as the TAP.7 TDIC and TDOC signal functions or alternate functions with a TAP.7 controller reset establishing the default signal function.

74.7.5 APU Functions

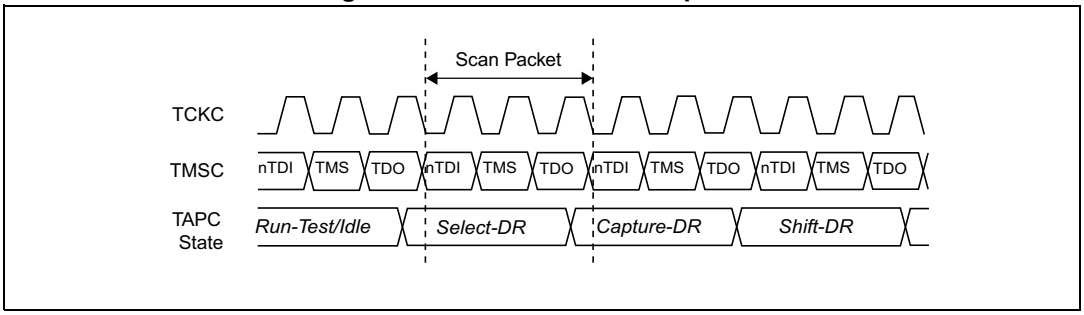
74.7.5.1 Bypass Active (BPA) Function

When the Standard Protocol is used, the APU connects the EPU inputs and outputs directly to the TAP.7 pins. If the TDIC and TDOC pin functions are not present (for example, a narrow TAP.7, or the TDIC and TDOC pins have an alternate function assigned to them), the APU provides a logic 1 to the EPU for unsupported pin functions and ignores the EPU's TDO output data.

74.7.5.2 Scan Packet Active (SPA) Function

The Scan Control function is encapsulated within the Scan Packet Active (SPA) state. With the Standard Protocol, the external tool and TAP.7 exchange the scan information associated with a single TAPC state machine state in one TCK period using the TMS(C), TDI(C), and TDO(C) pins. With the Advanced Protocol all or part of this information is exchanged serially within a Scan Packet (SP) as shown in *Figure 1973*. There is a one to one correspondence between SPs and TAPC state machine state changes, except when the SPA state precedes the CPA state. In this case the SP preceding a CP does not advance the TAPC state machine state.

Figure 1973. Scan Packet sequence

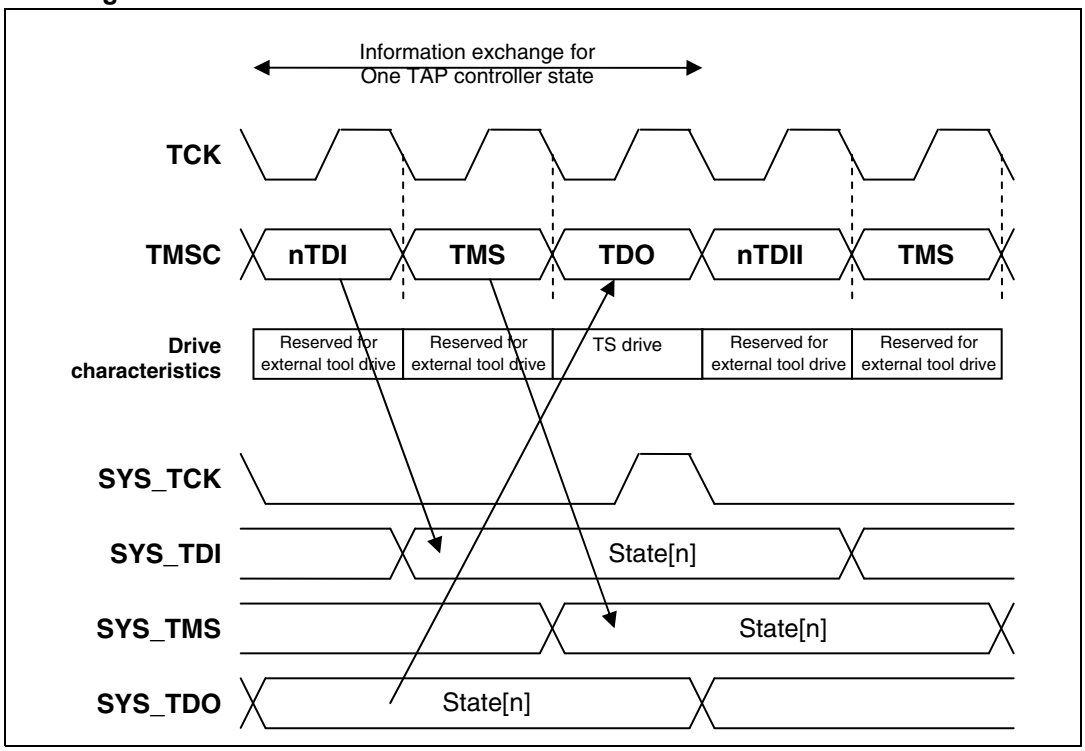


A comparison of the parallel transfer of TAP information with the Standard Protocol and the serialization and de-serialization of the TAP information using the TCKC and TMSC pins with the Advanced Protocol is shown in FIGURE. The IEEE 1149.1 signals between the EPU and STL are prefixed with SYS_.

The APU manages the serialization and de-serialization of the scan information. The APU converts the TDI and TMS information within a Scan Packet (SP) into the TMS and TDI information presented to the EPU. It converts the combination of the EPU TDO and EPU TDO_OE signals into the TDO information in the same SP. If the EPU indicates that the TDO pin drive is not enabled, logic 1 TDO data is created, otherwise the TDO data is the supplied by either the EPU or STL.

Information within a Scan Packet is first passed from the external tool to the TAP.7 followed by information passed from the TAP.7 to the external tool. In this example the scan format specifies the exchange of the TMS, TDI, and TDO TAPC information. Each TAPC state machine state takes three TCKC periods. Other scan formats specify the exchange of different mixes of information. Control information is added with the most flexible scan formats. Some scan formats send less information and consequently take fewer TCKC periods.

Figure 1974. Serialization and De-serialization of Scan Packet Information



In order to produce comparable TAPC state machine state rates with a TAP.1 and TAP.7 controller:

- Full-cycle timing between the external tool and TAP.7 allows operation at higher TCKC frequencies.
- The number of bits transferred to advance a TAPC state machine is minimized. (reaching as few as one for some scan formats).

The TAP.7 controller provides both full-cycle bit timing (falling TCKC edge to falling TCKC edge) and half-cycle bit timing (falling TCKC edge to rising TCKC edge), with a TAP.7 register used to define the type of timing used. Half-cycle timing is used as the default following a TAP.7 controller reset. The TCKC frequency is reduced to accommodate half-cycle timing.

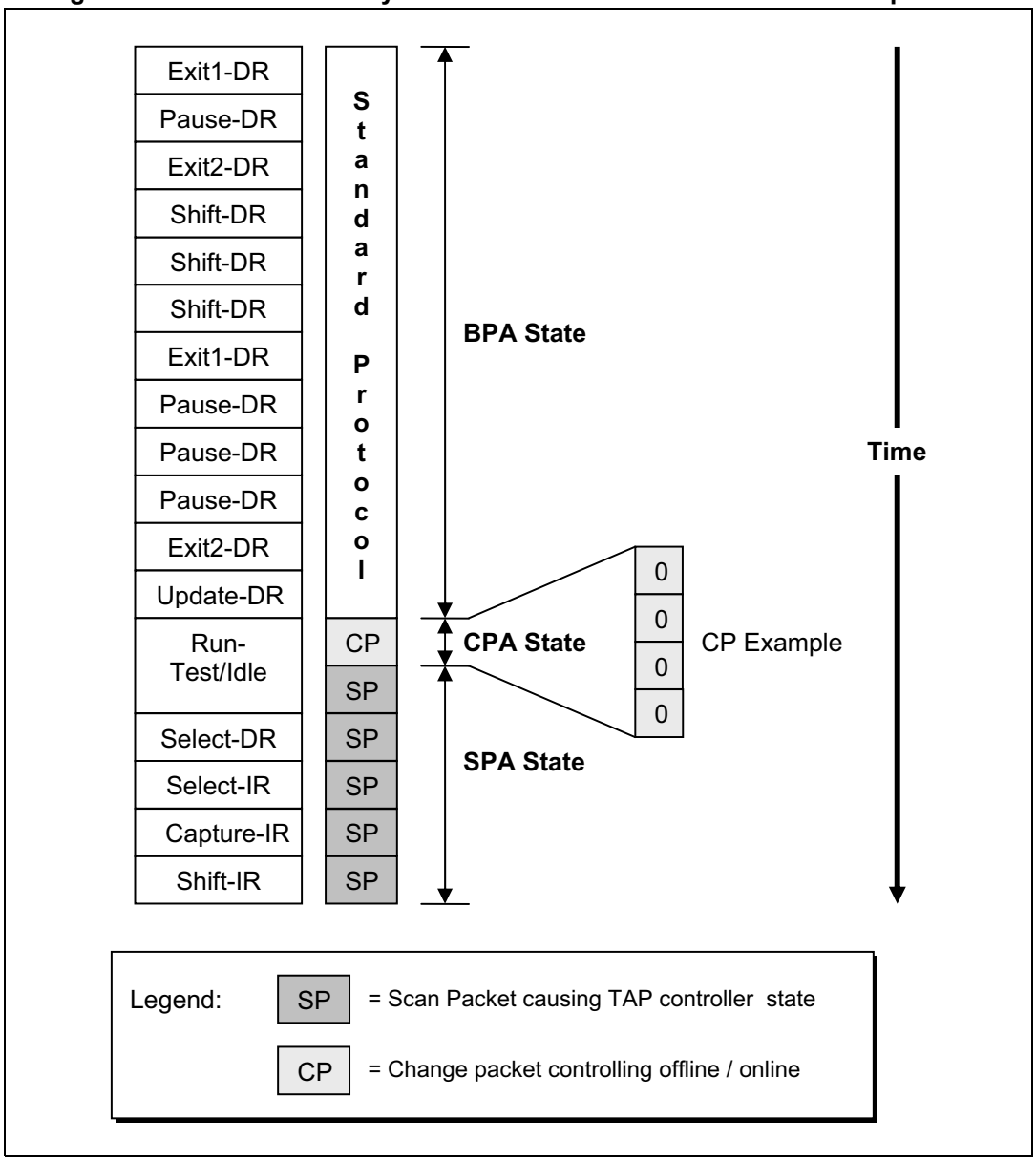
With full-cycle timing, the TMSC input and output data are both output with and sampled with the falling edge of the TCKC. This means that the entire TCKC period may be spent transferring a bit from the external tool to the TAP.7 or vice-versa. The TCKC frequency can therefore be maximized when full-cycle timing is used. With half-cycle timing, the APU samples the TMSC input data with the rising edge of the TCKC and outputs TMSC data with the falling edge of TCKC.

Minimizing the number of bits transferred per TAPC state machine state is affected by a number of factors. The information that must be transferred per TAPC state machine state varies for different test and debug applications. Also, with some test and debug applications, all TAPC state machine states do not require that the same amount of information be transferred. This is especially true with debug applications where specialized use of the TAP is more likely.

74.7.5.3 Control Process Active (CPA) Function

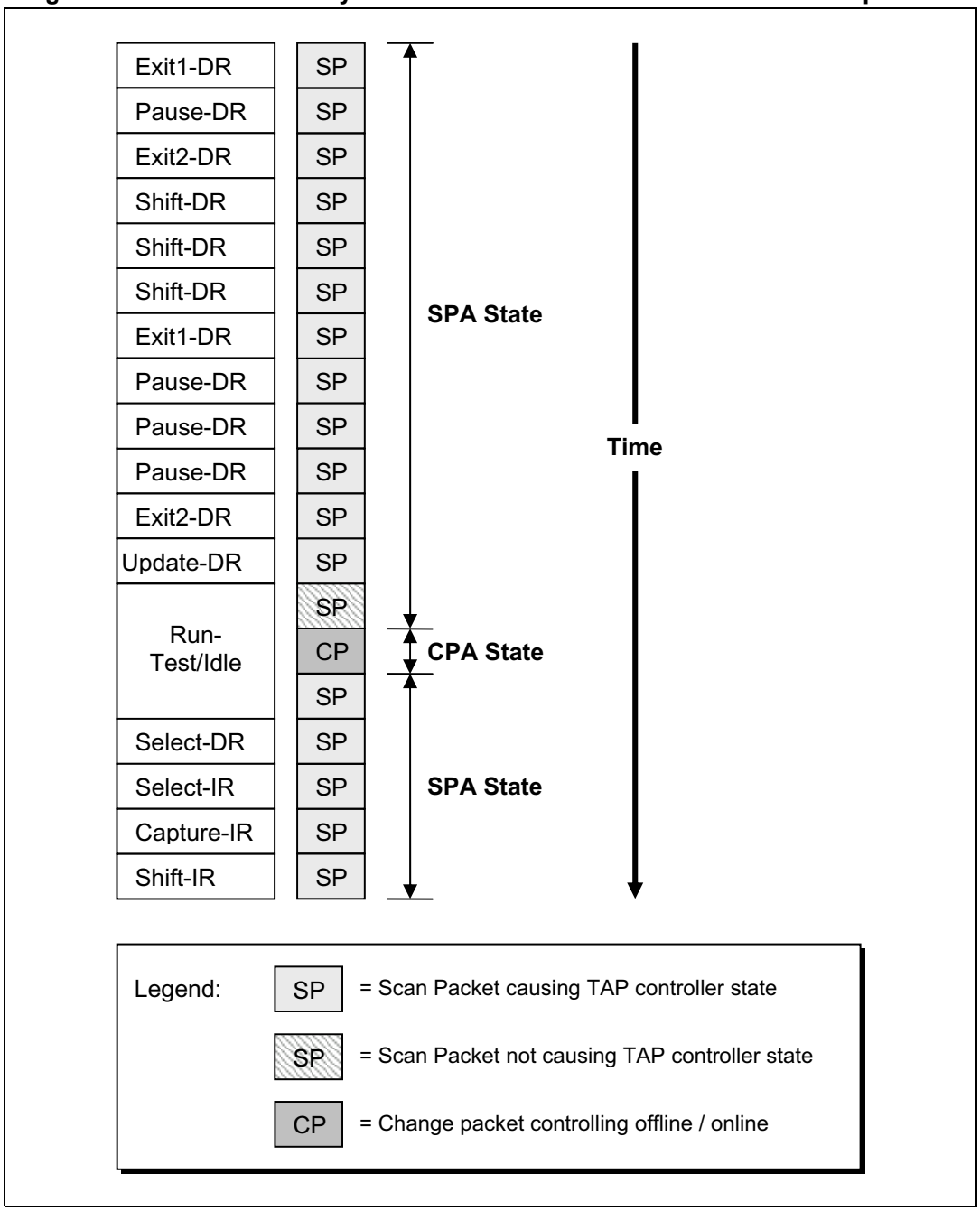
While using the Advanced Protocol, all TAP.7 configuration changes occur within the CPA state. All online and offline transitions also occur while in the CPA state. When entering the CPA state from the BPA or SPA state, the CPA state checks for the use of supported features affecting the Advanced Protocol. The CPA state is exited when the enabled features affecting the Advanced Protocol are supported.

Figure 1975. CPA State Entry from BPA State with Continued Online Operation



An SP preceding a CP does not advance the TAPC state machine state. This SP performs housekeeping functions that assure the contents of the SP preceding this SP have been fully utilized. This has significance when the SP contains control information that allows the STL to stall the completion of a SP.

Figure 1976. CPA State Entry from the SPA State with Continued Online Operation



When an unsupported feature is enabled, the TAP.7 controller is placed offline. This suspends the TAP.7 controller's operation at a point where its operation may later be synchronized to the remainder of the system. The APU stops advancing the EPU's and CLTAPC's TAPC state machine state while offline. It also suspends Data Channels activity while offline. The detection of the appropriate Online Activation Protocol while the TAP.7 controller is offline places the TAP.7 controller online and causes an exit from the CPA state to the SPA state. An offline to online transition initiates the use of a predefined set of mandatory features in all online TAP.7 controllers to assure they operate synchronously.

Figure 1977. CPA State Entry from the BPA State with Offline Operation

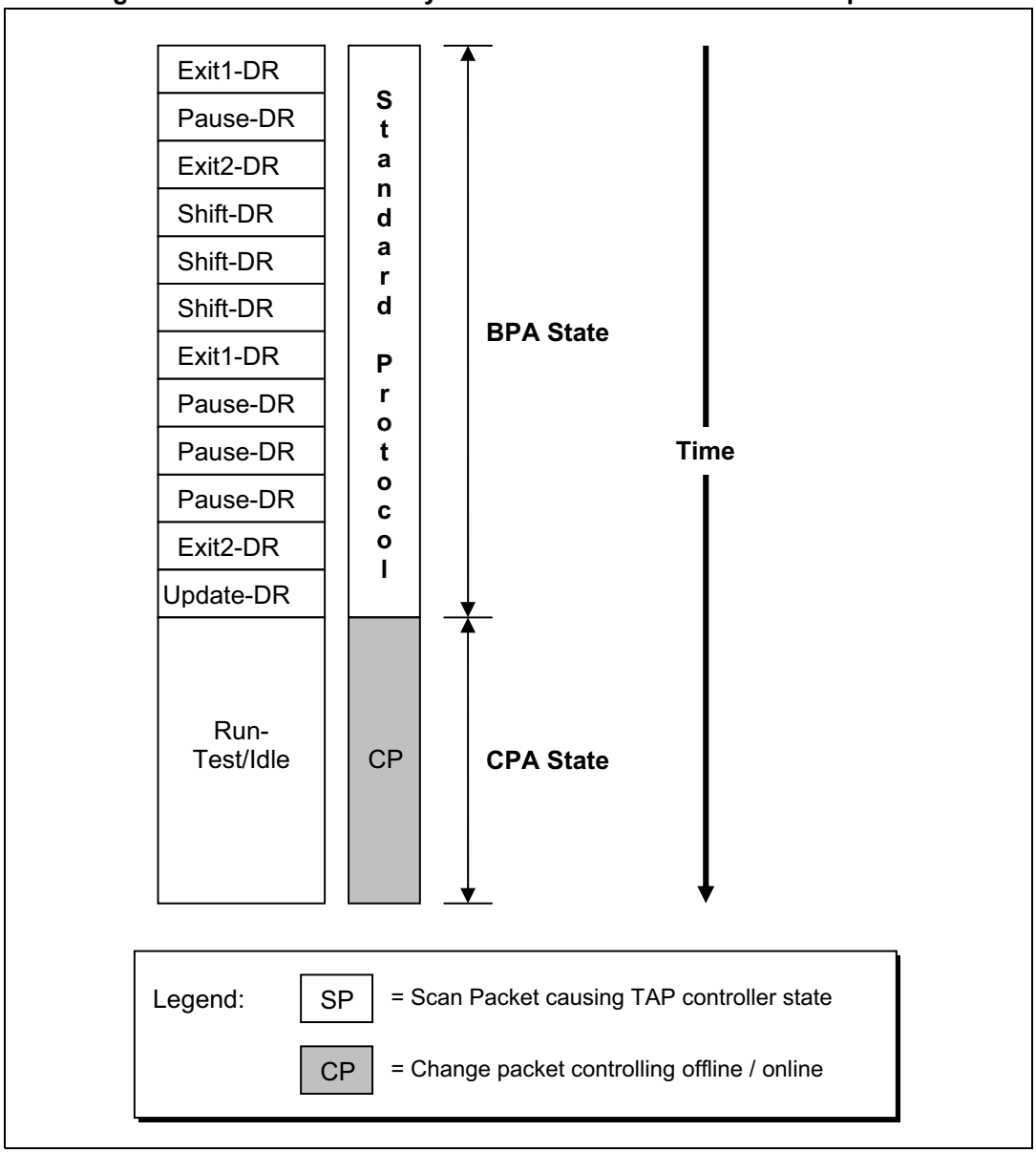
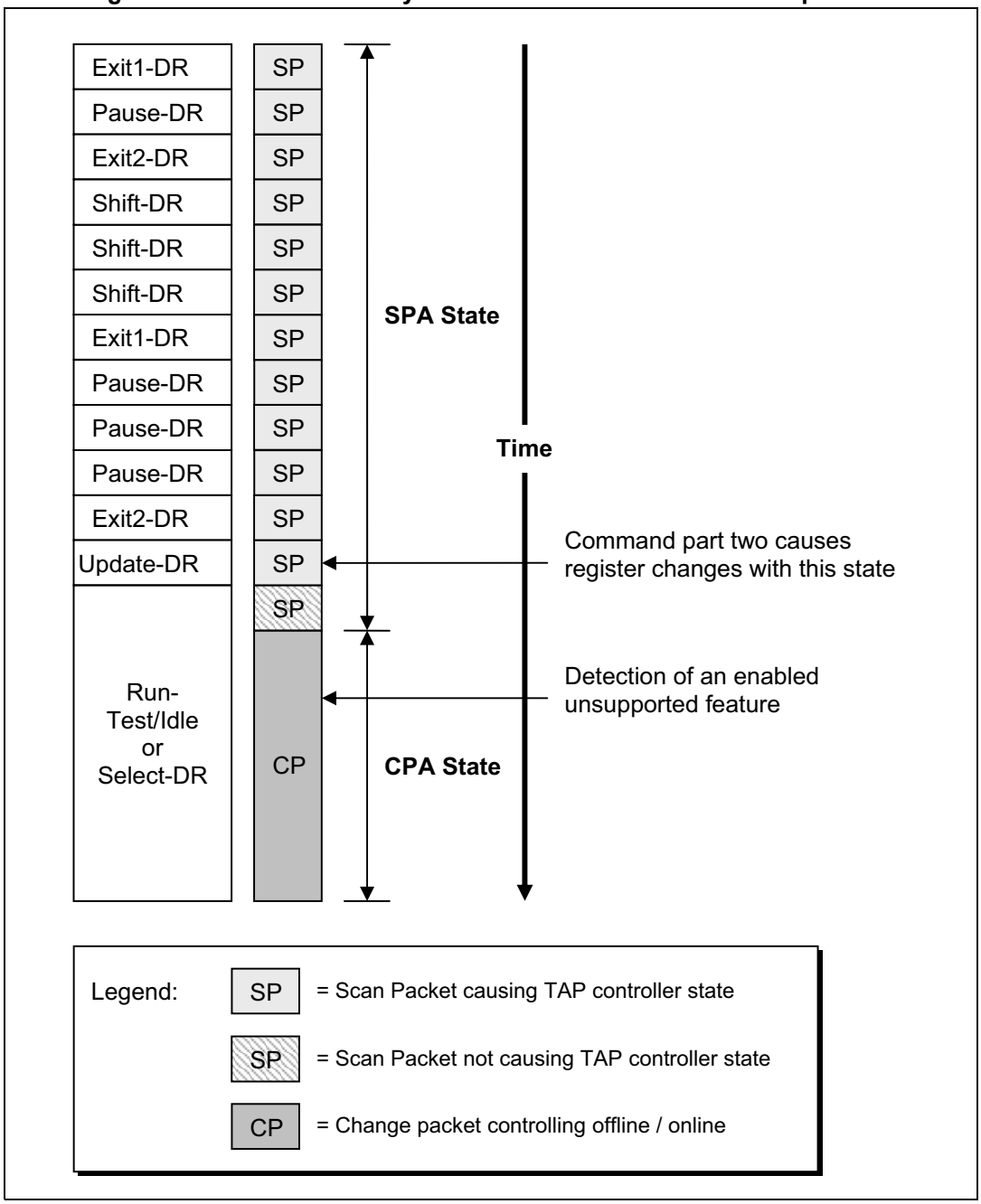


Figure 1978. CPA State Entry from the SPA State with Offline Operation



A TAP.7 controller reset places the TAP.7 controller offline when the offline at start-up option is used. From this point the TAP.7 controller awaits being placed online.

74.7.6 Configuration Change Packets (CP)

A Change Packet (CP) is composed of three element types:

- **Preamble** – A 1-bit value that is a don't care. This bit facilitates a Standard to Advanced Protocol change with some external tool implementations.
- **Body** – Two or more bits representing one or more CP directives:
  - **00** – CP_END
  - **01** – CP_NOP (extends the body by 1-bit)
  - **10** – CP_NOP (extends the body by 1-bit)
  - **11** – CP_RES (Generate a Type-3 TAP.7 controller reset coincident with the post amble bit)
- **Post amble** – A 1-bit value that is a don't care. This bi t facilitates and Advanced to Standard Protocol change with some external tool implementations.

A minimum-length CP has a Preamble, a 2-bit Body, and a Post amble. This can be as simple as four logic zero values.

74.7.7 Scan Packet

A Scan Packet defines the period of TMSC pin activity where control information exchanged by the external tool and TAP.7 controller affects the TAPC state, except when the SP precedes a Change Packet. The external tool should view the Scan Packet as:

- Exchanging only the necessary TAPC information (TMS, TDI, and TDO or some subset thereof).
- Causing an advance of the TAP.7 TAPC state when it is not followed by a Change Packet.
- Causing an advance of the STL TAPC state when it is not followed by a Change Packet and when the STL is coupled.

With this perspective, the external tool views Scan Packets as running a virtual TAPC state machine at the TAP.7 pins. The effects of each Scan Packet take effect when the Scan Packet completes. The real TAP.7 and STL controller state machine states may change state before or after the virtual state machine.

74.7.8 SP Format

A SP serially exchanges the TDI, TMS, and TDO or a subset thereof, between the external tool and TAP.7 controller. Control information may be added to this exchange. A SP has three parts:

- **Payload** – Data + Control information (all scan formats), the payload content is varied to trade-off performance and flexibility. RDY bit(s) within the payload are included with some scan formats to provide the STL a means to stall progression of the TAPC state.
- **Delay** – Control information (optional) providing separation of adjacent payloads in time. Delay elements support simple external tool implementations where a delay element is used to stall progression of the TAPC state while it develops a new SP following the completion of the prior payload.

The header elements, delay elements, and control information within the payload element is consumed by the TAP.7 controller. This and other control information within the SP payload is not visible to the EPU and STL TAPCs.

74.7.8.1 Payload Element Content

74.7.8.1.1 Input and Output Bit Frames

An input bit frame is sourced by the external tool and an output bit frame is sourced by the TAP.7 controller. The payload may contain an input bit frame, an output bit frame, or both. When an input bit frame arrives at the TAP.7 controller, the TDI and TMS information within it is presented in parallel to the TAP.1 controllers within the EPU and STL. The output bit frame begins when the input frame is completed. The payload is completed when the EPU and STL are ready to utilize this information and the TDO data needed to complete the output bit frame is available. This process is repeated for each SP.

With input bit frames, MScan and OScan scan formats have no control bits input bit frames.

With output bit frames:

- TDO information passed to the external tool is a logic 1 when the EPU indicates the TDO pin would be high impedance.
- RDY bits provide TAP.7 stall opportunities with MScan and some OScan formats.
- PC0 and PC1 bits are added to MScan scan formats to allow Wire-ANDed output.

74.7.8.1.2 Varying the content of input and output bit frames

The various scan formats (MScan, OScan0-7) provide a number of options for the serialization of scan information for use in a Star-2 scan topology. These scan formats vary the content of SPs to balance flexibility and performance for different use cases. The amount of information transferred per TAPC state is based on the constraints imposed by these use cases. The MScan and OScan scan formats determine the factors affecting the payload content as shown in *Table 1914*.

Table 1914. Factors determining the SP payload content

Scan format	Payload content determined by the:	
	TAPC state	Header
MScan	–	–
OScan	Yes	–

74.7.8.2 Delay Element Content

A delay element can be zero, one, two, or n bits in length. One and two bit delay elements are fixed length with neither the external tool or TAP.7 controller driving the TMSC pin. The TMSC pin is kept at the last value driven by either the external tool or TS. With an n-bit delay elements, the external tool drives the TMSC pin for the first n-1 bits and may drive the TMSC pin the nth bit. With an n-bit delay element the string of delay element bits is a series of directives virtually the same as those used by a CP (DLY_NOP, DLY_END, and DLY_RES).

74.7.8.3 Scan Formats

74.7.8.3.1 Overview

Scan formats utilizing the Advanced Protocol address different use cases. The amount of information transferred per TAPC state is based on the constraints imposed by these use cases. In one extreme two types of control information, the first allowing the stall of the completion of the Scan Packet, and the second supporting the Wire-AND of TAP.7 controller output are added to the TMS, TDI, and TDO information within the Scan Packet. In the other extreme, the external tool may minimize the information transferred for each TAPC state using its understanding of the application and transfer only the data of interest. In some cases the transfer of only one bit of information per TAPC is required. There are a number of scan formats that operate between these two extremes. The external tool chooses a scan format that best matches the performance/functionality needs of the application.

The scan formats added to specify the use of the Advance Protocol are listed below:

- **MScan** – Maximum flexibility.
- **OScan0-7** – Optimized for test and debug applications.

These scan formats provide seven different functions. Five factors have influenced the definition of these scan formats:

- Supporting the capabilities of the IEEE1149.x and IEEE 1532 Standards.
- Providing the STL and external tool an opportunity to stall the TAPC state progression.
- Providing a minimum pin count and scalable functionality.
- Maximizing the efficiency of scan operations utilized for applications debug operations.
- Providing a rich set of capability for high end applications debug.

The performance and flexibility of MScan and OScan scan formats are shown in *Table 1915* and *Table 1917*, respectively. The performance and flexibility of these scan formats are given subjective/approximate ratings in these tables. These ratings range from least to best. Use cases for these scan formats are also included in these tables.

74.7.8.3.2 MScan scan formats

The MScan scan format provides a universal scan function. It is the most flexible but lowest-performance scan format. The Scan Packets used by these scan formats are packed with data and control information and is the same for all TAPC states. This scan format accommodates the deployment of IEEE 1149.1 IP with non-compliant behavior within the STL. The STL is allowed to stall the TAPC state progression (to pace the arrival of Scan Packets with its ability to process them), and uses Wire-ANDed Output.

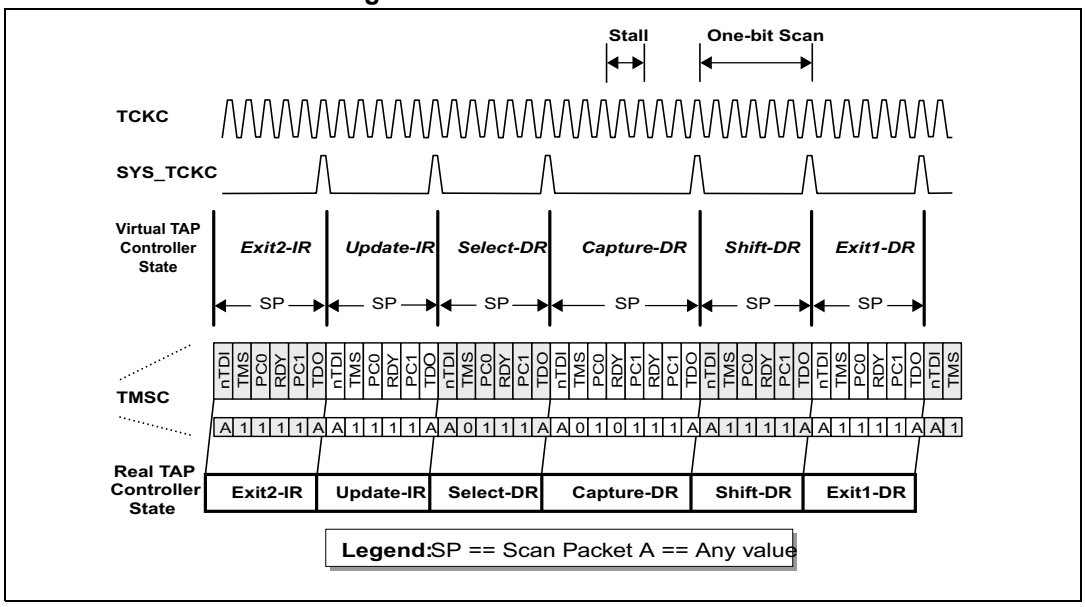
Table 1915. MScan scan format use case summary

Scan format	Supporting	Performance	Flexibility	TCKC source
MScan	STL Stalls	Least	Best	External tool or TS
	Test			
	Non-compliant behavior STL IP			
	Multi-chip debug communication			
	Controller ID Allocation			

Table 1916. MScan Scan Packet content

	Non-shift TAP controller states							Shift TAP controller states					
MScan	nTDI	TMS	PC0	RDY	PC1	TDO		nTDI	TMS	PC0	RDY	PC1	TDO

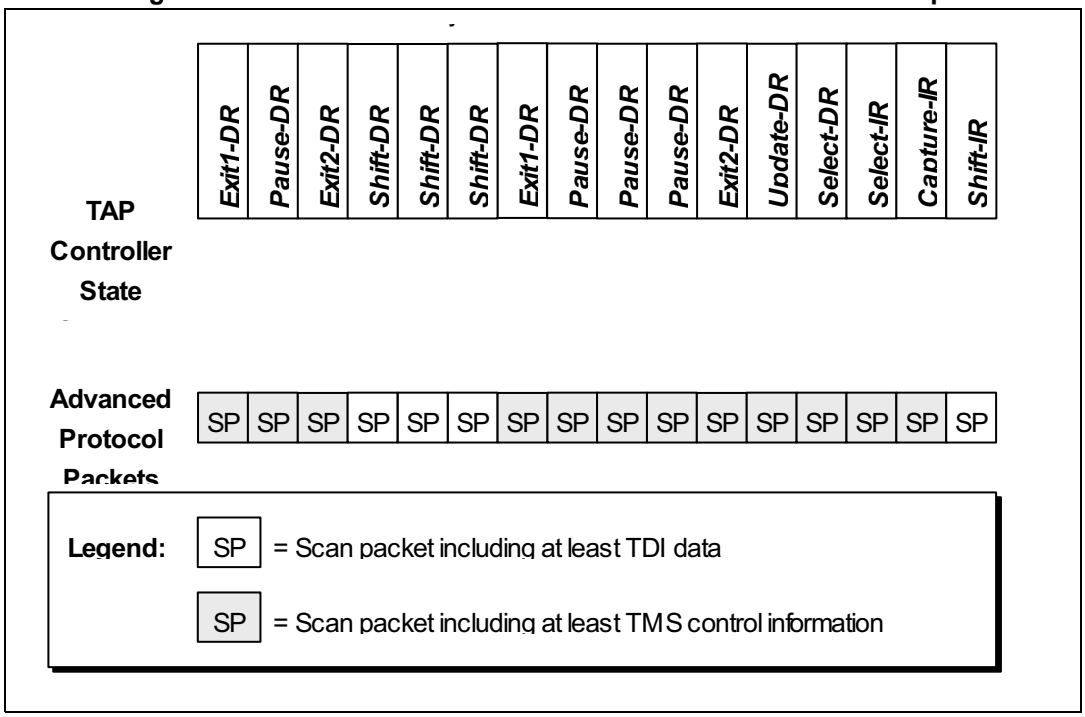
Figure 1979. MScan Transaction



74.7.8.3.3 OScan Scan Formats

The OScan scan formats provide optimization choices where no knowledge of the data content of the scan exchange is needed. The Scan Packet content of these scan formats is modulated by a combination of the scan format and the TAPC state for some OScan scan formats. For instance, TDI data and TDO data is not exchanged for non-shift TAPC states and included in shift TAPC states in certain OScan formats. The modulation of Scan Packet content with TAPC state varies by scan format.

Figure 1980. OScan Scan Packet content/TAPC state relationships



OScan scan formats replicate four functions in two forms:

- **OScan0 – OScan3:** Providing the best performance for a test clock source by the target system.
- **OScan 4 – OScan7:** Providing the best performance for a test clock sourced by the external tool.

The OScan4 – OScan7 scan formats are more efficient, as EOT escape sequences are used to identify an exit from the Shift-xR TAPC state. A Scan Packet associated with the Shift-xR TAPC state does not include a TMS bit. Instead, and EOT escape sequence overlaid on an nTDI bit causes an exit from shift TAPC state with no exit from this state occurring otherwise. A look at the 32-bit DR-Scan shows the impact of this optimization. A total of 96 bits are required when each Scan Packet associated with a Shift TAPC state when TMS is included in these Scan Packets while a total of 64 bits are required when TMS is not included in these Scan Packets. Assuming the EOT escape adds the equivalent of three bit periods, the number of TCK periods for this transfer is 67/96 or roughly 70% of the total without this optimization.

These scan formats may only be used with an external tool sourced TCKC as EOTs are utilized to improve its performance. The OScan0 – OScan3 Scan Formats always have a TMS bit included in the Scan Packet. The OScan scan format use case summary is shown in *Table 1917*.

Table 1917. OScan scan format use case summary

Scan Format	Supporting	Performance	Flexibility	TCKC Source
OScan0	Stalls	↓	Best	External tool
OScan1	Test		↑	
OScan2	Debug Performance			
OScan3	Debug Downloads (input only)	Best		
OScan4	Stalls	↓	Best	External tool or TS
OScan5	Test		↑	
OScan6	Debug Performance			
OScan7	Debug Downloads (input only)	Best		

Certain OScan optimizations are applied in a hierarchical manner.

- **Optimization I:** OScan1/OScan5 – RDY bit(s) and their trailing ONE bits are deleted from all Scan Packets.
- **Optimization II:** OScan2/OScan6 – Optimization I + nTDI and TDO bits are deleted from Scan Packets associated with non-shift TAPC states.
- **Optimization III:** OScan3/OScan7 – Optimization II + TDO bits are deleted from Scan Packets.
- **Optimization IV:** OScan4 – OScan7 – TMS bits are deleted from Scan Packets associated with shift TAPC states.

Table 1918. OScan Scan Format "Payload"

Scan Format	Non-shift TAP controller states					Shift TAP controller states			
OScan0	nTDI	TMS	RDY	TDO		nTDI	TMS	RDY	TDO
OScan1	nTDI	TMS	—	TDO		nTDI	TMS	—	TDO
OScan2	—	TMS	—	—		nTDI	TMS	—	TDO
OScan3	—	TMS	—	—		nTDI	TMS	—	—
OScan4	nTDI	TMS	RDY	TDO		nTDI	—	RDY	TDO
OScan5	nTDI	TMS	–	TDO		nTDI	—	—	TDO
OScan6	—	TMS	—	—	Note 1	nTDI	TMS	—	TDO
					Note 2	nTDI	—	—	TDO
OScan7	—	TMS	—	—		nTDI	—	—	—

Note 1: Following Capture-xR or Exit2-xR Scan Packet.

Note 2: Following Shift-xR Scan Packet.

74.8 Functional Description

This section combines the information given in the Ancillary Services, EPU Operation, and APU Operation sections into a single reference for performing common TAP.7 tasks.

74.8.1 Switching from Standard Protocol to Advanced Protocol

Upon exit of Type-0 to Type-4 reset, the CJTAG begins operation in Standard Protocol 4-pin mode. The following steps are required to move from Standard Protocol 4-pin mode to Advanced Protocol (either 4-pin or 2-pin) mode.

The first step required to move to the Advanced Protocol is to change the CJTAG control level to control level 2 via zero-bit scans (ZBS), as described in *Section 74.6.1.4: Control levels*.

After the control level is set to control level 2, TAP.7 commands can be executed as described in *Section 74.6.3: EPU Commands*. Execution of the Store Scan Format (STFMT) to set the scan format to any OSCAN0-7 or MSCAN format begins the switch from the Standard Protocol to the Advanced Protocol.

Once the STFMT command is executed selecting an Advanced Protocol scan format, the CJTAG transitions to the intermediate Control Protocol mode. The Control Protocol uses TMS inputs to control Configuration Change Packets as described in *Section 74.7.5.3: Control Process Active (CPA) Function* and *Section 74.7.6: Configuration Change Packets (CP)*. Loading a change packet body value of 00 (CP_END) results in the switch to the Advanced Protocol.

If Advanced Protocol 2-pin operation is desired, the APFC register should be written to a value of 10 or 11 prior to the execution of the STFMT command that selects an advanced scan format. The APFC register is written via the Store Conditional 2-bit (STC2) command described in *Section 74.6.3.5.1.3: Store Conditional 2-bit (STC2)*.

75 JTAG Data Communication (JDC)

75.1 Introduction

The JTAG Data Communication (JDC) module provides the capability to move register data between the IPS and JTAG domains. This facilitates communication between internal resources that access memory mapped register space and an external tool that accesses the JTAG port.

75.2 Overview

The JDC module consists of IPS accessible registers, JTAG accessible registers, and associated logic to coordinate movement of data from one register domain to another.

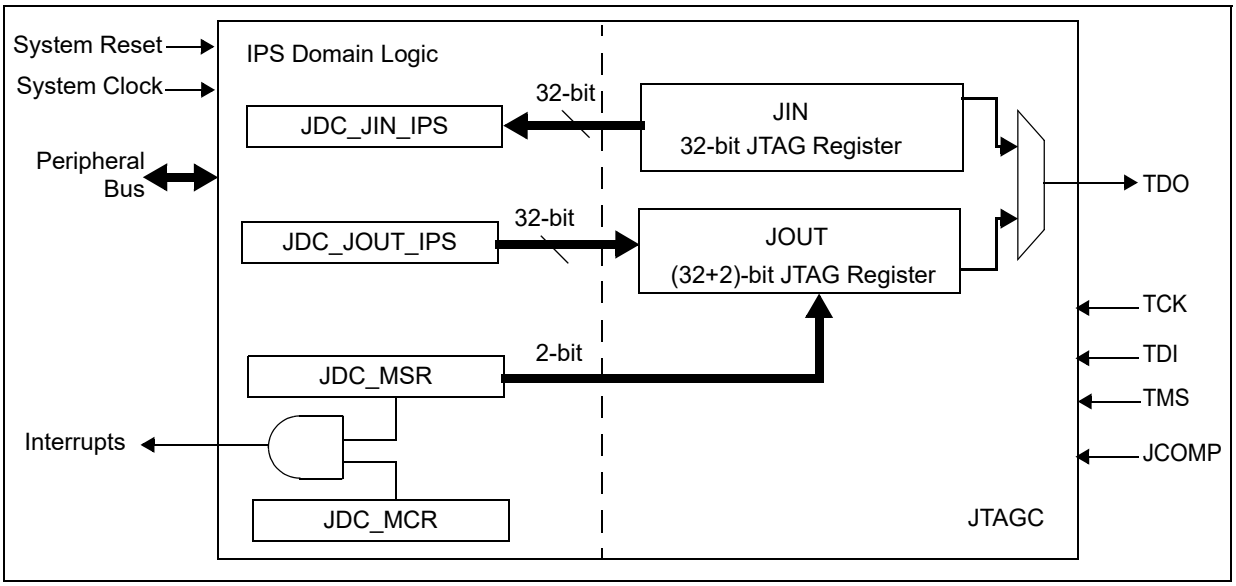
The JDC implements the following IPS data registers, occupying separate memory space:

- 32-bit memory mapped register that can be read or written via IPS (JDC_JOUT_IPS), and whose contents are ported out for capture into a JTAG register (JOUT) to be read via the JTAG port.
- 32-bit memory mapped register that can only be read via IPS (JDC_JIN_IPS), and whose contents are loaded from a JTAG register (JIN).

In addition to the data registers themselves, logic is implemented to indicate when new data has been shifted in via the JTAG port and is ready to be read from the JDC_JIN_IPS register, and when new data has been written to the JDC_JOUT_IPS register and is ready to be read via the JTAG port. The state of these flags is captured into a status register (JDC_MSR), and also provided to a JTAG register (JOUT) for tool visibility.

There is a single bus interface to port register data out to the JTAGC, and a single bus interface to port data in from the JTAGC. The architecture is shown in *Figure 1981*.

Figure 1981. JDC block diagram



75.3 Register definition

Table 1919 shows the JDC module registers. Four 32-bit registers are implemented. Only 32-bit accesses are valid. The effects of access that are not 32 bits are not defined.

Table 1919. JDC memory map

Address offset	Register	Access	Reset value	Location
0x0	Module Configuration Register (JDC_MCR)	R/W	0x0000_0000	*Section 75.3.1.1*
0x4	Module Status Register (JDC_MSR)	W1C	0x0000_0000	*Section 75.3.1.2*
0x8	JTAG Output IPS Data Register (JDC_JOUT_IPS)	R/W	0x0000_0000	*Section 75.3.1.3*
0xC	JTAG Input IPS Data Register (JDC_JIN_IPS)	Read Only	0x0000_0000	*Section 75.3.1.4*

75.3.1 Register descriptions

75.3.1.1 Module configuration register (JDC_MCR)

Figure 1982 shows the JDC_MCR. The JDC_MCR is used to enable the interrupt outputs of the JDC. This register is reset by system destructive reset.

Offset: 0x00 Access: User read/write

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

Figure 1982. Module Configuration Register (JDC_MCR)

The JDC_MCR is described in *Table 1920*.

Table 1920. JDC_MCR register field descriptions

Field	Description
15 JIN_IEN	JIN Interrupt Enable. 1 Setting of JDC_MSR[JIN_INT] bit asserts the JIN interrupt 0 Setting of JDC_MSR[JIN_INT] bit does not assert the JIN interrupt
31 JOUT_IEN	JOUT Interrupt Enable. 1 Setting of JDC_MSR[JOUT_INT] bit asserts the JOUT interrupt 0 Setting of JDC_MSR[JOUT_INT] bit does not assert the JOUT interrupt

75.3.1.2 Module status register (JDC_MSR)

Figure 1983 shows the format of the JDC_MSR. The JDC_MSR holds the JTAG register status and interrupt bits. This register is reset by system destructive reset.

Offset: 0x04 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	JIN_RDY	0	JIN_INT	0	0	0	0	0	0	0	0	0	0	0	0	0	JOUT_RDY	0	JOUT_INT
W																w1c																w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1983. Module Status Register (JDC_MSR)

The JDC_MSR is described in *Table 1921*.

Table 1921. JDC_MSR register field descriptions

Field	Description
13 JIN_RDY	JIN Ready (read only). 1 Set when new data is written to the JDC_JIN_IPS register 0 Cleared upon software read of JDC_JIN_IPS contents via IPS
15 JIN_INT	JIN Interrupt. 1 Set when new data is written to the JDC_JIN_IPS register 0 Cleared by writing logic 1
29 JOUT_RDY	JOUT Ready (read only). 1 Set when new data is written to the JDC_JOUT_IPS register 0 Cleared upon tool read of JOUT register via JTAG port
31 JOUT_INT	JOUT Interrupt. 1 Set when JOUT_RDY bit is cleared by tool reading JOUT register 0 Cleared by writing logic 1

75.3.1.3 JTAG output IPS data register (JDC_JOUT_IPS)

The JDC_JOUT_IPS register holds data written via IPS. The JDC_JOUT_IPS contents are ported out and captured into the JOUT register to be read via the JTAG port. *Figure 1984* shows the format of the JDC_JOUT_IPS register. This register is reset by system destructive reset.

Offset: 0x08 Access: User read/write

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

Figure 1984. JDC_JOUT_IPS register

The JDC_JOUT_IPS register is described in *Table 1922*.

Table 1922. JDC_JOUT_IPS register field descriptions

Field	Description
0:31 DATA	Contains JOUT_IPS data.

75.3.1.4 JTAG input IPS data register (JDC_JIN_IPS)

Data written to the JTAG input data register (JIN) via the JTAG port is held in the JDC_JIN_IPS register, where it can be read via IPS. *Figure 1985* shows the format of the JDC_JIN_IPS register. This register is reset by system destructive reset.

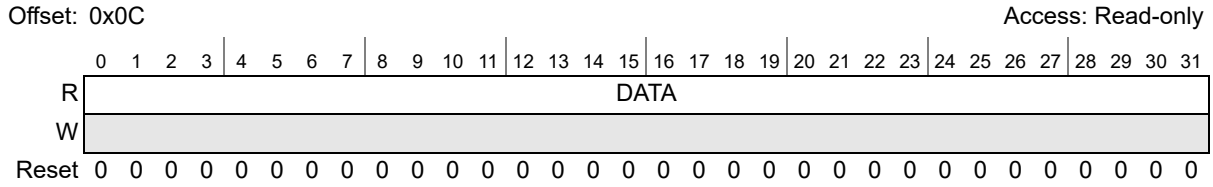


Figure 1985. JDC_JIN_IPS register

The JDC_JIN_IPS register is described in *Table 1923*.

Table 1923. JDC_JIN_IPS register field descriptions

Field	Description
0:31 DATA	Contains JIN_IPS data.

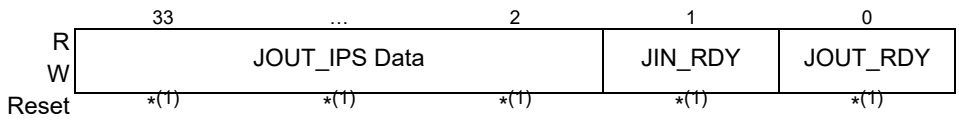
75.3.2 Non-Memory mapped register definition

The JDC also implements two JTAG accessible registers that are not memory mapped. One JTAG register is used to shift in data to be placed in the JDC_JIN_IPS register. The other JTAG register captures data from the JDC_JOUT_IPS register and ready status from the JDC_MSR to be shifted out via the JTAG port.

75.3.2.1 JTAG output data register (JOUT)

The JOUT register captures data from the JDC_JOUT_IPS register upon execution of the JOUT_READ JTAG instruction. It also holds the JDC_JIN_RDY and JDC_JOUT_RDY status bits. *Figure 1986* shows the format of the JOUT register. This register is reset by system destructive reset and JTAG reset.

Figure 1986. JOUT JTAG register



1. The reset value of the JOUT register is 34'b0.

The JOUT register is described in *Table 1924*.

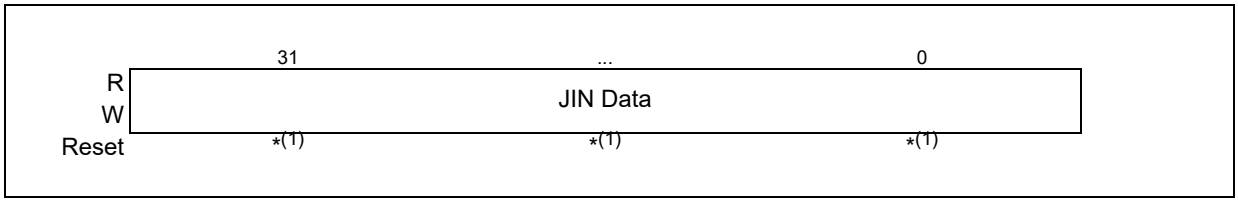
Table 1924. JOUT JTAG register field descriptions

Field	Description
JOUT_IPS Data	Data value from JOUT_IPS register
JIN_RDY	State of JIN_RDY bit from JDC_MSR
JOUT_RDY	State of JOUT_RDY bit from JDC_MSR

75.3.2.2 JTAG input data register (JIN)

Data is written to the JIN register via JTAG during execution of the JIN_WRITE JTAG instruction. The JIN data is later captured in the JDC_JIN_IPS register to be read via IPS. *Figure 1987* shows the format of the JIN register. This register is reset by system destructive reset and JTAG reset.

Figure 1987. JIN JTAG Register



1. The reset value of the JIN register is 32'b0.

The JIN register is described in *Table 1925*.

Table 1925. JIN JTAG register field descriptions

Field	Description
JIN Data	Contains data to be captured in JDC_JIN_IPS register upon exit of Update-DR state when executing WRITE_JIN JTAG instruction.

75.4 Functional description

The JDC module provides the ability to shift in data via the JTAG port and capture that data into a memory mapped register that can be accessed via IPS. It also provides the ability to capture data written to a memory mapped register into a JTAG shift register for output via the JTAG port. An overview of the module functionality is described below.

- A software write to the JOUT_IPS register sets the JOUT_RDY flag bit, indicating new data is available to be read from the JOUT register via the JTAG port. The state of the JOUT_RDY bit is reflected in the MSR and also ported out to the JOUT register. A JTAG read of the JOUT register via execution of the JOUT_READ instruction with a JOUT_RDY bit whose value is logic 1 indicates the register contains new data. The JOUT_RDY flag bit is cleared upon exit of the Capture-DR JTAG state during execution of the JOUT_READ instruction. Clearing the JOUT_RDY bit indicates to software that a new data value can be written to the JOUT_IPS register.
- A JTAG write to the JIN register via execution of the JIN_READ JTAG instruction updates the contents of the JDC_JIN_IPS register upon exit of the Update-DR state. An update of the JDC_JIN_IPS register sets the JIN_RDY flag bit, indicating new data

is available to be read via IPS. The state of the JIN_RDY bit is reflected in the MSR register and also ported out to the JOUT register. The JIN_RDY flag bit is cleared upon software read of the JDC_JIN_IPS register. A JTAG read of the JOUT register with a JIN_RDY value of logic 0 indicates new data can be written to the JIN register.

75.4.1 IPS register access

IPS access is available to the registers described in *Section 75.3.1*.

75.4.2 JTAG register access

Refer to the JTAGC documentation for information on how to access the JTAG registers.

75.4.2.1 JDC block instructions

The JDC block implements the IEEE 1149.1-2001 defined instructions listed in *Table 1926*. This section gives an overview of each instruction; refer to the IEEE 1149.1-2001 standard for more details. All undefined opcodes are reserved.

Table 1926. JTAG instructions

Instruction	Code[4:0]	Instruction Summary
Reserved	00001	Factory debug reserved
JOUT_READ	00010	Selects JOUT data register. Data from JOUT_IPS is captured into JOUT data register upon entry to Capture-DR state while JOUT_READ is active.
JIN_WRITE	01110	Selects JIN data register. Data from JIN is captured into JIN_IPS upon exit of Update-DR state while JIN_WRITE is active.
BYPASS	11111	Selects bypass register for data operations
Reserved	All other opcodes	Decoded to select bypass register

75.5 Use case example

A summary of how the JDC can be used to interact with the Hardware Security Module (HSM) to perform challenge/response password authorization is as follows:

1. After exit of reset, all registers are at their reset values. JIN_RDY bit value of 0 indicates the tool may write data to the JIN register. JOUT_RDY bit value of 0 indicates that software may write data to the JOUT_IPS register.
2. The JDC_MCR register is programmed to enable JOUT and JIN interrupts, or not.
3. Once the tool is ready to start the authorization process, it writes a value to the JIN register via execution of the JTAGC JIN_WRITE instruction.
4. Software monitors the state of the JIN_RDY bit or enables the JIN interrupt and uses the interrupt assertion as a trigger to start the authorization process.
5. Once the authorization process is started, software writes the first JOUT value to the JOUT_IPS register. This sets the JOUT_RDY bit value to 1.
6. Following the initial tool write to the JIN register, the tool polls the JOUT register to determine when software has provided a challenge word. Upon reading the JOUT register with JOUT_RDY bit set, the tool records the data value to be used to generate

the appropriate response. The read of the JOUT register clears the JOUT_RDY bit. The clearing of the JOUT_RDY bit also sets the JOUT_INT bit and asserts the JOUT interrupt if the JDC_MCR[JOUT_IEN] is set.

7. With the JOUT_RDY bit cleared, software may now provide the next challenge word.
8. Once the tool has generated a response word, it uses the value of the JIN_RDY bit from the JOUT register read to determine when it can perform a write to the JIN register to provide the response.
9. The HSM can provide additional challenge words and read back the corresponding response words as needed, repeating the process.

Simply providing a password without using the challenge/response protocol can be done in a similar way. The tool provides 32-bits of the password at a time with each write to the JIN register, and monitors the JIN_RDY bit value of the JOUT register to determine when the next JIN register value can be written. There is no need for software to write the JOUT_IPS register in this case.

76 Sequence Processing Unit (SPU)

76.1 Introduction

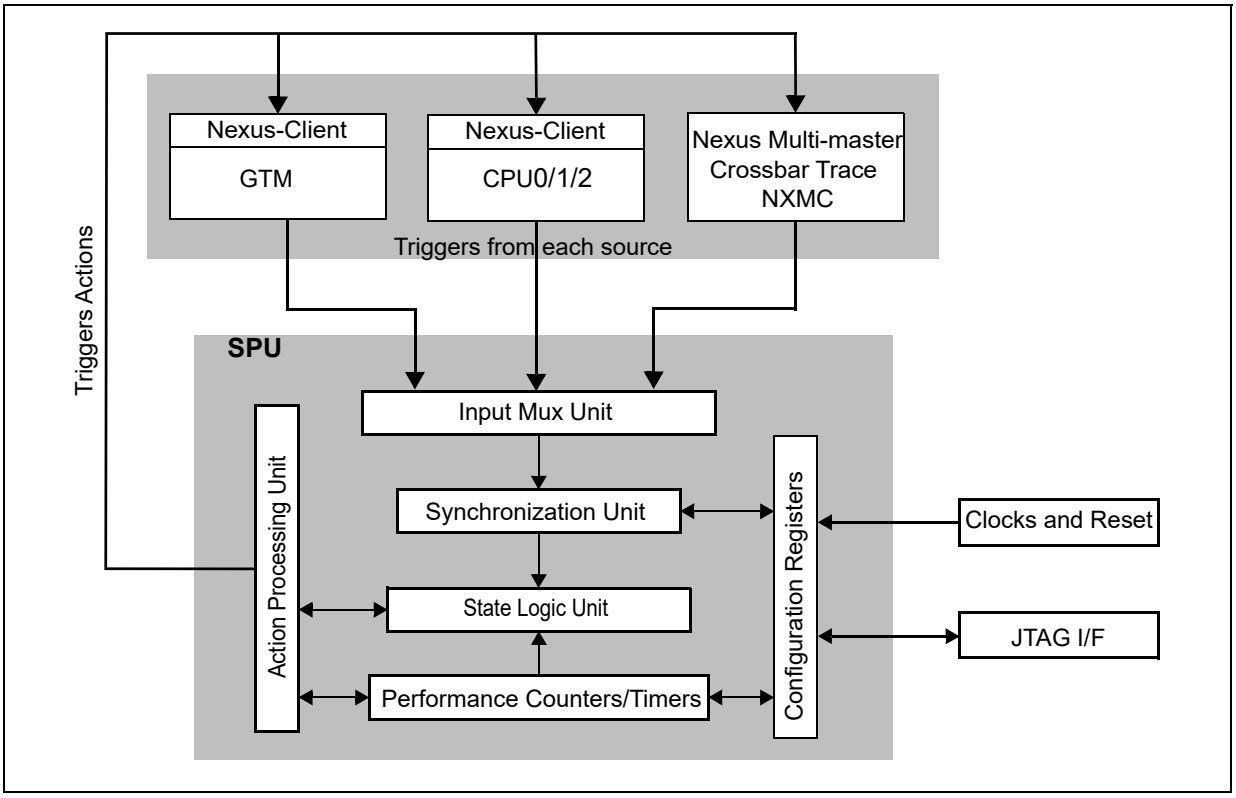
The Sequence Processing Unit (SPU) provides an on-device trigger functions similar to those found on a logic analyzer. In a complex SoC, performance monitor functions are available at two levels: system level and CPU level. Complex trigger and system performance monitor functions are implemented in the SPU module. System level performance monitor functions are integrated into the SPU complex trigger logic, and thus allow counting and timing of any debug trigger combinations supported by the SPU.

Various clients such as the cores, GTM, or crossbar switch, generate watchpoints and triggers when operating in their debug mode. The SPU collects these triggers (such as interrupt occurrence, address watchpoint, and so on) and uses them as conditions to sequence through states, with resultant actions (such as start/stop trace, start/stop counter, capture timebase, and so on). The SPU provides the capability to create complex debug triggers. By configuring each of the events to trigger specific actions, complex logic-analyzer-like behavior can be achieved, providing vital real-time visibility and debug ability of the activities of the system.

76.1.1 SPU block diagram

Figure 1988 shows a top-level block diagram of the SPU.

Figure 1988. SPU block diagram



76.1.2 Overview

The SPU is able to generate complex debug events, based upon input events from sources (CPUs, GTM, and system bus clients) throughout the SoC. The SPU has the ability to create a state machine to trigger a debug action. Single or multiple actions can be triggered by a state machine, which results in debug events being created. There are a selection of timers and counters available to assist comparison events.

Complex trigger support is implemented in the State Logic Unit (SLU). There are numerous inputs coming from different clients. Users can choose 63 inputs from a set of 209 inputs at the level one Mux. There are actually 64 inputs to the Muxes, but one of the inputs is permanently connected to a high level so users can only choose 63 of the inputs. Of these 63 inputs, users can select 16 inputs for each SLU, shown in *Figure 2027*.

One SLU consists of four input AND gates, the output of which are ORed together. One of the inputs of each AND gate has the option to invert it. Also, the output of each AND gate can be optionally inverted. Finally, the output of each SLU has the option of inversion. The block diagram for one SLU state is shown in *Figure 2028*. The SPU includes eight SLUs states.

Users can utilize these states to form a configurable state machine. This allows users to join states together with the if-then-else operations to create a sequence. Each state in a sequence has the ability to move to another state based on the true condition from the state logic, and the ability to route to another state based on the a false condition from the state logic. This is shown in *Figure 2029* and *Figure 2030*. At a given time, a maximum of four sequences are allowed. Each sequence can have one state or maximum of eight states. Each state can only be used in one unique sequence. The number of states in all the sequences cannot exceed eight. Each state has the ability to trigger one to four actions based on a true/false condition in the active sequence.

An example of typical complex trigger for the SPU would be: if FunctionA is entered, monitor CPU writes to VariableB. If CPU writes a value to VariableB which is 0x100, then enable trace for the GTM and halt CPU.

The SPU consists of the following sub-blocks:

- Input Mux Unit: receives the numerous triggers from various clients and reduces them down into a user selected set. The muxing unit selects sixteen inputs for each sequencing SLU from a set of 63 static inputs. This selection is done using two levels of muxing. At the first level of muxing, users select 63 out of 209 inputs. At the second level of muxing, out of these 63 inputs, one input can be selected for each of the four input AND gates. There are 128 64 $\times$ 1 Muxes to cater the needs of 128 inputs of all the eight states. This input mux selection logic is shown in *Figure 2027*.
- State Logic Units (SLU): responsible for performing operations on the selected input events to produce desired events. Each SLU has a control register that manages how the various inputs are combined to produce a desired event. Only one input of each four input AND gate can be independently inverted. The combined event can also be optionally inverted and, coupled with DeMorgan's law, allows for generalized signal combinations. This is shown in *Figure 2028*.
- Action Processing Unit: takes an action request from a SLU state and converts this into one or multiple actions. The action unit can be programmed to trigger various actions in response to each processed event from the SLU unit. The user can define the actions associated with each state from each SLU.
- Counters/Timers: the SPU implements 16 32-bit timer/counter functionality. Timer or counter selection is done via a configurable register in the SPU. Each counter/timer

has its own control register. A 32-bit comparator value can optionally be compared against the timer/counter. These registers are accessible via JTAG.

76.1.3 Features

The features of the SPU are as follows:

- clients:
  - 3 CPUs
  - 1 GTM
  - 2 System Bus clients (NXMC)
- Input watchpoints/triggers selection
- Input debug triggers for instruction execution selection
- Input interrupt selection
- Sequences and states criteria
  - Up to 4 user programmable number of active/simultaneous sequences
  - Up to 8 user programmable states that can be split between all four sequences
  - Each state can only be used in one unique sequence
  - A sequence can consist of a single state or any number of states up to 8
- State logic
  - Each state logic consist of sum of products (SOP)
  - SOP consist of four 4-input AND gates, and one 4-input OR gate
  - Optional inversion on one input of each AND gate of each state logic
  - Optional inversion on output of each AND gate
  - Optional inversion on output of OR gate of each state
  - States are joined together with if-then-else statement to create a sequence
- SLU events
  - Up to 8 user programmable derived events/actions
  - Any state can optionally trigger of 1 to 4 actions based on a true condition
  - Any state can optionally trigger of 1 to 4 actions based on a false condition
  - User selection of trigger actions generated in response to each programmed trigger
- SPU counters/timers
  - 16 32-bit counters/timers
  - Timer/counter configurability selection via a register
  - Start/increment when used as timer/counter respectively
  - Reset when used as timer/counter respectively
  - Counter value compare match
  - Counter overflow event status
  - Generation of a greater than condition against a 32-bit comparator value (no support for range condition from single counter)
  - Range support using two counters
  - Use of counter/timer registers to generate counted or timed triggers
  - Use of counter/timer registers to capture system performance statistics

  - Each timer/counter value has the ability to be inserted into the Nexus trace stream upon an SPU action
  - Access to timer/counter values via JTAG
- Prescaler for timer mode only
  - Configurable prescaler of /1, /4, /16 or /32 of the clock for timer operation inside the SPU
- Asynchronous interfaces
  - SPU to/from CPU (fastest clock)
  - SPU to/from NXMC
  - SPU to/from GTM
  - SPU to/from NAR
  - SPU to/from DCI
  - SPU to/from JTAGM
- Interfaces
  - CPU
  - GTM
  - System bus
  - NAR
  - DCI
- Configuration registers: 32-bit double word aligned configuration registers (byte and halfword access not supported)
- SPU actions: SPU block triggers various actions based on the derived events, these actions are defined in *Section 76.2*.

76.2 SPU actions

Sequence actions describe what is triggered. A sequence state may only have a GoTo action to determine the next sequence state. The complete set of actions are as follows:

- Start/stop trace for a source
- Start/stop/increment counter/timer counter/timer[0–15]
- Reset counter/timer[0–15]
- Halt1: single signal (EVTI0) that can be used to halt a pre-configured DCI group1
- Halt2: single signal (EVTI1) that can be used to halt a pre-configured DCI group2
- Single interrupt for INTC to interrupt a CPU: de-assert (cleared) via JTAGM
- Generate a pulse on the EVTO pin: output pulse over EVTO1 and EVTO2
- Start or Stop Performance Counters (PMC) for a CPU source: Start/Stop CPUn
- Optional trace group 1 (Start/Stop Trace a pre-configured set of trace signals): four (DTM,PTM,WTM,OTM) independent start/stop trace groups can be formed. One or more clients (CPUn and Nexus) can be part of a particular group.

Note: GoTo is not part of the Action list described above but it is implemented in the state logic to move from one state to another. If there is nothing defined for GoTo, it signifies the end of the sequence.

In the condition where a state (State n) updates a counter and the next state in the sequence (State n+1) evaluates the value of the same counter (for example, to compare if the counter is less-than a certain value), the update to the counter (from State n) is not applied before the counter evaluation in the next state (State n+1). This results in the old value of the counter being evaluated (in State n+1), rather than the desired updated value. This is due to hardware limitations of the SPU as the update to the counter does not complete until the clock cycle after the evaluation of the next state. This limitation is only applicable in Div1 mode and not in Div4 mode.

To workaround this hardware limitation, a delay should be inserted between the update to a counter and the evaluation of this counter. A dummy state can be used between the counter update (State n) and the counter evaluation (State n+1). A dummy state is a state with any input, that on both then/else conditions, the action proceeds to the next state (State n+1) where the counter evaluation occurs. The insertion of this dummy state provides sufficient time for the update of the counter to be applied, in time for the counter evaluation. The same outcome is also possible in a complex sequence with multiple states by scheduling states so that counter update and counter evaluation are not performed on consecutive states in a sequence.

76.3 Operation mode

76.3.1 SPU enable/disable

By default, the SPU is in disabled mode. In this state:

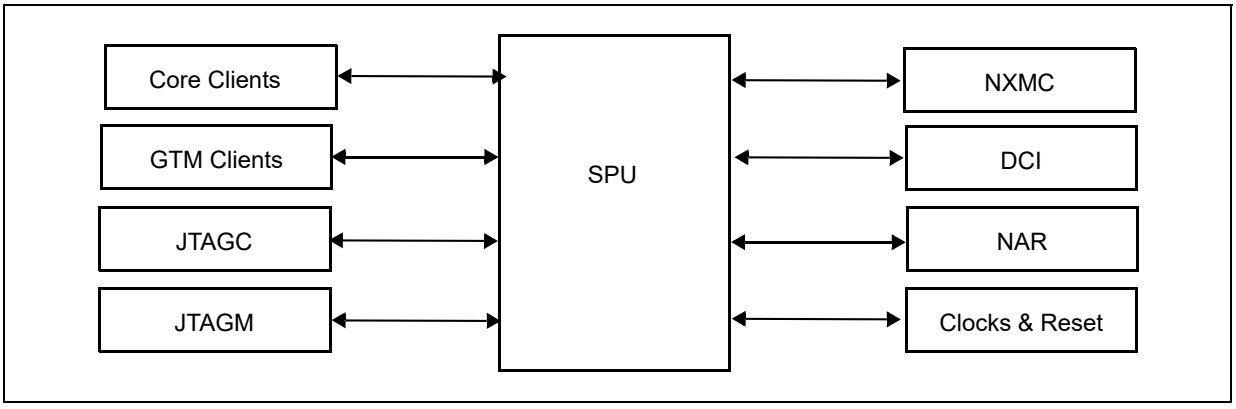
- All events/inputs are disabled such that no counting or other actions occur.
- All the SPU control registers are accessible.

The user must configure the control registers before enabling the SPU via the SPU Enable/Disable register.

76.4 SPU module interface

The SPU is connected to many clients such as the cores, GTM, system bus, NAR and Nexus clients. The SPU also interfaces to the DCI and the JTAG modules as shown in *Figure 1989*.

Figure 1989. SPU module interface



76.5 Register definition

The SPU contains 32-bit aligned registers (byte and half-word access are not supported). All configuration registers in the SPU are JTAG registers such that these registers are accessible via JTAG interface only. Write access to reserved locations have no impact while read access to reserved locations always return 0. These registers have Nexus reset and JTAG reset control.

Note: The SPU does not check for correctness of the programmed values in the registers and programmers must ensure that correct values are being written.

Table 1927 lists the memory mapped registers which control the operation of the SPU. There are two instructions for the SPU registers:

- Instruction to enable the SLU (first set of registers with offsets 0x01–0x49) registers is 4'h3.
- Instruction to enable the Counter Control Unit (second set of registers with offsets 0x01–0x72) is 4'h2.

Table 1927. SPU register summary

Offset	Register	Location
0x01	Level1 Mux Selection 0 (L1SEL0)	*Section 76.5.1.1.1*
0x02	Level1 Mux Selection 1 (L1SEL1)	*Section 76.5.1.1.2*
0x03	Level1 Mux Selection 2 (L1SEL2)	*Section 76.5.1.1.3*
0x04	Level1 Mux Selection 3 (L1SEL3)	*Section 76.5.1.1.4*
0x05	Level1 Mux Selection 4(L1SEL4)	*Section 76.5.1.1.5*
0x06	Level1 Mux Selection 5 (L1SEL5)	*Section 76.5.1.1.6*
0x07	Level1 Mux Selection 6 (L1SEL6)	*Section 76.5.1.1.7*
0x08	Level1 Mux Selection 7 (L1SEL7)	*Section 76.5.1.1.8*
0x09–0x0A	Reserved for future use	
0x0B	Input Trigger Level Detection 2 (ITLD2)	*Section 76.5.1.1.9*
0x0C	Input Trigger Level Detection 3 (ITLD3)	*Section 76.5.1.1.10*
0x0D	CPU0 Processor Except Vector Prefix (C0PEVP)	*Section 76.5.1.1.11*
0x0E	CPU1 Processor Except Vector Prefix (C1PEVP)	*Section 76.5.1.1.12*
0x0F	CPU2 Processor Except Vector Prefix (C2PEVP)	*Section 76.5.1.1.13*
0x10–0x11	Reserved for future use	
0x12	CPU0 Interrupt Priority Selection (C0PIS)	*Section 76.5.1.1.14*
13	CPU1 Interrupt Priority Selection (C1PIS)	*Section 76.5.1.1.15*
14	CPU2 Interrupt Priority Selection (C2PIS)	*Section 76.5.1.1.16*
0x15–0x16	Reserved for future use	
0x17	Level2 Mux State0 Selection 0 (L20SEL0)	*Section 76.5.1.2.1*
0x18	Level2 Mux State0 Selection 1 (L20SEL1)	*Section 76.5.1.2.2*
0x19	Level2 Mux State0 Selection 2 (L20SEL2)	*Section 76.5.1.2.3*

Table 1927. SPU register summary (continued)

Offset	Register	Location
0x1A	Level2 Mux State0 Selection 3 (L20SEL3)	Section 76.5.1.2.4
0x1B	Level2 Mux State1 Selection 0 (L21SEL0)	Section 76.5.1.2.1
0x1C	Level2 Mux State1 Selection 1 (L21SEL1)	Section 76.5.1.2.2
0x1D	Level2 Mux State1 Selection 2 (L21SEL2)	Section 76.5.1.2.3
0x1E	Level2 Mux State1 Selection 3 (L21SEL3)	Section 76.5.1.2.4
0x1F	Level2 Mux State2 Selection 0 (L22SEL0)	Section 76.5.1.2.1
0x20	Level2 Mux State2 Selection 1 (L22SEL1)	Section 76.5.1.2.2
0x21	Level2 Mux State2 Selection 2 (L22SEL2)	Section 76.5.1.2.3
0x22	Level2 Mux State2 Selection 3 (L22SEL3)	Section 76.5.1.2.4
0x23	Level2 Mux State3 Selection 0 (L23SEL0)	Section 76.5.1.2.1
0x24	Level2 Mux State3 Selection 1 (L23SEL1)	Section 76.5.1.2.2
0x25	Level2 Mux State3 Selection 2 (L23SEL2)	Section 76.5.1.2.3
0x26	Level2 Mux State3 Selection 3 (L23SEL3)	Section 76.5.1.2.4
0x27	Level2 Mux State4 Selection 0 (L24SEL0)	Section 76.5.1.2.1
0x28	Level2 Mux State4 Selection 1 (L24SEL1)	Section 76.5.1.2.2
0x29	Level2 Mux State4 Selection 2 (L24SEL2)	Section 76.5.1.2.3
0x2A	Level2 Mux State4 Selection 3 (L24SEL3)	Section 76.5.1.2.4
0x2B	Level2 Mux State5 Selection 0 (L25SEL0)	Section 76.5.1.2.1
0x2C	Level2 Mux State5 Selection 1 (L25SEL1)	Section 76.5.1.2.2
0x2D	Level2 Mux State5 Selection 2 (L25SEL2)	Section 76.5.1.2.3
0x2E	Level2 Mux State6 Selection 3 (L25SEL3)	Section 76.5.1.2.4
0x2F	Level2 Mux State6 Selection 0 (L26SEL0)	Section 76.5.1.2.1
0x30	Level2 Mux State6 Selection 1 (L26SEL1)	Section 76.5.1.2.2
0x31	Level2 Mux State6 Selection 2 (L26SEL2)	Section 76.5.1.2.3
0x32	Level2 Mux State6 Selection 3 (L26SEL3)	Section 76.5.1.2.4
0x33	Level2 Mux State7 Selection 0 (L27SEL0)	Section 76.5.1.2.1
0x34	Level2 Mux State7 Selection 1 (L27SEL1)	Section 76.5.1.2.2
0x35	Level2 Mux State7 Selection 2 (L27SEL2)	Section 76.5.1.2.3
0x36	Level2 Mux State7 Selection 3 (L27SEL3)	Section 76.5.1.2.4
0x37	Input/Output Inversion Control State0 (IOIC0)	Section 76.5.1.3.1
0x38	Input/Output Inversion Control State1 (IOIC1)	Section 76.5.1.3.1
0x39	Input/Output Inversion Control State2 (IOIC2)	Section 76.5.1.3.1
0x3A	Input/Output Inversion Control State3 (IOIC3)	Section 76.5.1.3.1
0x3B	Input/Output Inversion Control State4 (IOIC4)	Section 76.5.1.3.1
0x3C	Input/Output Inversion Control State5 (IOIC5)	Section 76.5.1.3.1

Table 1927. SPU register summary (continued)

Offset	Register	Location
0x3D	Input/Output Inversion Control State6 (IOIC6)	*Section 76.5.1.3.1*
0x3E	Input/Output Inversion Control State7 (IOIC7)	*Section 76.5.1.3.1*
0x3F	Sequence Control (SCTRL)	*Section 76.5.1.3.2*
0x40	State0 Goto Control (S0GC)	*Section 76.5.1.3.3*
0x41	State1 Goto Control (S1GC)	*Section 76.5.1.3.3*
0x42	State2 Goto Control (S2GC)	*Section 76.5.1.3.3*
0x43	State3 Goto Control (S3GC)	*Section 76.5.1.3.3*
0x44	State4 Goto Control (S4GC)	*Section 76.5.1.3.3*
0x45	State5 Goto Control (S5GC)	*Section 76.5.1.3.3*
0x46	State6 Goto Control (S6GC)	*Section 76.5.1.3.3*
0x47	State7 Goto Control (S7GC)	*Section 76.5.1.3.3*
0x48	SLU Status (SS)	*Section 76.5.1.4*
0x49	SPU Enable (SE)	*Section 76.5.1.5*
0x01	State0 True Action 0 (S0TA0)	*Section 76.5.1.7.1*
0x02	State0 True Action 1 (S0TA1)	*Section 76.5.1.7.2*
0x03	State1 True Action 0 (S1TA0)	*Section 76.5.1.7.1*
0x04	State1 True Action 1 (S1TA1)	*Section 76.5.1.7.2*
0x05	State2 True Action 0 (S2TA0)	*Section 76.5.1.7.1*
0x06	State2 True Action 1 (S2TA1)	*Section 76.5.1.7.2*
0x07	State3 True Action 0 (S3TA0)	*Section 76.5.1.7.1*
0x08	State3 True Action 1 (S3TA1)	*Section 76.5.1.7.2*
0x09	State4 True Action 0 (S4TA0)	*Section 76.5.1.7.1*
0x0A	State4 True Action 1 (S4TA1)	*Section 76.5.1.7.2*
0x0B	State5 True Action 0 (S5TA0)	*Section 76.5.1.7.1*
0x0C	State5 True Action 1(S5TA1)	*Section 76.5.1.7.2*
0x0D	State6 True Action 0 (S6TA0)	*Section 76.5.1.7.1*
0x0E	State6 True Action 1(S6TA1)	*Section 76.5.1.7.2*
0x0F	State7 True Action 0 (S7TA0)	*Section 76.5.1.7.1*
0x10	State7 True Action 1(S7TA1)	*Section 76.5.1.7.2*
0x11	State0 False Action 0 (S0FA0)	*Section 76.5.1.7.3*
0x12	State0 False Action 1 (S0FA1)	*Section 76.5.1.7.4*
0x13	State1 False Action 0 (S1FA0)	*Section 76.5.1.7.3*
0x14	State1 False Action 1(S1FA1)	*Section 76.5.1.7.4*
0x15	State2 False Action 0 (S2FA0)	*Section 76.5.1.7.3*
0x16	State2 False Action 1(S2FA1)	*Section 76.5.1.7.4*

Table 1927. SPU register summary (continued)

Offset	Register	Location
0x17	State3 False Action 0 (S3FA0)	*Section 76.5.1.7.3*
0x18	State3 False Action 1(S3FA0)	*Section 76.5.1.7.4*
0x19	State4 False Action 0 (S4FA0)	*Section 76.5.1.7.3*
0x1A	State4 False Action 1 (S4FA1)	*Section 76.5.1.7.4*
0x1B	State5 False Action 0 (S5FA0)	*Section 76.5.1.7.3*
0x1C	State5 False Action 1 (S5FA1)	*Section 76.5.1.7.4*
0x1D	State6 False Action 0 (S6FA0)	*Section 76.5.1.7.3*
0x1E	State6 False Action 1 (S6FA1)	*Section 76.5.1.7.4*
0x1F	State7 False Action 0 (S7FA0)	*Section 76.5.1.7.3*
0x20	State7 False Action 1 (S7FA1)	*Section 76.5.1.7.4*
0x21	DTM Trace Group Configuration (DTMTGC)	*Section 76.5.1.7.5*
0x22	PTM Trace Group Configuration (PTMTGC)	*Section 76.5.1.7.5*
0x23	OTM Trace Group Configuration (OTMTGC)	*Section 76.5.1.7.5*
0x24	WTM Trace Group Configuration (WTMTGC)	*Section 76.5.1.7.5*
0x25	Interrupt Status (INTS)	*Section 76.5.1.7.6*
0x26	Counter Control 0 (CCTRL0)	*Section 76.5.1.8*
0x27	Counter Control 1 (CCTRL1)	*Section 76.5.1.8*
0x28	Counter Control 2 (CCTRL2)	*Section 76.5.1.8*
0x29	Counter Control 3 (CCTRL3)	*Section 76.5.1.8*
0x2A	Counter Control 4 (CCTRL4)	*Section 76.5.1.8*
0x2B	Counter Control 5 (CCTRL5)	*Section 76.5.1.8*
0x2C	Counter Control 6 (CCTRL6)	*Section 76.5.1.8*
0x2D	Counter Control 7 (CCTRL7)	*Section 76.5.1.8*
0x2E	Counter Control 8 (CCTRL8)	*Section 76.5.1.8*
0x2F	Counter Control 9 (CCTRL9)	*Section 76.5.1.8*
0x30	Counter Control 10 (CCTRL10)	*Section 76.5.1.8*
0x31	Counter Control 11 (CCTRL11)	*Section 76.5.1.8*
0x32	Counter Control 12 (CCTRL12)	*Section 76.5.1.8*
0x33	Counter Control 13 (CCTRL13)	*Section 76.5.1.8*
0x34	Counter Control 14 (CCTRL14)	*Section 76.5.1.8*
0x35	Counter Control 15 (CCTRL15)	*Section 76.5.1.8*
0x36	Counter Compare 0(CCMP0)	*Section 76.5.1.9*
0x37	Counter Compare 1 (CCMP1)	*Section 76.5.1.9*
0x38	Counter Compare 2 (CCMP2)	*Section 76.5.1.9*
0x39	Counter Compare 3 (CCMP3)	*Section 76.5.1.9*

Table 1927. SPU register summary (continued)

Offset	Register	Location
0x3A	Counter Compare 4 (CCMP4)	*Section 76.5.1.9*
0x3B	Counter Compare 5 (CCMP5)	*Section 76.5.1.9*
0x3C	Counter Compare 6 (CCMP6)	*Section 76.5.1.9*
0x3D	Counter Compare 7 (CCMP7)	*Section 76.5.1.9*
0x3E	Counter Compare 8 (CCMP8)	*Section 76.5.1.9*
0x3F	Counter Compare 9 (CCMP9)	*Section 76.5.1.9*
0x40	Counter Compare 10 (CCMP10)	*Section 76.5.1.9*
0x41	Counter Compare 11 (CCMP11)	*Section 76.5.1.9*
0x42	Counter Compare 12 (CCMP12)	*Section 76.5.1.9*
0x43	Counter Compare 13 (CCMP13)	*Section 76.5.1.9*
0x44	Counter Compare 14 (CCMP14)	*Section 76.5.1.9*
0x45	Counter Compare 15 (CCMP15)	*Section 76.5.1.9*
0x46	Counter Compare Status (CCOMS)	*Section 76.5.1.10.1*
0x47	Counter Overflow Status (COS)	*Section 76.5.1.10.2*
0x48	Counter Capture Status (CCAPS)	*Section 76.5.1.10.3*

76.5.1 Registers description

76.5.1.1 Level1 input Mux configurations registers

A set of input Mux control registers is used to narrow down the large number of SPU watchpoints to a manageable set. There are sixty-four 8 × 1 (eight inputs and one output) multiplexers. Three bits per Mux are used to select one input out of 8 inputs. 8 Muxes are configured with each 32-bit register; in other words, 8 inputs out of 64 for each register. This provides the flexibility to the programmer to route any input trigger to any state input. The detail of this selection is shown in *Figure 2027*.

Note: Users cannot program MUX 63 in L1SEL7 (the 64th mux). All eight inputs of the 63rd mux are tied to high and the output is always high (1'b1).

76.5.1.1.1 Level1 Mux Selection 0 (L1SEL0)

Figure 1990 shows the format of the L1SEL0 register. The values for the fields of the L1SEL0 register are device dependent. For a description of these fields, see Calibration and Debug (or Debug and Trace) chapter that describes how the modules are configured.

Offset: 0x01 Access: Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	MUX 7			0	MUX 6			0	MUX 5			0	MUX 4		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	MUX 3			0	MUX 2			0	MUX 1			0	MUX 0		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1990. Level1 Mux Selection 0 (L1SEL0)

76.5.1.1.2 Level1 Mux Selection 1 (L1SEL1)

Figure 1991 shows the format of the L1SEL1 register. The values for the fields of the L1SEL1 register are device dependent. For a description of these fields, see Calibration and Debug (or Debug and Trace) chapter that describes how the modules are configured.

Offset: 0x02 Access: Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	MUX 15			0	MUX 14			0	MUX 13			0	MUX 12		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	MUX 11			0	MUX 10			0	MUX 9			0	MUX 8		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1991. Level1 Mux Selection 1 (L1SEL1)

76.5.1.1.3 Level1 Mux Selection 2 (L1SEL2)

Figure 1992 shows the format of the L1SEL2 register. The values for the fields of the L1SEL2 register are device dependent. For a description of these fields, see Calibration and Debug (or Debug and Trace) chapter that describes how the modules are configured.

Offset: 0x03 Access: Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	MUX 23			0	MUX 22			0	MUX 21			0	MUX 20		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	MUX 19			0	MUX 18			0	MUX 17			0	MUX 16		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1992. Level1 Mux Selection 2 (L1SEL2)

76.5.1.1.4 Level1 Mux Selection 3 (L1SEL3)

Figure 1993 shows the format of the L1SEL3 register. The values for the fields of the L1SEL3 register are device dependent. For a description of these fields, see Calibration and Debug (or Debug and Trace) chapter that describes how the modules are configured.

Offset: 0x04 Access: Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	MUX 31			0	MUX 30			0	MUX 29			0	MUX 28		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	MUX 27			0	MUX 26			0	MUX 25			0	MUX 24		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1993. Level1 Mux Selection 3 (L1SEL3)

76.5.1.1.5 Level1 Mux Selection 4 (L1SEL4)

Figure 1994 shows the format of the L1SEL4 register. The values for the fields of the L1SEL4 register are device dependent. For a description of these fields, see Calibration and Debug (or Debug and Trace) chapter that describes how the modules are configured.

Offset: 0x05 Access: Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	MUX 39			0	MUX 38			0	MUX 37			0	MUX 36		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	MUX 35			0	MUX 34			0	MUX 33			0	MUX 32		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1994. Level1 Mux Selection 4 (L1SEL4)

76.5.1.1.6 Level1 Mux Selection 5 (L1SEL5)

Figure 1995 shows the format of the L1SEL5 register. The values for the fields of the L1SEL5 register are device dependent. For a description of these fields, see Calibration and Debug (or Debug and Trace) chapter that describes how the modules are configured.

Offset: 0x06 Access: Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	MUX 47			0	MUX 46			0	MUX 45			0	MUX 44		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	MUX 43			0	MUX 42			0	MUX 41			0	MUX 40		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1995. Level1 Mux Selection 5 (L1SEL5)

76.5.1.1.7 Level1 Mux Selection 6 (L1SEL6)

Figure 1996 shows the format of the L1SEL6 register.The values for the fields of the L1SEL6 register are device dependent. For a description of these fields, see Calibration and Debug (or Debug and Trace) chapter that describes how the modules are configured.

Offset: 0x07 Access: Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	MUX 55			0	MUX 54			0	MUX 53			0	MUX 52		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	MUX 51			0	MUX 50			0	MUX 49			0	MUX 48		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1996. Level1 Mux Selection 6 (L1SEL6)

76.5.1.1.8 Level1 Mux Selection 7 (L1SEL7)

Figure 1997 shows the format of the L1SEL7 register. The values for the fields of the L1SEL7 register are device dependent. For a description of these fields, see Calibration and Debug (or Debug and Trace) chapter that describes how the modules are configured. All eight inputs of MUX 63 are tied to logic high (1'b1) and output is always high.

Offset: 0x08 Access: Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	MUX 63			0	MUX 62			0	MUX 61			0	MUX 60		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	MUX 59			0	MUX 58			0	MUX 57			0	MUX 56		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1997. Level1 Mux Selection 7 (L1SEL7)

76.5.1.1.9 Input triggers level detection control 2 (ITLD2)

Only a rising edge is detected on the outputs of the first 32 level1 Muxes and therefore these values are not programmable. The last 32 level1 Muxes (Mux 32–63) trigger level is programmable in the Input Triggers Level Detection Control 2 and 3 registers. *Figure 1998* shows the format of the ITLD2 register. All of the Mux fields in the ITLD2 register have the same description.

Offset: 0x0B Access: Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	Mux47		Mux46		Mux45		Mux44		Mux43		Mux42		Mux41		Mux40	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	Mux39		Mux38		Mux37		Mux36		Mux35		Mux34		Mux33		Mux32	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1998. Input triggers level detection control 2 (ITLD2)

The ITLD2 registers Mux fields are described in *Table 1928*.

Table 1928. ITLD2 register field descriptions

Field	Description
0:1 Mux*47*	Mux47 Output Signal Detection 00 Level is passed after synchronization (default) 01 Posedge detection 10 Toggle detection 11 Level is passed after synchronization
2:3 Mux*46*	Mux46 Output Signal Detection 00 Level is passed after synchronization (default) 01 Posedge detection 10 Toggle detection 11 Level is passed after synchronization
4:5 Mux*45*	Mux45 Output Signal Detection 00 Level is passed after synchronization (default) 01 Posedge detection 10 Toggle detection 11 Level is passed after synchronization
6:7 Mux*44*	Mux44 Output Signal Detection 00 Level is passed after synchronization (default) 01 Posedge detection 10 Toggle detection 11 Level is passed after synchronization
8:9 Mux*43*	Mux43 Output Signal Detection 00 Level is passed after synchronization (default) 01 Posedge detection 10 Toggle detection 11 Level is passed after synchronization
10:11 Mux*42*	Mux42 Output Signal Detection 00 Level is passed after synchronization (default) 01 Posedge detection 10 Toggle detection 11 Level is passed after synchronization

Table 1928. ITLD2 register field descriptions (continued)

Field	Description
12:13 Mux*41*	Mux41 Output Signal Detection 00 Level is passed after synchronization (default) 01 Posedge detection 10 Toggle detection 11 Level is passed after synchronization
14:15 Mux*40*	Mux40 Output Signal Detection 00 Level is passed after synchronization (default) 01 Posedge detection 10 Toggle detection 11 Level is passed after synchronization
16:17 Mux*39*	Mux39 Output Signal Detection 00 Level is passed after synchronization (default) 01 Posedge detection 10 Toggle detection 11 Level is passed after synchronization
18:19 Mux*38*	Mux38 Output Signal Detection 00 Level is passed after synchronization (default) 01 Posedge detection 10 Toggle detection 11 Level is passed after synchronization
20:21 Mux*37*	Mux37 Output Signal Detection 00 Level is passed after synchronization (default) 01 Posedge detection 10 Toggle detection 11 Level is passed after synchronization
22:23 Mux*36*	Mux36 Output Signal Detection 00 Level is passed after synchronization (default) 01 Posedge detection 10 Toggle detection 11 Level is passed after synchronization
24:25 Mux*35*	Mux35 Output Signal Detection 00 Level is passed after synchronization (default) 01 Posedge detection 10 Toggle detection 11 Level is passed after synchronization
26:27 Mux*34*	Mux34 Output Signal Detection 00 Level is passed after synchronization (default) 01 Posedge detection 10 Toggle detection 11 Level is passed after synchronization

Table 1928. ITLD2 register field descriptions (continued)

Field	Description
28:29 Mux*33*	Mux33 Output Signal Detection 00 Level is passed after synchronization (default) 01 Posedge detection 10 Toggle detection 11 Level is passed after synchronization
30:31 Mux*32*	Mux32 Output Signal Detection 00 Level is passed after synchronization (default) 01 Posedge detection 10 Toggle detection 11 Level is passed after synchronization

76.5.1.1.10 Input triggers level detection control 3 (ITLD3)

Only a rising edge is detected on the outputs of the first 32 level1 Muxes and so these values are not programmable. The last 32 level1 Muxes (Mux 32–63) trigger level is programmable in the Input Triggers Level Detection Control 2 and 3 registers. *Figure 1999* shows the format of the ITLD3 register.All of the Mux fields in the ITLD3 register have the same description.

Offset: 0x0C Access: Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	Mux63				Mux62				Mux61				Mux60			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	Mux55				Mux54				Mux53				Mux52			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47
R W	Mux51				Mux50				Mux49				Mux48			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63
R W	Mux47				Mux46				Mux45				Mux44			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1999. Input triggers level detection control 3 (ITLD3)

The ITLD3 registers Mux fields are described in *Table 1929*.

Table 1929. ITLD3 register field descriptions

Field	Description
0:1 Mux*63*	Mux63 Output Signal Detection 00 Level is passed after synchronization (default) 01 Posedge detection 10 Toggle detection 11 Level is passed after synchronization
2:3 Mux*62*	Mux62 Output Signal Detection 00 Level is passed after synchronization (default) 01 Posedge detection 10 Toggle detection 11 Level is passed after synchronization

Table 1929. ITLD3 register field descriptions (continued)

Field	Description
4:5 Mux*61*	Mux61 Output Signal Detection 00 Level is passed after synchronization (default) 01 Posedge detection 10 Toggle detection 11 Level is passed after synchronization
6:7 Mux*60*	Mux60 Output Signal Detection 00 Level is passed after synchronization (default) 01 Posedge detection 10 Toggle detection 11 Level is passed after synchronization
8:9 Mux*59*	Mux59 Output Signal Detection 00 Level is passed after synchronization (default) 01 Posedge detection 10 Toggle detection 11 Level is passed after synchronization
10:11 Mux*58*	Mux58 Output Signal Detection 00 Level is passed after synchronization (default) 01 Posedge detection 10 Toggle detection 11 Level is passed after synchronization
12:13 Mux*57*	Mux57 Output Signal Detection 00 Level is passed after synchronization (default) 01 Posedge detection 10 Toggle detection 11 Level is passed after synchronization
14:15 Mux*56*	Mux56 Output Signal Detection 00 Level is passed after synchronization (default) 01 Posedge detection 10 Toggle detection 11 Level is passed after synchronization
16:17 Mux*55*	Mux55 Output Signal Detection 00 Level is passed after synchronization (default) 01 Posedge detection 10 Toggle detection 11 Level is passed after synchronization
18:19 Mux*54*	Mux54 Output Signal Detection 00 Level is passed after synchronization (default) 01 Posedge detection 10 Toggle detection 11 Level is passed after synchronization
20:21 Mux*53*	Mux53 Output Signal Detection 00 Level is passed after synchronization (default) 01 Posedge detection 10 Toggle detection 11 Level is passed after synchronization

Table 1929. ITLD3 register field descriptions (continued)

Field	Description
22:23 Mux*52*	Mux52 Output Signal Detection 00 Level is passed after synchronization (default) 01 Posedge detection 10 Toggle detection 11 Level is passed after synchronization
24:25 Mux*51*	Mux51 Output Signal Detection 00 Level is passed after synchronization (default) 01 Posedge detection 10 Toggle detection 11 Level is passed after synchronization
26:27 Mux*50*	Mux50 Output Signal Detection 00 Level is passed after synchronization (default) 01 Posedge detection 10 Toggle detection 11 Level is passed after synchronization
28:29 Mux*49*	Mux49 Output Signal Detection 00 Level is passed after synchronization (default) 01 Posedge detection 10 Toggle detection 11 Level is passed after synchronization
30:31 Mux*48*	Mux48 Output Signal Detection 00 Level is passed after synchronization (default) 01 Posedge detection 10 Toggle detection 11 Level is passed after synchronization

76.5.1.1.11 CPU0 processor exception vector prefix (C0PEVP)

The SPU captures the exception vector from CPU0 when the exception enable signal is asserted. The exception vector decoding is defined in *Table 1957*. The SPU compares this captured vector with the value in the C0PEVP register. *Figure 2000* shows the format of the C0PEVP register.

Offset: 0x0D Access: Read/Write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	CPU0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 2000. CPU0 processor exception vector prefix (C0PEVP)

The C0PEVP register is described in *Table 1930*.

Table 1930. C0PEVP register field descriptions

Field	Description
16:31 CPU0	CPU0 processor exception vector Program this field with one of the vector values from *Table 1957* to select the corresponding processor exception. This field must use the byte swapped values in the CnPEVP register CPUn field value column in *Table 1957*.

76.5.1.1.12 CPU1 processor exception vector prefix (C1PEVP)

The SPU captures the exception vector from CPU1 when the exception enable signal is asserted. The exception vector decoding is defined in *Table 1957 on page 3366*. The SPU compares this captured vector with the value in the C1PEVP register. *Figure 2001* shows the format of the C1PEVP register.

Offset: 0x0E Access: Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	CPU1															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

Figure 2001. CPU1 processor exception vector prefix (C1PEVP)

The C1PEVP register is described in *Table 1931*.

Table 1931. C1PEVP register field descriptions

Field	Description
0:15 CPU1	CPU1 processor exception vector Program this field with one of the vector values from *Table 1957* to select the corresponding processor exception. This field must use the byte swapped values in the CnPEVP register CPUn field value column in *Table 1957*.

76.5.1.1.13 CPU2 processor exception vector prefix (C2PEVP)

The SPU captures the exception vector from CPU2 when the exception enable signal is asserted. The exception vector decoding is defined in *Table 1957*. The SPU compares this captured vector with the value in the C2PEVP register. *Figure 2002* shows the format of the C2PEVP register.

Offset: 0x0F Access: Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	CPU2															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

Figure 2002. CPU2 processor exception vector prefix (C2PEVP)

The C2PEVP register is described in *Table 1932*.

Table 1932. C2PEVP register field descriptions

Field	Description
0:15 CPU2	CPU2 processor exception vector Program this field with one of the vector values from *Table 1957* to select the corresponding processor exception. **Note:** This field must use the byte swapped values in the CnPEVP register CPUn field value column in *Table 1957*.

76.5.1.1.14 CPU0 interrupt priority selection (C0PIS)

Figure 2003 shows the format of the C0PIS register. The SPU compares the input CPR vector from CPU0 with the value programmed to the C0PIS register and generates a match signal if necessary. The typical use case would be for posedge detection on match (to detect when the priority level is set to the compare value) or high detection while match remains.

Offset: 0x12 Access: Read/Write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	CPU0							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 2003. CPU0 interrupt priority selection (C0PIS)

The C0PIS register field is described in *Table 1933*.

Table 1933. C0PIS register field descriptions

Field	Description
24:31 CPU0	CPU0 Priority Interrupt Selection

76.5.1.1.15 CPU1 interrupt priority selection (C1PIS)

Figure 2004 shows the format of the C1PIS register. The SPU compares the input CPR vector from CPU1 with the value programmed to the C1PIS register and generates a match signal if necessary. The typical use case would be for posedge detection on match (to detect when the priority level is set to the compare value) or high detection while match remains.

Offset: 0x13 Access: Read/Write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	CPU1							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 2004. CPU1 interrupt priority selection (C1PIS)

The C1PIS register is described in *Table 1934*.

Table 1934. C1PIS register field descriptions

Field	Description
24:31 CPU1	CPU1 Priority Interrupt Selection

76.5.1.1.16 CPU2 interrupt priority selection (C2PIS)

Figure 2005 shows the format of the C2PIS register. The SPU compares the input CPR vector from CPU2 with the value programmed to the C2PIS register and generates a match signal if necessary. The typical use case would be for posedge detection on match (to detect when the priority level is set to the compare value) or high detection while match remains.

Offset: 0x14 Access: Read/Write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	CPU2							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 2005. CPU2 interrupt priority selection (C2PIS)

The C2PIS register is described in *Table 1935*.

Table 1935. C2PIS register field descriptions

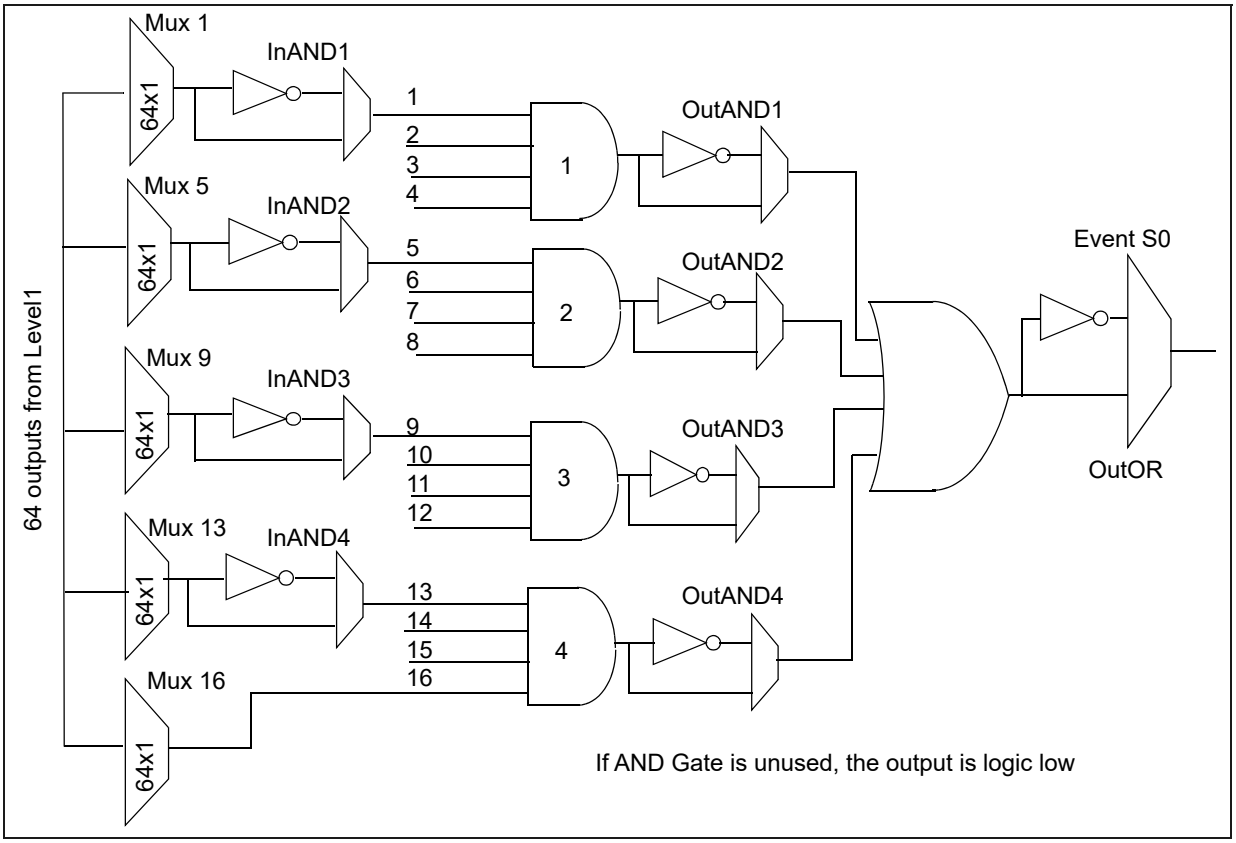
Field	Description
24:31 CPU2	CPU2 Priority Interrupt Selection

76.5.1.2 Level2 input Mux configurations registers

At level1, the set of input Mux control registers (described in *Section 76.5.1.1: Level1 input Mux configurations registers*) narrows down the large number of SPU watchpoints to a manageable set using sixty-four 8 × 1 multiplexers. At Level2, 16 inputs are selected for each SLU from the 64 level1 outputs using 16 64 × 1 multiplexers. Six bits per Mux select one input out of 64 (in fact 63) inputs. Each 32-bit register configures the AND gates of one state; therefore, eight registers are used to configure inputs for the 8 SLUs. All the 63 inputs are static for all the eight states. Any 16 inputs can be selected from the 63 input triggers for a state.

The detail selection is shown in the *Figure 2027* for State0. Similar logic is provided for States 1 to 7.

Figure 2006. Level2 Mux AND gates input pin routing for State0



Note: If any SLU AND gate is unused, then its output is logic low (1'b0). Thus, if all four inputs of a particular AND gate are at the default value 1'b1 (none of the inputs have been routed from the Level 2 mux to the inputs of AND gates), then the output of that AND gate is logic low. This provides protection to avoid the masking of other inputs to the OR gate.

76.5.1.2.1 Level2 Mux state n selection 0 (L2nSEL0)

Figure 2007 shows the format of the L2nSEL0 register where the state number, n = 0 to 7.

Offset: 0x17 +n*0x4 (n=0 to 7) Access: Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	FirstANDInput4						0	0	FirstANDInput3					
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	FirstANDInput2						0	0	FirstANDInput1					
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 2007. Level2 Mux state n selection 0 (L2nSEL0)

The L2nSEL0 register fields are described in *Table 1936*.

Table 1936. L2nSEL0 register field descriptions

Field	Description
2:7 FirstANDInput4	First AND Gate Input 4 Selection 000000 No Input is selected (tie to 1'b1) 000001 Level1 MUX 0 output 000010 Level1 MUX 1 output 000011 Level1 MUX 2 output … 111110 Level1 MUX 61 output 111111 Level1 MUX 62 output
10:15 FirstANDInput3	First AND Gate Input 3 Selection 000000 No Input is selected (tie to 1'b1) 000001 Level1 MUX 0 output 000010 Level1 MUX 1 output 000011 Level1 MUX 2 output … 111110 Level1 MUX 61 output 111111 Level1 MUX 62 output
18:23 FirstANDInput2	First AND Gate Input 2 Selection 000000 No Input is selected (tie to 1'b1) 000001 Level1 MUX 0 output 000010 Level1 MUX 1 output 000011 Level1 MUX 2 output … 111110 Level1 MUX 61 output 111111 Level1 MUX 62 output
26:31 FirstANDInput1	First AND Gate Input 1 Selection 000000 No Input is selected (tie to 1'b1) 000001 Level1 MUX 0 output 000010 Level1 MUX 1 output 000011 Level1 MUX 2 output … 111110 Level1 MUX 61 output 111111 Level1 MUX 62 output

76.5.1.2.2 Level2 Mux state n selection 1 (L2nSEL1)

Figure 2008 shows the format of the L2nSEL1 register where the state number, n = 0 to 7.

Offset: 0x18 +n*0x4 (n=0 to 7) Access: Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	SecondANDInput4						0	0	SecondANDInput3					
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	SecondANDInput2						0	0	SecondANDInput1					
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 2008. Level2 Mux state n selection 1 (L2nSEL1)

The L2nSEL1 register fields are described in *Table 1937*.

Table 1937. L2nSEL1 register field descriptions

Field	Description
2:7 SecondANDInput4	Second AND Gate Input 4 Selection 000000 No Input is selected (tie to 1'b1) 000001 Level1 MUX 0 output 000010 Level1 MUX 1 output 000011 Level1 MUX 2 output … 111110 Level1 MUX 61 output 111111 Level1 MUX 62 output
10:15 SecondANDInput3	Second AND Gate Input 3 Selection 000000 No Input is selected (tie to 1'b1) 000001 Level1 MUX 0 output 000010 Level1 MUX 1 output 000011 Level1 MUX 2 output … 111110 Level1 MUX 61 output 111111 Level1 MUX 62 output
18:23 SecondANDInput2	Second AND Gate Input 2 Selection 000001 Level1 MUX 1 output 000011 Level1 MUX 2 output … 111110 Level1 MUX 61 output 111111 Level1 MUX 62 output
26:31 SecondANDInput1	Second AND Gate Input 1Selection 000000 No Input is selected (tie to 1'b1) 000001 Level1 MUX 0 output 000010 Level1 MUX 1 output 000011 Level1 MUX 2 output … 111110 Level1 MUX 61 output 111111 Level1 MUX 62 output

76.5.1.2.3 Level2 Mux state n selection 2 (L2nSEL2)

Figure 2009 shows the format of the L2nSEL2 register where the state number, n = 0 to 7.

Offset: 0x19 n*0x4 (n=0 to 7) Access: Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	ThirdANDInput4						0	0	ThirdANDInput3					
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	ThirdANDInput2						0	0	ThirdANDInput1					
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 2009. Level2 Mux state n selection 2 (L2nSEL2)

The L2nSEL2 register fields are described in *Table 1938*.

Table 1938. L2nSEL2 register field descriptions

Field	Description
2:7 ThirdANDInput4	Third AND Gate Input 4 Selection 000000 No Input is selected (tie to 1'b1) 000001 Level1 MUX 0 output 000010 Level1 MUX 1 output 000011 Level1 MUX 2 output … 111110 Level1 MUX 61 output 111111 Level1 MUX 62 output
10:15 ThirdANDInput3	Third AND Gate Input 3 Selection 000000 No Input is selected (tie to 1'b1) 000001 Level1 MUX 0 output 000010 Level1 MUX 1 output 000011 Level1 MUX 2 output … 111110 Level1 MUX 61 output 111111 Level1 MUX 62 output

Table 1938. L2nSEL2 register field descriptions (continued)

Field	Description
18:23 ThirdANDInput2	Third AND Gate Input 2 Selection 000000 No Input is selected (tie to 1'b1) 000001 Level1 MUX 0 output 000010 Level1 MUX 1 output 000011 Level1 MUX 2 output … 111110 Level1 MUX 61 output 111111 Level1 MUX 62 output
26:31 ThirdANDInput1	Third AND Gate Input 1 Selection 000000 No Input is selected (tie to 1'b1) 000001 Level1 MUX 0 output 000010 Level1 MUX 1 output 000011 Level1 MUX 2 output … 111110 Level1 MUX 61 output 111111 Level1 MUX 62 output

76.5.1.2.4 Level2 Mux state n selection 3 (L2nSEL3)

Figure 2010 shows the format of the L2nSEL3 register where the state number, n = 0 to 7.

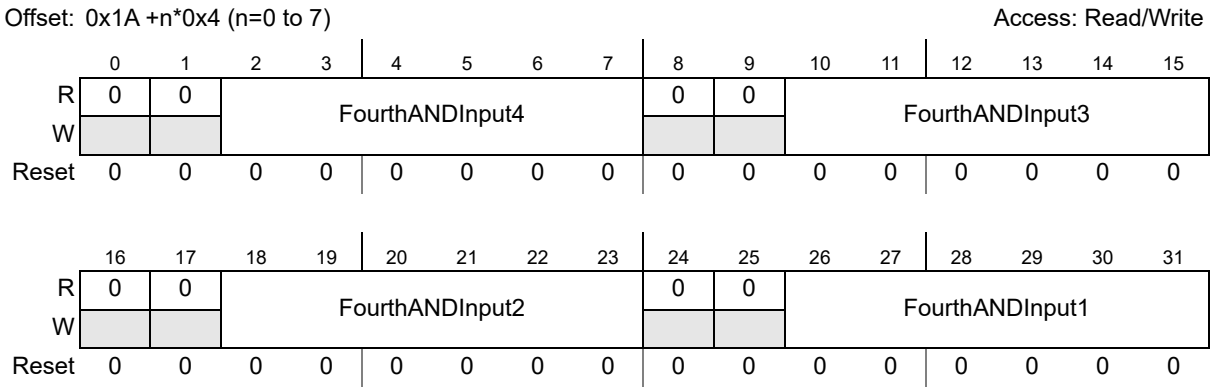


Figure 2010. Level2 Mux state n selection 3 (L2nSEL3)

The L2nSEL3 register fields are described in *Table 1939*.

Table 1939. L2nSEL3 register field descriptions

Field	Description
2:7 FourthANDInput4	Fourth AND Gate Input 4 Selection 000000 No Input is selected (tie to 1'b1) 000001 Level1 MUX 0 output 000010 Level1 MUX 1 output 000011 Level1 MUX 2 output … 111110 Level1 MUX 61 output 111111 Level1 MUX 62 output
10:15 FourthANDInput3	Fourth AND Gate Input 3 Selection 000000 No Input is selected (tie to 1'b1) 000001 Level1 MUX 0 output 000010 Level1 MUX 1 output 000011 Level1 MUX 2 output … 111110 Level1 MUX 61 output 111111 Level1 MUX 62 output
18:23 FourthANDInput2	Fourth AND Gate Input 2 Selection 000000 No Input is selected (tie to 1'b1) 000001 Level1 MUX 0 output 000010 Level1 MUX 1 output 000011 Level1 MUX 2 output … 111110 Level1 MUX 61 output 111111 Level1 MUX 62 output
26:31 FourthANDInput1	Fourth AND Gate Input 1 Selection 000000 No Input is selected (tie to 1'b1) 000001 Level1 MUX 0 output 000010 Level1 MUX 1 output 000011 Level1 MUX 2 output … 111110 Level1 MUX 61 output 111111 Level1 MUX 62 output

76.5.1.3 Sequence unit registers

76.5.1.3.1 Input/output Inversion control state n (IOICn)

An input control register is provided for each event generated by the SLU. This register is programmed to specify how the input signals to each SLU are combined to generate the resulting event for that particular sequence. In total, there are 16 inputs for each state and only one (the first) of the inputs can be used with or without inversion. The input can be left unused if not required for that state by selecting 000000 (No input selected option) in the L2nSEL registers. The output of each AND gate has the optional inversion control as well. The output can be routed directly or with inversion from each AND gate to the input of OR gate. *Figure 2011* shows the format of the IOICn register where the state number, n = 0 to 7.

Offset: 0x37 n*0x1 (n=0 to 7) Access: Read/Write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	OutOR	OutAND4	OutAND3	OutAND2	OutAND1	InAND4	InAND3	InAND2	InAND1
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 2011. Input/output Inversion control state n (IOICn)

The IOICn register fields are described in *Table 1940*.

Table 1940. IOICn register field descriptions

Field	Description
23 OutOR	Output OR GATE Control 0 Output is used directly (without inversion) for this event 1 Output is inverted for this event
24 OutAND4	Output of AND Gate 4 Control 0 Output is used directly (without inversion) for this event 1 Output is inverted for this event
25 OutAND3	Output AND Gate 3 Control 0 Output is used directly (without inversion) for this event 1 Output is inverted for this event
26 OutAND2	Output AND Gate 2 Control 0 Output is used directly (without inversion) for this event 1 Output is inverted for this event
27 OutAND1	Output AND Gate 1 Control 0 Output is used directly (without inversion) for this event 1 Output is inverted for this event
28 InAND4	Input 1 AND Gate 4 Control 0 Input 1 is used directly (without inversion) for this event 1 Input 1 is inverted for this event
29 InAND3	Input 1 AND Gate 3 Control 0 Input 1 is used directly (without inversion) for this event 1 Input 1 is inverted for this event
30 InAND2	Input 1 AND Gate 2 Control 0 Input 1 is used directly (without inversion) for this event 1 Input 1 is inverted for this event
31 InAND1	Input 1 AND Gate 1 Control 0 Input 1 is used directly (without inversion) for this event 1 Input 1 is inverted for this event

76.5.1.3.2 Sequence control (SCTRL)

This configuration register is provided to enable the sequence and to select the states that are part of a particular sequence. There can be at most four active sequences. Each sequence can have one state or a maximum of eight states. However, each state can only be used in one unique sequence. In other words, if one state has been selected for a sequence, then this state cannot be used for any other defined sequence. Each sequence has a start state and end state. *Figure 2012* shows the format of the SCTRL register.

Offset: 0x3F Access: Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	S4ST2			S4ST1			S4ST0	0	S3ST2			S3ST1			S3ST0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	S2ST2			S2ST1			S2ST0	0	S1ST2			S1ST1			S1ST0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 2012. Sequence control (SCTRL)

The SCTRL register fields are described in *Table 1941*.

Table 1941. SCTRL register field descriptions

Field	Description
1:3 S4ST2	Sequence 4 End State 000 State 0 is the Stop state for sequence 4 001 State 1 is the Stop state for sequence 4 010 State 2 is the Stop state for sequence 4 011 State 3 is the Stop state for sequence 4 100 State 4 is the Stop state for sequence 4 101 State 5 is the Stop state for sequence 4 110 State 6 is the Stop state for sequence 4 111 State 7 is the Stop state for sequence 4
4:6 S4ST1	Sequence 4 Start State(1) 000 State 0 is the Start state for sequence 4 001 State 1 is the Start state for sequence 4 010 State 2 is the Start state for sequence 4 011 State 3 is the Start state for sequence 4 100 State 4 is the Start state for sequence 4 101 State 5 is the Start state for sequence 4 110 State 6 is the Start state for sequence 4 111 State 7 is the Start state for sequence 4
7 S4ST0	Sequence 4 Enable/Disable(2) Once enabled, this bit is disabled at the end of the Sequence 4. 0 Sequence 4 is disabled 1 Sequence 4 is enabled

Table 1941. SCTRL register field descriptions (continued)

Field	Description
9:11 S3ST2	Sequence 3 End State 000 State 0 is the Stop state for sequence 3 001 State 1 is the Stop state for sequence 3 010 State 2 is the Stop state for sequence 3 011 State 3 is the Stop state for sequence 3 100 State 4 is the Stop state for sequence 3 101 State 5 is the Stop state for sequence 3 110 State 6 is the Stop state for sequence 3 111 State 7 is the Stop state for sequence 3
12:14 S3ST1	Sequence 3 Start State(1) 000 State 0 is the Start state for sequence 3 001 State 1 is the Start state for sequence 3 010 State 2 is the Start state for sequence 3 011 State 3 is the Start state for sequence 3 100 State 4 is the Start state for sequence 3 101 State 5 is the Start state for sequence 3 110 State 6 is the Start state for sequence 3 111 State 7 is the Start state for sequence 3
15 S3ST0	Sequence 3 Enable/Disable(2) Once enabled, this bit is disabled at the end of the Sequence 3. 0 Sequence 3 is disabled 1 Sequence 3 is enabled
17:19 S2ST2	Sequence 2 End State 000 State 0 is the Stop state for sequence 2 001 State 1 is the Stop state for sequence 2 010 State 2 is the Stop state for sequence 2 011 State 3 is the Stop state for sequence 2 100 State 4 is the Stop state for sequence 2 101 State 5 is the Stop state for sequence 2 110 State 6 is the Stop state for sequence 2 111 State 7 is the Stop state for sequence 2
20:22 S2ST1	Sequence 2 Start State(1) 000 State 0 is the Start state for sequence 2 001 State 1 is the Start state for sequence 2 010 State 2 is the Start state for sequence 2 011 State 3 is the Start state for sequence 2 100 State 4 is the Start state for sequence 2 101 State 5 is the Start state for sequence 2 110 State 6 is the Start state for sequence 2 111 State 7 is the Start state for sequence 2
23 S2ST0	Sequence2 Enable/Disable(2) Once enabled, this bit is disabled at the end of the Sequence 2. 0 Sequence 2 is disabled 1 Sequence 2 is enabled

Table 1941. SCTRL register field descriptions (continued)

Field	Description
25:27 S1ST2	Sequence 1 End State 000 State 0 is the Stop state for sequence 1 001 State 1 is the Stop state for sequence 1 010 State 2 is the Stop state for sequence 1 011 State 3 is the Stop state for sequence 1 100 State 4 is the Stop state for sequence 1 101 State 5 is the Stop state for sequence 1 110 State 6 is the Stop state for sequence 1 111 State 7 is the Stop state for sequence 1
28:30 S1ST1	Sequence 1 Start State 000 State 0 (Always)
31 S1ST0	Sequence1 Enable/Disable(2) Once enabled, this bit is disabled at the end of the Sequence 1. 0 Sequence 1 is disabled 1 Sequence 1 is enabled

1. Ideally, the start state for a sequence should be the state immediately after the stop state for the previous sequence. For example, if sequence 1 starts at state 0 and ends at state3, sequence 2 should start at state4.
2. Multiple active sequences such as Sequence 2, Sequence 3 and Sequence 4 must be programmed with Div4 option only.

76.5.1.3.3 State n goto control (SnGC)

SLU states are linked to each other based on if-then-else conditions. There are two conditions: true and false. A SLU state can remain in the same state or jump to any other state. The state goto control register is programmed to specify the GoTo a particular state from current state.

Figure 2013 shows the format of the SnGC register where the state number, n = 0 to 7.

Offset: 0x40 +n*0x1 (n= 0 to 7) Access: Read/Write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	NF			FE	NT			TE
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 2013. State n goto control (SnGC)

The SnGC register fields are described in *Table 1942*.

Table 1942. SnGC register field descriptions

Field	Description
24:26 NF	Next State on False Condition Defines the Goto next state in case of false condition from the present state. States may remain in the same state on a false condition if the same state is chosen as the next state. 000 Goto State 0 001 Goto State 1 010 Goto State 2 011 Goto State 3 100 Goto State 4 101 Goto State 5 110 Goto State 6 111 Goto State 7
27 FE	GOTO Control of State Under False Condition Defines the Goto control of a state under the false condition. If this bit is enabled, then the state moves to the next state based on bits[7:5]. If disabled, then this is the is end of a sequence when the sequence comes to this state. Actions associated with states under false condition are executed. Disabling this bit end the sequence under a false condition. 0 Disable Goto control 1 Enable Goto control
28:30 NT	Next State on True Condition Defines the Goto next state in case of true condition from the present state. States may remain in the same state on a true condition if the same state is chosen as the next state. 000 Goto State 0 001 Goto State 1 010 Goto State 2 011 Goto State 3 100 Goto State 4 101 Goto State 5 110 Goto State 6 111 Goto State 7
31 TE	GoTo Control of State Under True Condition Defines the Goto control of a state under the true condition. If this bit is enabled, then the State moves to the next state based on bits[3:1]. If disabled, then this is the end of a sequence when the sequence comes to this state. Actions associated with states under true condition are executed. Disabling this bit ends the sequence under a true condition. 0 Disable Goto control 1 Enable Goto control

76.5.1.4 SLU status register (SS)

The SLU status register indicates the states that have been hit/encountered in any of the active sequences. *Figure 2014* shows the format of the SS register.

Offset: 0x48 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	IM2	IM1	IM0	EM2	EM1	EM0	SAS4			SAS3
W							w1c	w1c	w1c	w1c	w1c	w1c	w1c			w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	SAS3		SAS2			SAS1			SH7	SH6	SH5	SH4	SH3	SH2	SH1	SH0
W	w1c		w1c			w1c			w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 2014. SLU status register (SS)

The SS register fields are described in *Table 1943*.

Table 1943. SS register field descriptions

Field	Description
6 IM2	Interrupt Match 2 This bit indicates the status of the match value on interrupt priority register of CPU2. 0 No match 1 Match occurred
7 IM1	Interrupt Match 1 This bit indicates the status of the match value on interrupt priority register of CPU1. 0 No match 1 Match occurred
8 IM0	Interrupt Match 0 This bit indicates the status of the match value on interrupt priority register of CPU0. 0 No match 1 Match occurred
9 EM2	Exception Match 2 This bit indicates the status of the processor exception vector match value of CPU2. 0 No match 1 Match occurred
10 EM1	Exception Match 1 This bit indicates the status of the processor exception vector match value of CPU1. 0 No match 1 Match occurred
11 EM0	Exception Match 0 This bit indicates the status of the processor exception vector match value of CPU0. 0 No match 1 Match occurred

Table 1943. SS register field descriptions (continued)

Field	Description
12:14 SAS4	Sequence 4 Active State Two consecutive read cycles are required to know the active state.These bits capture the active state of sequence 4 at the clock when these bits are read. The active state might be different in the previous clock and the next clock. 000 State0 is the active state 001 State1 is the active state 010 State2 is the active state 011 State3 is the active state 100 State4 is the active state 101 State5 is the active state 110 State6 is the active state 111 State7 is the active state
15:17 SAS3	Sequence 3 Active State Two consecutive read cycles are required to know the active state. These bits capture the active state of sequence 3 at the clock when these bits are read. The active state might be different in the previous clock and the next clock. 000 State0 is the active state 001 State1 is the active state 010 State2 is the active state 011 State3 is the active state 100 State4 is the active state 101 state5 is the active state 110 State6 is the active state 111 State7 is the active state
18:20 SAS2	Sequence 2 Active State Two consecutive read cycles are required to know the active state.These bits capture the active state of sequence 2 at the clock when these bits are read. The active state might be different in the previous clock and the next clock. 000 State0 is the active state 001 State1 is the active state 010 State2 is the active state 011 State3 is the active state 100 State4 is the active state 101 State5 is the active state 110 State6 is the active state 111 State7 is the active state
21:23 SAS1	Sequence 1 Active State Two consecutive read cycles are required to know the active state. These bits capture the active state of sequence 1 at the clock when these bits are read. The active state might be different in the previous clock and the next clock. 000 State0 is the active state 001 State1 is the active state 010 State2 is the active state 011 State3 is the active state 100 State4 is the active state 101 State5 is the active state 110 State6 is the active state 111 State7 is the active state

Table 1943. SS register field descriptions (continued)

Field	Description
24 SH7	State Hit 7 This bit is set when State7 is hit/encountered in any of the active sequences 0 State not hit 1 State hit
25 SH6	State Hit 6 This bit is set when State6 is hit/encountered in any of the active sequences 0 State not hit 1 State hit
26 SH5	State Hit 5 This bit is set when State5 is hit/encountered in any of the active sequences 0 State not hit 1 State hit
27 SH4	State Hit 4 This bit is set when State4 is hit/encountered in any of the active sequences 0 State not hit 1 State hit
28 SH3	State Hit 3 This bit is set when State3 is hit/encountered in any of the active sequences 0 State not hit 1 State hit
29 SH2	State Hit 2 This bit is set when State2 is hit/encountered in any of the active sequences 0 State not hit 1 State hit
30 SH1	State Hit 1 This bit is set when State1 is hit/encountered in any of the active sequences 0 State not hit 1 State hit
31 SH0	State Hit 0 This bit is set when State0 is hit/encountered in any of the active sequences 0 State not hit 1 State hit

76.5.1.5 SPU enable register (SE)

The SPU enable/disable register is used to enable and disable the SPU. If disabled, all inputs are masked. *Figure 2015* shows the format of the SE register.

Offset: 0x49 Access: Read/Write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	DIV	SED
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 2015. SPU enable register (SE)

The SE register fields are described in *Table 1944*.

Table 1944. SE register field descriptions

Field	Description
30 DIV	Div Option Select Div4 and Div1 programmable options based on number of active sequences. 0 Use Div4 option, when multiple up to four sequences are active or only one sequence is active which does not require the highest resolution 1 Use a Div1 option, when only one sequence (Sequence1) is active and highest resolution is required
31 SED	SPU Enable/Disable This bit should be set after programming all of the SPU configuration registers. 0 SPU is disabled 1 SPU is enabled

Note: Div option mapping with active sequences must be correctly programmed by the user. No hardware checks are implemented.

76.5.1.6 Sample and hold mechanism

SPU supports Div4/Div1 sample and state processing frequency (both are effectively the same). For the Div1 option, the sample and state processing frequency is the SPU clock frequency. Likewise, for the Div4 option, the sample and state processing frequency is 1/4th the SPU clock frequency.

The Div1 or Div4 is related to hold circuitry (among other circuits as well), but the sample circuitry is always Div1. Therefore, if Div4 is selected for hold circuitry, then four samples are taken and then latched into a hold circuitry (for the next four Div1 clocks, synchronized for all inputs).

This sample and hold mechanism is applicable for external signals only, but counters logic process differently. These external signals may be considered as selected conditions for any sequence, and for any of the states of any sequence. There is no difference based on state or sequence, as all inputs are treated the same regardless.

76.5.1.6.1 Functional limitations

If there are multiple true conditions in a Div4 period, only one condition is registered. Never are there any missed true conditions, except for extra true conditions in a Div4 period. Therefore, the user should select the resolution of SPU based on number of active sequences.

- If the signal changes faster than the sample and state frequency, then the SPU would probably only detect one event per the sample period (for example, if there are multiple true conditions in a Div4 period, only one condition is registered).
- If the signal changes faster than the sample and state frequency, then the SPU would only be able to process one event per state processing frequency.

76.5.1.6.2 Assumption

When using Div1, SPU counter operations are not available for state evaluation in the immediately next clock cycle. Therefore, special handling must be done in state evaluation during this delayed effect to assure that it does not create undesirable results. Refer to counter workaround description in *Section 76.2*.

When using Div4, however, there are no delayed effects when using SPU counter operations. This is because subsequent state evaluations are done four clocks apart, instead of every clock for Div1 selection. Therefore, counter workaround description in *Section 76.2*, becomes redundant with Div4 option.

76.5.1.7 SLU action unit registers

The action unit of the SPU is responsible for generating the action triggers supported by various actions, listed in *Section 76.2*. The action triggers are mapped to the events by programming the action registers. Each event can be programmed to trigger interrupts, watchpoints, trace signals, counter stop/reset/increment and many other general actions. *Table 1945* shows the bit mapping for all the possible actions divided in three groups: client based, counter based, system related.

Table 1945. SPU action decoder

Group bit [11:10]		Client/counter bit [9:5]		Action bit [4:1]			Enable bit [0]
ID	Name	ID	Name	ID	Name	Comment	
00	Client group	00000	CPU0	0000	No action	—	0=Stop 1=Start
		00001	CPU1	0001	DTM/gtm_gen0 trace	Valid for Cores, AHB, GTM and trace group actions	
		00010	CPU2	0010	OTM/gtm_gen1 trace		
		00011–00111	Reserved	0011	PTM/gtm_gen2 trace		
		01000	GTM	0100	WTM/gtm_gen3 trace		
		01001–01111	Reserved	0101	PMC1/NAR_supp_trig1	Valid for the cores as PMC Valid for NAR as the suppress trigger actions	For PMCn 0= Stop 1= Start For NAR suppress trigger = Don't care
		10000	NXMC1	0110	PMC2/NAR_supp_trig2		
		10001	NXMC2	0111	PMC3/NAR_supp_trig3		
		10010–10111	Reserved	1000	PMC4/NAR_supp_trig4		
		11000	NAR	1001	GTM_dtm_tr0	Valid for GTM	0=Stop 1=Start
		11001–11110	Reserved	1010	GTM_dtm_tr1		
				1011	GTM_ftm_tr0		
				1100	GTM_ftm_tr1		
		11111	Trace group(1)	1101–1111	Reserved	Valid for CPU cores and AHB	
01	Counter group	00000	Counter0	0000	No action	—	Don't care
		00001	Counter1	0001	Start timer/Incr counter	Valid for all counters/timers	
		00010	Counter2	0010	Stop timer		
		…	…	0011	Reset counter		
		01111	Counter15	0100	Capture		
		10000–11111	Reserved	0101–1111	Reserved		

Table 1945. SPU action decoder (continued)

Group bit [11:10]		Client/counter bit [9:5]		Action bit [4:1]			Enable bit [0]
ID	Name	ID	Name	ID	Name	Comment	
10	System group	00000	System action	0000	No action	—	Don't care
		00001–11111	Reserved	0001	Halt1_dci_evti	—	
				0010	Halt2_dci_evti	—	
				0011	—	—	
				0100	—	—	
				0101	—	—	
				0110	—	—	
				0111	Evto1	—	
				1000	Evto2	—	
				1001	NAR_sync_trigg	Valid for NAR	
				1010	Core INTR	—	
				1011–1111	Reserved	—	—
11	Reserved						

1. The valid actions for Trace group are OTM trace, PTM trace, DTM trace or WTM trace. No other actions can be selected with Trace group. The group can be defined using the Trace group configuration registers.

76.5.1.7.1 State n true action 0 (SnTA0)

The SPU can be programmed to enable up to four different actions against a true condition of every state. All the possible actions and their encoding are listed in *Table 1945*. A total of 12 bits are needed to code every action. The first two actions for true condition are defined in SnTA0 and the next two actions are defined in SnTA1.

Figure 2016 shows the format of the SnTA0 register where the state number, n = 0 to 7.

Offset: 0x01 + n*0x2 (n=0 to 7) Access: Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	ACTION2											
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	ACTION1											
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 2016. State n true action 0 (SnTA0)

The SnTA0 register fields are described in *Table 1946*.

Table 1946. SnTA0 register field descriptions

Field	Description
4:15 ACTION2	Action 2 to be performed for State n true condition The decoding of these bits is defined in *Table 1945*.
20:31 ACTION1	Action 1 to be performed for State n true condition The decoding of these bits is defined in *Table 1945*.

76.5.1.7.2 State n true action 1 (SnTA1)

The SPU can be programmed to enable up to four different actions against a true condition of every state. All the possible actions and their encoding are listed in *Table 1945*. A total of 12 bits are needed to code every action. The first two actions for true condition are defined in SnTA0 and the next two actions are defined in SnTA1. *Figure 2017* shows the format of the SnTA1 register where the state number, n = 0 to 7.

Offset: 0x02 + n*0x2 (n= 0 to 7) Access: Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	ACTION4											
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	ACTION3											
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 2017. State n true action 1 (SnTA1)

The SnTA1 register fields are described in *Table 1947*.

Table 1947. SnTA1 register field descriptions

Field	Description
4:15 ACTION4	Action 4 to be performed for State n true condition The decoding of these bits is defined in *Table 1945*.
20:31 ACTION3	Action 3 to be performed for State n true condition The decoding of these bits is defined in *Table 1945*.

76.5.1.7.3 State n false action 0 (SnFA0)

The SPU can be programmed to enable up to four different actions against a false condition of every state. All the possible actions and their encoding are listed in *Table 1945*. A total of 12 bits are needed to code every action. The first two actions for a false condition are defined in SnFA0 and the next two actions are defined in SnFA1. *Figure 2018* shows the format of the SnFA0 register where the state number, n = 0 to 7.

Offset: 0x11 + n*0x2 (n=0 to 7) Access: Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	ACTION2											
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	ACTION1											
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 2018. State n false action 0 (SnFA0)

The SnFA0 register fields are described in *Table 1948*.

Table 1948. SnFA0 register field descriptions

Field	Description
4:15 ACTION2	Action 2 to be performed for State n false condition The decoding of these bits is defined in *Table 1945*.
20:31 ACTION1	Action 1 to be performed for State n false condition The decoding of these bits is defined in *Table 1945*.

76.5.1.7.4 State n false action 1 (SnFA1)

The SPU can be programmed to enable up to four different actions against a false condition of every state. All the possible actions and their encoding are listed in *Table 1945*. A total of 12 bits are needed to code every action. The first two actions for false condition are defined in SnFA0 and the next two actions are defined in SnFA1. *Figure 2019* shows the format of the SnFA1 register where the state number, n = 0 to 7.

Offset: 0x12 + n*0x2 (n=0 to 7) Access: Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	ACTION4											
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	ACTION3											
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 2019. State n false action 1 (SnFA1)

The SnFA1 register fields are described in *Table 1949*.

Table 1949. SnFA1 register field descriptions

Field	Description
5:15 ACTION4	Action 4 to be performed for State n false condition The decoding of these bits is defined in *Table 1945*.
20:31 ACTION3	Action 3 to be performed for State n false condition The decoding of these bits is defined in *Table 1945*.

76.5.1.7.5 Trace group configuration registers (DTMTGC, PTMTGC, OTMTGC, WTMTGC)

There are four action trace groups and a configuration register for each trace group:

- DTM trace: clients are configured with the DTM trace group configuration (DTMTGC) register
- PTM trace: clients are configured with the PTM trace group configuration (PTMTGC) register
- OTM trace: clients are configured with the OTM trace group configuration (OTMTGC) register
- WTM trace: clients are configured with the WTM trace group configuration (WTMTGC) register

Each of these groups are specific to a trace. Any number of clients can be enabled for one trace using the action trace group.

Figure 2020 shows the format of the trace group configuration registers.

Offset: 0x21 (DTMTGC)
0x22 (PTMTGC)
0x23 (OTMTGC)
0x24 (WTMTGC)

Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	AHB2	AHB1
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	CPU2	CPU1	CPU0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 2020. Trace group configuration register format (DTMTGC, PTMTGC, OTMTGC, WTMTGC)

The trace group configuration register fields are described in *Table 1950*.

Table 1950. DTMTGC, PTMTGC, OTMTGC, WTMTGC field descriptions

Field	Description
14 AHB2	0 Disable trace for AHB2 1 Enable trace for AHB2
15 AHB1	0 Disable trace for AHB1 1 Enable trace for AHB1
29 CPU2	0 Disable trace for CPU2 1 Enable trace for CPU2
30 CPU1	0 Disable trace for CPU1 1 Enable trace for CPU1
31 CPU0	0 Disable trace for CPU0 1 Enable trace for CPU0

76.5.1.7.6 Interrupt status (INTS)

The status of the external interrupt request to the processor core through the JTAGM is registered in the Interrupt status (INTS) register. This status bit is cleared when the acknowledge is received from the JTAGM. *Figure 2021* shows the format of the INTS register.

Offset: 0x25 Access: Read/Write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	INT
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 2021. Interrupt status (INTS)

The INTS register fields are described in *Table 1951*.

Table 1951. INTS register field descriptions

Field	Description
31 INT	1 Interrupt status bit is cleared from JTAGM

76.5.1.8 Counters control n registers (CCTRLn)

There are 16 configurable counters/timers in the SPU. Configuration control of these counters is defined in the counter control registers. *Figure 2022* shows the format of the CCTRLn registers where the counter number, n = 0 to 15.

Offset: 0x26 + n*0x1 (n = 0 to 15) Access: Read/Write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	SW_RESET	PRESCALING			0	COUNT_INCR	MODE	ENABLE
W								0								
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 2022. Counters control n registers (CCTRLn)

The CCTRLn register fields are described in *Table 1952*.

Table 1952. CCTRLn register field descriptions

Field	Description
23	Reserved. This bit is for internal use only and must be set to 0.
24 SW_RESET	Reset Counter can be reset by software. 0 Software reset is disabled 1 Software reset
25:27 PRESCALING	Prescaling Divides the timer clock by a defined value. 000 No scaling, divide by 1 001 Divide by 4 010 Divide by 16 011 Divide by 32 100 Reserved 101 Reserved 110 Reserved 111 Reserved
29 COUNT_INCR	Count Increment 0 Increment the counter based on the actions generated from the sequences 1 Increment the counter based on the PMC input. The mapping of the PMC inputs with the counters is shown in *Figure 2034*.
30 MODE	Mode Select Select counter or timer operation. 0 Counter 1 Timer
31 ENABLE	Enable 0 Counter disabled 1 Counter enabled

76.5.1.9 Counters compare n registers (CCMPn)

A 32-bit comparator value can be programmed for each of the counters. This value is compared with the corresponding counter/timer to generate an event for the SPU. *Figure 2023* shows the format of the CCMPn registers where the counter number, n = 0 to 15.

Offset: 0x36 + n*0x1 (n = 0 to 15) Access: Read/Write

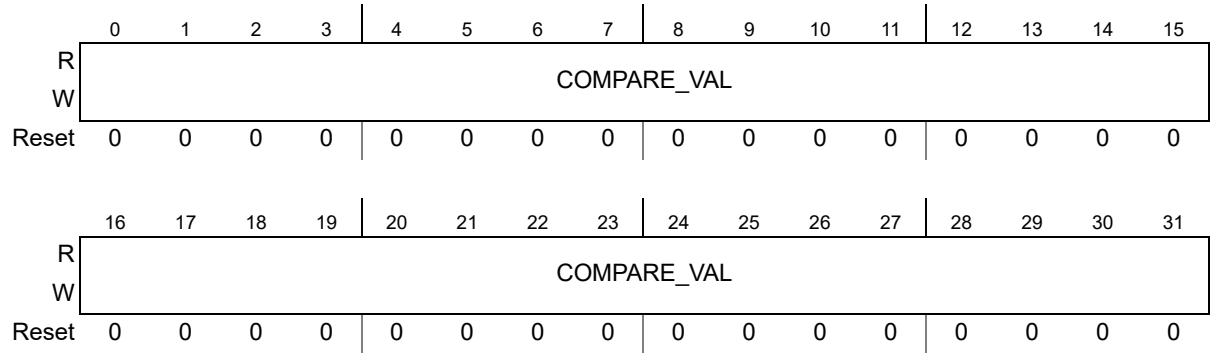


Figure 2023. Counters compare n registers (CCMPn)

The CCMPn register fields are described in *Table 1953*.

Table 1953. CCMPn register field descriptions

Field	Description
0:31 COMPARE_VAL	Compare value This value is compared with the corresponding counter/timer value to generate an event for the SPU.

76.5.1.10 Status registers

76.5.1.10.1 Counter compare status (CCOMS)

The counter compare status register shows the status of the comparisons for all the counters. The bits in this register can be cleared by writing a 1 to them. *Figure 2024* shows the format of the CCOMS register. All of the CC fields in the CCOMS register have the same description as shown in *Table 1954*.

Note: Counter compare values and other settings are not recommended to change dynamically, so w1c on CCOMS should be performed in either condition:

- *Counter is in disabled state (ENABLE bit of CCTRLn set to 0)*
- *SW-Reset bit (SW_RESET) is set*

Offset: 0x46 Access: Read/Write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	CC15	CC14	CC13	CC12	CC11	CC10	CC9	CC8	CC7	CC6	CC5	CC4	CC3	CC2	CC1	CC0
W	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 2024. Counter compare status (CCOMS)

The CCOMS register fields are described in *Table 1954*.

Table 1954. CCOMS register field descriptions

Field	Description
16 CC15	Counter 15 Compare 0 Counter status is not true 1 Counter status is true
17 CC14	Counter 14 Compare 0 Counter status is not true 1 Counter status is true
18 CC13	Counter 13 Compare 0 Counter status is not true 1 Counter status is true
19 CC12	Counter 12 Compare 0 Counter status is not true 1 Counter status is true
20 CC11	Counter 11 Compare 0 Counter status is not true 1 Counter status is true
21 CC10	Counter 10 Compare 0 Counter status is not true 1 Counter status is true
22 CC9	Counter 9 Compare 0 Counter status is not true 1 Counter status is true
23 CC8	Counter 8 Compare 0 Counter status is not true 1 Counter status is true
24 CC7	Counter 7 Compare 0 Counter status is not true 1 Counter status is true

Table 1954. CCOMS register field descriptions (continued)

Field	Description
25 CC6	Counter 6 Compare 0 Counter status is not true 1 Counter status is true
26 CC5	Counter 5 Compare 0 Counter status is not true 1 Counter status is true
27 CC4	Counter 4 Compare 0 Counter status is not true 1 Counter status is true
28 CC3	Counter 3 Compare 0 Counter status is not true 1 Counter status is true
29 CC2	Counter 2 Compare 0 Counter status is not true 1 Counter status is true
30 CC1	Counter 1 Compare 0 Counter status is not true 1 Counter status is true
31 CC0	Counter 0 Compare 0 Counter status is not true 1 Counter status is true

76.5.1.10.2 Counter overflow status (COS)

Whenever a counter overflow occurs, the corresponding counter overflow status register bit is set. The bits in this register can be cleared by writing a 1 to them. *Figure 2025* shows the format of the COS register. All of the CO fields in the COS register have the same description as shown in *Table 1955*.

Offset: 0x47 Access: Read/Write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	CO15	CO14	CO13	CO12	CO11	CO10	CO9	CO8	CO7	CO6	CO5	CO4	CO3	CO2	CO1	CO0
W	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 2025. Counter overflow status (COS)

The COS register fields are described in *Table 1955*.

Table 1955. COS register field descriptions

Field	Description
16 CO15	Counter 15 Overflow 0 Counter did not overflow 1 Counter overflow occurred
17 CO14	Counter 14 Overflow 0 Counter did not overflow 1 Counter overflow occurred
18 CO13	Counter 13 Overflow 0 Counter did not overflow 1 Counter overflow occurred
19 CO12	Counter 12 Overflow 0 Counter did not overflow 1 Counter overflow occurred
20 CO11	Counter 11 Overflow 0 Counter did not overflow 1 Counter overflow occurred
21 CO10	Counter10 Overflow 0 Counter did not overflow 1 Counter overflow occurred
22 CO9	Counter 9 Overflow 0 Counter did not overflow 1 Counter overflow occurred
23 CO8	Counter 8 Overflow 0 Counter did not overflow 1 Counter overflow occurred
24 CO7	Counter 7 Overflow 0 Counter did not overflow 1 Counter overflow occurred
25 CO6	Counter 6 Overflow 0 Counter did not overflow 1 Counter overflow occurred
26 CO5	Counter 5 Overflow 0 Counter did not overflow 1 Counter overflow occurred
27 CO4	Counter 4 Overflow 0 Counter did not overflow 1 Counter overflow occurred
28 CO3	Counter 3 Overflow 0 Counter did not overflow 1 Counter overflow occurred
29 CO2	Counter 2 Overflow 0 Counter did not overflow 1 Counter overflow occurred

Table 1955. COS register field descriptions (continued)

Field	Description
30 CO1	Counter 1 Overflow 0 Counter did not overflow 1 Counter overflow occurred
31 CO0	Counter 0 Overflow 0 Counter did not overflow 1 Counter overflow occurred

76.5.1.10.3 Counter capture status (CCAPS)

Whenever a counter value is captured, the corresponding capture status register bit for that counter is set. If the counter capture value causes a FIFO overflow, the captured value is lost and the status bit is not set. The bits in this register can be cleared by writing a 1 to them. *Figure 2026* shows the format of the CCAPS register.

Offset: 0x48 Access: Read/Write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	CC15	CC14	CC13	CC12	CC11	CC10	CC9	CC8	CC7	CC6	CC5	CC4	CC3	CC2	CC1	CC0
W	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 2026. Counter capture status (CCAPS)

The CCAPS register fields are described in *Table 1956*.

Table 1956. CCAPS register field descriptions

Field	Description
16 CC15	Counter 15 Captured 0 Counter not captured 1 Counter captured
17 CC14	Counter 14 Captured 0 Counter not captured 1 Counter captured
18 CC13	Counter13 Captured 0 Counter not captured 1 Counter captured
19 CC12	Counter 12 Captured 0 Counter not captured 1 Counter captured

Table 1956. CCAPS register field descriptions (continued)

Field	Description
20 CC11	Counter 11 Captured 0 Counter not captured 1 Counter captured
21 CC10	Counter 10 Captured 0 Counter not captured 1 Counter captured
22 CC9	Counter 9 Captured 0 Counter not captured 1 Counter captured
23 CC8	Counter 8 Captured 0 Counter not captured 1 Counter captured
24 CC7	Counter 7 Captured 0 Counter not captured 1 Counter captured
25 CC6	Counter 6 Captured 0 Counter not captured 1 Counter captured
26 CC5	Counter 5 Captured 0 Counter not captured 1 Counter captured
27 CC4	Counter 4 Captured 0 Counter not captured 1 Counter captured
28 CC3	Counter 3 Captured 0 Counter not captured 1 Counter captured
29 CC2	Counter 2 Captured 0 Counter not captured 1 Counter captured
30 CC1	Counter 1 Captured 0 Counter not captured 1 Counter captured
31 CC0	Counter 0 Captured 0 Counter not captured 1 Counter captured

76.6 Functional description

76.6.1 Input Mux unit

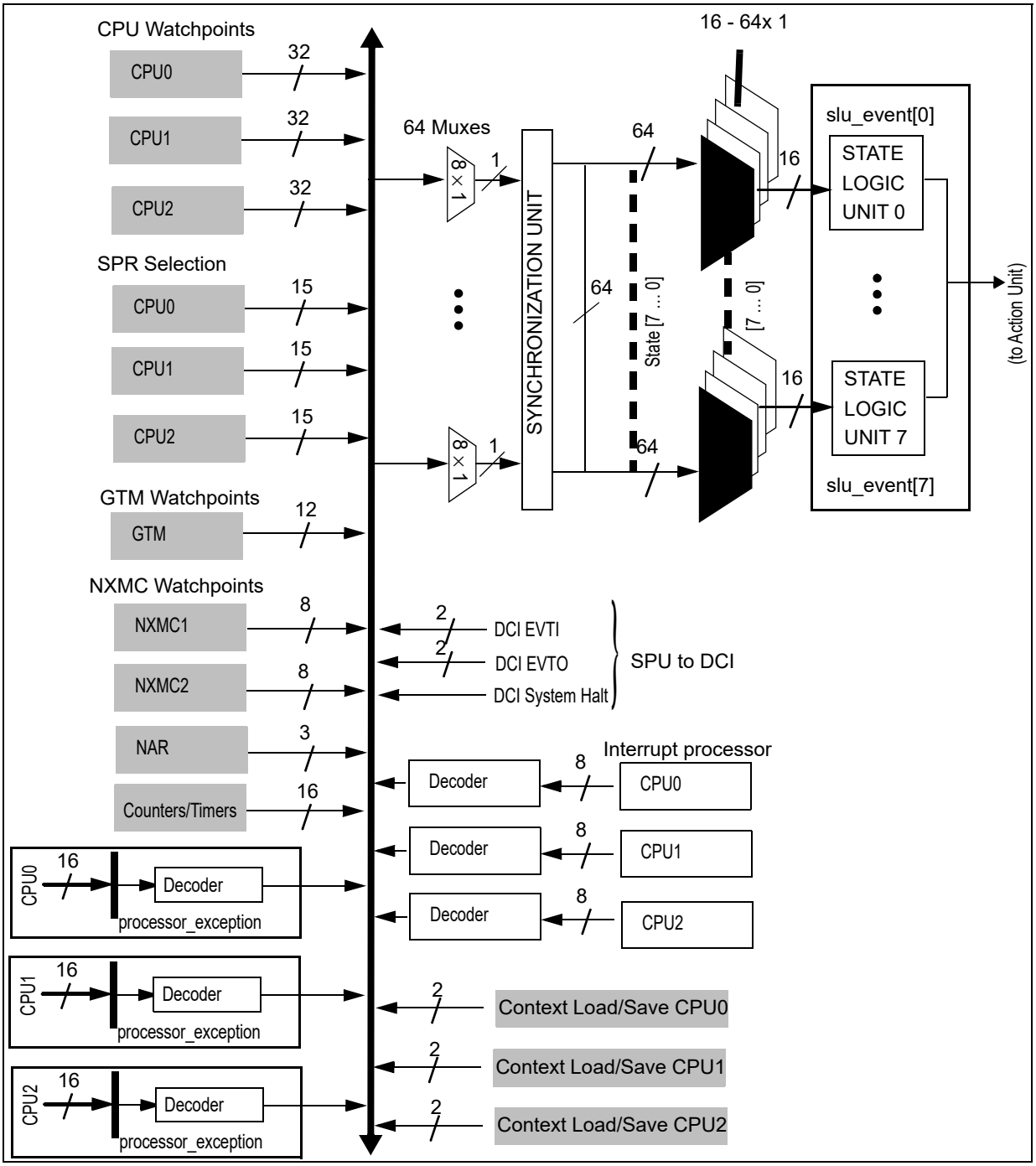
The SPU receives numerous inputs in the form of complex triggers from various clients to create a predefined sequence. One state logic can accommodate 16 inputs at one point of time. Therefore, an input Muxing unit becomes essential for reducing the debug and performance events to a manageable set of 16 inputs for each SLU. In addition to client

input signals, there are certain performance events which are fed directly to the different counters inside the SPU.

The selection of 16 inputs for each state logic is done at two level of muxing both of which are shown in *Figure 2027*.

- Level 1 Muxing: At this level, there are 209 inputs. Out of these possible inputs, 63 inputs are selected. This selection is done using an 8:1 Mux. Therefore, sixty-four 8×1 muxes are required. The set of 63 watchpoints are passed to the second level of muxing. The selection mechanism is shown in *Figure 2027*.
- Level 2 Muxing: At this level, 16 inputs are selected out of the 63 possible inputs for each SLU. These 63 inputs consist of multiple set of 8 watchpoints from different clients, processor exception signals, interrupt from clients, and event IN/OUT signals. In addition, certain events from counters are used as feedback in the present state logic of each state.
  At level two there are 16 banks of 64:1 muxes for the selection of 16 inputs for one state logic. The inputs to these Muxes are expected to be somewhat unique. In general, care should be take to choose signal assignments that allow for key use cases to be achieved without encountering concurrency restrictions. Since there are eight such states, 16×8 banks of 64:1 mux are required. The selection mechanism is shown in *Figure 2027*.

Figure 2027. Input Mux unit block diagram



76.6.2 Processor exception vector encoding

Active-high input signals from the CPU indicate the particular exception being processed when the CPU exception enable signal is asserted. These signals can be used for debug triggering mechanisms. The SPU captures the CPU exception vector whenever the CPU exception enable signal is asserted and the SPU uses the exception vector as an input

condition as shown in *Table 1957*. The CPU exception vector values from the CPUs must be byte swapped before they are programmed to the CPUn field of the CnPEVP (n = 0 to 2) registers as shown in *Table 1957*.

Table 1957. Processor exception processing conditions

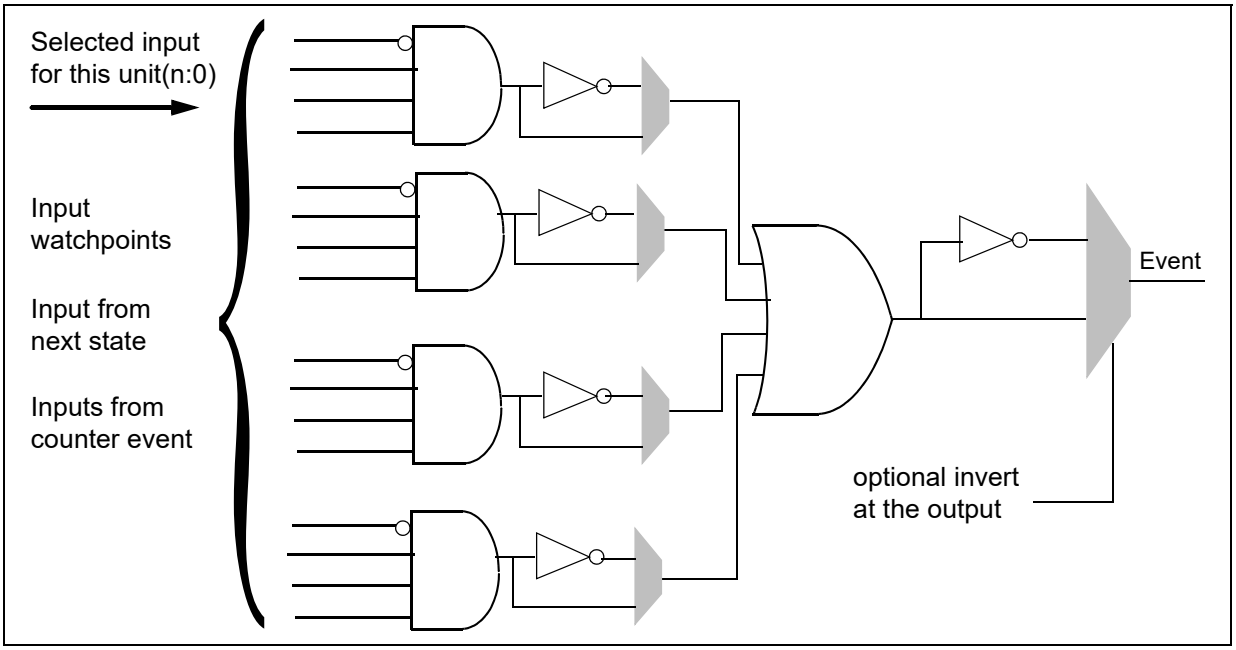
CPU exception vector	CnPEVP register CPUn field value	Exception condition
0x0000	0x0000	Critical Input: autovectored
0x0010	0x1000	Machine check
0x0020	0x2000	Data storage
0x0030	0x3000	Instruction storage
0x0040	0x4000	External input: autovectored
0x0050	0x5000	Alignment
0x0060	0x6000	Program trap
0x0061	0x6100	Program illegal
0x0062	0x6200	Program privileged
0x0070	0x7000	Performance monitor
0x0080	0x8000	System call
0x0090	0x9000	Debug
0x00A0	0xA000	EFPU data exception
0x00B0	0xB000	EFPU round exception
0xVVVV3, low two bits = 01	0xVVVV3, high two bits = 01	External input: vectored
0xVVVV3, low two bits = 01	0xVVVV3, high two bits = 01	Critical input: vectored

76.6.3 State logic unit (SLU)

Complex triggers and system performance monitor functions are implemented in the SLU. The SPU creates debug events based upon states in a sequence. A sequence can consist of a single state, or any number of states up to 8. Each state consists of combinational logic allowing AND/OR operations on inputs from the trigger source unit. Single or multiple (up to 8) actions can be triggered by a state machine.

A logic diagram of one SLU state is shown in *Figure 2028*.

Figure 2028. SLU block diagram



Each state consists of AND gates, the outputs of which are ORed together.

- Each AND gate can have up to 4 inputs, selected from the trigger source unit by the user.
- The user has an option to invert one input signal into the AND gate.
- The OR gate has a user selectable option to invert the output.
- The output of the OR gate is used as the trigger to move to the next state in the sequence.

The SLUs are responsible for performing operations on the selected input events to produce derived events. Each SLU has a control register that manages how the various inputs are combined to produce a derived event.

An event is triggered from each state based on all the possible selected inputs for each state. This event is used to trigger an Action.

Note: With only one optional inversion at the input of each AND gate and optional inversion at the output of the OR gate, many combinations are not possible.

76.6.4 Sequence formation

Event sequencing can be accomplished with the if-then-else operations. The next event can be fed back as a necessary condition for the current event. For example, event5 can be designated as a necessary condition for event4 in case test condition fails in event5. Event5 can be in turn designated as necessary for event6 in case test condition passes, and so on. In this way, events can be chained together to create a sequence.

To create complex triggers, a configurable state machine is implemented. This allows the user to join states together with if-then-else operations to create a sequence. The SPU supports up to four simultaneous sequences, however only eight states are supported regardless of the number of active sequences, and each state can only be used in one unique sequence.

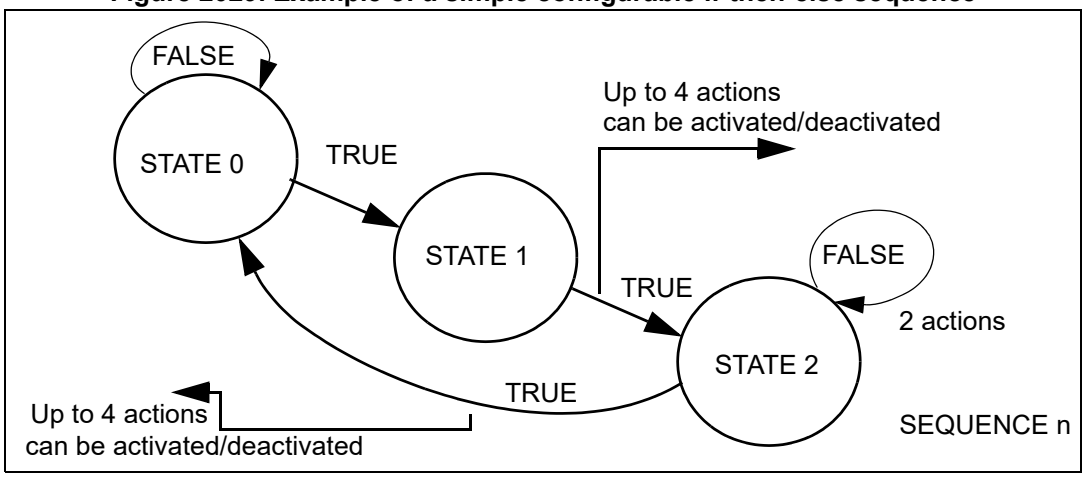
For example, the following state configurations are possible:

- 1 active sequence, with 8 states
- 2 active sequences, each with 4 states
- 2 active sequences, the first with 3 states and the second with 5 states
- 4 active sequences, each with 2 states
- 4 active sequences, each with 1 state

Each sequence has the following options which are shown in *Figure 2029*:

- The ability to optionally trigger one or more actions based on a true condition from any state in the sequence.
- The ability to optionally trigger one or more actions based on a false condition from any state in the sequence.
- Each state in a sequence has the ability to route to another state on a true condition from the state logic, and the ability to route to a different state based on a false condition from the state logic.

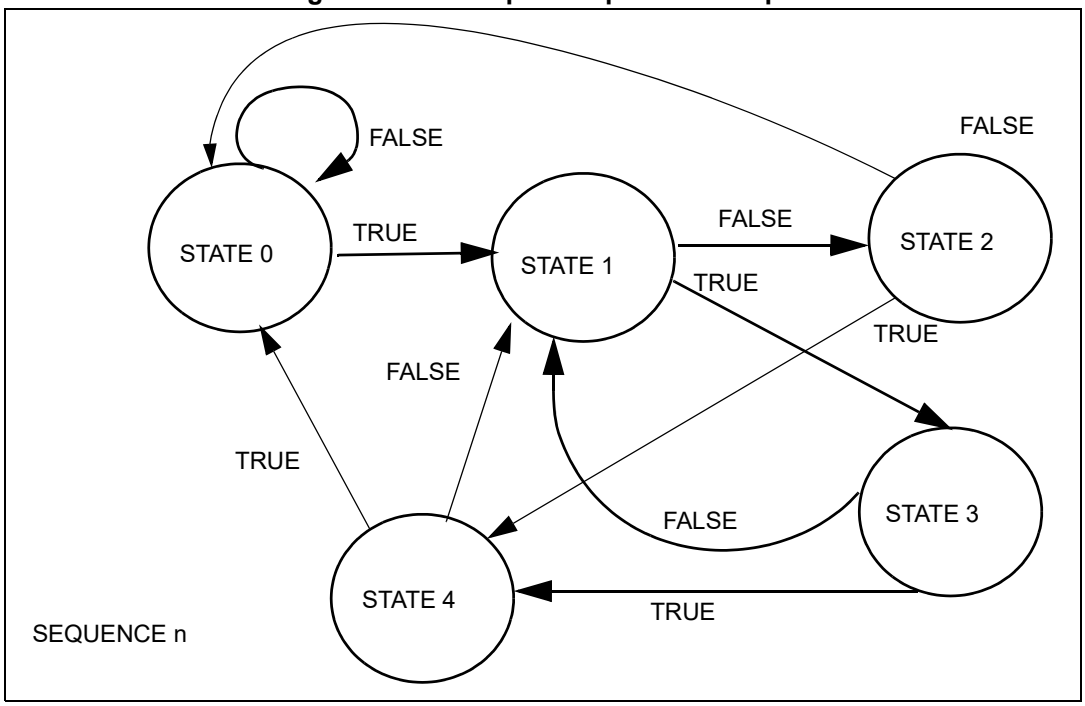
Figure 2029. Example of a simple configurable if-then-else sequence



76.6.4.1 Complex sequence example

A user can create a complex sequence using five states as shown in the state transition diagram in *Figure 2030*.

Figure 2030. Complex sequence example



76.6.4.2 Typical sequence example

For this example, the user would like to create a timestamp in the trace stream each time a function is entered and stop the trace signal on the 12th occurrence of the function. This use case can be implemented as follows:

- State 0 takes an instruction address compare as its input. On a match of the instruction address to a specific function, called function_x, the sequence moves to State 1. Upon this action, State0 is configured to insert a timestamp into the trace stream and also increment an SPU counter, say counter[0].
- The sequence is now in State 1. State 1 takes an input from counter[0]. Counter[0] has been pre-configured to create a match event when the counter value is equal to 12.
  - If the value of counter[0] is less than 12, then the sequence moves back to State0.
  - If the value of couner[0] is equal to 12, then the sequence completes and an action from State1 disables the trace signal.

76.6.4.3 Sequence status register description

A status register providing feedback on each sequence is provided to allow debug tools to identify the source of a debug event from the SPU. The status register consists of two field types:

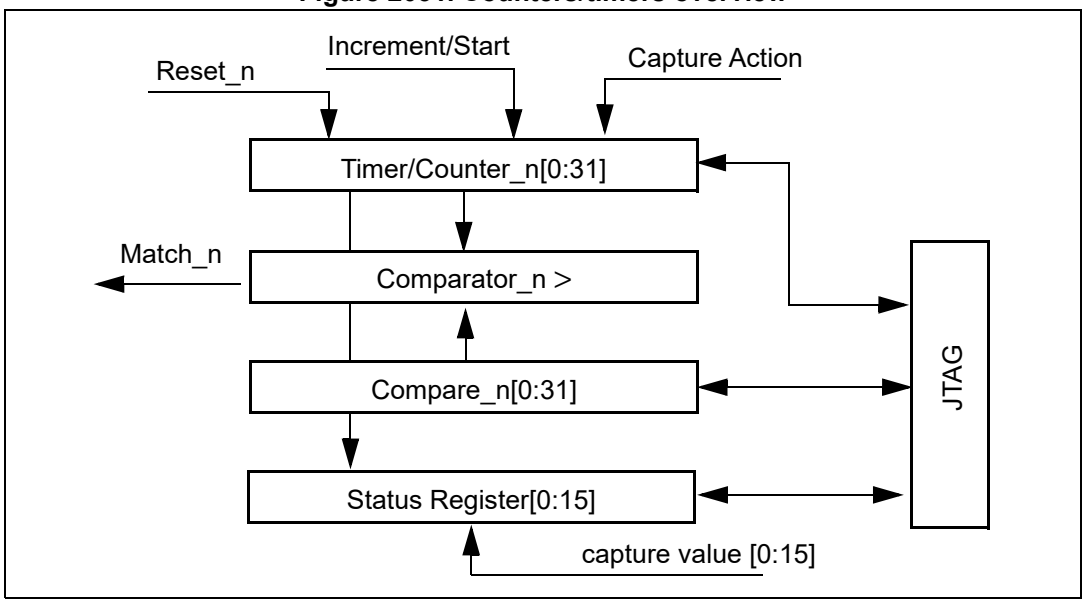
- States: each state has a sticky bit that is set upon a true match of the state logic. This sticky bit is cleared upon a write to this register
- Sequence status: each sequence has a field indicating the active state of the state machine. This is a read only field

76.6.5 Performance counters/timers unit

The 32-bit SPU counters may be used to delay events, count distances, count latencies or count events as shown in *Figure 2031*. Each counter can be configured as either a counter or a timer via the configuration register. The basic counting operation is straightforward: once enabled (default is disabled mode), the counter increments using the platform clock whenever the selected input is driven to a logic 1.

Access (via the JTAG interface) to each of the counter/timer registers allows the configuration of the timer/counter, the compare registers and global SPU timer/counter status registers. These registers are accessible via the JTAG interface even when the SPU is in disabled mode.

Figure 2031. Counters/timers overview



76.6.5.1 Counter events and actions

Each counter is capable of detecting one event condition, such as compare value match. The compare value match is used for triggering actions in the SPU. Each counter can be reset in response to a reset action generated from the SPU. Capture registers are available for capturing the counter state in response to a SPU trigger condition. This capture information is updated in the status register, which is accessible to users via JTAG interface. These status registers are type write 1 to clear (W1C).

There are two classes of signals, actions and events:

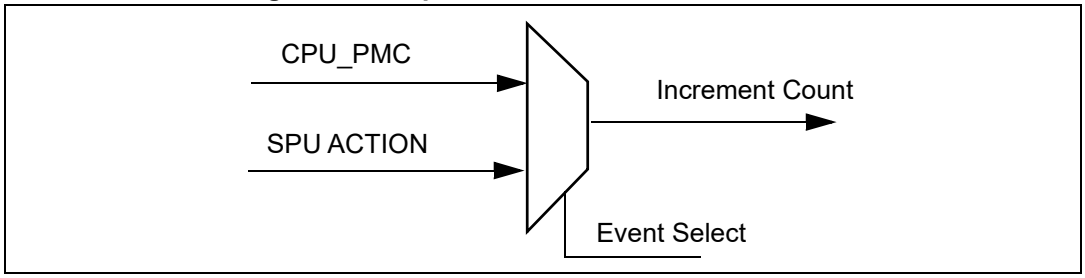
- Actions: each timer/counter has an interface from the action unit to start/increment and reset as appropriate. The start/increment signal is a single signal, with the operation selected in the SPU configuration registers. The generation of the appropriate count/reset signal is generated in the SPU.

  On an action to insert the timer/counter value into the trace stream, the value stored in the timer/counter is buffered inside the SPU and routed to the appropriate SoC block to insert the value into the trace stream one by one.
- Event: on a match event where a timer/counter value matches the value of the respective compare register, a signal is sent to the trigger source unit to indicate this.

76.6.5.2 Counter input selection at SPU level

Counter increment inputs derive from the SPU action or direct from the CPU performance measurement counters (CPU_PMC), as shown in *Figure 2032*. These inputs include signals from the primary SPU muxes or SPU events of counters.

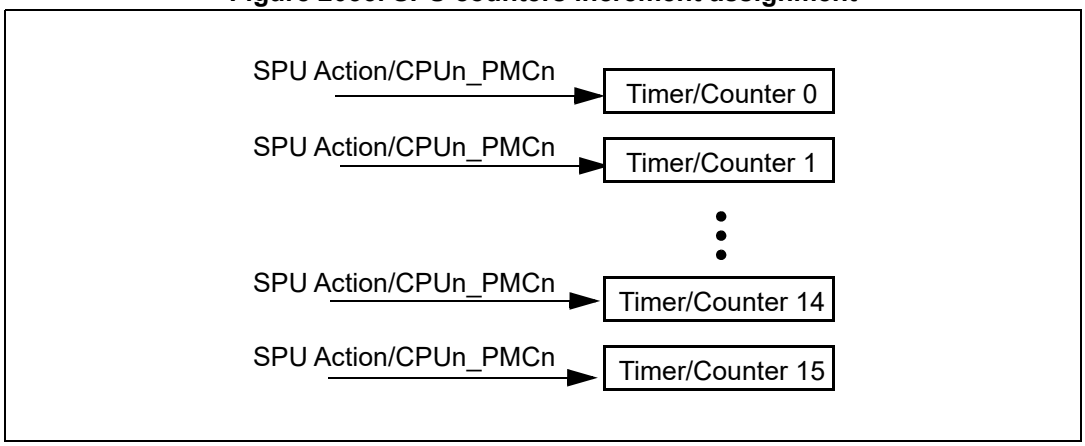
Figure 2032. nput selection for count increment



76.6.5.3 Counters increment assignment

When configured as a counter, the counter has an increment signal. This can optionally be routed from the SPU action unit, or directly from a CPU performance counter signal. The assignment of which signal is used to increment a counter is handled in the SPU, and is configurable by user. The assignment of increment signals to counters is shown in *Figure 2033*.

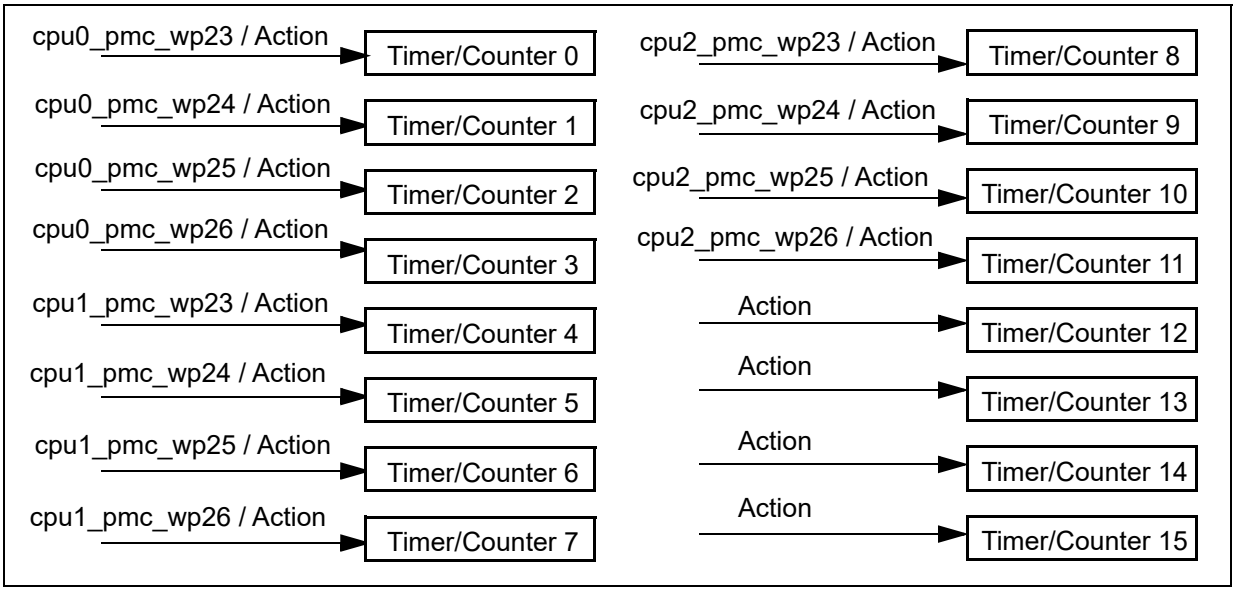
Figure 2033. SPU counters increment assignment



76.6.5.4 CPU PMC mapping to counters

Each of the CPUs in the system has four performance measurement counters (PMC). The outputs (overflow signals) from these counters (of CPUn) are routed to the SPU debug unit. The counters inside the SPU can be configured to count the pulses from the PMCs. There is a one-to-one mapping between the PMC inputs and the counters that they trigger. Their mapping is shown in the *Figure 2034*.

Figure 2034. CPU PMC mapping inside the SPU unit



76.6.6 Action unit

The action unit block of the SPU is responsible for the generating the action triggers supported by various actions. A state has true and false (if-else-than) condition. Four simultaneous actions can be generated under true/false condition of a particular state. The list of the actions is given in the *Section 76.2*. The actions triggers are mapped to the events by programming the set of SPU group action control for events registers**.** Each event can be programmed to trigger interrupts, timestamp watch points, trace signals, PMCs, start/reset/increment counters and many other general actions.

76.6.7 Optional trace group

User defined trace group can be defined through the configuration register. Configuration registers, defined in *Table 1950*, are available where the user can group the different traces from same or different clients.

76.6.8 Format for CKSRC & CKDATA Values

The format of the CKSRC is defined for the SPU Peripheral (as In-Circuit trace client):

Table 1958. CKSRC format

ICT CLIENTS	CKSRC Value
SPU Counter Trace Client	b00_0000
Reserved	b00_0001–b11_1111

The format for the CKDATA field for the SPU Counter trace client is:

Table 1959. CKDATA format

CKDATA	
CKDATA[CTR_Index]	CKDATA[CTR_VALUE]
4 bits	32-bits

77 Buddy Device—JTAG Master (JTAGM)

77.1 Introduction

The JTAG Master (JTAGM) is a module that is able to act as JTAG master inside the device. The module has a parallel interface that can exchange data with another serial communication module (such as the LFAST module) or via customer software.

The data transferred to this module is transformed to produce TCK, TMS, TDI and TRST outputs and to accept TDO inputs. The JTAGM is connected in the device to allow these five signals to connect to the JTAGC as if the JTAG data is coming from an outside tool. The JTAGM generates all required JTAG scan chains to allow software and high speed serial communication access to all JTAG mapped resources.

77.1.1 Overview

The JTAGM is the master to drive JTAG signals from within the device. It has options to receive parallel data from software through the IPS interface or from the LFAST through the parallel interface. The JTAGM has the capability to differentiate between data from software and LFAST. It then sends this data on TDI, TMS and TCK to the DCI. The JTAGM also receives serial data through TDO from the DCI and transfers this data to the LFAST through another parallel interface. The JTAGM has the following registers:

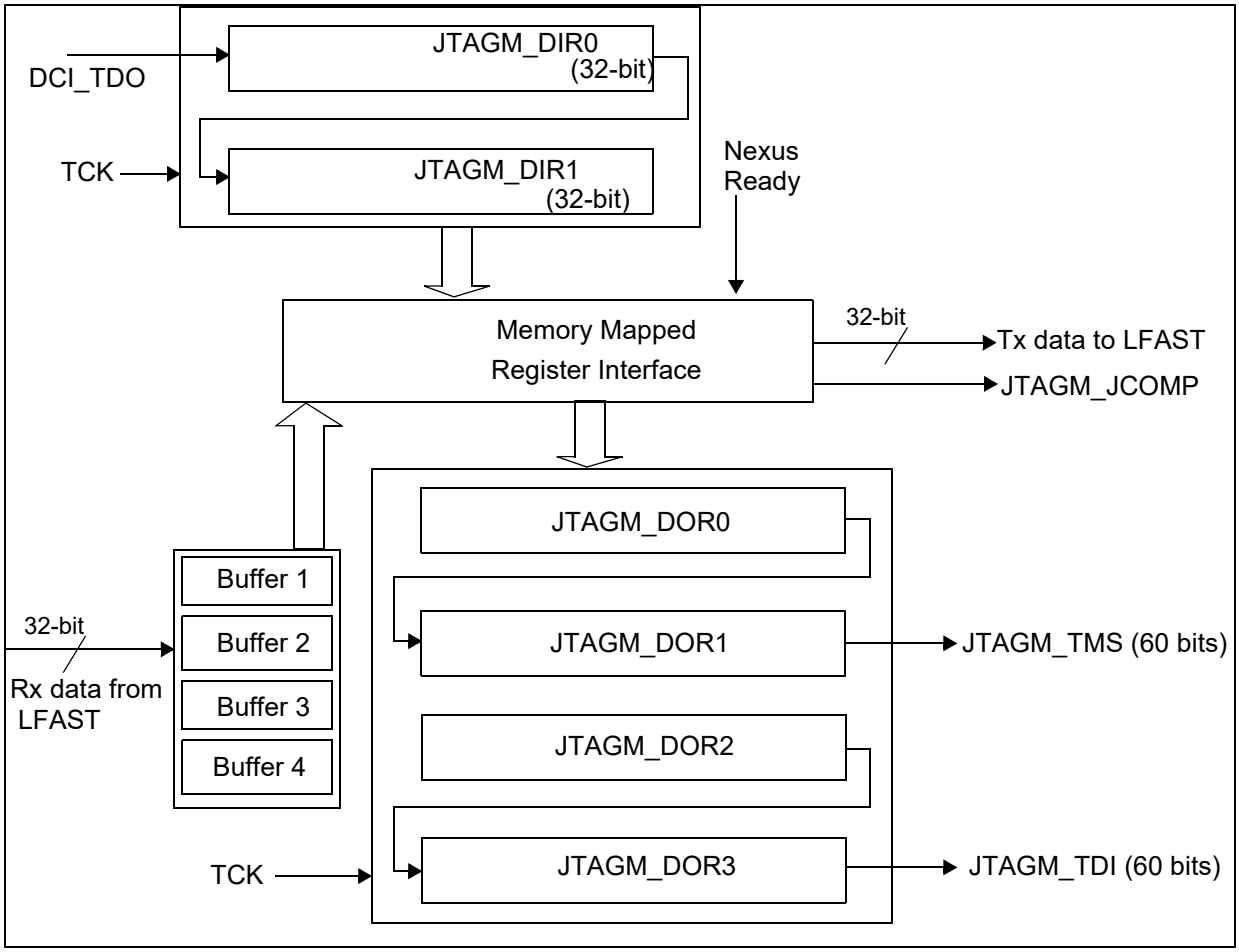
- Configuration register
- Status register
- Four data output registers
- RXCRC receive register
- Two data input registers

All these registers, described in detail in *Section 77.4: Memory map and register definition*, are memory mapped and can be accessed by software.

The clocks for the JTAGM are derived from the system clock. The JTAGM also samples the ready signal from the Nexus, to provide a more efficient handshake to the external tool to read and write any memory mapped address location within the device.

Figure 2035 shows the block diagram of the JTAGM.

Figure 2035. JTAGM block diagram



77.1.2 Feature description

The main features of the JTAGM are the following:

- Provides efficient handshake to the external tool to read any memory mapped address location within the device
- Provides software the option to write data for driving JTAG
- Receives/provides JTAG data from/to the LFAST

77.2 Functional description

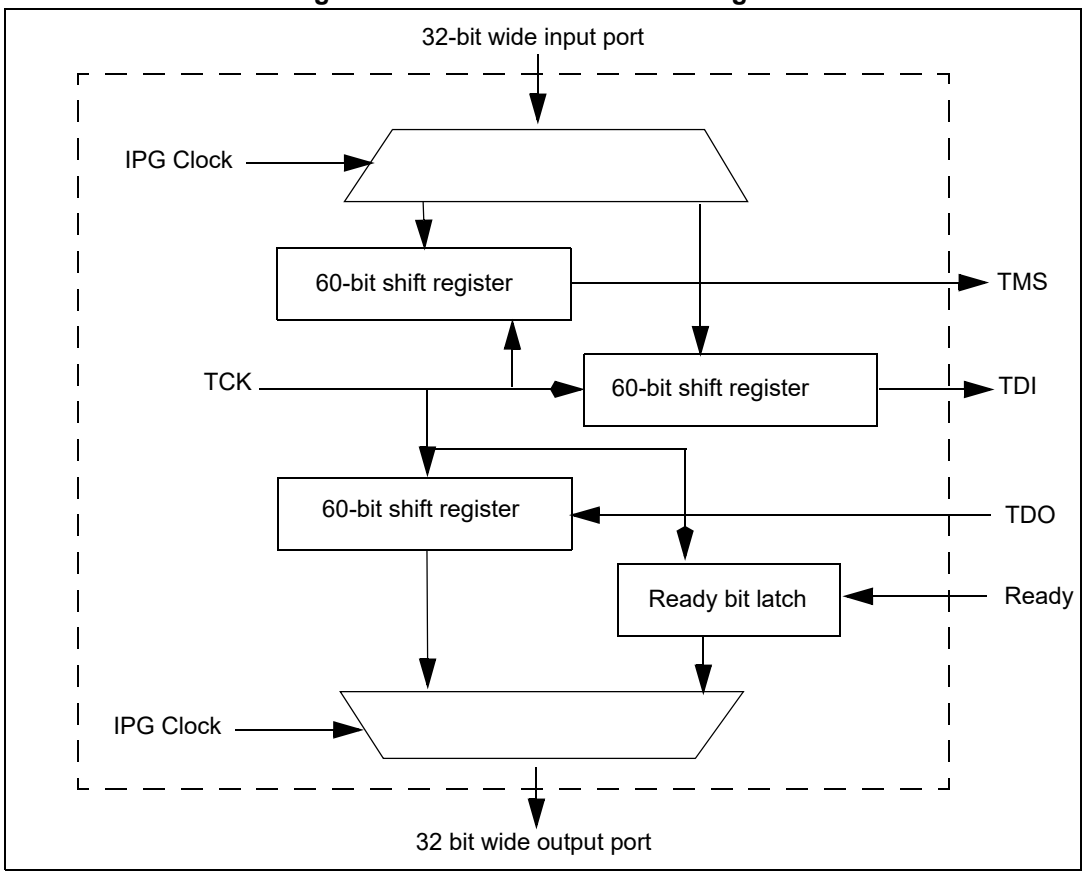
77.2.1 Module architecture

The JTAGM is simple in concept. TCK is generated from the device system clock. Data is pushed to the modules via a 32-bit wide parallel interface. The data is in the form of 60 bits of TMS data and 60 bits of TDI data. This data is the logical state to be driven onto the TMS and TDI output pins on each of the 60 TCK cycles. During these 60 TCK cycles, the TDO pin is sampled and the 60 bits of sampled data is transferred to the 32-bit wide module interface. This concept allows complete flexible generation on TMS and TDI data and

capture of TDO data. The only limitation is that JTAG signals can only be generated in 60 TCK cycles packets. Extra cycles are wasted in idle cycles at the end of a scan chain.

Figure 2036 shows the flow diagram of the LFAST to DCI connection.

Figure 2036. LFAST-to-DCI flow diagram



77.2.2 JTAGM JCOMP usage in LFAST

JTAGM is used to drive JTAG sequences in software mode and in LFAST mode. The JTAGM_MCR[jtagm_JCOMP] bit is provided to reset all TAP controllers by generating a JCOMP signal to JTAGC.

The guidelines to be considered for the usage of JTAGM_MCR[jtagm_JCOMP]:

- Do not clear the JTAM_MCR[jtagm_JCOMP] bit while in LFAST mode as this operation is not guaranteed
- If for some reasons during the LFAST mode there is a loss of communication, one of the following should be done (instead of clearing the jtagm_JCOMP bit):
  - Generate the escape sequence and then, once the device is back in JTAG mode, generate a test-logic-reset state via the TCK and TMS signals. This resets all TAP controllers but does not affect the client debug configuration
  - Generate the escape sequence and then, once the device is back in JTAG mode, force the JCOMP pin low. This resets all TAP controllers as well as the client debug configuration

Note: The escape sequence is generated by forcing LFAST LVDS ports RX+ and RX- to zero.

77.2.3 JTAGM data error detection

To improve the integrity of the JTAG operation, data passed to and from the JTAGM includes an 8-bit CRC only in LFAST mode. The data sent to the JTAGM consists of 60 bits of TMS data, 60 bits of TDI and an 8-bit CRC. The JTAGM recalculates the 8-bit CRC for the first 120 bits of data and compares the result with the received CRC. If the two match, the data is assumed to be valid and the JTAGM converts this data into a valid JTAG frame.

If the CRCs do not match, the module sets the CRC error status bit (CRC_err) in the JTAGM status register (JTAGM_SR) and sends a standard 96-bit response frame to the LFAST channel B, with all 60 data bits set to 0. The CRC error status bit is automatically cleared as soon as the response frame has been sent to the LFAST.

Data sent from the JTAGM to the LFAST also includes an 8-bit CRC. The data sent from the JTAGM is in the form of a 96-bit frame consisting of 60 bits of TDO data + 4 bits of zero-padding + 32 bits of the status register. The 8-bit CRC value is calculated for the first 88 bits of the 96-bit frame and inserted into the least significant 8 bits of the status register; which is also the last 8 bits of the 96-bit frame. The tool is responsible for checking the CRC of the received data and taking corrective action.

77.2.4 JTAGM message tagging

The JTAGM, in conjunction with the LFAST, is pipelined to allow several JTAGM messages to be sent from the tool before any response is received back. This allows better utilization of the LFAST bandwidth. To enable the tool to associate individual messages with the correct response, the messages are tagged. The tool sends data messages on LFAST channels E, F, G and H in sequence. The JTAGM module returns responses on the same channel they were received. For example, if the tool sends a data command on channel E, JTAGM returns the response on channel E; if the tool sends a data command on channel F, the JTAGM returns the response on channel F; and so on. This way, the tool can ensure a response was generated by a specific command.

Configuration commands to write/read the configuration and status registers within the JTAGM are sent on channel A and response data is also sent back on channel A.

77.2.5 JTAGM Ready signal

The Nexus Read/Write Access (RWA) module allows an external tool to read and write any memory mapped address location within the device via JTAG commands. Read accesses to memory take a finite amount of time and a JTAG read of the data register too soon results in erroneous data being read back. There is a bit in a JTAG register to show the read data is available. In conventional JTAG mode, the status of this ready bit is also provided back to the tool via a dedicated pin to allow the tool to know data is ready and the JTAG read register can be accessed. With the LFAST interface providing pipelined data to the JTAGM, this technique is not available. Two interlocked mechanisms are provided within the JTAGM that use this ready bit to allow optimized data reads without ready bit errors.

77.2.5.1 Status register ready bit

A bit is provided in the JTAGM status register to indicate the status of the ready signal. To provide this ready signal functionality, the ready signal is monitored. If the ready signal has

toggled between the start of the previous 60 TCK shift period and start of the current 60 TCK shift period, the ready bit in the JTAGM status register is set. Otherwise the ready bit in the status register is cleared.

The entire content of the 32-bit status register is appended to the 60 bits of JTAG TDO data captured along with 4 bits of 0-padding and sent as output on the parallel output port as a 96-bit payload on channel A. This provides back to the tool, an indication that the data read in the current transaction is valid.

77.2.5.2 Inter-JTAG frame gap timer

To operate in conjunction with the status register ready bit, a configurable timer is provided to force a programmable gap between the end of one 60-bit JTAG frame and the start of the next. This timer is configured via the JTAGM configuration register from 1 to 64 TCKs in 1 TCK intervals. Furthermore, if the ready signal asserts during this inter-frame gap period, the gap timer is aborted and the next JTAG frame starts immediately.

77.2.6 $\overline{\text{EVTO}}$ and $\overline{\text{EVTI}}$ signals

The standard Nexus EVTI (Event In) and EVTO (Event Out) signals are optionally present on dedicated package pins and are used by an external tool in conjunction with the JTAG interface to pass information, events and triggers between the tool and the MCU.

The JTAGM as well as providing a higher speed alternative to the standard JTAG interface, also allows control and status of the EVTI and EVTO signals to be embedded into the serial protocol. The EVTI0 and EVTI1 signals can be asserted by writing control bits in the Module Configuration Register (JTAGM_MCR). The JTAGM can be configured by bits in the Module Configuration Register (JTAGM_MCR) to respond to a rising/falling/both edge transition of either EVTO0 or EVTO1. The response is either an unsolicited serial message to the tool when in LFAST mode (DTM = 0); or a CPU interrupt when in software mode (DTM = 1).

The unsolicited LFAST message is sent on LFAST channel B to allow the tool to distinguish the message from normal JTAG response data (channels E, F, G & H) and Config Command Responses (channel A). The CPU interrupt is gated by an Interrupt enable bit, in MCR and reported by Interrupt Status flags in the Status Register (JTAGM_SR). The status bits stay set until cleared by writing to the clear bits in the JTAG_SR.

Asserting the evti0_assert bit in the JTAGM_MCR register can trigger debug mode in the DCI. Asserting the evti1_assert bit in the JTAGM_MCR register can trigger debug mode in the DCI. The DTM bit in the JTAGM_MCR register can put the DCI in LFAST or SW mode.

Note: EVTI generation and EVTO detection bits of JTAGM can be used by debugger to detect EVTI assertion and to generate EVTO. This method is useful when EVTI/O pins are not available on SoC PADs.

77.2.7 JTAGM configuration and status monitoring

The JTAGM module requires some level of configuration and status monitoring. The JTAGM includes a 32-bit configuration register (JTAGM_MCR) and a 32-bit status register (JTAGM_SR, which includes the ready bit mentioned in *Section 77.2.5.1: Status register*

ready bit). The configuration register is accessed via the parallel input port by a 128-bit payload arriving on channel A. The 128-bit payload is made from the following:

- 32-bit mask of which bits within the configuration register should be written
- 32-bit value to be written to the configuration register
- 24-bit mask of which bits within the status register should be written
- 8 bits of zero-padding
- 24-bits of data to be written to the status register
- 8-bit CRC (The least significant 8 bits of the status register cannot be written because they contain automatically generated CRC data)

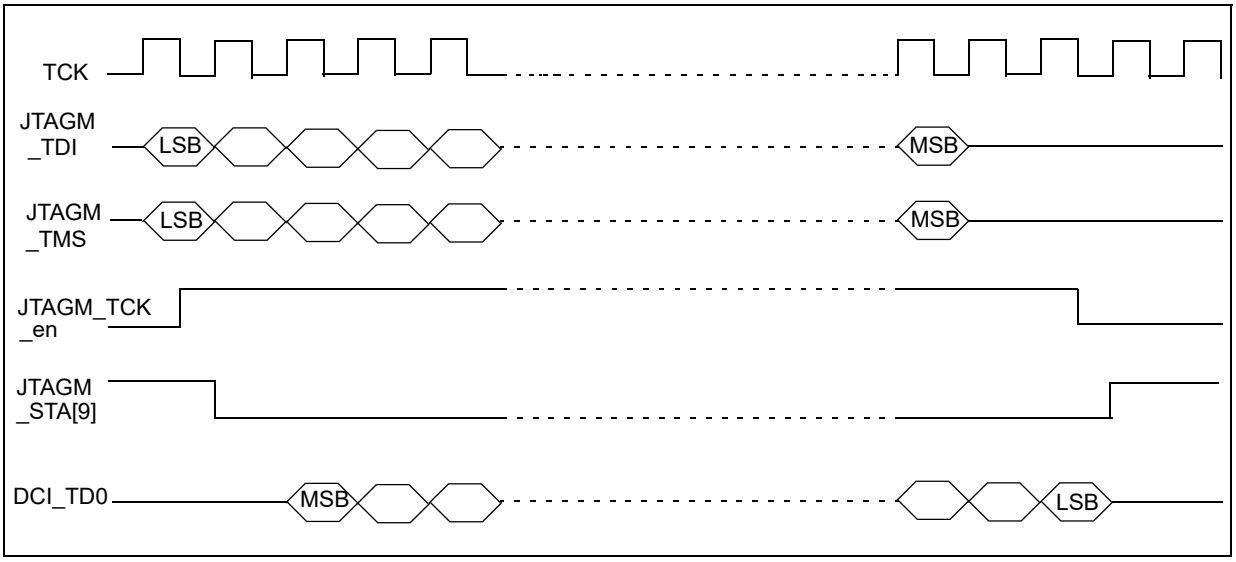
The masked data is transferred to the 32-bit configuration register and 32-bit status register. The JTAGM module responds to this configuration/status write by outputting a 96-bit payload to the parallel output port to channel number A. The 96-bit payload is made up of the 32-bit value of the configuration register plus 32 bits of zero-padding plus 32-bit value of the status register (including the least significant 8 bits which are an 8-bit CRC calculated on the previous 88 bits).

While the current messages are being transmitted from the second buffering (memory mapped registers), the first buffers are continuously filled up. The status register also includes status bits provided by the LFAST module and the DCI module. The JTAGM requests data from the LFAST only when at least one internal buffer is free.

77.2.8 JTAGM to DCI serial interface

Figure 2037 shows the data transfer timings between JTAGM and DCI.

Figure 2037. JTAGM to DCI serial interface diagram



77.2.9 JTAGM data handling and CRC calculation in LFAST mode

Figure 2038 shows the bit ordering when a data frame is transmitted from LFAST to JTAGM for subsequent transmission to DCI.

Figure 2038. Bit ordering for LFAST to JTAGM transfer

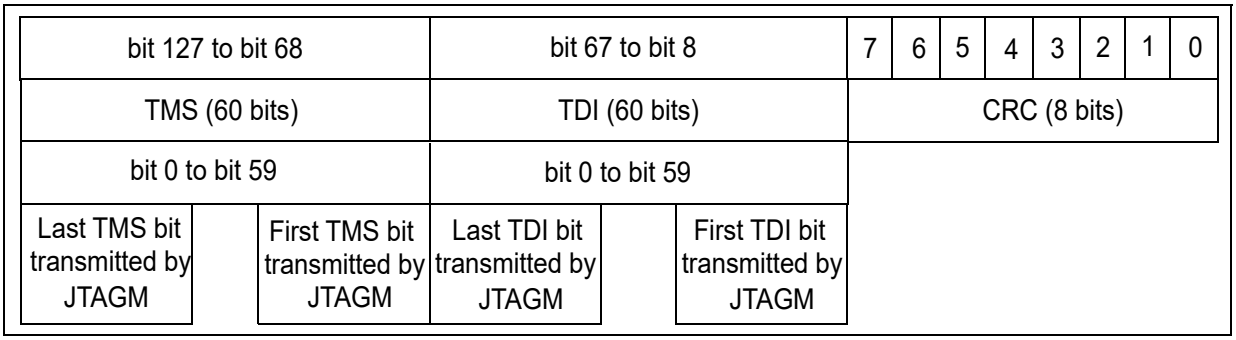
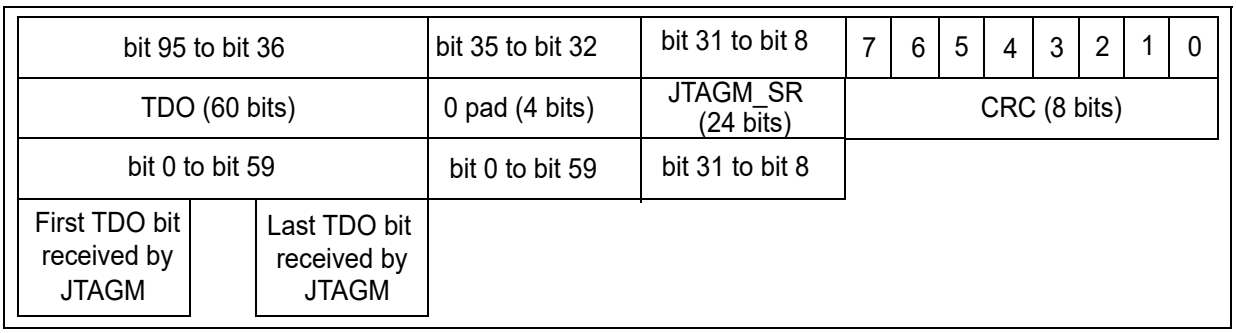


Figure 2039 shows the bit ordering when a data frame is transmitted from JTAGM to LFAST consisting of TDO data received from DCI.

Figure 2039. Bit ordering for JTAGM to LFAST transfer



The CRC calculation is described in following example.

The data transferred from LFAST to JTAGM (in four frames) is as follows:

- D0—F7B3_D591
- D1—E6A2_C486
- D2—A2C4_80F7
- D3—B3D5_9183

These are supplied to CRC block starting from D0 to D3. The CRC calculation begins on D0 with the MSB (left bit) going first. D3 is supplied as B3D5_9100 (after masking CRC).

JTAGM extracts TMS and TDI information from the data received as follows:

- TMS = F7B3_D591_E6A2_C48
- TDI = 6A2C_480F_7B3D_591

JTAGM transmits the data to DCI starting from LSB (right bit going first). Corresponding to this TMS and TDI transmission, JTAGM receives TDO data from DCI and stores it as first bit to most significant location in TDO frame (shown in *Figure 2039*).

77.2.10 Software interface

It is possible for software to write the 120 bits of JTAG data to be output and to read the 60 bits of data. The JTAGM module has an IPS interface.

77.3 Modes of operation

There are no special modes of operation. The JTAGM works normally in the debug mode.

77.4 Memory map and register definition

Table 1960 contains the memory map of the JTAGM registers.

Table 1960. Memory map

Offset	Register	Location
0x00	Module Configuration Register (JTAGM_MCR)	*Section 77.4.1.1*
0x04	Status Register (JTAGM_SR)	*Section 77.4.1.2*
0x08	Data Out Register 0 (JTAGM_DOR0)	*Section 77.4.1.3*
0x0C	Data Out Register 1 (JTAGM_DOR1)	*Section 77.4.1.4*
0x10	Data Out Register 2 (JTAGM_DOR2)	*Section 77.4.1.5*
0x14	Data Out Register 3 (JTAGM_DOR3)	*Section 77.4.1.6*
0x18	Receive CRC Register (JTAGM_RXCRC)	*Section 77.4.1.7*
0x1C	Data Input Register 0 (JTAGM_DIR0)	*Section 77.4.1.8*
0x20	Data Input Register 1 (JTAGM_DIR1)	*Section 77.4.1.9*

77.4.1 Register descriptions

77.4.1.1 Module configuration register (JTAGM_MCR)

This register contains the status bits for the current transfer. *Figure 2040* shows the format of the JTAGM_MCR. In the Emulation and Debug Device (ED), all bits of this register need to be programmed in the Buddy Device instance of the JTAGM. The only exception is the SIE bit, which always needs to be programmed in the Production Device instance of the JTAGM, regardless of being ED or not.

Offset 0x00 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	SWRESET	evto0_sense		evto1_sense		evto_IE	evti0_assert	evti1_assert	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	inter_jtag_frame_timer						0	SIE	IIE	TCKSEL			jtagm_JCOMP	DTM
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0

Figure 2040. JTAGM_MCR register

The JTAGM_MCR register is described in *Table 1961*.

Table 1961. JTAGM_MCR register field descriptions

Field	Description
0 SWRESET	Software Reset. Writing a 1 resets the state machine and counters inside JTAGM. It is a self clearing bit. Writing a 0 has no effect. **Note:** After being written to “1” , this bit is auto cleared within 3 TCK cycles.
1:2 evto0_sense	Determines JTAGM response to activity on $\overline{\text{EVTO}}$0 00 No action is taken by the JTAGM in response to any $\overline{\text{EVTO}}$0 edges (default setting) 01 evto0_edge status bit is set whenever a falling edge is detected on $\overline{\text{EVTO}}$0 10 evto0_edge status bit is set whenever a rising edge is detected on $\overline{\text{EVTO}}$0 11 evto0_edge status bit is set whenever a falling or rising edge is detected on $\overline{\text{EVTO}}$0
3:4 evto1_sense	Determines JTAGM response to activity on $\overline{\text{EVTO}}$1 line 00 No action is taken by the JTAGM in response to any $\overline{\text{EVTO}}$1 edges (default setting) 01 evto1_edge status bit is set whenever a falling edge is detected on $\overline{\text{EVTO}}$1 10 evto1_edge status bit is set whenever a rising edge is detected on $\overline{\text{EVTO}}$1 11 evto1_edge status bit is set whenever a falling or rising edge is detected on $\overline{\text{EVTO}}$1
5 evto_IE	Enables $\overline{\text{EVTO}}$ triggered JTAGM interrupt 0 No JTAGM interrupt is generated In response to any $\overline{\text{EVTO}}$0 or $\overline{\text{EVTO}}$1 activity 1 JTAGM interrupt is generated when evto0_edge or evto1_edge are set
6 evti0_assert	EVTI0 Assert 0 Set jtagm_evti0 low/asserted 1 Set jtagm_evti0 high/de-asserted
7 evti1_assert	EVTI1 Assert 0 Set jtagm_evti1 low/asserted 1 Set jtagm_evti1 high/de-asserted **Note:** In the SW mode (when DTM =1) , regardless of the state of “evti1_assert” bit the internall “evti” signal remains de-asserted. Though this bit can be written, writing to this bit has no effect .
18:23 inter_jtag_frame_timer	TCK delay 000000 0 TCK cycle 000001 1 TCK cycle … 111111 63 TCK cycles
25 SIE	SPU Interrupt Enable(1) 0 JTAGM does not generate an interrupt to the CPU 1 JTAGM generates an interrupt to the CPU if the SPU interrupt request is asserted
26 IIE	Idle Interrupt Enable 0 JTAGM does not generate an interrupt to the CPU upon completion of a 60-bit JTAG transfer 1 JTAGM generates an interrupt to the CPU upon completion of a 60-bit JTAG transfer **Note:** Before enabling a JTAGM idle interrupt by setting JTAGM_MCR[IIE] = 0b1, set the Idle bit (JTAGM_SR[Idle]) to 0b1 to clear any previous interrupts.

Table 1961. JTAGM_MCR register field descriptions (continued)

Field	Description
27:29 TCKSEL	TCK Divider division factor control value. (TCK Divider divides the clock input to JTAGM IP to generate TCK out(input to DCI)). 000 TCK Divider division factor is 1 001 TCK Divider division factor is 2 010 TCK Divider division factor is 3 011 TCK Divider division factor is 4 100 TCK Divider division factor is 5 101 TCK Divider division factor is 6 110 TCK Divider division factor is 7 111 TCK Divider division factor is 8 **Note:** It should be taken care that the maximum frequency of TCK input to DCI, should be equal or less than half of system clock.
30 jtagm_JCOMP	JTAG reset 0 JCOMP low/asserted 1 JCOMP high/not asserted (default)
31 DTM	Data Transfer Mode. 0 Data for JTAG transferred by LFAST 1 Data for JTAG transferred by software

1. The BD JTAGM instance does not receive signals from the SPU. If required, this feature needs to be enabled in the PD instance of the JTAGM, even when JTAG generation in LFAST mode is handled by the BD JTAGM.

77.4.1.2 Status register (JTAGM_SR)

77.4.1.2.1 LFAST mode

Figure 2041 shows the format of the JTAGM_SR in LFAST mode. In the ED, all bits of this register need to be accessed in the BD instance of the JTAGM. The exceptions are the bits related to the SPU interrupts (SPU_INT, SPU_INT_CLR), which always need to be accessed from the PD JTAGM, regardless of being ED or not.

Offset 0x04 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	overrun	0	lfast_status						dci_status							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	evto0_edge	evto1_edge	SPU_INT_CLR	SPU_INT	CRC_err	Nexus_err	Idle	NR	CRC							
W																
Reset	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0

Figure 2041. JTAGM_SR register in LFAST mode

77.4.1.2.2 SW mode

Figure 2042 shows the format of the JTAGM_SR in SW mode.

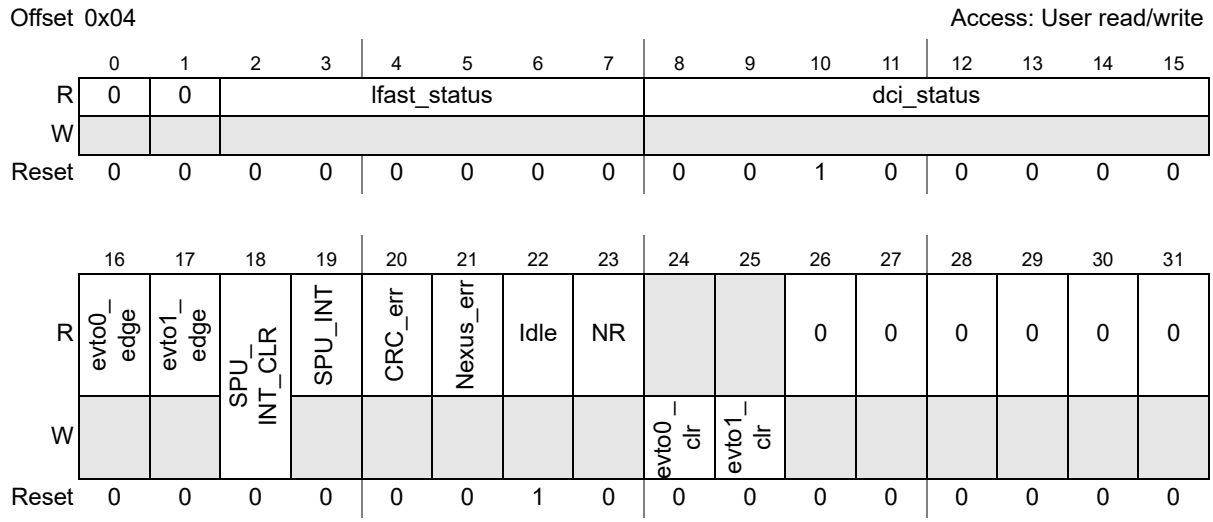


Figure 2042. JTAGM_SR register in SW mode

The JTAGM_SR register is described in *Table 1962*.

Table 1962. JTAGM_SR register field descriptions

Field	Description
0 overrun	Set by hardware when the JTAGM overwrites the first Tx data with the second Tx data. Writing 1 clears this bit. This bit is set only in LFAST mode.
2:7 lfast_status	Status information from the LFAST to be used by software. The composition of these bits are as follows: bit 2 Indicates either data frame or ping frame was transmitted successfully bit 3 Indicates Tx data Interface not enabled and a frame is ready to be transmitted bit 4 Indicates Rx data FIFO overflow bit 5 Indicates reception of frame with invalid frame payload size bit 6 Indicates reception of frame with invalid ICLC code bit 7 Indicates reception of illegal LCT frame
8:15 dci_status	Status information from the DCI. The composition of these bits is as follows: bit 8 Reserved to 0 bit 9 DCI LVDS port enable bit 10 DCI JTAG port enable bit 11 DCI LVDS safe mode bit 12 DCI JTAG safe mode bit 13 DCI LVDS escape mode bit 14 DCI enable LFAST bit 15 DCI tool present , set when a debug tool is connected

Table 1962. JTAGM_SR register field descriptions (continued)

Field	Description
16 evto0_edge	In LFAST mode (DTM = 0) a channel B message is sent on LFAST and the bit is automatically cleared. In software mode (DTM = 1), the bit stays set until cleared by writing to evto0_clr bit 0 No activity sensed on $\overline{\text{EVTO}}$0 1 Activity defined by evto0_sense detected on $\overline{\text{EVTO}}$0
17 evto1_edge	In LFAST mode (DTM = 0) a channel B message is sent to the LFAST and the bit is automatically cleared. In software mode (DTM = 1), the bit stays set until cleared by writing to evto1_clr bit 0 No activity sensed on $\overline{\text{EVTO}}$1 1 Activity defined by evto1_sense detected on $\overline{\text{EVTO}}$1
18 SPU_INT_CLR	Write 1 clears the SPU_INT(1) This is a self clearing bit. If SPU_INT is not set then SPU_INT_CLR remains set until SPU_INT is set, and both are cleared.
19 SPU_INT	This bit is set when the SPU interrupt request input rising edge is detected. Writing a 1 to SPU_INT_CLR resets this bit. (1)
20 CRC_err	CRC error bit. Set when the calculated CRC does not match the last received CRC. This bit is cleared when the error packet is sent to the LFAST on channel B.
21 Nexus_err	Nexus error bit. It is set when there is an error, else it is 0.
22 Idle	Idle bit. Idle bit is cleared while a JTAG frame is in progress. This bit is set to ‘1’ when a JTAG frame ends. Writing a 1 to the Idle bit clears the Idle_interrupt in SW Mode. A read always gives the JTAG frame status but does not give the write value.
23 NR	Status of Ready bit from Nexus RWA. This bit is automatically cleared when a new JTAG message is started. 0 RDY has not asserted indicating Nexus RWA is not ready 1 RDY has asserted indicating a Nexus RWA has completed bus access and is ready for the data register to be read **Note:** In LFAST mode(when DTM =0) , this bit has no effect and acts as a “dont care” . This bit should not be used by the application in the LFAST mode
24:31 CRC	Calculated CRC. CRC calculated on the information received. LFAST Mode: Bit 24:31 reads the CRC. Write has no effect. SW Mode: – Bit 24:31 reads as 0x00. Writing a 0 to any bit have no effect. – Writing 1 on bit 24 clears evto0_edge – Writing 1 on bit 25 clears evto1_edge

1. The BD JTAGM instance does not receive signals from the SPU. If required, this bit needs to be read from the PD instance of the JTAGM, even when JTAG generation in LFAST mode is handled by the BD JTAGM.

77.4.1.3 Data out register 0 (JTAGM_DOR0)

Figure 2043 shows the format of the JTAGM_DOR0.

Offset 0x08 Access: User read/write

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

Figure 2043. JTAGM_DOR0 register

The JTAGM_DOR0 register is described in *Table 1963*.

Table 1963. JTAGM_DOR0 register field descriptions

Field	Description
0:31 TMS_HIGH	Higher word of data for TMS, bits 59-28. **Note:** If application software generated data is required to be written to the JTAGM Data Output Registers then the permissions to write to these registers should be handled within the software to ensure no genuine JTAG Data required is overwritten.

77.4.1.4 Data out register 1 (JTAGM_DOR1)

Figure 2044 shows the format of the JTAGM_DOR1.

Offset 0x0C Access: User read/write

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

Figure 2044. JTAGM_DOR1 register

The JTAGM_DOR1 register is described in *Table 1964*.

Table 1964. JTAGM_DOR1 register field descriptions

Field	Description
0:27 TMS_LOW	Lower word of data for TMS, bits 27–0. TMS_LOW[0] is shifted out first. **Note:** If application software generated data is required to be written to the JTAGM Data Output Registers then the permissions to write to these registers should be handled within the software to ensure no genuine JTAG Data required is overwritten.

77.4.1.5 Data out register 2 (JTAGM_DOR2)

Figure 2045 shows the format of the JTAGM_DOR2.

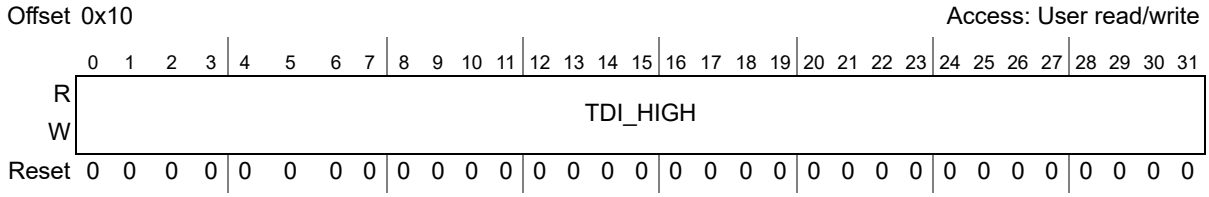


Figure 2045. JTAGM_DOR2 register

The JTAGM_DOR2 register is described in *Table 1965*.

Table 1965. JTAGM_DOR2 register field descriptions

Field	Description
0:31 TDI_HIGH	Higher word of data for TDI, bits 59–28. **Note:** If application software generated data is required to be written to the JTAGM Data Output Registers then the permissions to write to these registers should be handled within the software to ensure no genuine JTAG Data required is overwritten.

77.4.1.6 Data out register 3 (JTAGM_DOR3)

Figure 2046 shows the format of the JTAGM_DOR3.

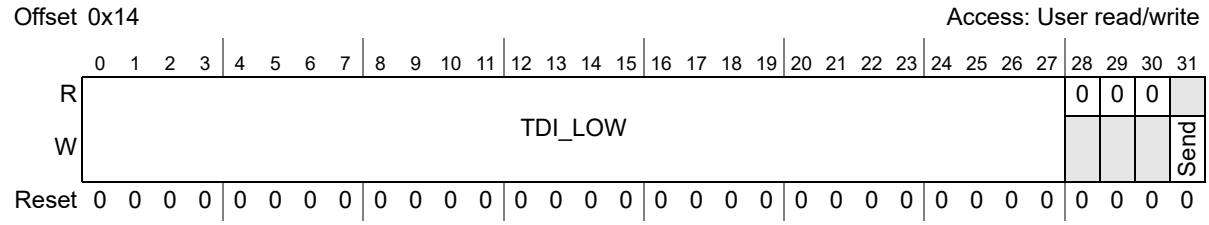


Figure 2046. JTAGM_DOR3 register

The JTAGM_DOR3 register is described in *Table 1966*.

Table 1966. JTAGM_DOR3 register field descriptions

Field	Description
0:27 TDI_LOW	Lower word of data for TDI, bits 27–0. TDI_LOW[0] is shifted out first. **Note:** If application software generated data is required to be written to the JTAGM Data Output Registers then the permissions to write to these registers should be handled within the software to ensure no genuine JTAG Data required is overwritten.
31 Send	Send bit. When this bit is set, data is sent to the DCI. It is a self-clearing bit and is always read as 0. Writable by software only when DTM = 1

77.4.1.7 Receive CRC register (JTAGM_RXCRC)

Figure 2047 shows the format of the JTAGM_RXCRC register.

Offset 0x18 Access: User read/write

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

Figure 2047. JTAGM_RXCRC register

The JTAGM_RXCRC register is described in *Table 1967*.

Table 1967. JTAGM_RXCRC register field descriptions

Field	Description
24:31 CRC	Received CRC. Refers to the CRC of the received LFAST message.

77.4.1.8 Data input register 0 (JTAGM_DIR0)

Figure 2048 shows the format of the JTAGM_DIR0.

Offset 0x1C Access: Read

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

Figure 2048. JTAGM_DIR0 register

The JTAGM_DIR0 register is described in *Table 1968*.

Table 1968. JTAGM_DIR0 register field descriptions

Field	Description
0:31 TDO_LOW	Lower word of data received on TDO, bits 0–31

77.4.1.9 Data input register 1 (JTAGM_DIR1)

Figure 2049 shows the format of the JTAGM_DIR1.

Offset 0x20 Access: Read

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

Figure 2049. JTAGM_DIR1 register

The JTAGM_DIR1 register is described in *Table 1969*.

Table 1969. JTAGM_DIR1 register field descriptions

Field	Description
0:27 TDO_HIGH	Higher word of data received on TDO, bits 32–59

78 Debug Zipwire

78.1 Overview

The Debug SIPI and Debug LFAST modules work together as a single unit called Debug Zipwire. The Debug LFAST portion of the two modules allows for high speed debugging capabilities.

The Debug LFAST operates in slave mode configuration. Please see the SIPI and LFAST chapters for detailed information on module configuration and functionality.

78.2 Debug Zipwire Overview

Dedicated Zipwire functionality is available for debug use allowing support for increased bandwidth read/write operations to the chip memory space performed over the debug interface. This Debug Zipwire function is available on both production and ED devices. The modules used to support Debug Zipwire are the Debug SIPI module, the Debug LFAST module and the LFAST Switch. The Debug Zipwire functionality does not use the LFAST or SIPI modules which support the Zipwire interprocessor bus. The only resource shared between the Debug and interprocessor bus Zipwire interfaces is the use of a single shared LFAST PLL on the Production Device (PD).

The Debug Zipwire interface allows a connected tool to perform 8-bit, 16-bit or 32-bit reads and writes to any 32-bit address in the microcontroller. The Zipwire architecture is fully pipelined and supports multiple outstanding commands to allow maximum use of the serial link bandwidth.

The serial link runs at 320 Mbaud using LVDS physical layer. One LVDS pair for Tx and another pair for Rx, with a separate 20 MHz reference clock for a total of five pins.The Debug Zipwire interface does not support streaming mode commands.

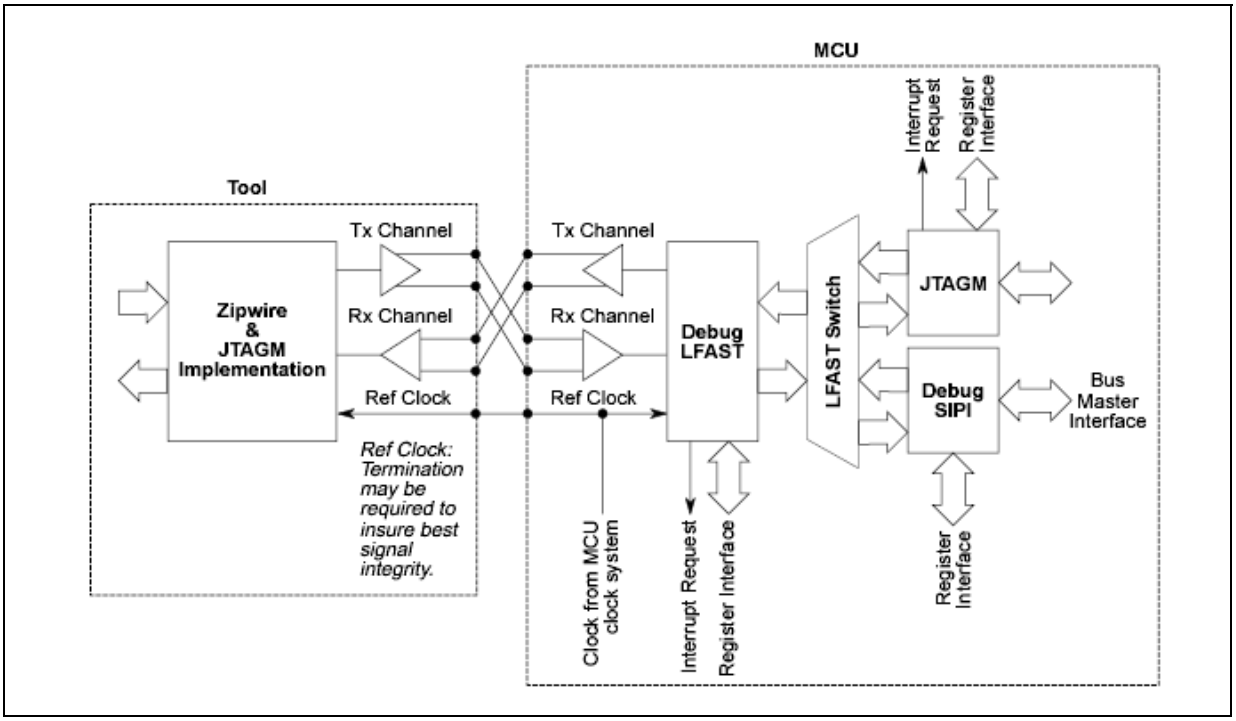
For single random access, 32-bit read, from any 32-bit address location, the latency is approximately 1 µs.

78.3 Debug Zipwire Block Diagram

Figure 2050 shows a tool connected to the MCU via five wires. These five pins are shared with the standard JTAG pins, but for simplicity this is not shown in the diagram. The MCU Debug Zipwire only supports the Target mode of Zipwire. The Debug SIPI Bus Master Interface is used for all Target Mode transactions. The LFAST Register Interface is used for Initial Setup.

The diagram shows the LFAST Reference clock coming from the MCU clock system.

Figure 2050. Debug Zipwire connection diagram



78.4 Architecture

Debug Zipwire is composed of several modules and system resources. The physical layer is LVDS with 1.2 V common mode voltage and 200 mV swing.The pads are shared with JTAG signals.

The transport layer is a protocol called LFAST. LFAST is an asynchronous prototcol, using non-return to zero encoding. The LFAST protocol is composed of the following:

- A fixed 16-bit sync frame to allow the receiver to detect the optimal point to sample the incoming data.
- Followed by an 8-bit LFAST header, that defines the channel number and the size of the LFAST payload.
- Finally the payload, which can be between 8 and 288 bits.

The application layer protocol is called Serial Inter Processor Interface (SIPI). Debug SIPI runs on top of Debug LFAST and is fully encapsulated within the Debug LFAST payload. Debug SIPI uses three fixed sizes of Debug LFAST payload in Debug Zipwire:

- 32-bit
- 64-bit
- 96-bit

The SIPI protocol implements a suite of commands initiated by a tool, for reading and writing any 32-bit address location in a connected MCU. Only the tool can initiate commands. The MCU only responds to commands initiated by the tool and never creates any unsolicited messages.

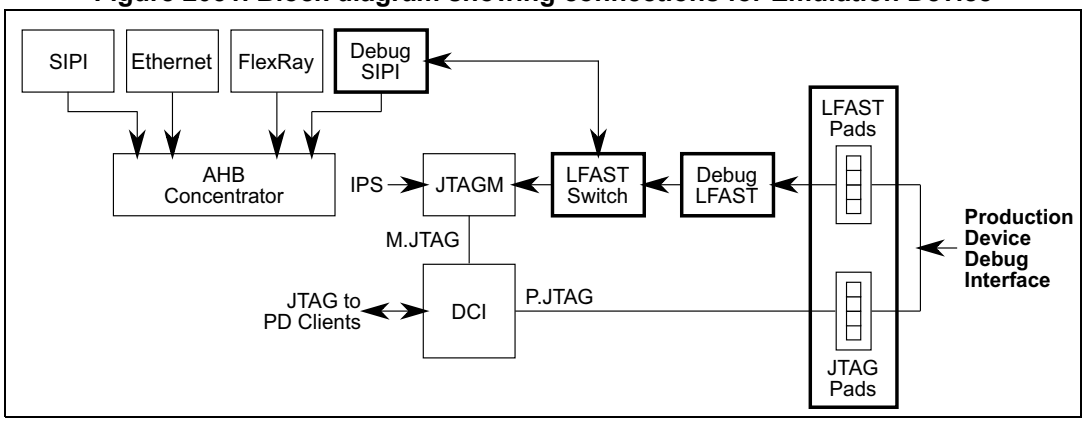
For Debug Zipwire, a reduced version of the SIPI module called Debug SIPI is used. The Debug SIPI module implements only the recipient part of the SIPI protocol (called Target

mode). The Debug SIPI module is a bus master on the low speed XBAR. As the target MCU for a command from the inititating tool, Debug SIPI can perform read and write accesses to any address location within the target MCU. The software running on the local MCU, should configure the system MPU to allow/deny memory accesses to MCU memory and resources as required.

78.5 Debug Zipwire implementation

Figure 2051 shows the modules on the production device which support the Debug Zipwire function, along with their interconnections to the debug interface signals.

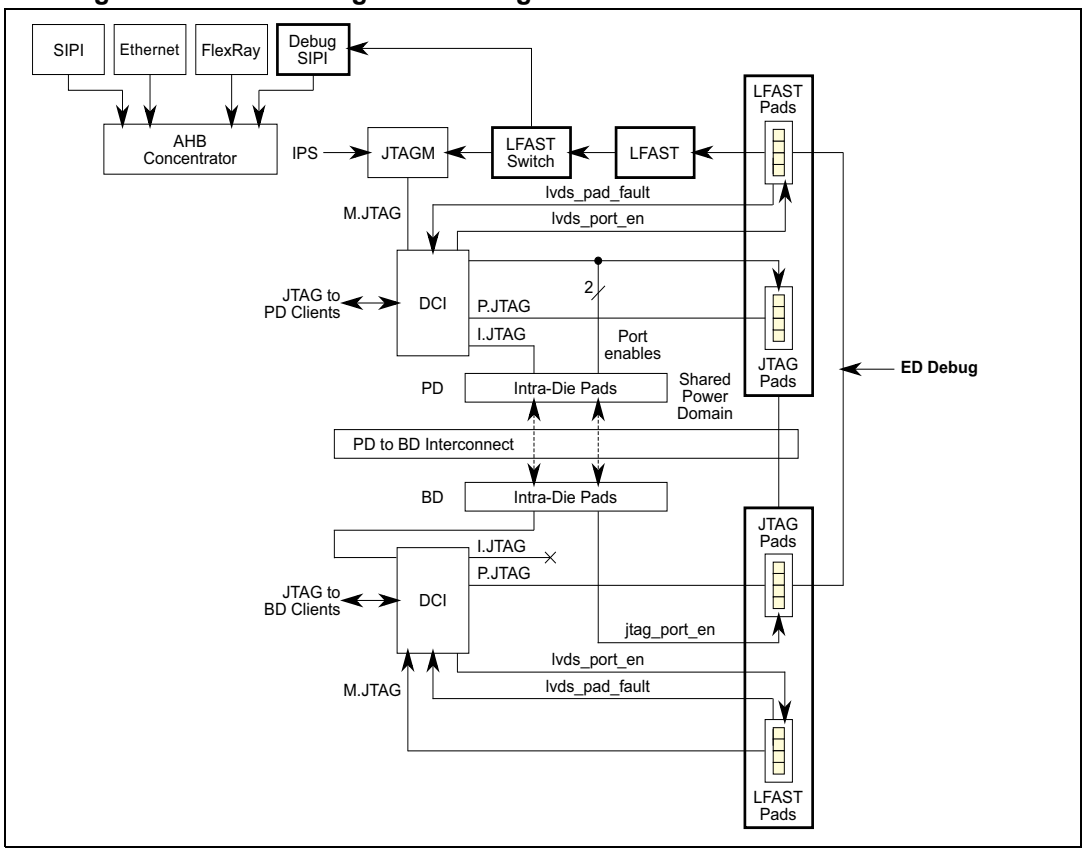
Figure 2051. Block diagram showing connections for Emulation Device



Internal connections to other debug modules which are used to support other debug interface protocols such as JTAG and JTAG over LFAST are also shown. The Debug SIPI module is connected to the Debug LFAST module via an LFAST switch module, which controls routing of debug LFAST traffic to either the Debug SIPI or the JTAGM module.

Figure 2052 shows the equivalent connections for ED devices utilizing the LFAST switch.

Figure 2052. Block diagram showing connections for the Emulation Device



78.6 LFAST clocking

The LFAST implementation includes a dedicated PLL for multiplying a low speed clock source up to 320 MHz for the transmission and reception of data. This PLL is used to support both interprocessor bus Zipwire and Debug Zipwire so care must be taken to ensure that any tool configuration of this resource is compatible with application usage of Interprocessor Bus Zipwire. The high speed asynchronous nature of LFAST, requires that both the local node and remote node, operate from the same, stable, low jitter clock sources. This common low speed clock is referred to as the Reference Clock.

The MCU includes clock multiplexing to allow the source of the LFAST reference clock to come from one of three sources via a programmable clock divider:

- XOSC
- Output of PLL0 (PLL0:PHI)
- External pin

The most expedient source for the reference clock is the 20 MHz crystal oscillator (XOSC) from one MCU. This 20 MHz clock can be used as the LFAST Reference clock for the local LFAST. This same 20 MHz clock must be exported from the local MCU to the remote MCU, to be used as the LFAST reference at the remote MCU.

The 20 MHz clock may be used in the following ways:

- Shared between the MCUs using a dedicated reference clock pin on each MCU
- Divided by 2 and shared as a 10 MHz reference
- Shared as the fundamental clock for the MCU system clock and the LFAST reference clock

The LFAST PLL for generating the 320 MHz clock, is shared between LFAST for SIPI.

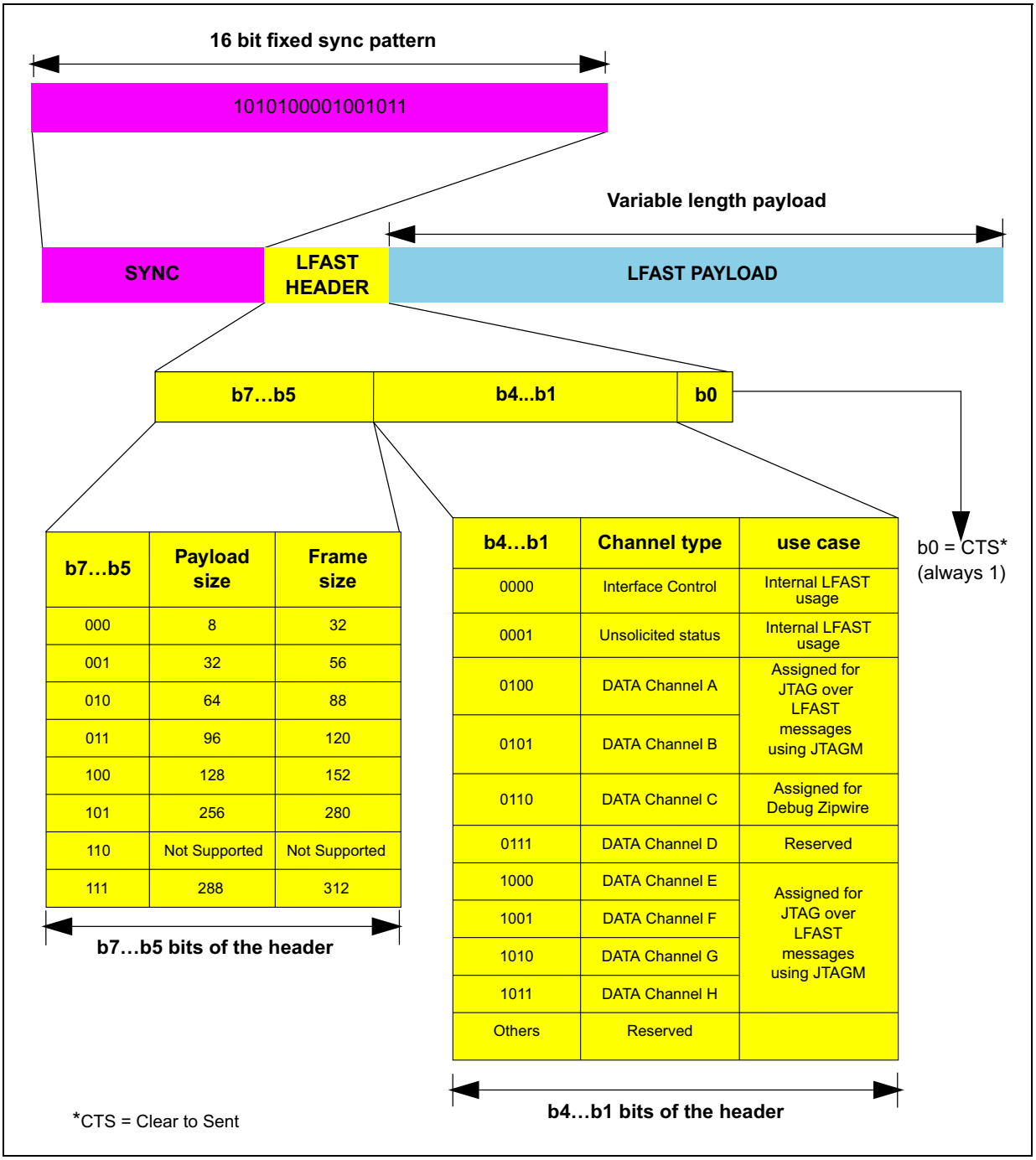
78.7 LFAST switch

Incoming messages on the Debug LFAST interface can be either for JTAG over LFAST use through the JTAGM or debug Zipwire messages. Differentiation of these messages is via use of LFAST channel number. DATA channel C is used to identify high speed Debug SIPI accesses, while JTAG over LFAST messages are supported using LFAST channels A, B, E, F, G, H as shown in *Figure 2053*.

The LFAST SWITCH module is used to direct LFAST messages based on the above scheme, with JTAG over LFAST messages being routed to the JTAGM IP block and debug Zipwire messages being routed to the Debug SIPI block.

The LFAST switch has no accessible registers and requires no user configuration.

Figure 2053. LFAST channel allocation



b7…b5	Payload size	Frame size
000	8	32
001	32	56
010	64	88
011	96	120
100	128	152
101	256	280
110	Not Supported	Not Supported
111	288	312

b4…b1	Channel type	use case
0000	Interface Control	Internal LFAST usage
0001	Unsolicited status	Internal LFAST usage
0100	DATA Channel A	Assigned for JTAG over LFAST messages using JTAGM
0101	DATA Channel B	
0110	DATA Channel C	Assigned for Debug Zipwire
0111	DATA Channel D	Reserved
1000	DATA Channel E	Assigned for JTAG over LFAST messages using JTAGM
1001	DATA Channel F	
1010	DATA Channel G	
1011	DATA Channel H	
Others	Reserved	

78.8 Debug SIPI module

The debug SIPI module is used to support high bandwidth read/write accesses over the LFAST debug interface. It is dedicated for debug use and is entirely separate from the SIPI module used to support interprocessor bus zipwire interface.

The debug SIPI characteristics are shown in *Table 1970*.

Table 1970. Debug SIPI feature set

Feature	Debug implementation
Number of channels supported	1
Initiator support	No
Target support	Yes
Block transfer support	No
Chip XBAR connection	M1 on XBAR_1

The debug SIPI module issues read and write accesses via the AHB concentrator connected to physical master port 2 on XBAR_1, which is shared with the ethernet, FlexRay and Zipwire SIPI bus masters.

The debug SIPI memory mapped registers are accessible via PBRIDGE_0 (off platform - slot 10) with base address FFFD_4000h.

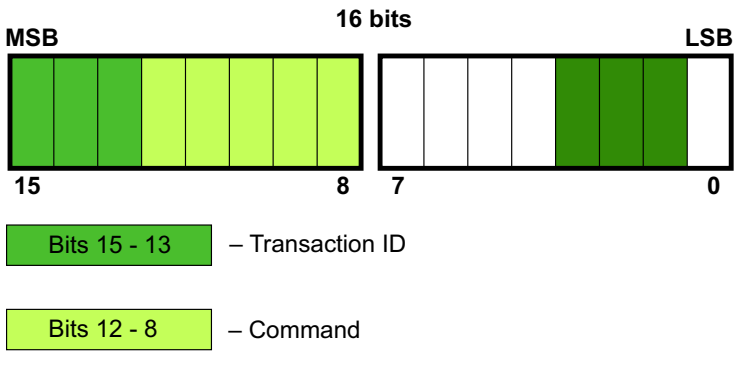
The debug SIPI has Logical Bus Master Assignment of 6 for SMPU0 and SMPU1, an NXMC trace source ID (SRC) of 1101b and a trace master ID (MASTER) = 1011b.

The debug SIPI supports only target mode SIPI operations on a single channel with a reduced command set. Commands supported include read (8/16/32 bits) and write (8/16/32 bits) along with the appropriate responses. Other SIPI commands are unimplemented as noted in *Table 1971*.

The debug SIPI will be capable of:

- receiving a 64-bit command on LFAST channel C and responding with either a 64-bit response or 32-bit error response on LFAST channel C.
- receiving a 96-bit command on LFAST channel C and responding with either a 32-bit acknowledge or 32-bit error response on LFAST channel C.

Figure 2054. Debug SIPI header



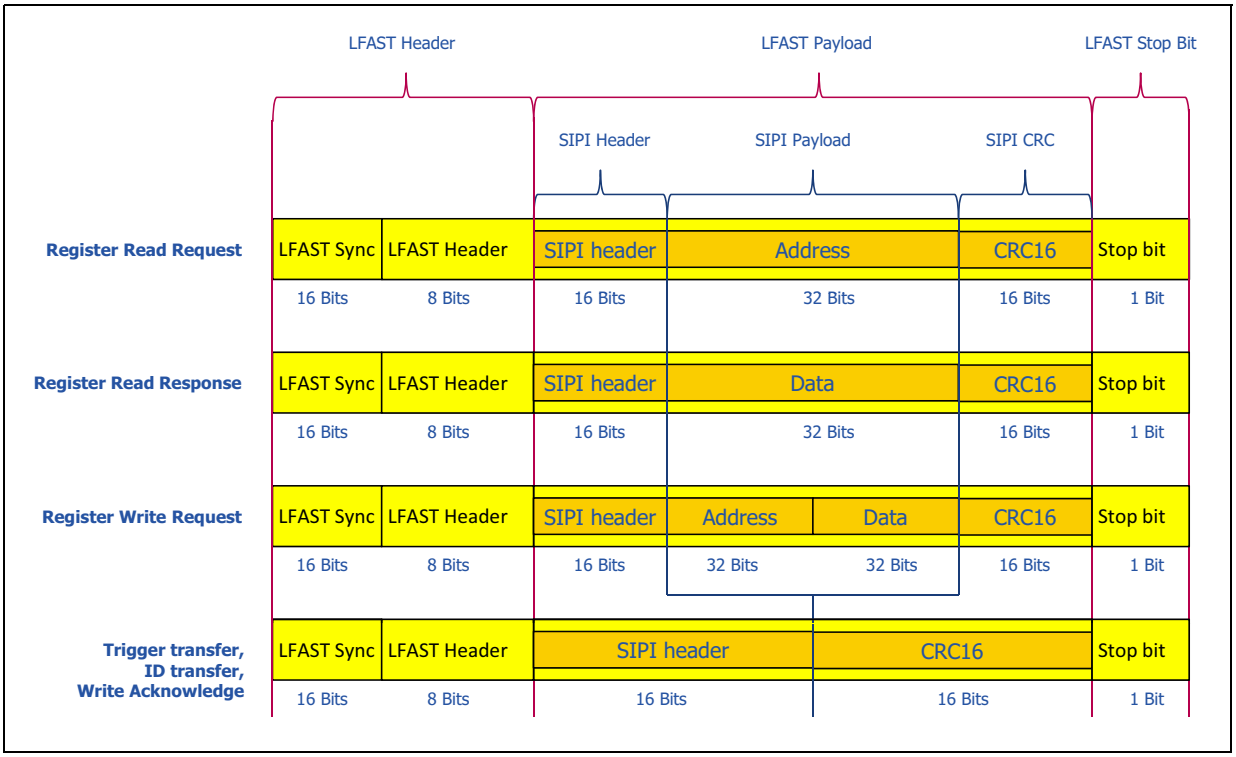
Note: See Figure 2053 for the description of bits[7:0]

Table 1971. SIPI header command coding

Bits[12:8] (hex)	Command	Payload Size	Message Direction at chip (SIPI Target)
00	Read 8 bits (Received by SIPI from LFAST)	64	Rx
01	Read 16 bits (Received by SIPI from LFAST)	64	Rx
02	Read 32 Bits (Received by SIPI from LFAST)	64	Rx
03	Reserved	—	—
04	Write 8 bits with ACK (Received by SIPI from LFAST)	96	Rx
05	Write 16 bits with ACK (Received by SIPI from LFAST)	96	Rx
06	Write 32 bits with ACK (Received by SIPI from LFAST)	96	Rx
07	Reserved	—	—
08	ACK – OK (Issued to LFAST from SIPI)	32	Tx
09	ACK – Fault (Issued to LFAST from SIPI)(1)	32	Tx
0A	Read Answer – OK (Issued to LFAST from SIPI)	64	Tx
0B	Reserved	—	—
0C	Unimplemented – Trigger comm and with ACK	32	—
0D	Reserved	—	—
…	…	…	…
11	Reserved	—	—
12	ID Register Read Request	32	Rx
13	Reserved	—	—
…	…	…	…
16	Reserved	—	—
17	Unimplemented – Stream Data with ACK (32 bytes)	288	—
18	Reserved	—	—
…	…	…	…
1F	Reserved	—	—

1. ACK – Fault is also referred to as “NACK” and “ACK Error” in the HSSL/SIPI Interprocessor Bus Protocol Specification.

Figure 2055. Zipwire message format



Note: Identification Register Read is the same as the Register Read response format.

78.8.1 ED power supply constraints

For ED devices the following power domain user constraints must be observed:

- The Vdd_hv_jtag_osc and Vdd_hv_io_bd supplies need to be connected on the application PCB.
- For cold start operation where a tool must be able to communicate via JTAG and read/write ED RAM contents, Vdd_lv_stby, Vdd_hv_jtag_osc and Vdd_hv_io_bd must remain powered.
- For cold start operation where only ED RAM contents have to be maintained, only Vdd_lv_stby must remain powered.

78.9 Debug Zipwire interconnections

Debug LFAST has the following connections:

- Tx and Rx connections to the pads via the SIUL and the MSCR registers.
- Peripheral Bridge interface (PBRIDGE) — Allows software to read and write configuration registers.
- Tx Data Port/Rx Data Port — Connected to the LFAST Switch. This allows LFAST data to be dynamically routed to/from either Debug SIPI for Debug Zipwire operation or JTAGM for JTAG over LFAST operation.
- Interrupt Request connections — Allows the module to flag to the CPU when it requires servicing.

LFAST Switch has the following connections:

- Tx Data Port/Rx Data port directly connected to Debug SIPI.
- Tx Data Port/Rx Data port directly connected to JTAGM.
- Tx Data Port/Rx Data port directly connected to Debug LFAST.
- LFAST switch has no IPS interface; no registers; no interrupts.

Debug SIPI has the following connections:

- Peripheral Bridge interface (PBRIDGE) — Allows software to read and write configuration registers and SIPI Interface Registers. For the Debug SIPI, a restricted set of registers is implemented. In use, no configuration of these registers is needed from either s/w or a connected tool, as the default settings should be usable for typical operation.
- Crossbar master port — Allows SIPI to execute requested commands to read and write MCU address space.
- Tx Data Port/Rx Data Port — Directly connected to the LFAST module. Allows received data to be efficiently transferred from LFAST and transmit data to be transferred to LFAST.

79 Debug Serial Interprocessor Interface (SIPI)

79.1 Introduction

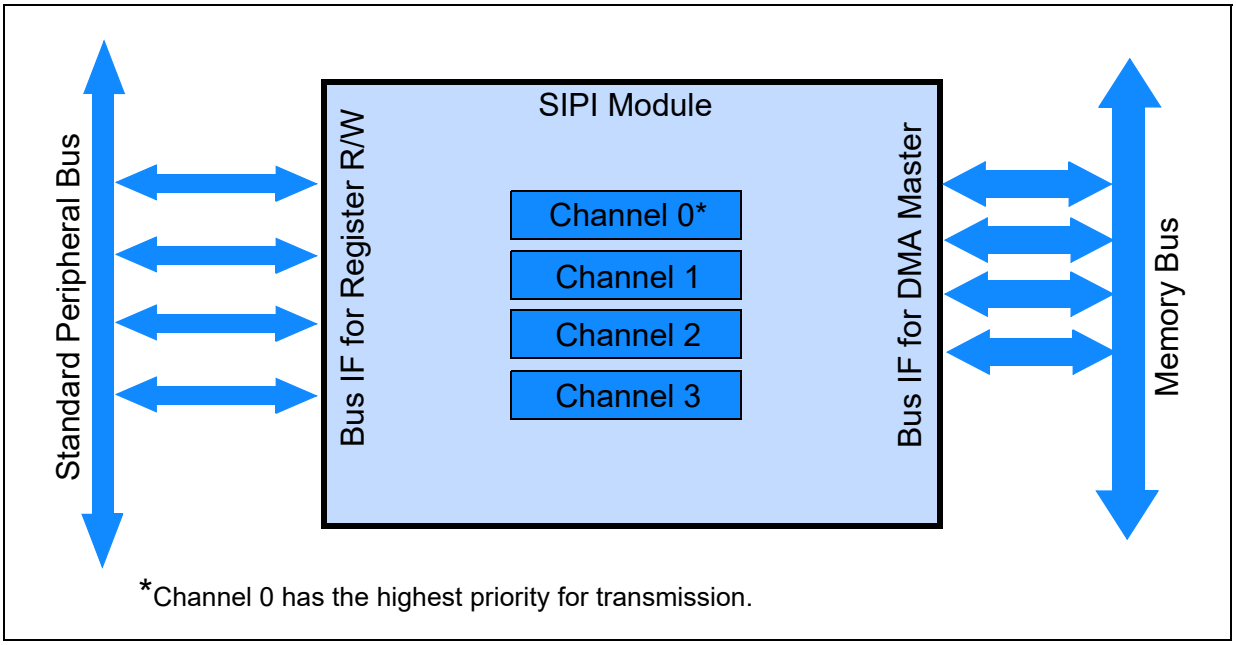
The Debug Serial Interprocessor Interface (SIPI) is an application layer protocol that runs on top of the LFAST module. It is used by the local device to access the shared memory of a remote device. Debug SIPI defines point-to-point full duplex communication between two nodes, where LFAST works as a physical medium of communication between both the devices.

Debug SIPI implements the Target mode of a communication protocol in hardware. Debug SIPI can interpret 'read' and 'write' commands received from an LFAST module and convert them into bus accesses to local memory; then send back to the LFAST module either read data or a write acknowledge response. This provides a mechanism for an external device, such as an external tool, to read and write the address map of the local MCU. The bandwidth of these read/write commands is approximately 10 MBytes/second when accessing random addresses using 32-bit transfers.

79.2 Overview

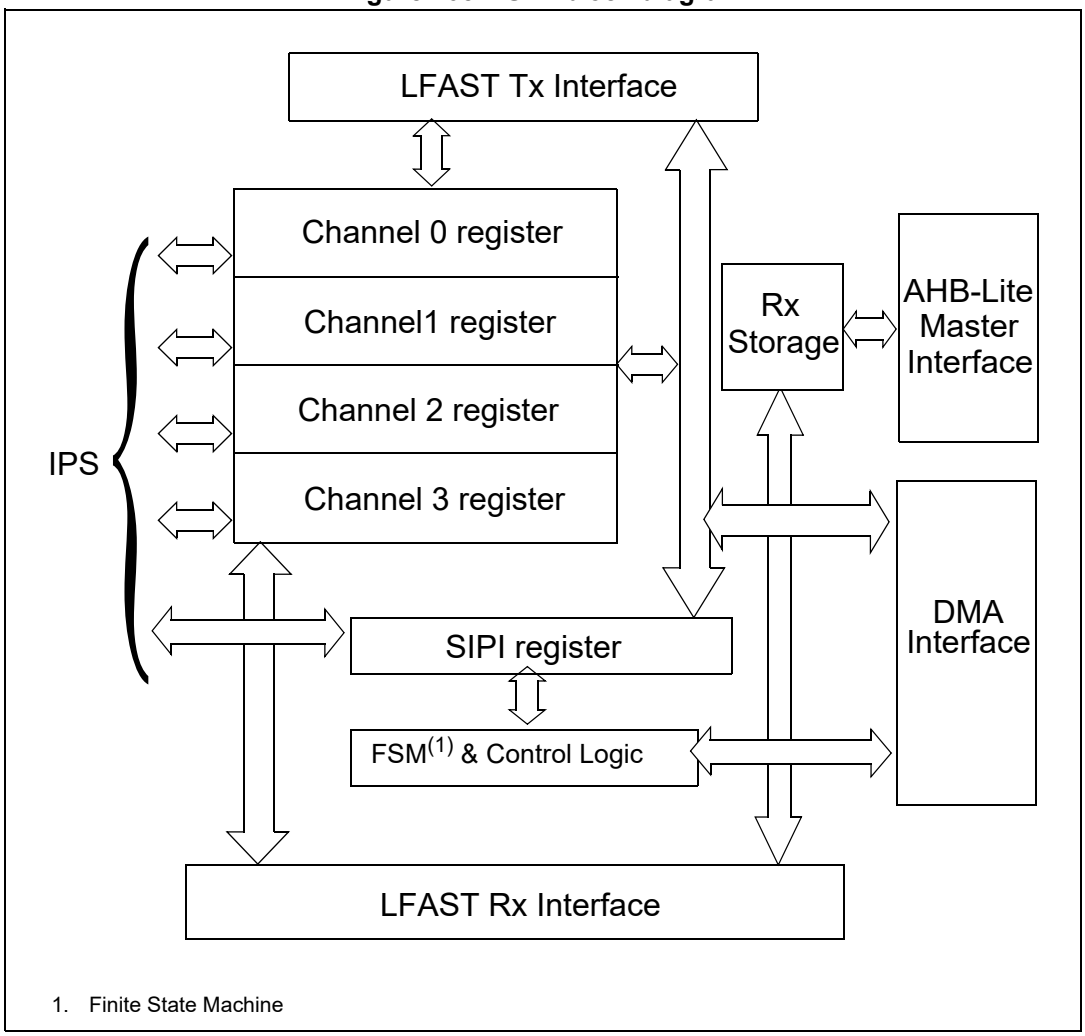
Debug SIPI can access memory mapped resources directly through its AHB master interface. Debug SIPI has four channels, 0, 1, 2, 3. Payload width for all channels is 32 bits.

Figure 2056. SIPI module



79.3 Debug SIPI block diagram

Figure 2057. SIPI block diagram



1. Finite State Machine

79.4 Feature description

This section describes the features of the Debug SIPI module.

79.4.1 Main features

- Supports 8, 16 and 32-bit read and write commands
- Command/response includes a 16-bit CRC to ensure data integrity
- Command/response include a 2-bit tag to allow tracking of data to commands in a pipelined system
- Command/response include a 3-bit channel identifier to allow multiple Initiators to send commands to a single target node
- Command/response includes a 16-bit header that defines, the function, channel and tag
- Every command generates a response to allow a 'sliding window' flow control protocol to be implemented
- AHB master interface that is used by target node to access memory mapped resources

79.4.2 Standard features

- IPS bus interface
- AHB master interface
- LFAST Tx/Rx interfaces
- Cyclic Redundancy Check error detection (CRC16)

79.5 Debug SIPI operation from reset

When the Debug SIPI module exits reset, it is operational and target mode is enabled without the need to configure the control registers.

79.6 Functional description

79.6.1 External signals

The Debug SIPI has no chip external signals.

79.6.2 Frame format

All frames have the same general format:

- 16-bit header
- Address, address and data, or nothing
- 16-bit CRC

There are 2 main groups of command; read and write. Within those 2 groups are three read/write formats:

- 32-bit
- 16-bit
- 8-bit

Each command generates one of three different responses:

- Read data
- Write acknowledge
- Error

There is one additional command that requests the module ID from the Target node. The ID is a unique 32-bit ID that is specific to a particular device. It is normally the same as the JTAG ID.

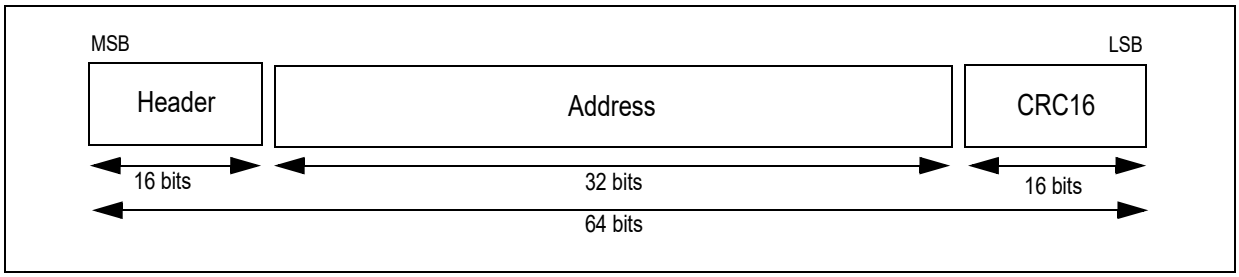
The sections below illustrate the four different frame formats that are used to implement all the commands and responses.

The number of frames depends on the data buffers (associated with every channel), the frame data is stored in data buffers at the initiator. Address and data are both transmitted in the same order in which they are stored.

Note: SIPI module architecture is always big endian but the storage in host memory depends on endianness architecture of host memory.

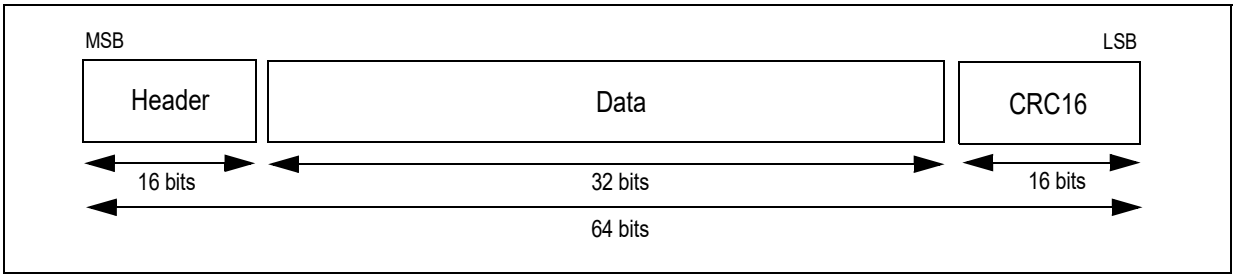
79.6.2.1 Register read request

Figure 2058. SIPI register read request



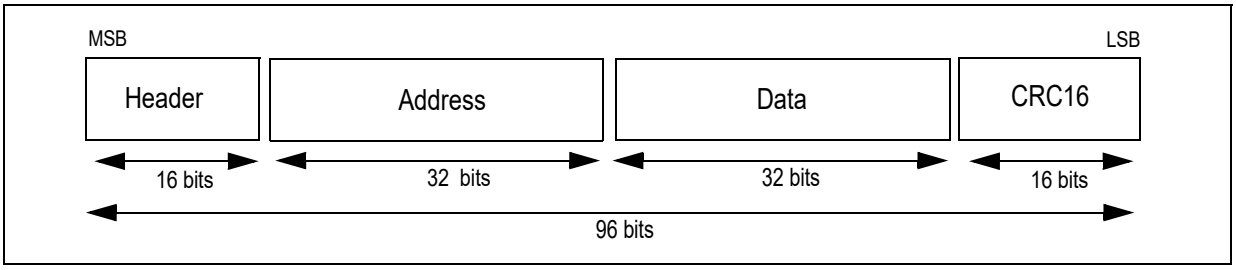
79.6.2.2 Register read response, ID request response

Figure 2059. SIPI read response



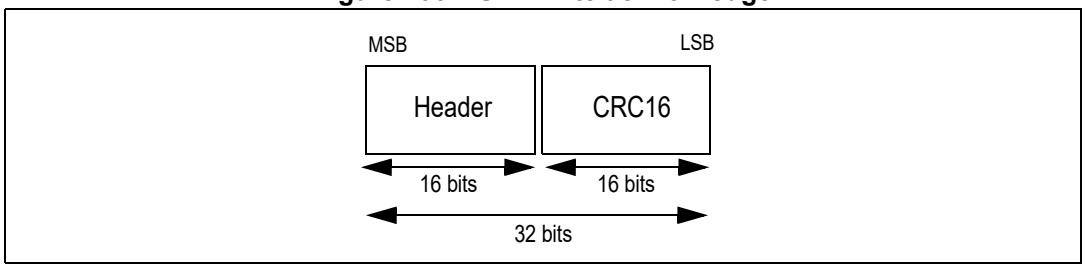
79.6.2.3 Register write request

Figure 2060. SIPI register write request



79.6.2.4 ID transfer, write acknowledge

Figure 2061. SIPI write acknowledge



79.6.2.5 Header field

Header field contains 16 bits of configuration information. MSB will be transmitted first.

79.6.2.5.1 Debug SIPI header coding

Figure 2062, *Table 1972*, and *Figure 2063* show how the Debug SIPI header bits are coded.

Figure 2062. SIPI header coding

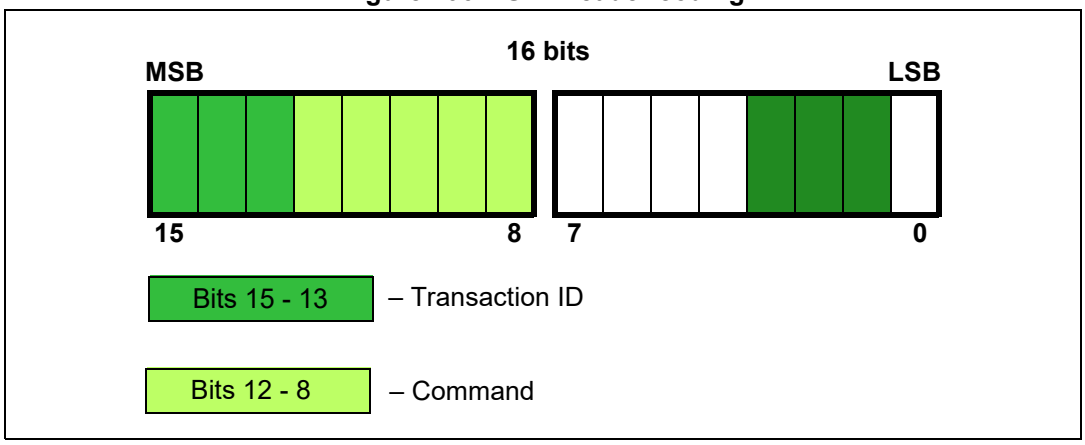


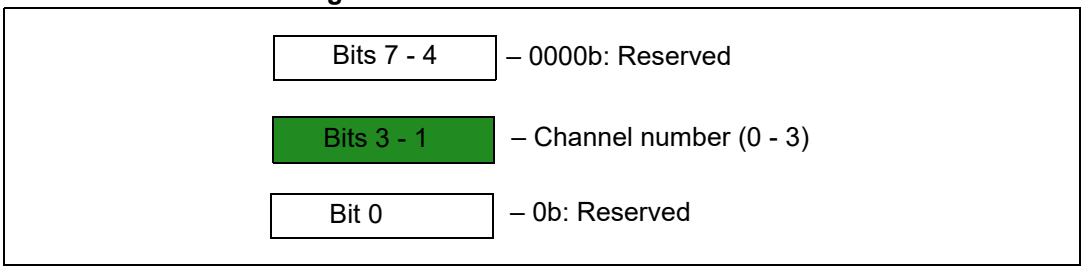
Table 1972 shows the command coding for the bits[12:8] of the header.

Table 1972. SIPI header command coding

b[12:8] (hex)	Command	Payload Size
0x00	Read 8 bits	64
0x01	Read 16 bits	64
0x02	Read 32 Bits	64
0x03	Reserved	—
0x04	Write 8 bits with ACK	96
0x05	Write 16 bits with ACK	96
0x06	Write 32 bits with ACK	96
0x07	Reserved	—
0x08	ACK – OK	32
0x09	ACK – Fault	32
0x0A	Read Answer – OK	64
0x0B	Reserved	—
0x0C	Reserved	—
0x0D	Reserved	—
…	…	…
0x11	Reserved	—
0x12	ID Register Read Request	32
0x13	Reserved	—
…	…	…
0x16	Reserved	—
0x17	Reserved	—
0x18	Reserved	—
…	…	...
0x1F	Reserved	—

Figure 2063 shows the content of bits[7:0] in the SIPI header.

Figure 2063. SIPI header channel field



79.6.2.6 Address field

The address field is 32 bits wide with the MSB transmitted first.

79.6.2.7 Payload field

Table 1973 shows the payload sizes of LFAST frames.

Table 1973. Payload size of LFAST channel frame

LFAST code for payload size	Debug SIPI Code	LFAST payload size (bits)(1)	LFAST payload size (bytes)
000b	—	8	1
001b	—	32	4
010b	010b	64	8
011b	011b	96	12
100b	100b	128	16
101b	101b	256	32
111b	111b	288	36

1. Only 32, 64, 96 and 288 bits sizes are used for SIPI.

Table 1974 shows the coding of LFAST channel for a given Debug SIPI channel code.

Table 1974. Converted coding of LFAST channel code for Debug SIPI headers

LFAST channel code	Debug SIPI channel code	Data channel(1)
0100b	**100b**	**Data channel A**
0101b	**101b**	**Data channel B**
0110b	**110b**	**Data channel C**
0111b	**111b**	**Data channel D**
1000b	**000b**	**Data channel E (not used by Debug SIPI)**
1001b	**001b**	**Data channel F (not used by Debug SIPI)**
1010b	**010b**	**Data channel G (not used by Debug SIPI)**
1011b	**011b**	**Data channel H (not used by Debug SIPI)**

1. Debug SIPI channel 0 sends all commands on LFAST channel A, commands received on LFAST channel A, are processed and the response sent back on LFAST channel A.
Debug SIPI channel 1 sends all commands on LFAST channel B, commands received on LFAST channel B, are processed and the response sent back on LFAST channel B.
Debug SIPI channel 2 sends all commands on LFAST channel C, commands received on LFAST channel C, are processed and the response sent back on LFAST channel C.
Debug SIPI channel 3 sends all commands on LFAST channel D, commands received on LFAST channel D, are processed and the response sent back on LFAST channel D.

79.6.2.8 CRC field

CRC field is 16 bits wide with calculation always enabled using CRC-16-CCITT syndrome (x16 + x12 + x5 + 1). The data stream is transmitted with the MSB sent first.

79.6.2.8.1 CRC examples

Example 1 – 32-bit write

- Header = 0x260A
- Address = 0x1122_3344
- Data = 0xCCDD_EEFF
- CRC = 0xBF7D

Example 2 – 32-bit read

- Header = 0x420C
- Address = 0x89AB_CDEF
- CRC = 0x6B80

Example 3 – Event command

- Header = 0x6C0E
- CRC = 0xB286

79.7 Transfer types

This section describes the available transfer types of the Debug SIPI module. The Debug SIPI frame is inserted inside the payload of the LFAST frame as shown in the figures of the following examples.

79.7.1 Read transfer

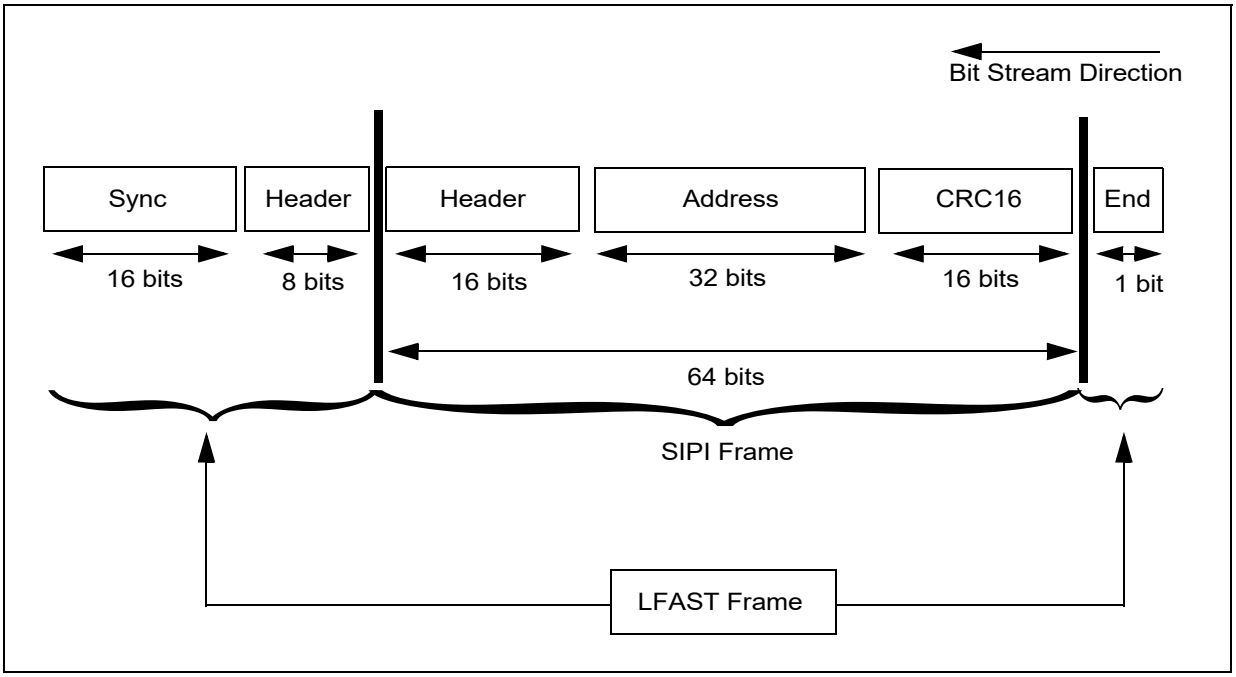
A Read transfer can be of two types:

- Read request transfer (at initiator node)
- Read response transfer (at target node)

79.7.1.1 Register read request transfer

If there is a read request transfer, the initiator node will send header, address and CRC bits as shown in *Figure 2064*.

Figure 2064. Read request transfer (showing LFAST frame encapsulation)



79.7.2 Register read answer transfer

In response to a read request, Debug SIPI sends header, payload and CRC16. Data transfer could be in 8-bit, 16-bit or 32-bit modes (refer to *Figure 2065*). In the case of 8-bit or 16-bit modes, copies of the SIPI data is sent in the payload (refer to *Figure 2066*).

Figure 2065. Read answer transfer (LFAST frame encapsulation is not shown)

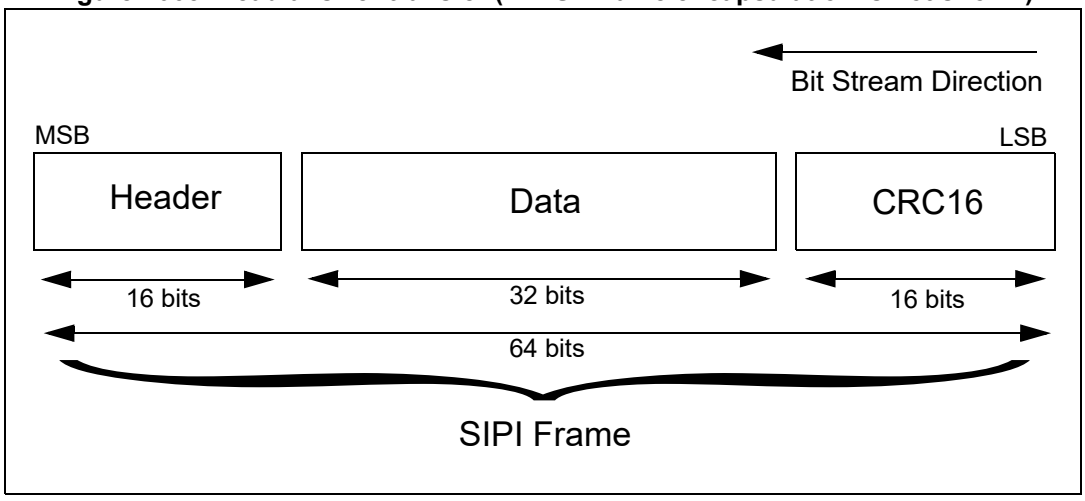
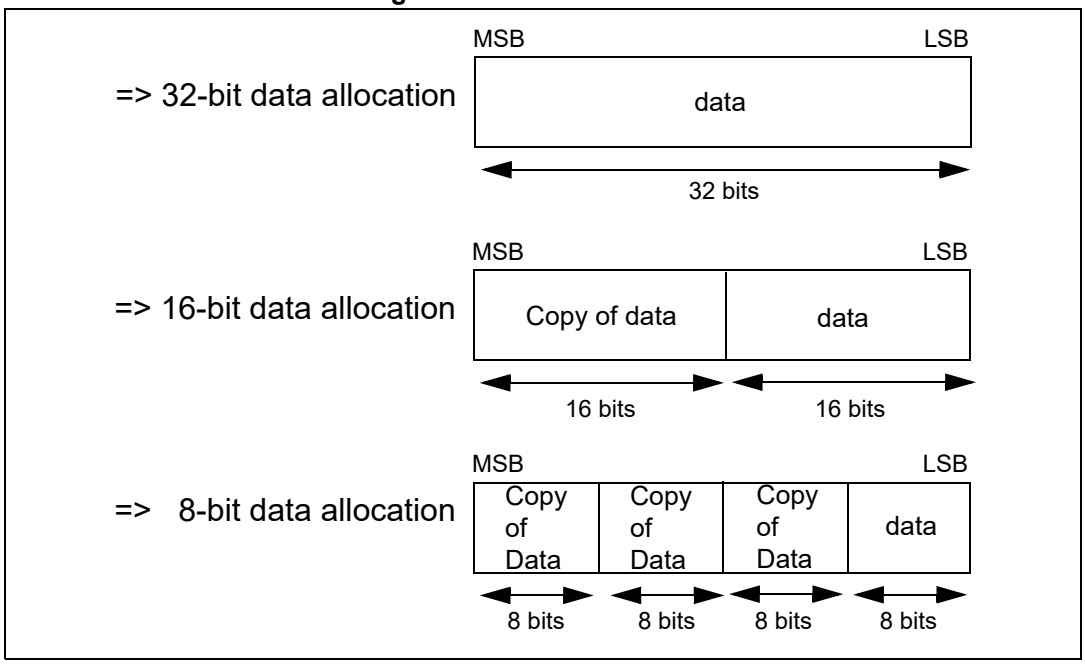


Figure 2066. Data allocation



Note: For an 8-bit transfer, address bits [31:2] are used as address and bits [1:0] are used as byte enables (big endian):

Bits[1:0]:

- *00: Byte 3 enabled (MSB)*
- *01: Byte 2 enabled*
- *10: Byte 1 enabled*
- *11: Byte 0 enabled (LSB)*

For a 16-bit transfer, address bits [31:1] are used as address and bit [0] is used as half word enable (big endian).

Bit[0]:

- *0: Half-word 1 enabled (MSB)*
- *1: Half-word 0 enabled (LSB)*

Note: Byte Ordering depends from address written in CAR and transfer width:
Example:

- *CDR content: 0x12345678*
- *CAR content: see table columns*

Table 1975. Target Memory content dependent from CAR content

	Target Memory Byte Address	CAR = 0x02040	CAR = 0x02041	CAR = 0x02042	CAR = 0x02043
32-bit transfer	0x02040	12	12	12	12
	0x02041	34	34	34	34
	0x02042	56	56	56	56
	0x02043	78	78	78	78

Table 1975. Target Memory content dependent from CAR content (continued)

	Target Memory Byte Address	CAR = 0x02040	CAR = 0x02041	CAR = 0x02042	CAR = 0x02043
16-bit transfer	0x02040	12	56	xx	xx
	0x02041	34	78	xx	xx
	0x02042	xx	xx	12	56
	0x02043	xx	xx	34	78
8-bit transfer	0x02040	12	xx	xx	xx
	0x02041	xx	34	xx	xx
	0x02042	xx	xx	56	xx
	0x02043	xx	xx	xx	78

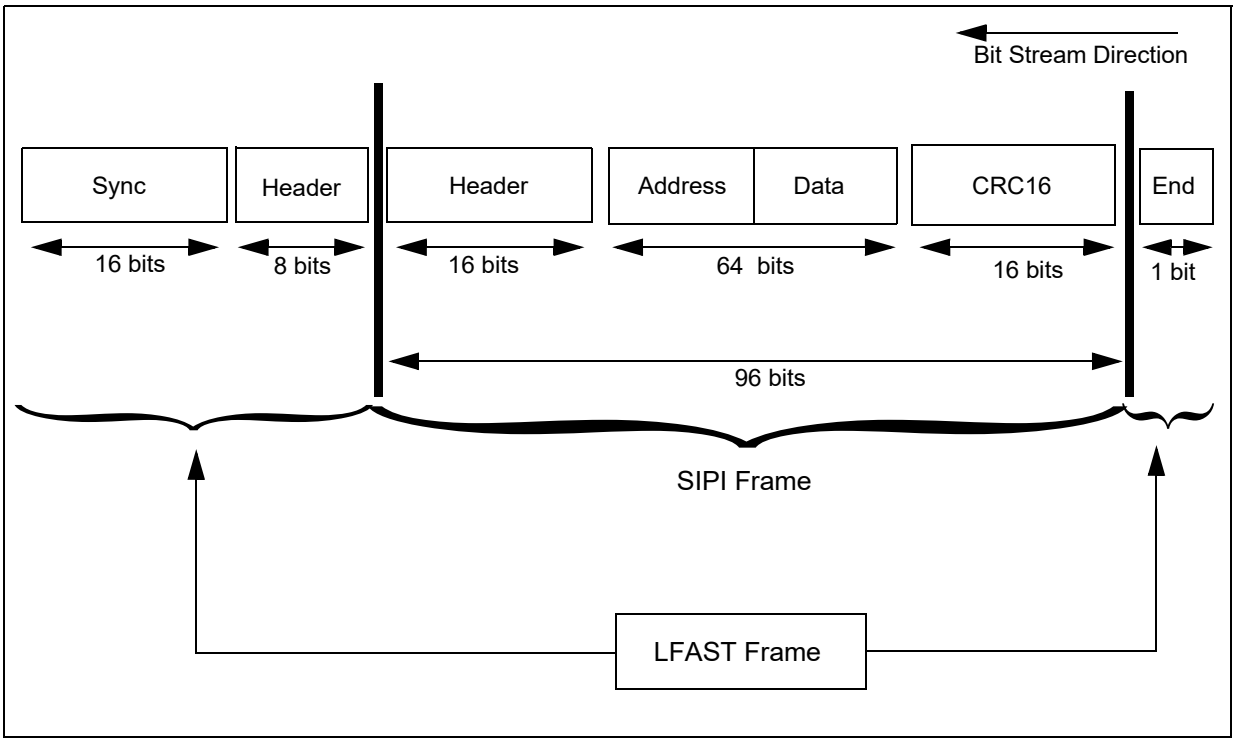
79.7.3 Register Write transfer

Register Write transfer can be of type:

- Normal Write transfer – channels 0, 1, 2 and 3

A Register Write transfer can be done through Normal Write transfer - channels 0, 1, 2 and 3 (refer to *Section 79.7.3.1: Normal write transfer*) as shown in *Figure 2067*.

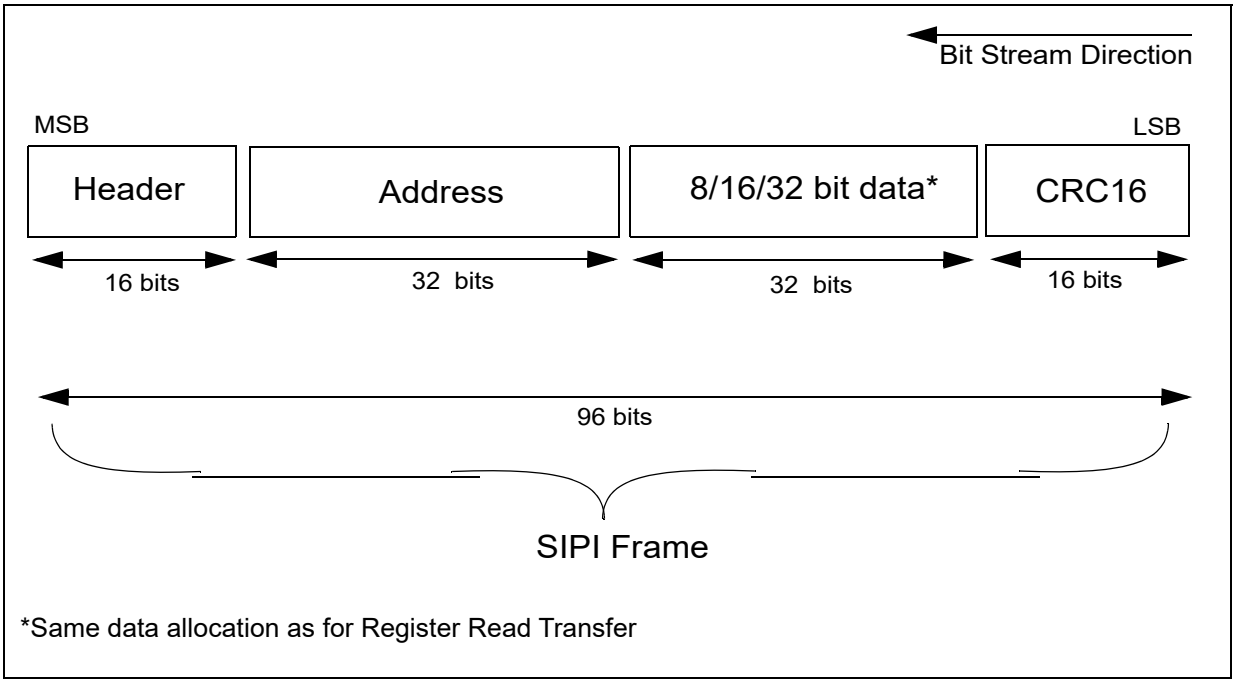
Figure 2067. Register write transfer (showing LFAST frame encapsulation)



79.7.3.1 Normal write transfer

A normal write transfer contains header, address, data (32-bit) and 16-bit CRC as shown in *Figure 2068*.

Figure 2068. Write transfer (LFAST frame encapsulation is not shown)



Note: In the case of an 8-bit transfer, address bits [31:2] are used as address and bits [1:0] are used as a byte enable:

Bits[1:0]:

- *00 – byte 3 enabled (MSB)*
- *01 – byte 2 enabled*
- *10 – byte 1 enabled*
- *11 – byte 0 enabled (LSB)*

In the case of a 16-bit transfer, address bits [31:1] are used as address and bit [0] is used as a half-word enable:

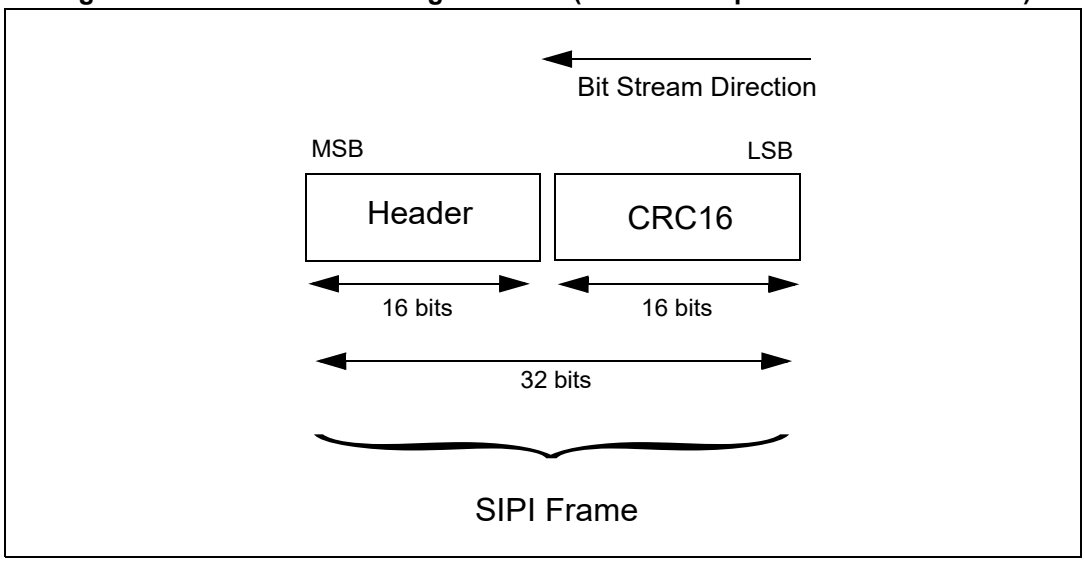
Bit[0]:

- *0 – half-word 1 enabled (MSB)*
- *1 – half-word 0 enabled (LSB)*

79.7.4 Write acknowledge transfer

A write acknowledge transfer contains only header and CRC bits (refer to *Figure 2069*). The CRC bits are calculated on the header field.

Figure 2069. Write acknowledge transfer (LFAST encapsulation is not shown)

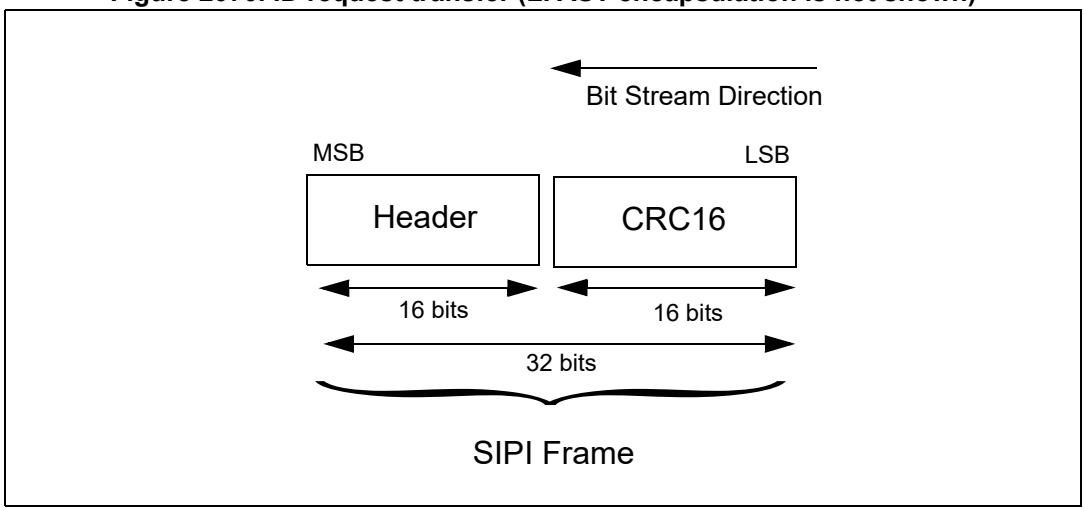


79.7.5 ID request response

79.7.5.1 ID request transfer

An ID request is transmitted by the initiator node as shown in *Figure 2070*.

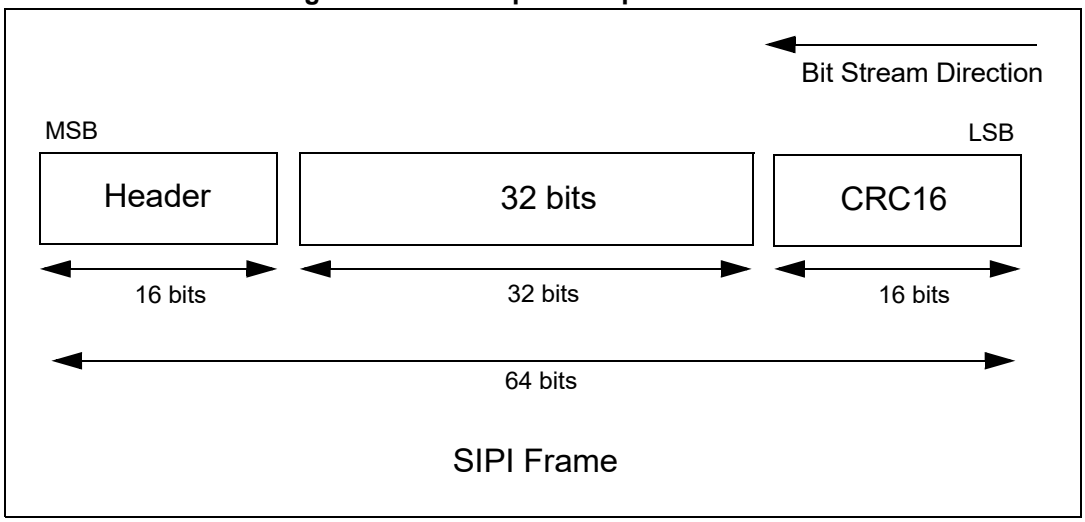
Figure 2070. ID request transfer (LFAST encapsulation is not shown)



79.7.5.2 ID request response transfer

An ID request response is transmitted by the target node as shown in *Figure 2071*.

Figure 2071. ID request response transfer



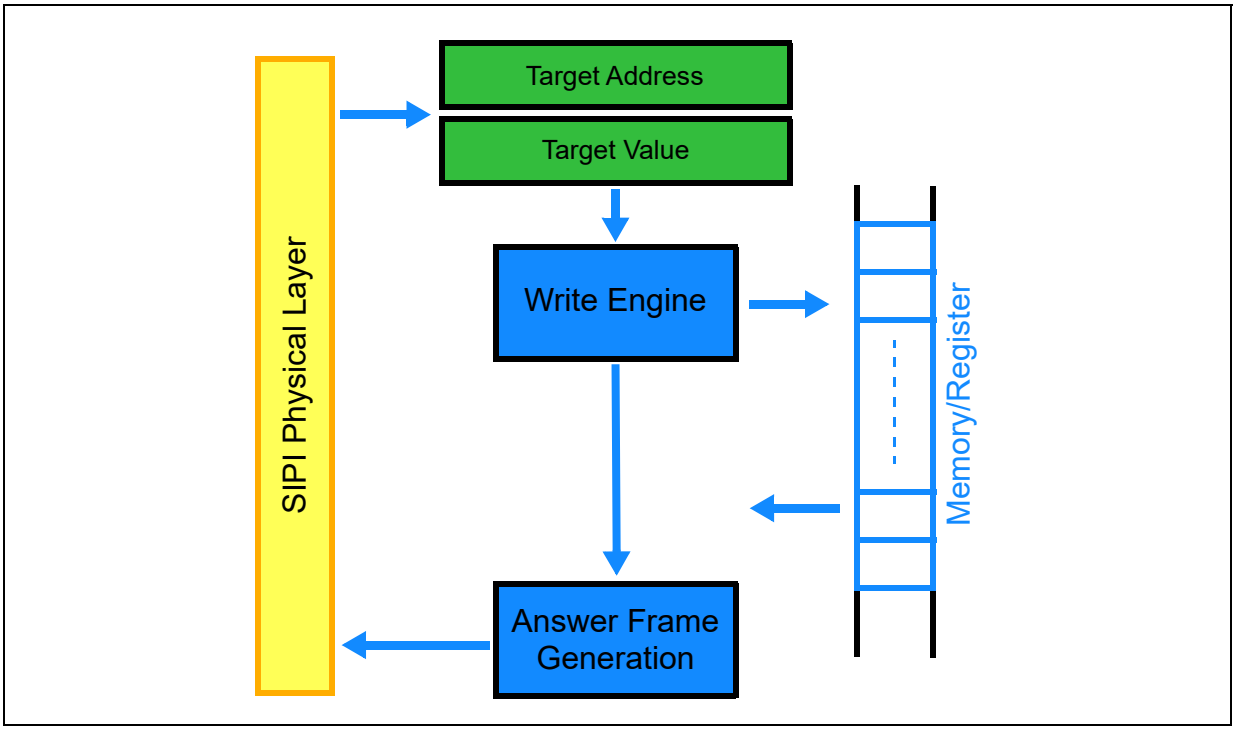
Note: The ID request response contains the value of the CHIP ID in place of data.

79.8 Transfer API and flow charts

On a single write transfer request reception (refer to *Figure 2072*):

1. Debug SIPI will place the address, data and control information on its AHB master interface.
2. When the process in completed, the Debug SIPI will generate an acknowledge frame and send it back to the LFAST.

Figure 2072. SIPI single register write API – Flow chart



On multiple write transfer request reception:

1. Debug SIPI will place the address, data and control information on its AHB Master Interface.
2. When the process in completed, Debug SIPI will generate an acknowledge frame and send it back to the LFAST.

On single read transfer request reception:

1. Debug SIPI will place the address and control information on its AHB Master Interface.
2. When the process in completed, Debug SIPI will send read response back to LFAST.

On multiple read transfer request reception:

1. Debug SIPI then will start reading data through its AHB interface.
2. When the transfer is completed/all data registers are full, it will calculate CRC and send the response frame back to LFAST.

79.9 CRC calculation

Example 1

If header is 0xAABB, address is 0x1122_3344 and data is 0xCCDD_EEFF. Then CRC calculation will take place as follows:

1. The CRC seed is initialized by 0xFFFF_FFFF.
2. All the data will be mirrored before sending to CRC engine (for example, MSB will be sent as LSB).
3. So the header will be sent as 0xDD55_0000, byte_enable = 0xC.
4. Address will be sent as 0x22CC_4488, byte_enable = 0xF.
5. Data will be sent as 0xFF77_BB33, byte enable = 0xF.

79.10 Memory map and register description

Table 1976. SIPI memory map

Offset from SIPI base (hex)	Register	Section/Page
0x0000–0x009B	Reserved	
0x009C	SIPI Module Configuration Register (SIPI_MCR)	*Section 79.10.1.1*
0x00A0	SIPI Status Register (SIPI_SR)	*Section 79.10.1.2*
0x00A4–0x00B0	Reserved	

79.10.1 Register descriptions

79.10.1.1 SIPI Module Configuration Register (SIPI_MCR)

The SIPI_MCR is a 32-bit global configuration register of the SIPI.

Offset: 0x009C Access: User read/write

	0	11	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	FRZ	0	HALT	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	1	INIT(1)	MOEN	SR
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0

1. Can only be written when SIPI_MCR[MOEN] = 1.

Figure 2073. SIPI Module Configuration Register (SIPI_MCR)

Table 1977. SIPI_MCR field descriptions

Field	Description
0 FRZ	Freeze Enable The FRZ bit specifies the SIPI behavior when Debug mode is requested at the MCU level. When FRZ is asserted, the SIPI is enabled to enter Freeze mode. Negation of this bit field causes SIPI to exit Freeze mode. 0 Not enabled to enter Freeze mode 1 Enabled to enter Freeze mode
2 HALT	Halt Mode Enable Assertion of this bit puts SIPI into Freeze mode. No Rx or Tx is performed in the SIPI until this bit is cleared. If this bit is enabled in during Tx or Rx communications, the current activity will finish, then the SIPI will enter Freeze mode. 0 No Freeze mode request. 1 Enters Freeze mode if FRZ bit is asserted.
29 INIT	Setting this bit puts the module in initialization mode. This bit should be cleared by software. Most register bits are configured using this bit. The SIPI_MCR[MOEN] bit needs to be set first, and then the INIT bit can be set and both bits cannot be enabled together. 0 Normal Mode 1 Initialization Mode
30 MOEN	Module Enable This bit should be set or cleared by software. When this bit is negated, future SIPI Tx transfers are disabled.
31 SR	Soft Reset Setting this bit clears all status and error registers and FSMs are moved to idle state. This bit is automatically cleared by hardware once the reset operation is complete. Statical configuration, data and address registers remain unchanged.

79.10.1.2 SIPI Status Register (SIPI_SR)

The SIPI_SR is the global status register of SIPI.

Offset: 0x00A0 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	FRZACK	LPMACK	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

Figure 2074. SIPI Status Register (SIPI_SR)

Table 1978. SIPI_SR field descriptions

Field	Description
0 FRZACK	Freeze Mode Acknowledge This read-only bit indicates that SIPI is in Freeze mode. The Freeze mode request cannot be granted until current transmission or reception processes have finished. The software can poll the FRZACK bit to find when the SIPI has actually entered Freeze mode. If Freeze mode is requested while SIPI is in any of the low power modes, then the FRZACK bit will only be set when the low power mode is exited. 0 SIPI not in Freeze mode 1 SIPI in Freeze mode
1 LPMACK	Low Power Mode Acknowledge This read-only bit indicates that SIPI is in Disable Mode. Disable mode can not be entered until all current transmission or reception processes have finished. The CPU can poll the LPMACK bit to know when SIPI has actually entered low power mode. 0 SIPI not in any low power modes 1 SIPI is in Disable mode

80 LVDS Fast Asynchronous Serial Transmission – High Speed Debug

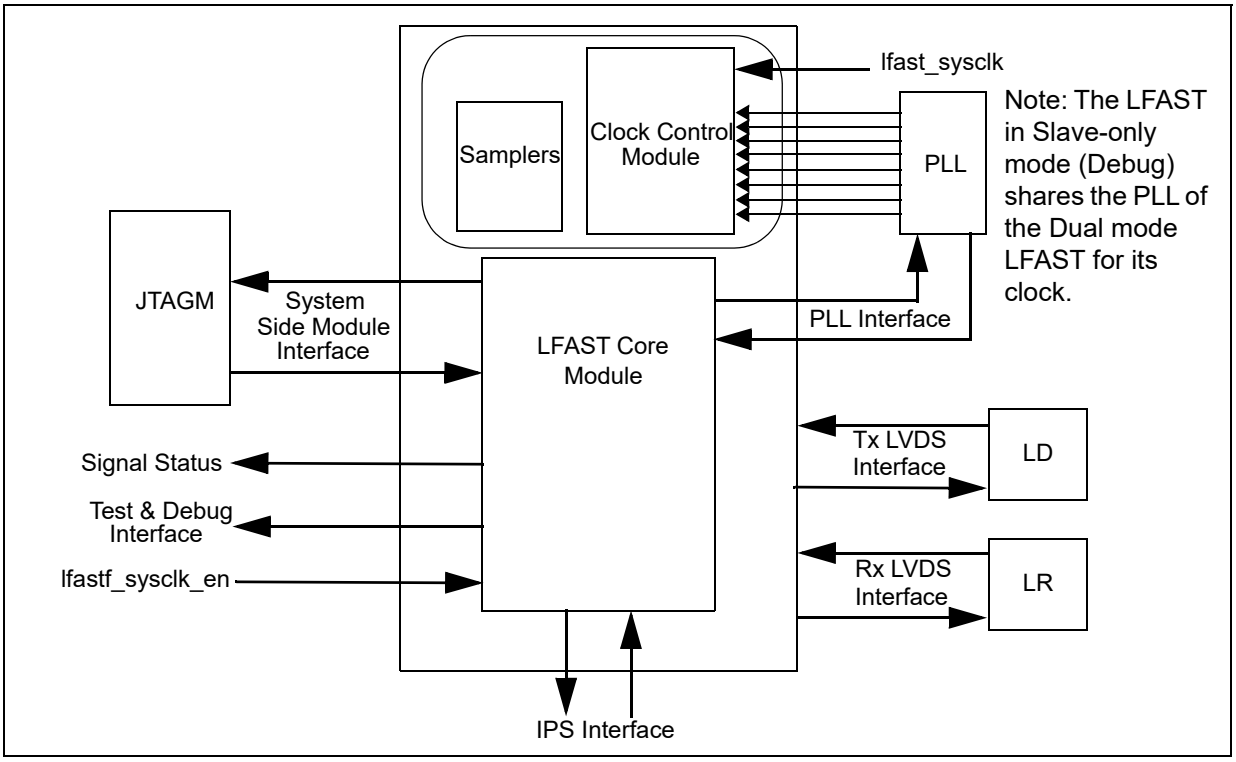
80.1 Introduction

This chapter describes the specifications of the LVDS Fast Asynchronous Serial Transmission (LFAST) module. LFAST is used in slave only mode enabling high speed debug communications through the JTAG port.

80.2 Block diagram

Figure 2075 depicts LFAST interaction with other modules on the device.

Figure 2075. LFAST block diagram



80.3 External signals

LFAST is a six pin interface with the following signals after enabling high speed debug mode:

- lfast_sysclk_en
  - Interface enable signal in LFAST Slave Only mode
- lfast_sysclk
  - Reference clock of the LFAST master and slave
- txdatap/txdatan
  - Differential transmit (Tx) interface pair
- rxdatap/rxdatan
  - Differential receive (Rx) interface pair

LFAST interface is an asynchronous high speed LVDS interface.

80.3.1 LFAST operating data rates

The change of data rate is controlled by the LFAST master by issuing appropriate Interface Control Logical Channel (ICLC) packets to the LFAST slave. Henceforth, the data rate 6.5 Mbps/5 Mbps (lfast_sysclk ÷ 4 or lfast_sysclk ÷ 2) is referred to as low data rate, and the high data rate is in the Data Sheet.

80.4 LFAST frame structure

LFAST frames are transmitted and received across a serial interface using a packeted asynchronous communication format.

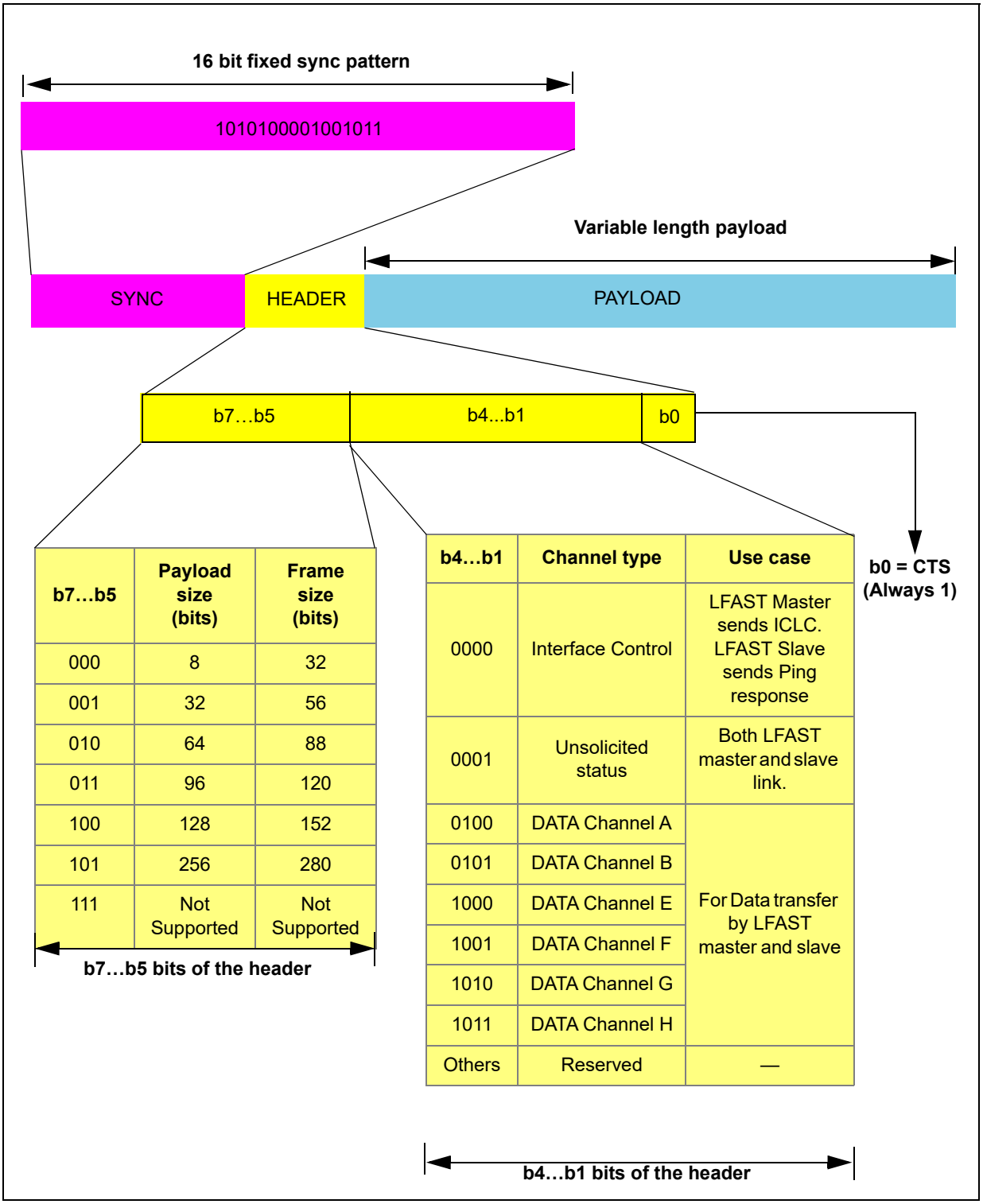
A LFAST frame is made up of three fields:

- Sync pattern
- Header
- Payload

Sync pattern and header fields are of fixed length. A sync pattern is used to synchronize incoming data stream in LFAST module.

Header field of the frame distinguishes various types of data and control transferred and also contains information about the length of the payload. The payload field is the actual data that is transferred across the channel. See *Figure 2076* for details of the LFAST frame.

Figure 2076. LFAST frame structure

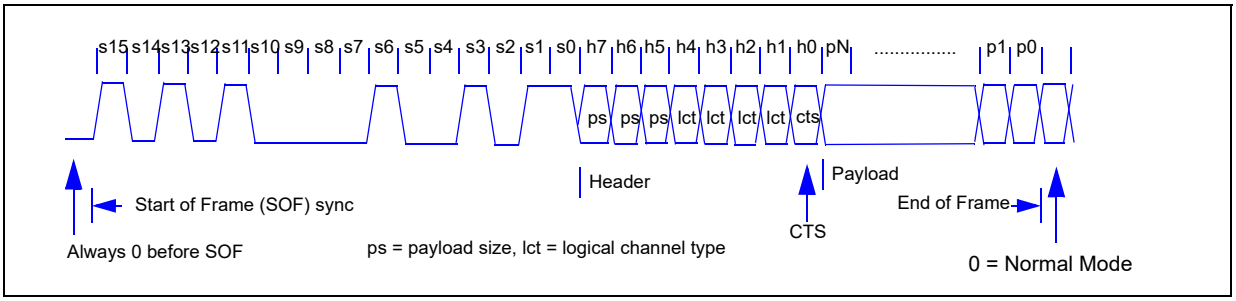


b7…b5	Payload size (bits)	Frame size (bits)
000	8	32
001	32	56
010	64	88
011	96	120
100	128	152
101	256	280
111	Not Supported	Not Supported

b4…b1	Channel type	Use case
0000	Interface Control	LFAST Master sends ICLC. LFAST Slave sends Ping response
0001	Unsolicited status	Both LFAST master and slave link.
0100	DATA Channel A	For Data transfer by LFAST master and slave
0101	DATA Channel B	
1000	DATA Channel E	
1001	DATA Channel F	
1010	DATA Channel G	
1011	DATA Channel H	
Others	Reserved	—

The same protocol is used on both transmit and receive interfaces for communications of data, control and status information. A synchronization pattern (16 bits) and header (8 bits) are present in every frame.

Figure 2077. Serial frame structure



The frame structure is shown in *Figure 2077* and consists of the following:

- **16-bit synchronization pattern:** Used for clock synchronization and pattern recognition. The frame synchronization code at the start of every frame is a unique reserved word that is used to identify whether the data received is the start of the frame and for synchronization (clock phase extraction) with the stream data. A synchronous sequence is used to give a high quality auto-correlation. The LFAST 16-bit synchronization sequence = 1010_1000_0100_1011b = A84Bh.
- **Header - 3 MSB (b7 - b5):** Defines the payload size according to *Table 1979*.

Table 1979. Header payload sizes

b7-b5 (bin)	Payload size (bits)	Frame size (bits)
000	8	32
001	32	56
010	64	88
011	96	120
100	128	152
101	256	280
110	288	312
111	—	—

- **Header - (b4 - b1):** Defines the logical channel types, which indicate the type of payload that the frame carries. How the payload field of a frame is used for any other logical channel type is system side module specific except in the case of the interface control logical channel type.
- **Header - (b0):** CTS on the both LFAST master and slave devices. In slave only mode the CTS field will always be 1.
- **Payload: Content dependent upon frame type.**
- **Bit after frame:** This bit determines entry into Sleep mode (1 = Sleep mode, 0 = normal mode).

80.5 Features

- Supports slave only mode
- Supports asynchronous data transfer rates up to the maximum data rated shown in the product Datasheet
- Transmits and receives data and unsolicited frames
- Receives ICLC frames
- Supports processor controlled transfer of ICLC frame with 8-bit payload size to implement the data rate changes and test modes
- Supports flow control using sliding window protocol
- Provides transmit of data frame length with 96 bits of payload size and reception of data frame with 128 bits of payload size
- Provides configurable frame length for unsolicited frame with variable payload sizes of 8, 32, 64, 96, 128, 256 or 288 bits
- Supports PLL configuration (such as feedback loop divider) through registers
- Supports LVDS configuration through registers
- Supports multiple loopback modes for checking the physical interface
- Supports automatic ping response generation
- Supports for detection of unsupported channel number and unsupported payload size

80.6 Memory map and register definition

All registers are 32-bit wide.

80.6.1 Memory map

Table 1980. LFAST memory map

Offset from LFAST_BASE (hex)	Register	Access	Reset Value (hex)	Location
0000	LFAST Mode Configuration Register (MCR)	R/W	0x0000_0010	*Section 80.6.2.1*
0004	LFAST Speed Control Register (SCR)	R/W	0x0001_0000	*Section 80.6.2.2*
0008	LFAST Correlator Control Register (COCR)	R/W	0x0000_000E	*Section 80.6.2.3*
000C	LFAST Test Mode Control Register (TMCR)	R/W	0x0000_0000	*Section 80.6.2.4*
0010	LFAST Auto Loopback Control Register (ALCR)	R/W	0x0000_0000	*Section 80.6.2.5*
0014	LFAST Rate Change Delay Control Register (RCDCR)	R/W	0x000F_0000	*Section 80.6.2.6*
0018	LFAST Wakeup Delay Control Register (SLCR)	R/W	0x1201_5F02	*Section 80.6.2.7*
001C	Reserved			
0020	LFAST Ping Control Register (PICR)	R/W	0x0000_80CA	*Section 80.6.2.8*
0024–002C	Reserved			
0030–0038	Reserved – Access will not generate errors			

Table 1980. LFAST memory map (continued)

Offset from LFAST_BASE (hex)	Register	Access	Reset Value (hex)	Location
003C	Reserved			
0040	LFAST LVDS Control Register (LCR)	R/W	0x0000_5024	*Section 80.6.2.9*
0044	LFAST Unsolicited Tx Control Register (UNSTCR)	R/W	0x0000_0000	*Section 80.6.2.10*
0048	LFAST Unsolicited Tx Data Registers 8 (UNSTDR8)	R/W	0x0000_0000	*Section 80.6.2.11*
004C	LFAST Unsolicited Tx Data Registers 7 (UNSTDR7)	R/W	0x0000_0000	*Section 80.6.2.11*
0050	LFAST Unsolicited Tx Data Registers 6 (UNSTDR6)	R/W	0x0000_0000	*Section 80.6.2.11*
0054	LFAST Unsolicited Tx Data Registers 5 (UNSTDR5)	R/W	0x0000_0000	*Section 80.6.2.11*
0058	LFAST Unsolicited Tx Data Registers 4 (UNSTDR4)	R/W	0x0000_0000	*Section 80.6.2.11*
005C	LFAST Unsolicited Tx Data Registers 3 (UNSTDR3)	R/W	0x0000_0000	*Section 80.6.2.11*
0060	LFAST Unsolicited Tx Data Registers 2 (UNSTDR2)	R/W	0x0000_0000	*Section 80.6.2.11*
0064	LFAST Unsolicited Tx Data Registers 1 (UNSTDR1)	R/W	0x0000_0000	*Section 80.6.2.11*
0068	LFAST Unsolicited Tx Data Registers 0 (UNSTDR0)	R/W	0x0000_0000	*Section 80.6.2.11*
006C–007C	Reserved			
0080	LFAST Global Status Register (GSR)	R	0x0000_0000	*Section 80.6.2.12*
0084–0090	Reserved			
0094	LFAST Data Frame Status Register (DFSR)	R	0x0000_0000	*Section 80.6.2.13*
0098	LFAST Tx Interrupt Status Register (TISR)	R/W	0x0000_0000	*Section 80.6.2.14*
009C	LFAST Rx Interrupt Status Register (RISR)	R/W	0x0000_0000	*Section 80.6.2.15*
00A0	LFAST Rx ICLC Interrupt Status Register (RIISR)	R/W	0x0000_0000	*Section 80.6.2.16*
00A4	LFAST PLL and LVDS Status Register (PLLLSR)	R	0x0002_0003	*Section 80.6.2.17*
00A8	LFAST Unsolicited Rx Status Register (UNSRSR)	R/W	0x0000_0000	*Section 80.6.2.18*
00AC	LFAST Unsolicited Rx Data Register 8 (UNSRDR8)	R	0x0000_0000	*Section 80.6.2.19*
00B0	LFAST Unsolicited Rx Data Register 7 (UNSRDR7)	R	0x0000_0000	*Section 80.6.2.19*
00B4	LFAST Unsolicited Rx Data Register 6 (UNSRDR6)	R	0x0000_0000	*Section 80.6.2.19*
00B8	LFAST Unsolicited Rx Data Register 5 (UNSRDR5)	R	0x0000_0000	*Section 80.6.2.19*
00BC	LFAST Unsolicited Rx Data Register 4 (UNSRDR4)	R	0x0000_0000	*Section 80.6.2.19*
00C0	LFAST Unsolicited Rx Data Register 3 (UNSRDR3)	R	0x0000_0000	*Section 80.6.2.19*
00C4	LFAST Unsolicited Rx Data Register 2 (UNSRDR2)	R	0x0000_0000	*Section 80.6.2.19*

Table 1980. LFAST memory map (continued)

Offset from LFAST_BASE (hex)	Register	Access	Reset Value (hex)	Location
00C8	LFAST Unsolicited Rx Data Register 1 (UNSRDR1)	R	0x0000_0000	*Section 80.6.2.19*
00CC	LFAST Unsolicited Rx Data Register 0 (UNSRDR0)	R	0x0000_0000	*Section 80.6.2.19*

80.6.2 Register descriptions

80.6.2.1 Mode Configuration Register (MCR)

Offset: 0x0000 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	LSSEL
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	DRFEN	RXEN	TXEN	0	0	0	0	0	0	0	0	TXARBD	0	0	DRFRST(1)	DATAEN
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0

1. Set by user software and cleared by system hardware.

Figure 2078. Mode Configuration Register (MCR)

Table 1981. MCR field descriptions

Field	Description
15 LSSEL	Selects the fraction of sysclk in Low Speed Select mode Refer to *Section 80.10.2: Slow speed clock* for more detailed description. 0 Low Speed Mode in which where lfast_sysclk input is used to generate 4 phases of lfast_sysclk/2. 1 Low Speed Mode in which where lfast_sysclk input is used to generate 4 phases of lfast_sysclk/4.
16 DRFEN	LFAST Enable This bit enables/disables the reception and transfer of LFAST device. 0 LFAST is immediately disabled. All current/pending requests are terminated and the Tx and Rx data FIFOs are flushed. If this bit is cleared in the middle of a transmit/receive operation, then that operation is terminated immediately and nothing is transmitted/received further. All the programmable registers retain their values and status registers are cleared to their reset values. Registers read/write operations can be performed through the IPS Bus. 1 LFAST is Enabled.

Table 1981. MCR field descriptions (continued)

Field	Description
17 RXEN	LFAST Receiver Enable This bit controls the reception of the frames and decoding on the LFAST device. This bit also disables the Rx LVDS LR. 0 Receiver Interface is disabled. If this bit is cleared during a data transfer, the current frame is received and then the Rx block is disabled. After the Rx block is disabled, all new frames from LFAST peer device are ignored. System Side Module Rx interface is not disabled by this bit. 1 Receiver Interface is Enabled.
18 TXEN	LFAST Transmitter Enable This bit controls the transmission of frames from the LFAST device and disables the Tx LVDS LD. This bit can also be set and cleared by LFAST slave hardware on reception of an ICLC command frame. 0 LFAST transmitter Interface is disabled. No new request is accepted but ongoing request is served. 1 Not allowed. Do not set when in slave only mode.
27 TXARBD	Tx Arbiter Disable This bit enables/disables the Tx block arbiter. Current frame transfer is completed, but new frame requests are ignored. 0 Enable Tx arbiter and framer. When enabled it takes all the frame request and services based on priority. 1 Disable Tx arbiter and framer. All frame requests are ignored.
30 DRFRST	LFAST Soft Reset This bit is automatically cleared after reset. 0 No soft reset. 1 Soft reset to LFAST is asserted. When set it causes a reset of the LFAST module; all the registers will be reset to their default values and all the FIFOs will be flushed.
31 DATAEN	DATA Frame Enable This bit enables/disables the transmission and reception of data frames between the LFAST master and slave devices. 0 Data frame transmission and reception is disabled. Tx data frame requests are ignored by the transmitter. Frame with LCT of data frame is ignored by the receiver. 1 Data frame transmission and reception is enabled. Tx data frame requests are serviced by the transmitter. Frame with LCT of data frame is received and placed into the Rx data FIFO.

80.6.2.2 Speed Control Register (SCR)

The SCR is used to configure the Rx and Tx data rate of the LFAST.

Offset: 0x0004 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	DRMD(1)
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	RDR	0	0	0	0	0	0	0	TDR
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. Only writable when MCR[DRFEN] = 0.

Figure 2079. Speed Control Register (SCR)

Table 1982. SCR field descriptions

Field	Description
15 DRMD	Data Rate Controller mode Defines the mode set for LFAST slave device by the software or the LFAST master. This bit should not be modified by the software and left as SCR[DRMD] =1.
23 RDR	Receiver Data Rate This bit defines the receiver data rate. This bit should not be modified by the software.
31 TDR	Transmit Data Rate This bit defines the transmitter data rate. This bit should not be modified by the software.

80.6.2.3 Correlator Control Register (COCR)

The COCR is used to select the sampler data path and the number of bits of correlation to be used.

Offset: 0x0008 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	SMPSEL(1)								0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	CORRTH(1)			PHSSEL(1)
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	0

1. Only writable only when MCR[RXEN] = 0.

Figure 2080. Correlator Control Register (COCR)

Table 1983. COCR field descriptions

Field	Description
0:7 SMPSEL	Sampler Data Path Selector (overrides the correlator selection) Defines the sampler data path to be activated at all the times. All the bits should be 0 (00h) for Sampler Data Path to be selected by the correlator. In Low Speed mode only Sampler Data Paths 0-3 are valid. 0x00 Sampler Data Path selected by correlator 0x01 Sampler Data Path 0 selected 0x02 Sampler Data Path 1 selected 0x04 Sampler Data Path 2 selected 0x08 Sampler Data Path 3 selected 0x10 Sampler Data Path 4 selected 0x20 Sampler Data Path 5 selected 0x40 Sampler Data Path 6 selected others Sampler Data Path 7 selected
28:30 CORRTH	Correlator threshold level. Defines the correlation threshold level. 000 9 Bits of correlation 001 10 Bits of correlation …. 110 15 Bits of correlation 111 16 Bits of correlation
31 PHSSEL	Polyphase 8 or 4 phase selection Defines the number of phases for the polyphase generator used. This bit is ignored in low speed mode since only 4 Phases are used in low speed mode. In High Speed mode phase 0, 2, 4 and 6 are used when 4 phase mode is selected. 0 8 phases 1 4 phases

80.6.2.4 Test Mode Control Register (TMCR)

The TMCR enables and configures the LFAST clock test and loopback modes

Offset: 0x000C Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	CLKTST	LPON	0	0	0	0	0	LPMOD(1)		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	LPFRMTH(1)															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. Writable only when TMCR[LPON] = 0.

Figure 2081. Test Mode Control Register (TMCR)

Table 1984. TMCR field descriptions

Field	Description
6 CLKTST	Clock Test mode The software can define when the clock test mode is enabled. This bit can also be set and cleared by hardware on reception of an ICLC command. This field can only be written when MCR[DRFEN] = 1. 1 Clock Test mode enabled 0 Clock Test mode disabled
7 LPON	Loopback mode Logic Enable The software can define when the loopback logic is enabled. This bit can also be written by LFAST slave hardware on reception of an ICLC command. This field can only be written when MCR[DRFEN] = 1. 1 Loopback mode is enabled. 0 Loopback mode is disabled.
13:15 LPMOD	Loopback mode Defines the type of loopback mode enabled. 000 Rx loopback 001 Rx LVDS loopback 010 Tx loopback without automatic frame generation 011 Tx loopback with automatic frame generation 100 Tx LVDS loopback (external) with automatic frame generation Others - Reserved (should not be written by the software)
16:31 LPFRMTH	Loopback check mode valid pass frames threshold value. Defines the number of frames to verify before setting GCR[LPFPDV] when running in Automatic Loopback Frame mode. The loopback frame is considered pass when the payload is CBh, header is 13h and sync is valid. This mode is valid only when TMCR[LPMOD] = 011b or TMCR[LPMOD] = 100b. 0x0000: Reserved (Not to be used) 0x0001: Check 1 frame have correct sync, header and payload. … 0xFFFE: Check 65534 frames have correct sync, header and payload. 0xFFFF: Check 65535 frames have correct sync, header and payload.

80.6.2.5 Auto Loopback Control Register (ALCR)

Offset: 0x0010 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	LPCNTEN(1)
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	LPFMCNT(1)															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. Writable only when TMCR[LPON] = 0.

Figure 2082. Auto Loopback Control Register (ALCR)

Table 1985. ALCR field descriptions

Field	Description
15 LPCNTEN	Auto Loopback Frame Transmission Count Enable Enables fixed number of auto predefined loopback frame transmission. 0 Infinite predefined loopback frame transmission enabled 1 Fixed count of predefined loopback frame transmission enabled
16:31 LPFMCNT	Auto Loopback Frame Transmission Count Defines the number of predefined auto loopback frames to be sent. The predefined loopback frame has payload CBh, header 13h and valid sync. This mode is valid if TMCR[LPMOD] = 011b or TMCR[LPMOD] = 100b. 0x0000: Reserved (Not to be used) 0x0001: Send 1 frame with correct sync, header and payload. ... 0xFFFE: Send 65534 frames with correct sync, header and payload. 0xFFFF: Send 65535 frames with correct sync, header and payload.

80.6.2.6 Rate Change Delay Control Register (RCDCR)

Offset: 0x0014 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	DRCNT(1)			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1

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

1. Only writable when MCR[DRFEN] = 1.

Figure 2083. Rate Change Delay Control Register (RCDCR)

Table 1986. RCDCR field description

Field	Description
12:15 DRCNT	Data Rate Controller Counter Value Defines the number of cycles of Phase 0 clock needed by the Tx interface Data rate change controller to switch from one speed mode to another. The arbitrator ignores all requests during this period. 0x0: 0 cycles 0x1: 1 cycles ... 0xE: 14 cycles 0xF: 15 cycles

80.6.2.7 Wakeup Delay Control Register (SLCR)

Offset: 0x0018 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	HSCNT(1)								0	0	0	0	LSCNT(1)			
W																
Reset	0	0	0	1	0	0	1	0	0	0	0	0	0	0	0	1

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	HWKCNT(1)								0	0	0	0	LWKCNT(1)			
W																
Reset	0	1	0	1	1	1	1	1	0	0	0	0	0	0	1	0

1. Only writable when MCR[DRFEN] = 0.

Figure 2084. Wakeup Delay Control Register (SLCR)

Table 1987. SLCR field descriptions

Field	Description
0:7 HSCNT	High Speed Sleep mode Exit Time Defines 1/4 of the number of the high speed clock cycle wait after the negation of the LVDS LD sleep signal, and before the start of new frame. This wait is the summation of settling time of the Line Driver (LD) after negation of its sleep signal, and the wakeup time of the LR of the peer LFAST. 0x00: cycle 0x01: 1 cycle … 0x12: 18 cycles (200 ns + 8 cycles of High speed clock) … 0xFE: 254 cycles 0xFF: 255 cycles
12:15 LSCNT	Low Speed Sleep mode Exit Time Defines 1/4 of the number of Low speed clock cycle wait after the negation of LVDS LD sleep signal, and before the start of new frame. This wait is the summation of settling time of LD after negation of LVDS LD sleep signal, and the wakeup time of the LR of the peer LFAST. 0x0: 0 cycle 0x1: 1 cycle (200ns + 1 cycle of Low speed clock) … 0xE: 14 cycles 0xF: 15 cycles
16:23 HWKCNT	Wake Up time for the LD Defines the 1/4 of the number of High speed clock cycles used during the wakeup of the LD from shutdown mode. This is time required by the LD to move from moving from Shutdown to Normal State in High speed mode. 0x00: 0 cycles 0x01: 1 cycle … 0x5F: 95 cycles (1.18 us) … 0xFF: 255 cycles Maximum
28:31 LWKCNT	Wake Up time for the LD Defines the 1/4 of the number Low speed clock used during the wakeup of the LD from shutdown mode. This is time required by the LD to move from Shutdown to Normal State in Low speed mode. 0x0: 0 cycle 0x1: 1 cycle 0x2: 2 cycles (1.18 us) … 0xE: 14 cycles 0xF: 15 cycles

80.6.2.8 Ping Control Register (PICR)

Offset: 0x0020 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	PNGREQ(1)
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	PNGAUTO(2)	0	0	0	0	0	0	0	PNGPYLD(2)							
W																
Reset	1	0	0	0	0	0	0	0	1	1	0	0	1	0	1	0

1. Set by user software and cleared by system hardware.
2. Only writable when MCR[DRFEN] = 0.

Figure 2085. Ping Control Register (PICR)

Table 1988. PICR field descriptions

Field	Description
15 PNGREQ	Ping Response Frame Request This bit is set to initiate the transfer of Ping response frame. Cleared after transmission of Ping response frame. This field can only be written when MCR[DRFEN] = 1. 1 Ping response frame transmission request is queued. 0 No pending Ping response frame transmission request.
16 PNGAUTO	Ping Response Enable Defines when ping response should be automatically sent on reception of Ping ICLC frame from LFAST master This bit should not be modified by the software.
24:31 PNGPYLD	Defines the LFAST slaves ping reply frame payload content.

80.6.2.9 LVDS Control Register (LCR)

Offset: 0x0040 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	SWWKLD(1)	SWSLPLD(1)	SWWKLR(1)	SWSLPLR(1)	SWOFFLD	SWONLD	SWOFFLR	SWONLR
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	LVRXOFF	LVTXOE	TXCMUX	LVRFEN	LVLPEN	0	0	0	0	0	LVRXOP_TR	Reserved	LVRXOP_BR	LVTXOP	LVCKSS	LVCKP
W																
Reset	0	1	0	1	0	0	0	0	0	0	1	0	0	1	0	0

1. Set by user software and cleared by system hardware.

Figure 2086. LVDS Control Register (LCR)

Table 1989. LCR field description

Field	Description
8 SWWKLD	Software signal to take LVDS LD out of Sleep mode This bit can be changed only when DRFEN is high. 0 No effect 1 LVDS LD will be taken out of sleep (provided no other source is trying to put it in sleep).
9 SWSLPLD	Software signal to put LVDS LD into Sleep mode This bit can be changed only when DRFEN is high. 0 No effect 1 LVDS LD will be put in sleep.
10 SWWKLR	Software signal to take LVDS LR out of Sleep mode This bit can be changed only when DRFEN is high. 0 No effect 1 LVDS LR will be taken out of sleep (provided no other source is trying to put it in sleep).
11 SWSLPLR	Software signal to put LVDS LR into Sleep mode This bit can be changed only when DRFEN is high. 0 No effect 1 LVDS LR will be put in sleep (provided no other source is trying to wake it up).
12 SWOFFLD	Software signal to turn OFF the LVDS LD This bit can be changed only when DRFEN is high. This bit should not be modified by the software.

Table 1989. LCR field description (continued)

Field	Description
13 SWONLD	Software signal to turn ON the LVDS LD This bit can be changed only when DRFEN is high. This bit should not be modified by the software.
14 SWOFFLR	Software signal to turn OFF the LVDS LR This bit can be changed only when DRFEN is high. This bit should not be modified by the software.
15 SWONLR	Software signal to turn ON the LVDS LR This bit can be changed only when DRFEN is high. This bit should not be modified by the software.
16 LVRXOFF	Indicates the value driven onto LVDS LR output when in shutdown mode.
17 LVTXOE	LVDS LD output buffer enable 0 LVDS LD output buffer enable is disabled. 1 LVDS LD output buffer enabled
18 TXCMUX	Tx and Clock Mux The bit can be used to bring out PLL Phase 0 clock on Tx LVDS pad. 0 No effect 1 PLL Phase 0 clock will be brought out to Tx LVDS pad.
19 LVRFEN	LVDS pad reference enable 0 LVDS reference pad disabled 1 LVDS reference pad enabled
20 LVLPEN	Tx LVDS internal loopback enable The internal loopback is not intended for board level functionality, so this feature should not be implemented. 0 Tx LVDS normal mode enabled 1 Tx LVDS internal loopback mode enabled
26 LVRXOP_TR	Used to enable or disable the on-chip receiver termination resistor in LFAST mode (applies to LVDS pad use for LFAST only). 0 Disable on-chip LFAST receiver termination 1 Enable on-chip LFAST receiver termination
28 LVRXOP_BR	Used to set the bias current for the receiver in LFAST mode. It is recommended to always write 1 to this bit when using the LFAST interface. Writing 0 will allow for a small power savings during lower baud rates (applies to LVDS pad use for LFAST only). 0 Use for LFAST receiver baud rates less than maximum baud rate 1 Required for LFAST receiver maximum baud rate
29 LVTXOP	Control signal for LFAST and Micro-Second Bus selection 0 Micro-Second Bus selection 1 LFAST Bus selection

Table 1989. LCR field description (continued)

Field	Description
30 LVCKSS	LVDS Clock Sync Select This bit is used to adjust the LVDS data sampling to the duty cycle of the clock. It is recommended that a value of zero is used in all cases. A value of one is reserved for specific devices/revisions with different clock timing. 0 Normal clock used to sample the LVDS data 1 Adjusted clock used to sample the LVDS data
31 LVCKP	LVDS clock select This bit is used for selecting use of direct pll clock or inverted pll clock in LVDS. 0 Direct pll clock to be used inside the LVDS 1 Inverted pll clock to be used inside the LVDS

80.6.2.10 Unsolicited Tx Control Register (UNSTCR)

Offset: 0x0044 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	USNDRQ(1)
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	UNSHDR(2)						
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. Set by user software and cleared by system hardware.
2. Only writable when UNSTCR[USNDRQ] = 0.

Figure 2087. Unsolicited Tx Control Register (UNSTCR)

Table 1990. UNSTCR field descriptions

Field	Description
15 USNDRQ	Tx Unsolicited send request. This field can only be written when MCR[DRFEN] = 1. 0 No valid Unsolicited frame exists 1 Valid Unsolicited frame exists for transmission
25:31 UNSHDR	Tx Unsolicited message header

80.6.2.11 LFAST Unsolicited Tx Data Registers (UNSTDR*n*)

Offset: 0x0048 (UNSTDR8)
0x004C (UNSTDR7)
0x0050 (UNSTDR6)
0x0054 (UNSTDR5)
0x0058 (UNSTDR4)
0x005C(UNSTDR3)
0x0060 (UNSTDR2)
0x0064 (UNSTDR1)
0x0068 (UNSTDR0)

Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	UNTXD(1)															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	UNTXD(1)															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. Only writable when UNSTCR[USNDRQ] = 0.

Figure 2088. Unsolicited Tx Data Registers (UNSTDR8–UNSTDR0)

Table 1991. UNSTDR8–UNSTDR0 field descriptions

Field	Description
0:31 UNTXD	Unsolicited Transmit Data 8–0 This represents 9 registers for Unsolicited transmit data. The first bit to transmitted as part of the payload will be from UNTXD8[31], second bit from UNTXD8[30], and so on. So the last bit transmitted will be from UNTXD0[0] in case of 288 bit payload.

80.6.2.12 Global Status Register (GSR)

Offset: 0x0080 Access: User read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	DUALMD	0	0	0	0	0	0	0	0	0	0	0	0	LRMD	LDSM	DRSM
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	LPTXDN	LPFPDV	LPCPDV	LPCHDV	LPCSDV
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 2089. Global Status Register (GSR)

Table 1992. GSR field descriptions

Field	Description
0 DUALMD	Indicates the LFAST module is in Dual mode 0 LFAST Module Slave only mode 1 LFAST Module in dual mode
13 LRMD	Indicates if the Rx Controller is idle/active and that the Rx clocks are enabled. In functional mode this will always be active. 0 Rx Controller is in Idle state. 1 Rx Controller is active.
14 LDSM	Transmit Interface Data Rate Status Indicates the current speed rate of the Tx controller. 0 Data rate of LOW speed mode 1 Data rate of HIGH speed mode
15 DRSM	Receive Interface Data Rate Status Indicates the current speed rate of the Rx controller. 0 Data rate of LOW speed mode 1 Data rate of HIGH speed mode
27 LPTXDN	Auto loopback frame transmission count reached The Count of frame is defined by *Auto Loopback Control Register (ALCR)* LPFMCNT. 0 Auto loopback frame transmission count not reached 1 Auto loopback frame transmission count reached
28 LPFPDV	Loopback frame pass threshold reached 0 Pass frame threshold not reached 1 Pass frame threshold achieved

Table 1992. GSR field descriptions (continued)

Field	Description
29 LPCPDV	Valid payload received during loopback check mode Indicates whether the Loop back frame received payload is 0xCB. 0 Payload received is not 0xCB. 1 Payload received is 0xCB.
30 LPCHDV	Valid header received during loopback check mode Indicates whether the Loop back frame received header is 0x13. 0 Header received is not 0x13. 1 Header received is 0x13.
31 LPCSDV	Valid synchronization received 0 Valid Synchronization pattern not detected 1 Valid Synchronization pattern detected

80.6.2.13 Data frame Status Register (DFSR)

Offset: 0x0094 Access: User read only

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	RXDCNT						0	0	0	0	0	RXFCNT		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R			TXDCNT						0	0	0	0	0	TXFCNT		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 2090. Data frame Status Register (DFSR)

Table 1993. DFSR field descriptions

Field	Description
2:7 RXDCNT	Unread Rx Frame Data Count Indicates the number of unread data stored in the Rx Data FIFO.
13:15 RXFCNT	Unread Rx Frame Count Indicates the number of unread data frames stored in the Rx Data FIFO.
18:23 TXDCNT	Unread Tx Frame Data Count Indicates the number of unread data stored in the Tx Data FIFO.
29:31 TXFCNT	Unread Tx Frame Count Count of pending Data Frames programmed by System Side Module.

80.6.2.14 Tx Interrupt Status Register (TISR)

Offset: 0x0098 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	TXIEF	TXOVF
W															w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	TXPNGF	0	TXUNSF	TXICLCF	TXDTF
W												w1c		w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 2091. Tx Interrupt Status Register (TISR)

Table 1994. TISR field descriptions

Field	Description
14 TXIEF	Tx Data Interface not enabled Tx Data Interface not enabled and a frame is ready to be transmitted. 0 Interrupt event has not occurred. 1 Interrupt event has occurred.
15 TXOVF	Transmit Data FIFO Overflow Interrupt 0 Interrupt event has not occurred. 1 Interrupt event has occurred.
27 TXPNGF	Ping response frame transmitted interrupt 0 Interrupt event has not occurred. 1 Interrupt event has occurred.
29 TXUNSF	Unsolicited Frame transmitted Interrupt 0 Interrupt event has not occurred. 1 Interrupt event has occurred.
30 TXICLCF	ICLC Frame transmitted Interrupt 0 Interrupt event has not occurred. 1 Interrupt event has occurred.
31 TXDTF	Data Frame transmitted Interrupt 0 Interrupt event has not occurred. 1 Interrupt event has occurred.

80.6.2.15 Rx Interrupt Status Register (RISR)

Offset: 0x009C Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	RXUOF	0	0	RXUFF	RXOFF	RXSZF	RXICF	RXLCEF
W									w1c			w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	RXCTSF	RXDF	RXUNSF	0
W													w1c	w1c	w1c	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 2092. Rx Interrupt Status Register (RISR)

Table 1995. RISR field descriptions

Field	Description
8 RXUOF	Unsolicited frame register overflow Indicates existing unsolicited frame has not been read and a new unsolicited frame has arrived. 0 Interrupt event has not occurred. 1 Interrupt event has occurred.
11 RXUFF	Rx Data FIFO Underflow 0 Interrupt event has not occurred. 1 Interrupt event has occurred.
12 RXOFF	Rx Data FIFO Overflow 0 Interrupt event has not occurred. 1 Interrupt event has occurred.
13 RXSZF	Frame with unsupported frame size received 0 Interrupt event has not occurred. 1 Interrupt event has occurred: On reception of frame with payload size for a frame other than mentioned in *Table 2008*.
14 RXICF	Invalid ICLC code Received 0 Interrupt event has not occurred. 1 Interrupt event has occurred.
15 RXLCEF	Invalid Logical Channel Type 0 Interrupt event has not occurred. 1 Interrupt event has occurred: On reception of frame other than mentioned in *Table 2008*.
28 RXCTSF	Frame with CTS bit Low Received 0 Interrupt event has not occurred. 1 Interrupt event has occurred.

Table 1995. RISR field descriptions (continued)

Field	Description
29 RXDF	Data frame received 0 Interrupt event has not occurred. 1 Interrupt event has occurred.
30 RXUNSF	Unsolicited Frame received 0 Interrupt event has not occurred. 1 Interrupt event has occurred.

80.6.2.16 Rx ICLC Interrupt Status Register (RIISR)

Offset: 0x00A0 Access: User read/write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	ICPFF	ICPSF	ICPRF	ICTOF	ICLPF	ICCTF	ICTDF	ICTEF	ICRFF	ICRSF	ICTFF	ICTSF	ICPOFF	ICPONF
W			w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 2093. Rx ICLC Interrupt Status Register (RIISR)

Table 1996. RIISR field descriptions

Field	Description
18 ICPFF	Ping Frame Response Payload mismatch 0 Interrupt event has not occurred. 1 Interrupt event has occurred.
19 ICPSF	Ping Frame Response successful 0 Interrupt event has not occurred. 1 Interrupt event has occurred.
20 ICPRF	ICLC Ping Frame Request received 0 Interrupt event has not occurred. 1 Interrupt event has occurred.
21 ICTOF	ICLC frame for Test mode off received 0 Interrupt event has not occurred. 1 Interrupt event has occurred.
22 ICLPF	ICLC frame for Loopback On received 0 Interrupt event has not occurred. 1 Interrupt event has occurred.
23 ICCTF	ICLC frame for Clk Test mode received 0 Interrupt event has not occurred. 1 Interrupt event has occurred.

Table 1996. RIISR field descriptions (continued)

Field	Description
24 ICTDF	ICLC frame for LFAST Slaves Tx Interface Disable received 0 Interrupt event has not occurred. 1 Interrupt event has occurred.
25 ICTEF	ICLC frame for LFAST Slaves Tx Interface Enable received 0 Interrupt event has not occurred. 1 Interrupt event has occurred.
26 ICRFF	ICLC frame for LFAST Slaves Rx Interface fast mode switch received 0 Interrupt event has not occurred. 1 Interrupt event has occurred.
27 ICRSF	ICLC frame for LFAST Slaves Rx Interface slow mode switch received 0 Interrupt event has not occurred. 1 Interrupt event has occurred.
28 ICTFF	ICLC frame for LFAST Slaves Tx Interface fast mode switch received 0 Interrupt event has not occurred. 1 Interrupt event has occurred.
29 ICTSF	ICLC frame for LFAST Slaves Tx Interface slow mode switch received 0 Interrupt event has not occurred. 1 Interrupt event has occurred.
30 ICPOFF	ICLC frame for PLL OFF received 0 Interrupt event has not occurred. 1 Interrupt event has occurred.
31 ICPONF	ICLC frame for PLL ON received 0 Interrupt event has not occurred. 1 Interrupt event has occurred.

80.6.2.17 PLL and LVDS Status Register (PLLLSR)

Offset: 0x00A4 Access: User read

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	LRSLPS	LDSLPS	LDPDS	LRPDS
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1

Figure 2094. PLL and LVDS Status Register (PLLLSR)

Table 1997. PLLLSR field descriptions

Field	Description
28 LRSLPS	This bit indicates the real time status of the LR sleep signal. 0 LR sleep signal is negated. 1 LR sleep signal is asserted.
29 LDSLPS	This bit indicates the real time status of LD sleep signal. 0 LD power sleep signal is negated. 1 LD power sleep signal is asserted.
30 LDPDS	This bit indicates the real time status of LD power down signal. When asserted, LD is put in the power down state. 0 LD power down signal is negated. 1 LD power down signal is asserted.
31 LRPDS	This bit indicates the real time status of LR power down signal. When asserted, LR is put in the power down state. 0 LR power down signal is negated. 1 LR power down signal is asserted.

80.6.2.18 Unsolicited Rx Status Register (UNSRSR)

Offset: 0x00A8 Access: read/write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	URXDV	0	0	0	0	0	URPCNT		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 2095. Unsolicited Rx Status Register (UNSRSR)

Table 1998. UNSRSR field descriptions

Field	Description
23 URXDV	Unsolicited data valid Indicates a valid frame exists in the Unsolicited Data registers.
29:31 URPCNT	Rx Unsolicited payload Indicates the number of bytes of valid Rx unsolicited data payload present in the *Unsolicited Rx Data Registers (UNSRDRn)*. The bitfield coding is same as b7-b5 in *Table 1979*.

80.6.2.19 Unsolicited Rx Data Registers (UNSRDR*n*)

Offset: 0x00AC (UNSRDR8), 0x00B0 (UNSRDR7), 0x00B4 (UNSRDR6), 0x00B8 (UNSRDR5), 0x00BC (UNSRDR4), 0x00C0 (UNSRDR3), 0x00C4 (UNSRDR2), 0x00C8 (UNSRDR1), 0x00CC (UNSRDR0)

Access: User Read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	UNRXD															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	UNRXD															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 2096. Unsolicited Rx Data Registers (UNSRDR8–UNSRDR0)

Table 1999. UNSRDR8–UNSRDR0 field descriptions

Field	Description
0:31 UNRXD	Unsolicited Receive Data This represents 9 registers for Unsolicited received data.It is read only register. The first bit received as part of the payload will be stored at UNRXD8[31], second bit at UNRXD8[30], and so on. So the last bit will be stored at UNRXD0[0] in case of 288-bit payload.

80.6.3 Register safety classification requirements

This module is classified as NoSaMo (Non-Safety relevant Module) for safety requirements (Refer to the “Functional safety” chapter for register classification details. All registers of the LFAST module are classified as non-safety relevant.

The MCR[IPGDBG] fulfills the safety requirements for monitoring of the debug freeze signal.

80.7 Functional description

80.7.1 Startup procedure

The LFAST requires a sequence of operations before it can be used for communication. The procedure to enable LFAST for proper operation is listed below.

80.7.1.1 LFAST slave interface startup procedure

1. After POR the RCDCR are programmed according to the LVDS parameter in the device datasheet.
2. The PLLCR[30] is programmed with the configuration parameters of the PLL.
3. The LCR is programmed with the configuration parameters of the LVDS.
4. MCR[DATAEN] fields is programmed to select the LFAST mode.
5. Write MCR[TXARBD] = 0 to enable Tx of frames.
6. Write MCR[DRFEN] = 1 to enable LFAST.
7. Write MCR[RXEN] = 1 to enable LFAST Rx.
8. On the assertion of the LFAST interface enable the signals LR power down and LR disable are negated.
9. When an ICLC frame with payload 0x31 is received the hardware writes RIISR[ICTEF] = 1 and MCR[TXEN] = 1.
10. The ping frame is sent when an ICLC frame with payload 0x00 frame is received and the hardware writes RIISR[ICPRF] = 1.

Speed mode change:

11. The PLL starts when an ICLC frame with payload 0x02 is received and the hardware writes RIISR[ICPONF] = 1.
12. The speed of the Tx interface is changed when an ICLC frame with payload 0x80 is received and the hardware writes RIISR[ICTFF] = 1. This will cause SRC[TDR] = 1.
13. The speed of the Rx Interface is changed when an ICLC frame with payload 0x10 is received and the hardware writes RIISR[ICRFF] = 1. This will cause SCR[RDR] = 1.
14. The ping frame is sent when an ICLC frame with payload 0x00 is received and the hardware writes RIISR[ICPRF] = 1.

80.7.2 LFAST Interface enable signal (lfast_sysclk_en)

The lfast_sysclk_en signal comes from the DCI module. It is asserted by the DCI internal state machine when switching from JTAG to LFAST and vice-versa can safely be done. The detailed effects of lfast_sysclk_en on the module are:

LFAST interface enable (lfast_sysclk_en) negation:

1. If LCR[SWONLD] = 0 the LD LVDS pad is powered down.
2. If LCR[SWONLR] = 0 the LR LVDS pad is powered down.
3. The LFAST slave exits the test mode (for example, loopback mode or clock test mode). Then the software writes MCR[CLKTST] = 0 and MCR[LPON] = 0.
4. The LFAST slave Rx and Tx interfaces are set to Low Speed mode. Then write SCR[TDR] = 0 and SCR[RDR] = 0.
5. The Tx and Rx interface controllers are disabled.
6. The status registers are cleared.

LFAST interface enable (lfast_sysclk_en) assertion:

30. The PLLCR is shared between the LFAST – Interprocessor Communications and the LFAST – High Speed Debug modules. The programming values for the PLLCR in the LFAST – Interprocessor Communications are also used by the LFAST – High Speed Debug module.

1. The LD LVDS pad is power up, if all conditions are met:
   – LCR[SWOFFLD] = 0.
   – LCR[SWONLD] = 1.
   – MCR[TXEN] = 1.
   – When frame ICR[ICLCPLD] = 0x31 is received.
2. LR LVDS pad is powered up if all conditions are met:
   – LCR[SWOFFLR] = 0.
   – LCR[SWONLR] = 1.
   – MCR[RXEN] = 1.

80.7.3 Line Receiver (LR)

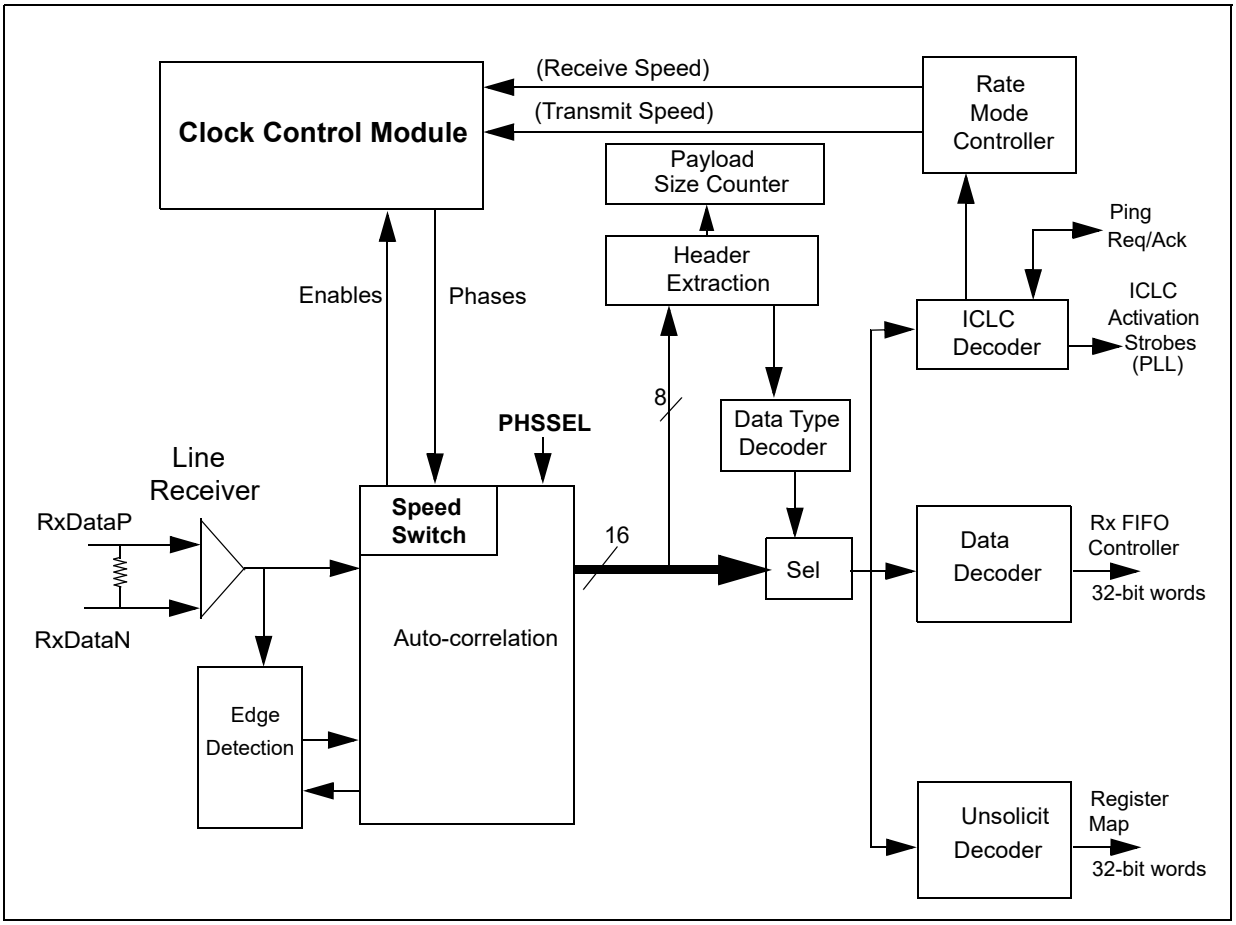
80.7.3.1 Introduction

The LR detects the voltage swing on the differential pair and converts it to a CMOS logic level that feeds the adaptive auto-correlation block. The received data is sampled using the best of the 8 (default) or 4 (alternative setting) possible sampling edges from the high speed clock or 4 phases using the low speed clock. The sampling edge is chosen by checking which of the 8/4 Correlators provide the maximum correlation. If more than one sampling edge provides the maximum then the state machine will choose the sampling edge based on a defined selection algorithm.

After the correct sampling edge is chosen, the remaining unused sampling edges are turned off. Once the header is received the length of the frame (payload size) and logical channel definition can be obtained. The logical channel definition will determine the data type of the payload and therefore will determine the destination for the payload. Decoding of the ICLC commands from the payload is required in order to extract the interface control, rate mode and PLL commands.

Figure 2097 shows the block diagram of Uplink controller.

Figure 2097. Top level receive controller



80.7.3.2 Edge-detection and auto-correlation

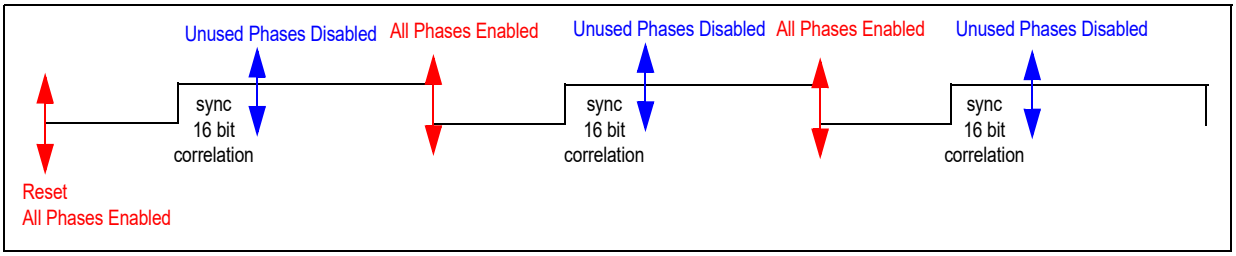
The edge-detection and auto-correlation are described together, as the mode setting of the auto-correlation block impacts largely on whether the edge-detection circuitry is used or not. The edge-detection will be performed first if required before auto-correlation.

80.7.3.2.1 Auto-correlation modes

80.7.3.2.1.1 Hunt correlation mode

On reset the Receive Controller will always come up in Hunt Correlation mode. In this mode all the Correlators are enabled and the Receive Controller is always hunting for the synchronization pattern. The phase enables (either 8 or 4) to the external clock control module are always high. There is no edge detection for the first bit of the synchronization pattern. Hunt Correlation mode is considered the safest mode as the Receive Controller is always checking for the synchronization pattern, but it is the mode that consumes the most power. In Hunt Correlation mode the correlation is performed over all 16 bits of the synchronization pattern.

Figure 2098. Hunt correlation mode



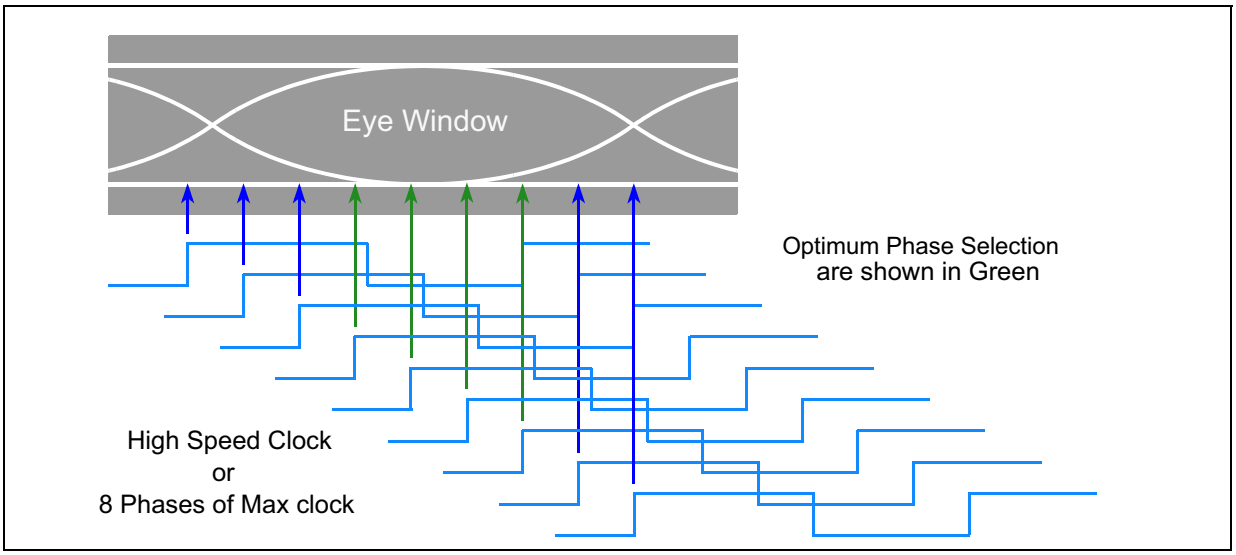
80.7.3.2.2 Edge detection

The edge detector is disabled in Hunt Correlation mode.

80.7.3.2.3 Auto-correlation

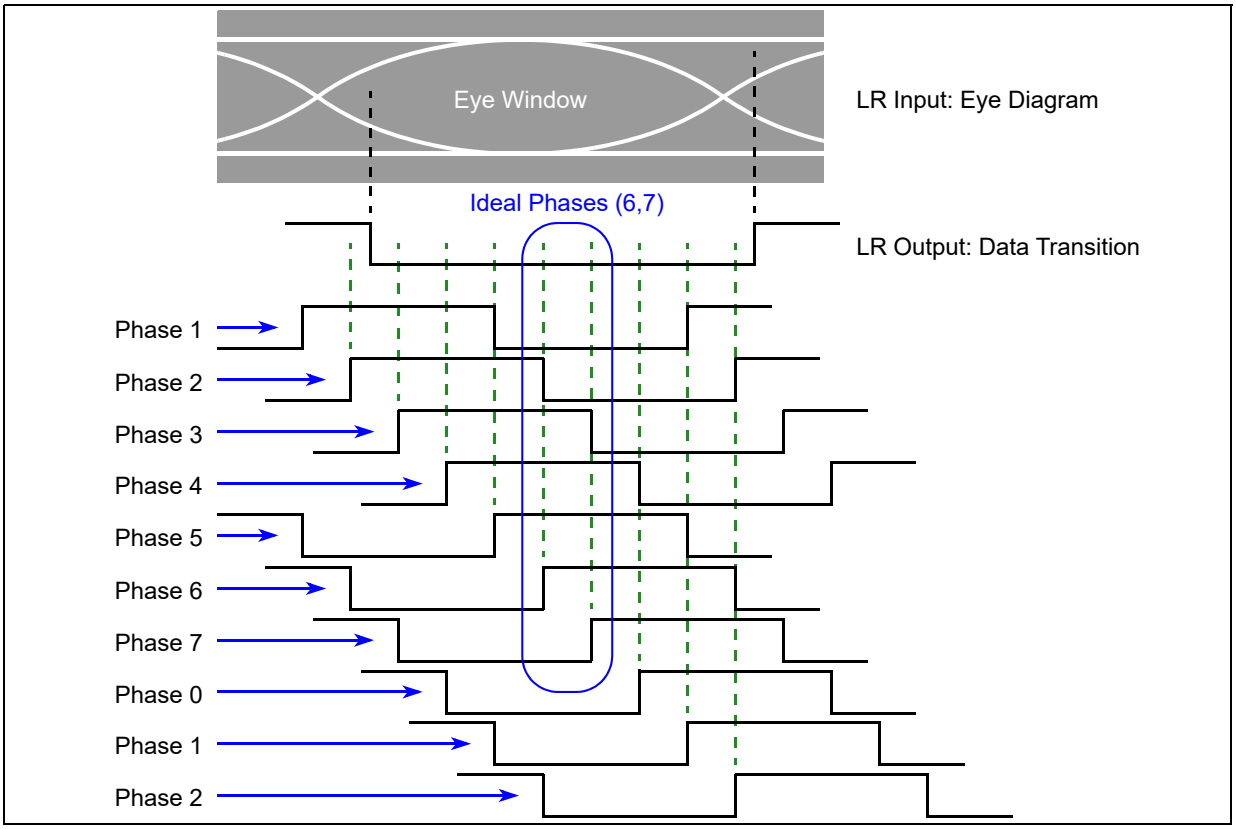
The data transmission between the 2 devices LD and LR is asynchronous in nature. Hence the Receive Controller does not have the knowledge about the correct clock phase to be used for extracting the data. The task of the auto-correlation (or synchronization) scheme is to estimate the best clock phase (8 or 4) for extraction of data as shown in *Figure 2099*.

Figure 2099. High speed 8 phase selection



The objective of the auto-correlation is to select the clock phase that occurs closest to the center of the eye diagram window. For 8 Phase clock alignment the worst case selection is when the clock edge is just after the LR output transition. The 8 clock phases will sample the correct value but the middle clock phases are the ideal selection. If one of the middle phases are selected then the minimum distance to the LR transition is 3 clock phases as shown in *Figure 2100*.

Figure 2100. High speed 8 phase clock alignment example



Each of the 8 high speed phases are 45 degrees separated. For 4 phases, whether high or low speed, the phases are 90 degrees separated but can have different phases enabled as shown in *Figure 2101* and *Figure 2102*.

Figure 2101. High speed 4 phase clock alignment example

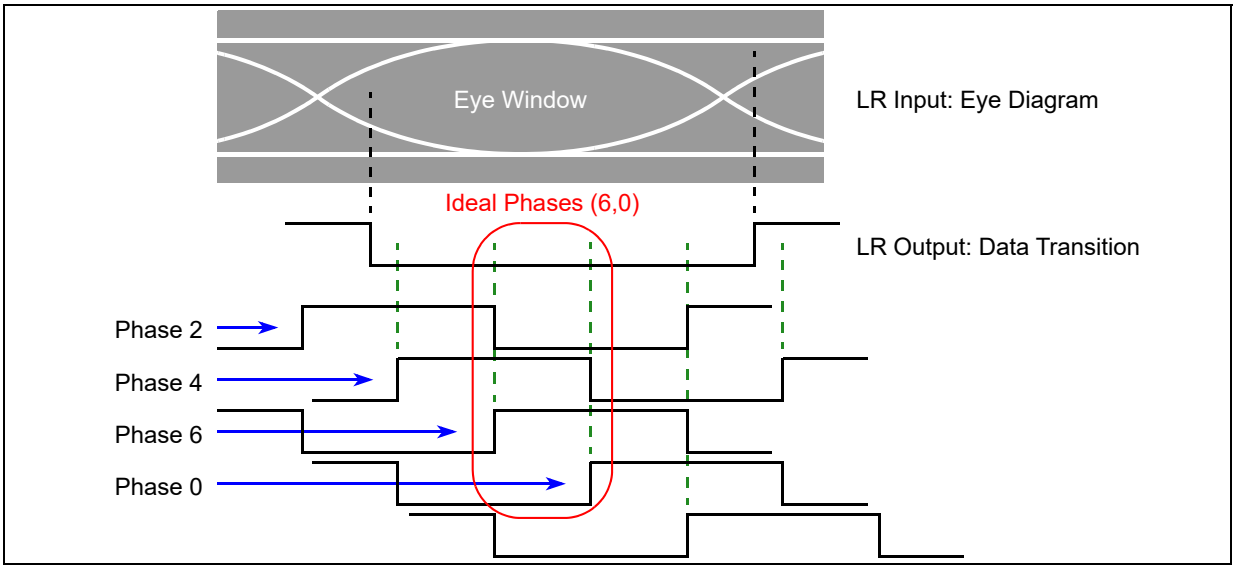
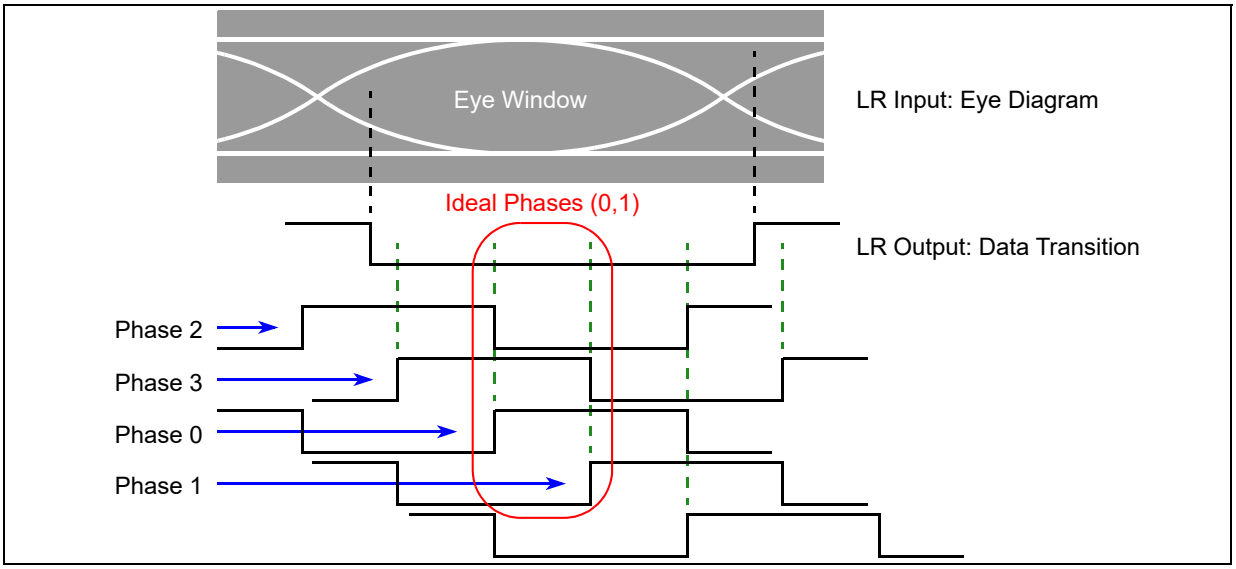


Figure 2102. Low speed 4 phase clock alignment example



The auto-correlation and selection of the correct clock phase starts after the edge detection finds a 0 to 1 transition. The high speed 8 or 4 phases from the PLL and the low speed 4 phases (generated inside the Clocking Module) are muxed inside the clocking module. The LFAST interface block will determine which phases are to be enabled and disabled. The only time the interface does not choose the phases is when the Clocking Module overrides the phase enables using COCR[SMPSEL].

The input data path can be sampled by different samplers depending on the configuration:

- High Speed 8 Phases: Samplers 0,1,2,3,4,5,6,7
- High Speed 4 Phases: Samplers 0,2,4,6
- Low Speed 4 Phases: Samplers 0,1,2,3

Each sampler block samples the data path with a different clock phase from the Clocking Module, and re-samples it to an intermediate phase as shown in *Table 2000*, *Table 2001*, and *Table 2002* before re-sampling it to Phase 0. The intermediate phase is used to make static timing between the initial phase and the final phase 0. The final phase, Phase 0 is chosen so all the clock trees after the sampler are clocked by the same clock.

Table 2000. High speed 8 phase selection - sampling procedure

Samplers	Initial Sample	Intermediate Sample	Final Sample
0	Phase 0	Phase 0	Phase 0
1	Phase 1	Phase 0	Phase 0
2	Phase 2	Phase 0	Phase 0
3	Phase 3	Phase 0	Phase 0
4	Phase 4	Phase 2	Phase 0
5	Phase 5	Phase 2	Phase 0
6	Phase 6	Phase 4	Phase 0
7	Phase 7	Phase 4	Phase 0

Table 2001. High speed 4 phase selection - sampling procedure

Samplers	Initial Sample Phase	Intermediate Sample Phase	Final Sample Phase
0	Phase 0	Phase 0	Phase 0
1	Disabled	Disabled	Disabled
2	Phase 2	Phase 0	Phase 0
3	Disabled	Disabled	Disabled
4	Phase 4	Phase 2	Phase 0
5	Disabled	Disabled	Disabled
6	Phase 6	Phase 4	Phase 0
7	Disabled	Disabled	Disabled

For High speed 4 Phase selection, See *Table 2001*, Samplers 1, 3, 5, 7 are disabled. In the Phase Select algorithm Phase 1 is mapped to Phase0, Phase 3 is mapped to Phase 2, Phase 5 is mapped to Phase 4, Phase 7 is mapped to Phase 6 so it simplifies the algorithm and allows the algorithm to be the same independent of 4 or 8 Phases in high speed.

Table 2002. Low speed 4 phase selection - sampling procedure

Samplers	Initial Sample Phase	Intermediate Sample Phase	Final Sample Phase
0	Phase 0	Phase 0	Phase 0
1	Phase 1	Phase 0	Phase 0
2	Phase 2	Phase 0	Phase 0
3	Phase 3	Phase 0	Phase 0
4	Disabled	Disabled	Disabled
5	Disabled	Disabled	Disabled
6	Disabled	Disabled	Disabled
7	Disabled	Disabled	Disabled

For low speed 4 Phase selection, See *Table 2002*, Samplers 4, 5, 6, 7 are disabled.The first four Samplers (0,1,2,3) of the low 4 phase speed selection algorithm are the same as the four samplers required for the high 8 phase speed. Therefore the same architecture can be used for both high speed and low speed with the other four data paths disabled for low speed.

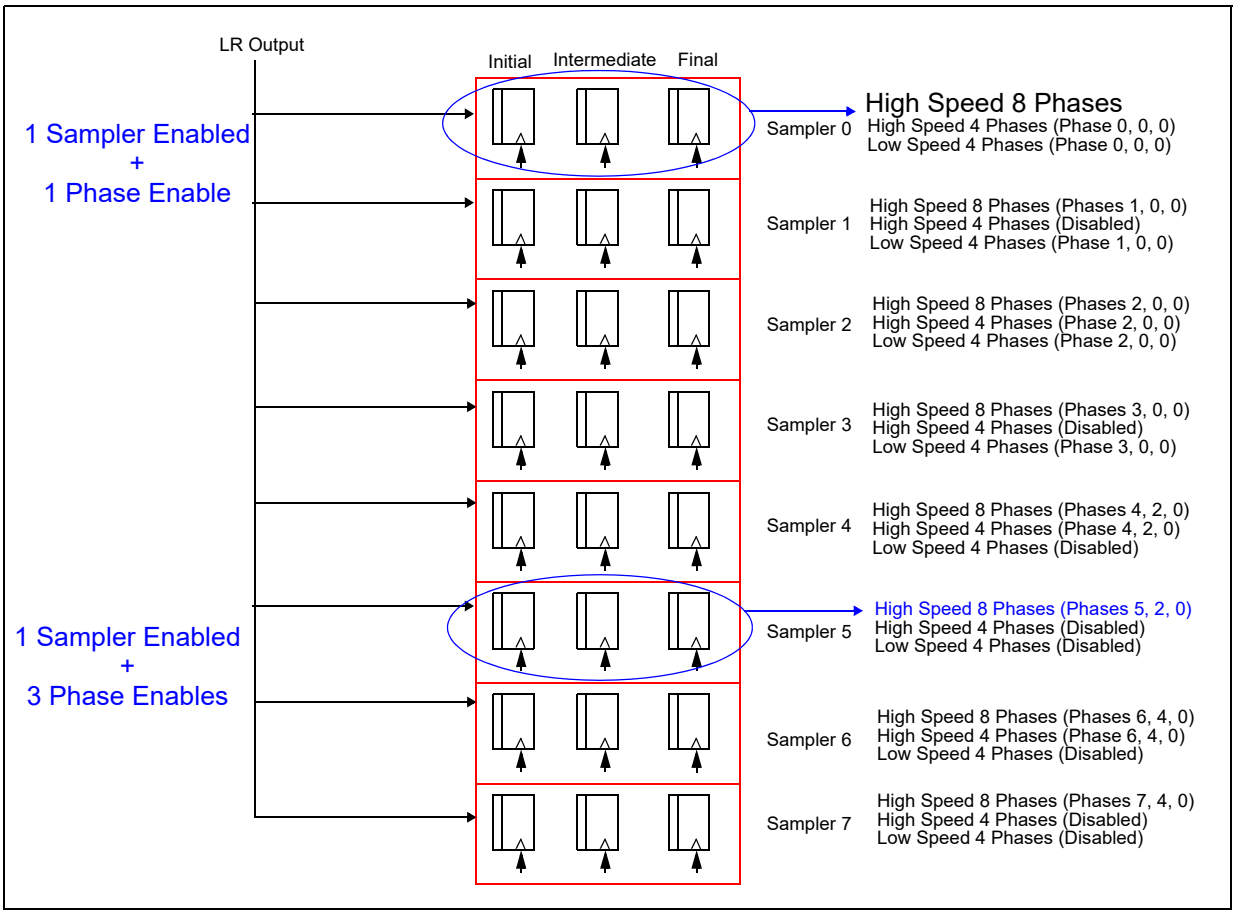
80.7.3.2.4 Sampler block and phase enable and disable

There are 8 data sampler paths in total inside the auto-correlation block, each data sampler has 3 sampling registers with the possibility of each register being clocked by a different phase (in example, Sampler 5 for high speed can have Phases 5, 2 and 0). Therefore after the correct data sampler has been selected for either high or low speed, the block can turn off all the sampler paths except for one, therefore 3 out of 24 registers will remain enabled while the others are disabled, as shown in *Figure 2103*.

After correlation and the correct data sampler has been selected all the other data samplers whose initial phase does not match the select phase are disabled. The selected sampler will then keep the required phases needed enabled, this can be max 3 phases (for instance Sampler 5 has Phases 5, 2, 0) or min 1 phase (in example, Sampler 0 has all Phase 0 content).

The end result is that the Rx Controller can disable the required number of unused sampler registers and disable any unnecessary phases from the Clocking Module by de-asserting the respective phase enables.

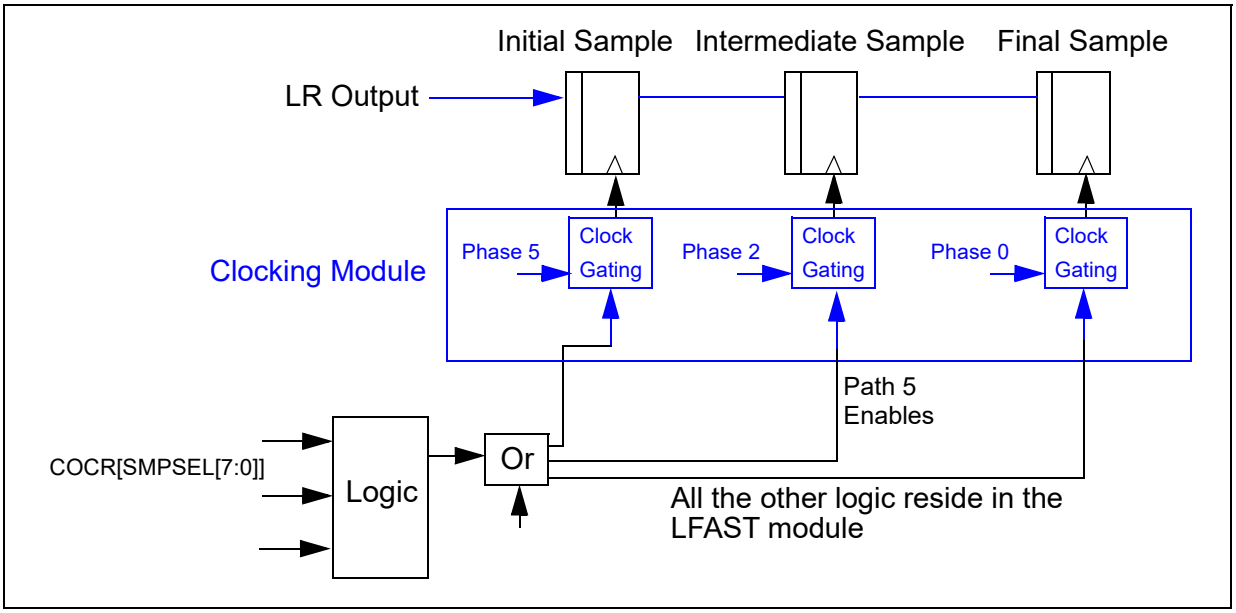
Figure 2103. Samplers, each sampler has 3 registers



In order to achieve the sampler and phase enable requirements each Sampler block will require the logic as shown in *Figure 2104*.

The Clock Gating Element resides inside the Clocking Module block. The interface will provide the enables to the Clocking Module and the Clocking Module will in return provide the individual clocks to the sampling registers.

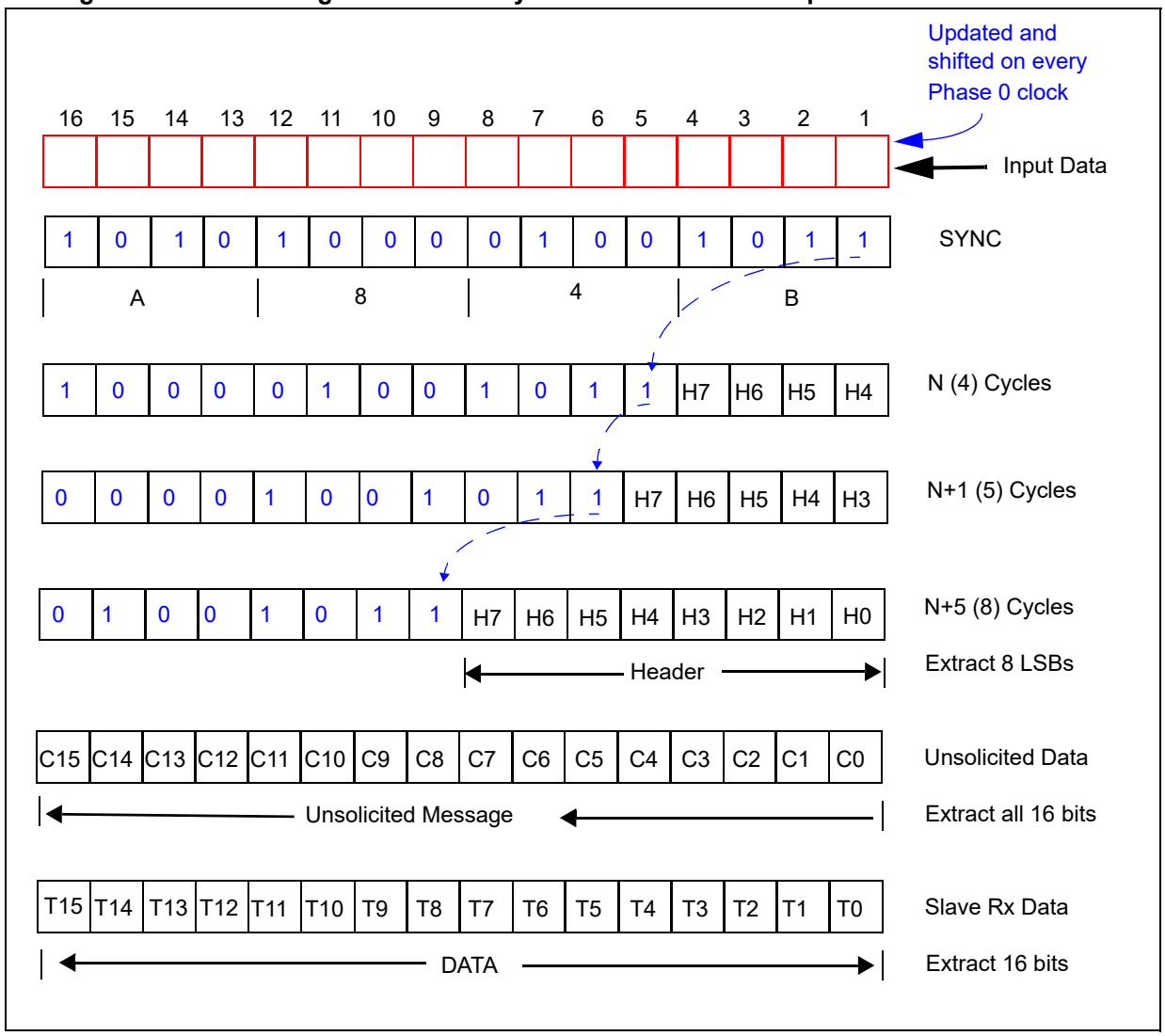
Figure 2104. Sampler 5 Logic



80.7.3.3 Header and Payload extraction

Once the Phases/Samplers are chosen after Synchronization and Correlation the next part of the flow is the header extraction. The Receive Controller will output a 16-bit word, bit shifted every Phase 0 clock as shown in *Figure 2105*.

Figure 2105. Extracting Header and Payload from the 16-bit output from Auto-correlation



80.7.3.3.1 Line Receiver states

The LD has the following states:

- Shutdown
  The LR enters in the shutdown state when the power down signal is high. In this state the LR regulator is supplying a Vdd level, however the LR is not enabled. In this state the LR holds its off value. Once power down signal is negated, a predefined settling time is required before the LR may be used for communication. During the settling time the peer LFAST device should not start its transfer.
- Sleep
  The LR enters in the Sleep state when the sleep signal is high. In the sleep state, the LR is enabled, but held in a power-saving state. Sleep mode may be used during inter-frame gaps that are long compared to the frame durations but not long enough to allow the interface(s) or high-speed clock generators to be powered down completely. In this

mode the receiver current consumption is reduced. Hysteresis in the LR ensures that it continues to present a 1 to the LFAST module.

- Normal
  In the Normal state, the LR is primed for reception. Receiver converts differential signals to a single ended logic signal.

Table 2003. Line Receiver state

LR State	LFAST Slave Triggers (at least one of the following)
Shutdown	– LFAST interface enable assertion/negation – Programing of LCR[SWOFFLR] or LCR{SWONLR] – Programing MCR[RXEN] and MCR[TXEN]
Sleep	– Programing of LCR[SWSLPLR] or LSC[SWWKLR]
Normal	Absence of above mentioned conditions

80.7.4 Transmit controller

80.7.4.1 Introduction

The Transmit Controller uses Rx information, status information and error control data and codes them into the appropriate frame structure for transmission to the LD. This includes the synchronization and header, in preparation for the LD, which transmits the frame to peer LFAST IC at either low or high speed. The Transmit Controller creates a frame structure that is converted to a serial format for the LD.

An arbitration block will determine which message has higher priority data, such as unsolicited, ICLC, ping. The data frames are extracted from the FIFO controller, the unsolicited message from the register block, the CTS messages from the Receive Controller and the ping response from the register block.

The Tx interface has two speed modes:

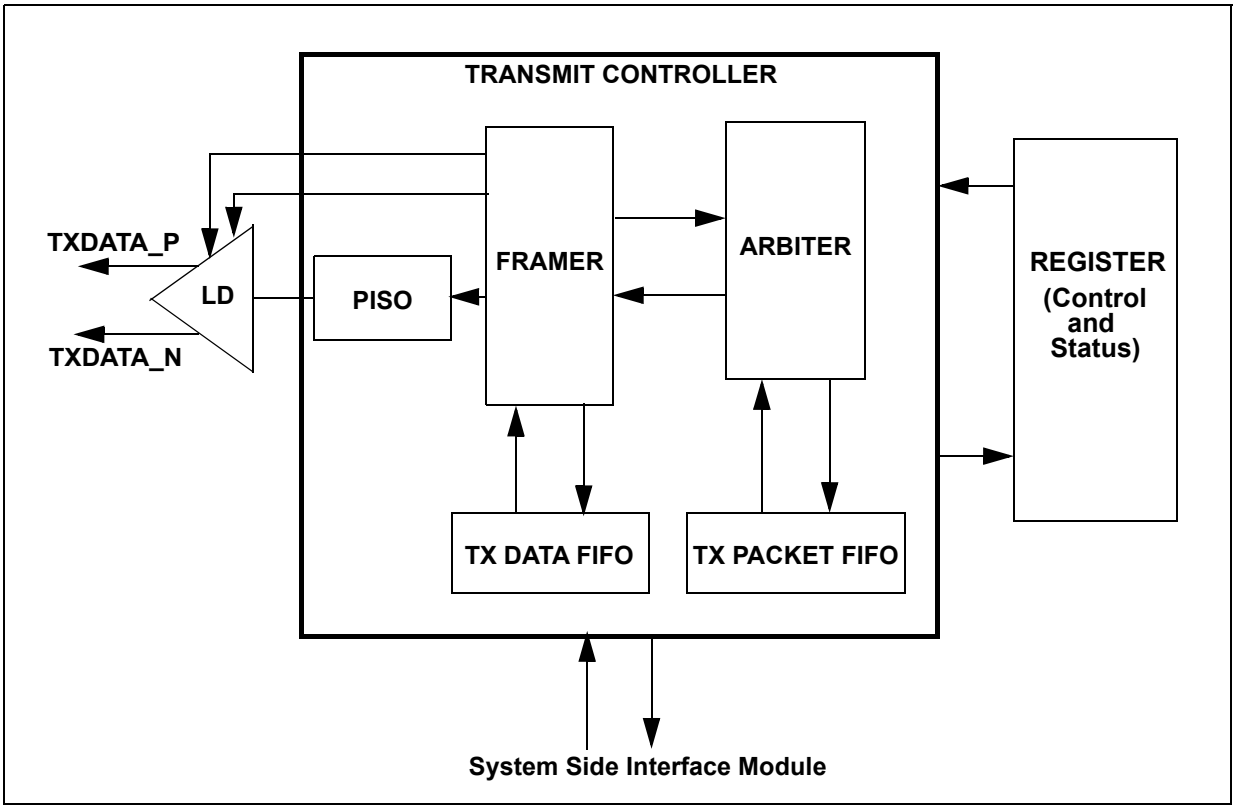
- Low Speed
- Fast Speed

The Transmit Controller consists of the following blocks as shown in *Figure 2106*.

- Arbitrator
- Framer
- Request Clock Control

The arbitrator will grant access to the framer from the request that has the highest priority. The framer block will extract the payload data from the granted request source and build the frame (adding synchronization or header if required). The frame is fed into a PISO (Parallel In Serial Out) and eventually sent to the LD.

Figure 2106. Transmit controller connections



80.7.4.2 Arbitration

The arbitration block prioritizes between requests from multiple sources like:

- Software programmable registers
- System side interface FIFO
- Rx interface controller
- LFAST Slave: LFAST interface enable (lfast_sysclk_en)

The level of priority for each request is shown in *Table 2004*.

Table 2004. Priority levels for the Transmit controller

Request	LFAST slave priority	Triggering conditions (at least one of the listed)
LFAST interface enable (lfast_sysclk_en)	1	– LFAST interface enable (lfast_sysclk_en) is asserted – LFAST interface enable (lfast_sysclk_en) is negated
Tx Interface disabled	2	– MCR[DRFEN] = 0 – MCR[TXEN] = 0 – MCR[DRFRST] = 1 – MCR[TXARBD] = 1 – LFAST Slave: ICLC frame with payload for “Disable Rx interface” received

Table 2004. Priority levels for the Transmit controller (continued)

Request	LFAST slave priority	Triggering conditions (at least one of the listed)
Tx Interface speed mode change	3	– SCR[TDR] is modified – LFAST Slave: ICLC frame with payload for changing Rx interface speed received
Loopback frame in Loopback mode	4	– TMCR[LPON] = 1 – LFAST Slave: ICLC frame with payload for loopback mode enable received **Note:** Valid only for following TMCR[LPMOD] settings: LPMOD[2:0] = 011b LPMOD[2:0] = 100b
Ping response request	5	– LFAST Slave: ICLC frame with payload for ping request received and PICR[PNGAUTO] = 1 – LFAST Slave: PICR[PNGREQ] = 1
Unsolicited frame request	6	– Valid only if Last Frame with header b0 = 1 received from the peer LFAST – UNSTCR[USNDRQ] = 1
Data frame from Tx FIFO	7	– Tx FIFO contains one or more frames – Last Frame with header b0 = 1 received from the peer LFAST device – MCR[DATAEN] = 1
Clock mode test	8	– TMCR[CLKTST] = 0 – LFAST Slave: ICLC frame with payload for clock test mode enable received

Once the arbitrator has granted access to a request, the Framer will commence building the frame. Any new requests for frame or data rate change will not be granted access until the framer has finished transmitting the current frame. If a data rate change request for the Tx interface is received while the Transmit controller is in the middle of sending a frame then the rate change request to the Clocking Module will be delayed. This allows for the frame to be completed before the change in speed mode. This prevents any speed mode change during the transmission of a frame. Once the speed mode request is sent to the Clocking Module by the Tx Interface Controller, it will then not allow any new requests to be processed for a specific time period defined by the bit field RCDCR[DRCNT].

Table 2005. Line driver states

LD state	LFAST slave triggers
Shutdown	– LFAST interface enable (lfast_sysclk_en) negation/assertion – Programing LVDS[SWOFFLD] and LVDS[SWONLD] – ICLC command from LFAST master – Programing MCR[TXEN] and MCR[DRFEN]
Sleep	– Programing LVDS[SWSLPLD] and LVDS[SWWKLD]
Normal	Absence of above mentioned conditions

80.7.5 Frames supported

Table 2006 provides the frames supported and their permissible payload sizes

Table 2006. Frames supported by LFAST interfaces

Frame Type	LCT Code	Supported by 1. LFAST Master Tx 2. LFAST Slave Rx	Supported by 1. LFAST Slave Tx 2. LFAST Master Rx	Payload Size (bits)
Data Frame	Rx – 0100b,1000b – 1011b Tx – All	YES	YES	TX: 96 RX:128
Unsolicited Frame	0001b	YES	YES	8, 32, 64, 96, 128, 256 and 288
ICLC Frame	0000b	YES	NO(1)	8
CTS Frame	n/a	NO	NO	n/a
Reserved	All others	—	—	—

1. Except the ICLC PING response frame.

80.7.6 Frame flow

The following sections describe Data Frame Flow, ICLC Flow and Rx Unsolicited Data Flow.

80.7.6.1 Data flow

LFAST supports the transfer of data between master and slave LFAST devices.

80.7.6.1.1 Data transmit

System Side Module is the initiator of all the Tx data frame to LFAST peer device. Data frame transmit is triggered whenever the Tx Data FIFO has at least one valid frame. The MCR[DATAEN], MCR[DRFEN] and MCR[TXEN] bits, should be set to enable the transfer of Tx data.

If MCR[DATAEN] = 0, then the valid frames in the in Tx data FIFO will be ignored for transmission. The Payload Size and channel type of each frame is specified by the System Side Module interface during transfer of the frame to the LFAST interface. The frame header is stored in the Tx packet FIFO and the payload in the Tx data FIFO.Whenever the Tx FIFO has at least one frame then a data transmit request is made to the Tx arbiter of the LFAST. Tx block arbitrates the data transmit request and schedules it depending on the priority of all the pending transmit requests. When data request is scheduled by Tx block, the required data is fetched from Tx data FIFO. The number of frames present in the Tx FIFO for transmission is indicated by the bitfield DFSR[TXFCNT].

80.7.6.1.1.1 Programing model for Tx data transmit

1. Program MCR[DATAEN] = 1, MCR[TXEN] = 1 and MCR[DRFEN] = 1 to enable the Tx path of LFAST device.
2. Select the desired clock rate at which data should be transmitted, using ICLC transfers and appropriate programing of SCR[TDR] bits.
3. Frame present in TX FIFO (Data and Packet FIFO) are sent.
4. TISR[TXDTF] = 1 after each frame transfer.

80.7.6.1.2 Data receive

LFAST master and slave supports reception of data frame. The received frame is determined to be of data frame type by decoding channel type field of the header present in the received frame. The MCR[DATAEN], MCR[RXEN] and MCR[DRFEN] bits should be set to enable the data frame reception.When MCR[DATAEN] = 0 the received data frames will be ignored and will not be placed in the Rx data FIFO.

The Rx data frames received by Rx block are stored in the Rx FIFO. Whenever the frame is received in the Rx FIFO the System Side Module is indicated by assertion of LFAST Rx FIFO ready signal. The frame size and the Channel type is passed to the LFAST. The frame boundaries are indicated by start of frame and end of frame signals. The number of unread frames in the Rx Data FIFO are indicated by DFSR[RXFCNT]. When Rx DATA FIFO is full and cannot accommodate current frame completely, then the remaining data of the Rx frame is discarded.

80.7.6.2 Unsolicited flow

80.7.6.2.1 Unsolicited frame transmit flow

The software is the initiator for unsolicited frames to the LFAST peer device. The unsolicited frame header and payload is programmed into the Unsolicited Data and Control registers. Once the Payload is programmed UNSTCR[USNDRQ] is set, generating a request for unsolicited frame transfer to the Tx arbiter.

80.7.6.2.1.1 Programing model for unsolicited frame transmit

1. Program MCR[TXEN] = 1 and MCR[DRFEN] = 1 in the *Mode Configuration Register (MCR)* to enable the Tx path.
2. Select the desired clock rate at which data should be transmitted, using ICLC transfers and appropriate programing of SCR[TDR].
3. Read the UNSTCR[USNDRQ].
   - If UNSTCR[USNDRQ] = 1 then wait for either of the following:
     - UNSTCR[USNDRQ] = 0
     - TISR[TXUNSF] = 1.
   - If UNSTCR[USNDRQ] = 0 then:
     - Program the unsolicited payload in UNSTDR0–UNSTDR8, and can be written up to a payload of frame of a maximum of 288 bits.
     - Program the unsolicited frame header in UNSTCR[UNSHDR].
     - Program UNSTCR[USNDRQ] = 1.
4. UNSTCR[USNDRQ] is cleared by the Tx Block after the frame transfer.
5. TISR[TXUNSF] = 1 when the frame is transmitted.

80.7.6.2.2 Unsolicited frame receive flow

Whenever an Unsolicited frame is received from the peer device, the following steps are performed:

If UNSRSR[URXDV] = 0 then:

1. The payload size field of the header is first saved into the UNSRSR[URPCNT].
2. The payload is stored in the UNSTDR0–UNSTDR8.
3. UNSRSR[URXDV] = 1 and the RISR[RXUNSF] = 1 indicating the successful reception of the frame.

If UNSRSR[URXDV] = 1 then:

1. The current unsolicited frame is ignored.
2. RISR[RXUOF] = 1.

Typical steps by the processor after RISR[RXUNSF] = 1 is as follows:

1. The processor reads UNSRSR to get the payload size of the frame.
2. It then reads the complete frame by reading UNSRDR8–UNSRDR0 for the payload size as received in the frame.

80.7.6.3 ICLC flow

The ICLC (Interface Control Logical Channel) is defined as a separate logical channel type, which is mainly meant for implementing the data rate change in the LFAST interface and initiating the test modes.

80.7.6.3.1 .ICLC data receive flow

When the bits 4 to 1 of a receive frame are 0000b it indicates that the payload is an ICLC. ICLC payloads are always 8 bits.

80.7.6.3.1.1 Ping request ICLC

The LFAST slaves ICLC decoder will decode the ping request and set RIISR[ICPRF] = 1. The software can also write to PICR[PNGREQ] to indicate to the Tx block that a ping response frame needs to be transmitted. The PICR[PNGAUTO] indicates whether Tx block can respond automatically to the request by the LFAST master ICLC Ping Request frame.The Tx block will arbitrate the ping response frame request and send a ping frame with ping data defined by bitfield PICR[PNGPLYD]. Once the ping response frame has been sent the Tx block will set TISR[TXPNGF] and clear PICR[PNGREQ], if set.

80.7.6.3.1.2 LFAST Slaves RxData Interface Slow/Fast ICLC

The LFAST Slaves Rx Interface has two speed modes supported: slow (Low speed) and fast (High speed). The ICLC command will write RIISR[ICRSF] = 1 (Low speed) or RIISR[ICRFF] = 1 (High speed).

80.7.6.3.1.3 LFAST Slaves TxData Interface Slow/Fast ICLC

The LFAST slaves Tx Interface Controller has two speed modes: slow (Low speed), and fast (High speed). The ICLC command will write RIISR[ICTSF] = 1 (Low speed) or RIISR[ICTFF] = 1 (High speed).

80.7.6.3.1.4 Enable/Disable LFAST Slaves TxData Interface ICLC

The ICLC commands, Enable RxData Interface and Disable RxData Interface are decoded and the appropriate enable/disable signal sent to the Tx block Controller. These ICLC command writes RIISR[ICTEF] = 1 and RIISR[ICTDF] = 1. The ICLC Tx enable command will write MCR[TXEN] = 1.

No frames can be sent on the LFAST Slave Tx interface until the either LFAST master has enabled it via an ICLC frame or software programs MCR[TXEN] = 1.

80.7.6.3.1.5 Clock Test mode ICLC

This ICLC command is decoded, and then is used to write TMCR[CLKTST] = 1. This indicates to the Tx interface to output an alternating pattern of 1 and 0 at the currently configured clock rate. The Rx interface generates RIISR[ICCTF] = 1 on reception of this ICLC command.

The exit from this mode happens on reception of Test mode off ICLC command.

80.7.6.3.1.6 Loopback payload on ICLC

This ICLC command is decoded and TMCR[LPON] = 1 to indicate to the Tx Block that the received payload is to be sent back. The exit from this mode happens on reception of Test mode off ICLC command. The Rx interface causes RIISR[ICLPF] = 1 on reception of this ICLC command.

80.7.6.3.1.7 Test mode off ICLC

This ICLC command is decoded and TMCR[LPON] and TMCR[CLKTST] are cleared to indicate to the Tx block to exit from loopback mode or clock test mode.

Another option is for the payload loopback option to remain enabled until negated on the toggling of LFAST interface enable (lfast_sysclk_en).

80.7.7 Test and debug support

The test and debug interface helps to debug the LFAST module. These signals can be brought out for ease of validation.

80.7.7.1 Loopback test mode

The loopback function allows to verify the correct operation of the physical interface and the basic checks for the LFAST module without a peer device.

There are certain prerequisites before entering the Loopback mode:

- LD and LR should be turned on.
- Tx and Rx mode should be enabled.
- Interrupts needed for software should be enabled.
- Both Tx and Rx interface should be in same speed mode.
- For Automatic Test mode TMCR[LPFRMTH] should be programmed.

The LFAST module supports four loopback modes, defined by TMCR[LPMOD].

Table 2007. Loopback modes

LPMOD[2:0]	Mode selected
000	Rx Loopback (default)
001	Rx LVDS Loopback
010	Tx Loopback without Automatic frame generation
011	Tx Loopback with Automatic frame generation
100	Tx LVDS Loopback (external) with Automatic frame generation

The TMCR[LPON] bit controls the state of the loopback function, and the default setting is 0 (off). This can be written by the software, or in the case of the LFAST slave, the TMCR[LPON] bit can be modified by the Rx interface controller on decoding of a valid ICLC loopback on or Test mode off commands. The LPMOD should not be changed when the LPON bit is set.

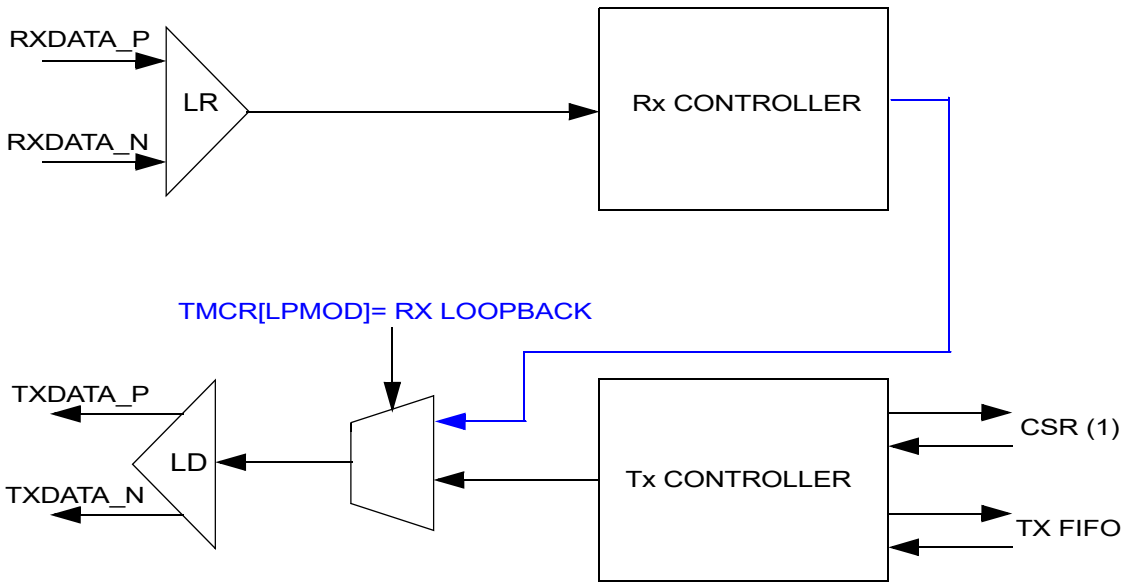
In all loopback modes, except Tx loopback without automatic frame generation, the decoding of frame and error flags is stopped. Only decoding of following is done:

- ICLC Turn Test mode Off frame.
- CTS bit of all frames.
- Automatic Loopback frame decoding (only in Automatic frame generation mode).

80.7.7.1.1 Rx Loopback mode

For Rx loopback mode the output of the LR is passed to the LD with manipulation via the Rx and Tx Interface controllers. Bit b0 of the header is guaranteed to be asserted when the incoming data from the other device is looped back. In Rx Loopback mode the Rx Interface controller operates in normal mode, decoding the frames (header, payloads). This allows the LFAST Master to control the Loopback mode on and off by sending ICLC commands.

Figure 2107. Rx Loopback mode



1. CSR stands for Control and Status Registers (used to configure and indicate the status of the Tx/Rx controller).

Entry to Rx Loopback mode:

1. Software programs TMCR[LPMOD] = 000b.
2. Loopback can be turned on by either of the following methods:
   - Software programs TMCR[LPON] = 1.
   - For Slave Only: - Reception of ICLC Frame with Payload 0xFF (Loopback mode on), from LFAST Master

Exit from Rx Loopback mode can be done by any of the following methods:

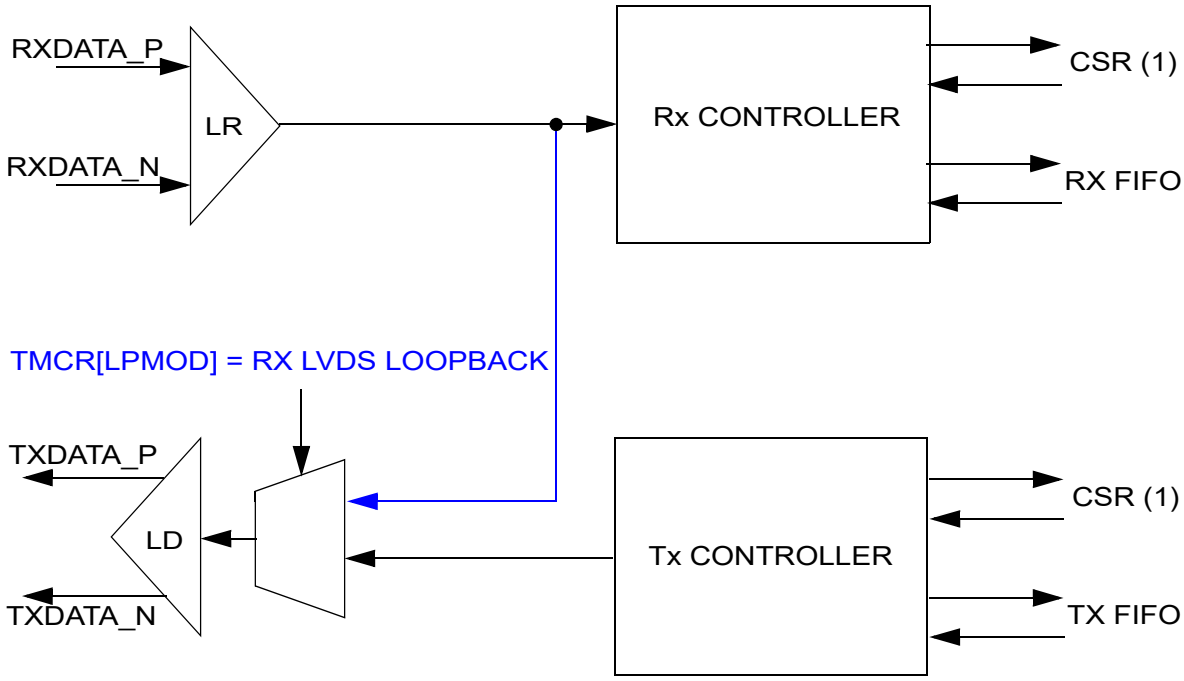
- Software programs TMCR[LPON] = 0
- For LFAST Slave: - Transmission of ICLC Frame with payload 0x38 (Test mode off), from LFAST Master
- For LFAST Slave: - Deassertion of the LFAST interface enable (lfast_sysclk_en).

The peer LFAST device is required to maintain at least 2-bit IFG between two Loopback frames in this mode.

80.7.7.1.2 Rx LVDS Loopback mode

This loopback mode is provided to verify and characterize the LVDS pads. In this loopback mode the data received by LFAST on Rx LVDS input is loopback to Tx LVDS output, bypassing LFAST. For LVDS loopback the output of the LR is passed to the LD via “Rx LVDS LoopBack Mux”. Bit 0 of the header cannot be guaranteed to be asserted when the incoming data from the other device is looped back. In this loopback, the Rx Interface Controller operates in normal mode, decoding the frames (header, payloads) but the Tx Interface Controller is ignored.

Figure 2108. Rx LVDS LoopBack mode



1. CSR stands for Control and Status Registers (used to configure and indicate the status of the Tx/Rx controller).

Entry to Rx LVDS Loopback mode:

1. Software programs TMCR[LPMOD] = 001b.
2. Loopback can be turned ON by either of the following methods:
   - The software writes TMCR[LPON] = 1.
   - For Slave Only: - Reception of ICLC Frame with Payload = 0xFF (Loopback mode ON), from LFAST Master.

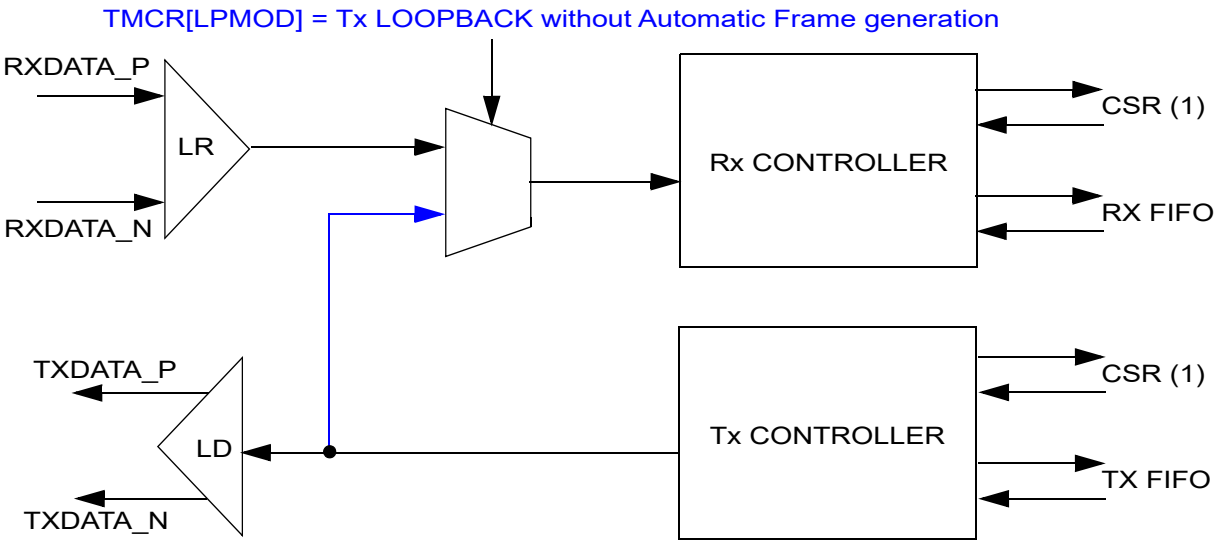
Exit from Rx LVDS Loopback mode can be done by any of the following methods:

- Software programs TMCR[LPON] = 0.
- For LFAST Slave:
  - Transmission of ICLC frame with payload 0x38 (Test mode off), from LFAST Master.
  - Deassertion of the LFAST interface enable (lfast_sysclk_en).

80.7.7.1.3 Tx loopback mode without automatic frame generation

This loopback mode is provided to verify the LFAST functionality, if LVDS pads are not functional. In this loopback mode the data transmitted by LFAST on Tx LVDS output is loopbacked internally on Rx LVDS input, bypassing LVDS pads.

Figure 2109. Tx LoopBack mode without automatic frame generation



1. CSR stands for Control and Status Registers (used to configure and indicate the status of the Tx/Rx controller).

Entry to Tx Loopback without Automatic frame generation mode:

1. Software programs TMCR[LPMOD] = 010b.
2. Loopback can be turned on by either of the following methods:
   - Software programs TMCR[LPON] = 1.
   - For Slave Only: Reception of ICLC frame with payload 0xFF (Loopback mode on), from LFAST Master

Exit from Tx Loopback without Automatic frame generation mode can be done by any of the following methods:

- Software programs TMCR[LPON] = 0.
- For LFAST Slave: Reception of ICLC frame with payload 0x38 (Test mode off), from Tx interface (using unsolicited frame).
- For LFAST Slave: Deassertion of the LFAST interface enable (lfast_sysclk_en).

All frames and error flags are decoded in this mode.

80.7.7.1.4 Automatic frame generation

The LFAST module supports two Loopback modes where predefined frames are generated automatically by the Tx Interface and Rx Interface indicates its successful reception by dedicated signals and status registers. These modes are defined for BIST like check for the LFAST, with minimal software intervention.

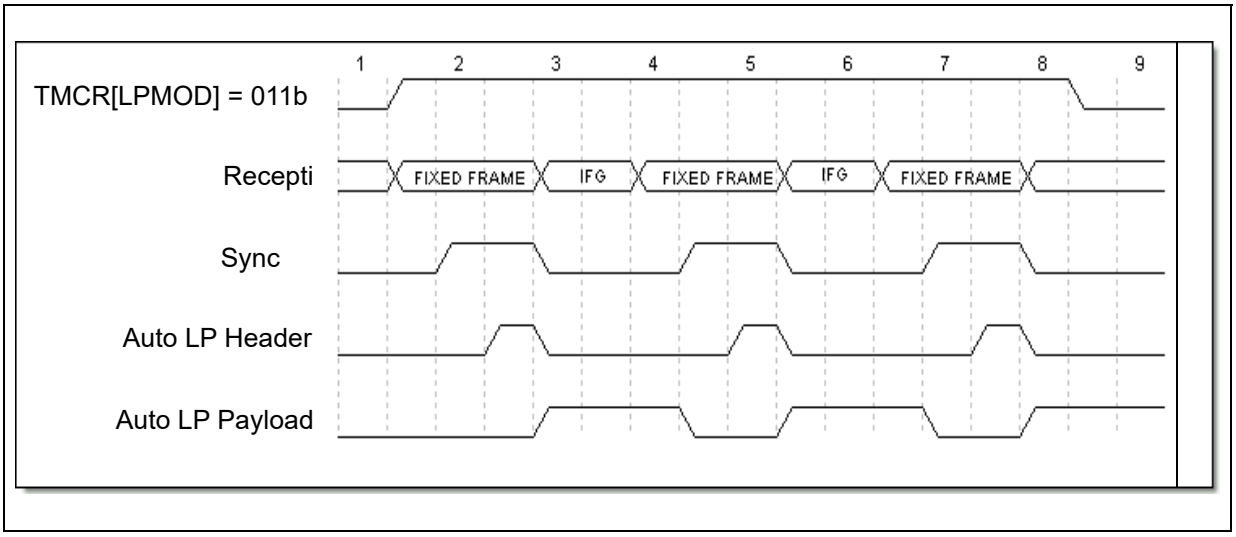
The Control register used for these modes are:

- ALCR[LPCNTEN] defines whether fixed number of auto loopback frames to be transmitted.
- ALCR[LPFMCNT] defines the number of loopback frames to be transmitted if ALCR[LPCNTEN] = 1.

The status signals and register used for these modes are:

- GSR[LPCSDV]: Valid Synchronization received.
- GSR[LPCHDV]: Frame with Header of 0x13 received.
- GSR[LPCPDV]: Frame with Payload of 0xCB received.
- GSR[LPTXDN]: Number of Auto Loopback frame transmitted with valid Synchronization, Header (13h) and Payload (CBh) is equal to ALCR[LPFMCNT].

Figure 2110. Automatic Loopback test status signal timings



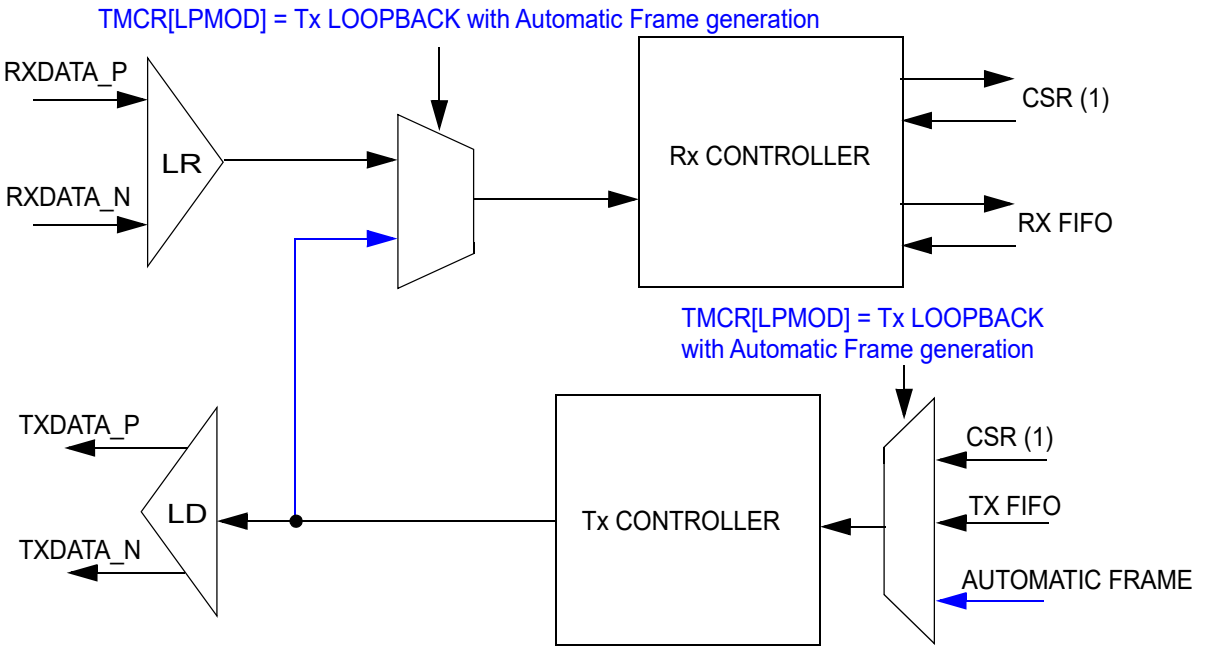
The two Loopback with automatic frame generation modes are:

- Tx Loopback with Automatic frame generation.
- Tx LVDS (external) with Automatic frame generation.

80.7.7.1.4.1 Tx LoopBack mode with automatic frame generation

When TMCR[LPMOD] = 011b (Tx Loopback mode with Automatic frame generation) the frames are generated by the Tx Interface. This mode helps to validate the Tx and Rx Path with minimal intervention from the software. The frames generated have fixed header of 0x13 and payload of 0xCB. The successful reception of this frame is indicated by the status signals and registers.

Figure 2111. Tx LoopBack mode with automatic frame generation



1. CSR stands for Control and Status Registers (used to configure and indicate the status of the Tx/Rx controller).

Entry to Tx Loopback with automatic frame generation mode:

1. Software programs TMCR[LPMOD] = 011b.
2. Loopback can be turned on by either of the following methods:
   – Software programs TMCR[LPON] = 1.
   – For Slave Only: Reception of ICLC frame with payload 0xFF (Loopback mode on), from the LFAST Master.

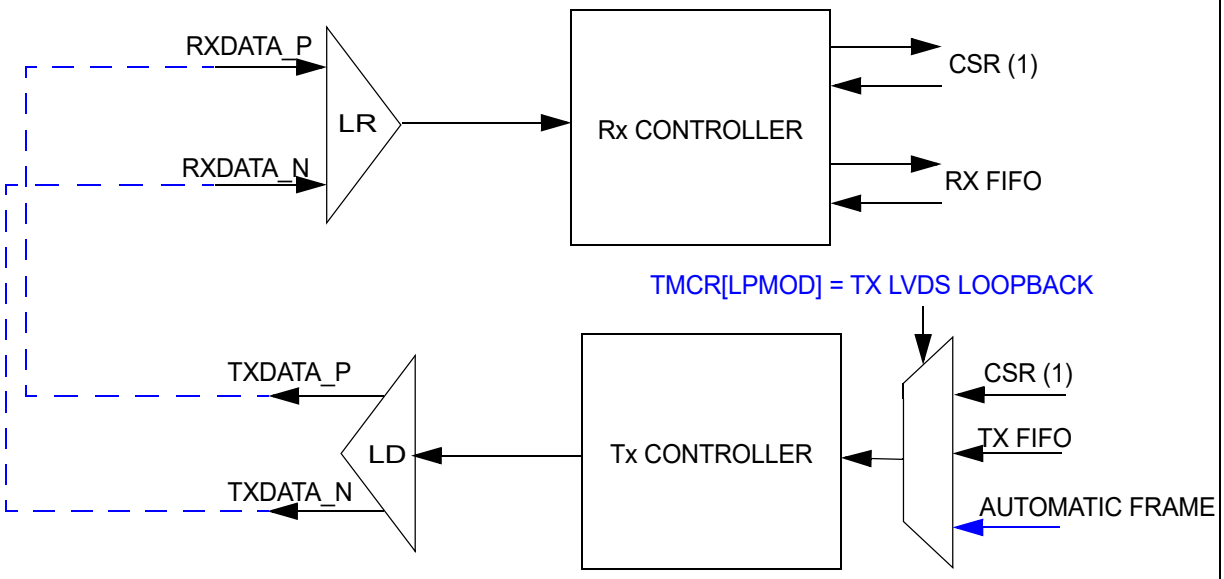
Exit from Tx Loopback with Automatic frame generation mode can be done by any of the following methods:

- Software programs TMCR[LPON] = 0.
- For LFAST Slave: Deassertion of the LFAST interface enable (lfast_sysclk_en).

80.7.7.1.4.2 Tx LVDS loopback (external) mode with automatic frame generation

In this mode the LD/Tx Controller is used to test the LR/Rx Controller. The idea is to use a test frame (predefined) from the Tx Controller out through the LD, loopback completed on the DUT-board (LD connected to LR via a transmission line), back into the LR, synchronized and correlated in the Rx Controller and compared to what was sent in the Downlink controller.

Figure 2112. Tx LVDS Loopback (external) mode with automatic frame generation



1. CSR stands for Control and Status Registers (used to configure and indicate the status of the Tx/Rx controller).

Entry to Tx LVDS loopback with automatic frame generation mode:

1. Software programs TMCR[LPMOD] = 100b.
2. Loopback can be turned on by either of the following methods:
   – Software programs TMCR[LPON] = 1.
   – For Slave Only: Reception of ICLC frame with payload 0xFF (Loopback mode on) from LFAST Master

Exit from Tx LVDS Loopback with automatic frame generation mode can be done by any of the following methods:

- Software programs TMCR[LPON] = 0.
- For LFAST Slave: Deassertion of the LFAST interface enable (lfast_sysclk_en).

80.7.7.2 Clock test mode

The bit TMCR[CLKTST] enables or disables the Clock Test mode of the LFAST module. In this mode the LFAST sends fixed pattern out on the LD at the current configured RxData clock rate. It is a RWM bitfield and the default setting is 0 (off). This bit can be set under one of the following conditions:

- Software programs TMCR[CLKTST].
- For Slave Only: Reception of ICLC frame with payload 0x34 (Clock Test mode on) from LFAST Master.

The Tx Controller will send out a pattern of alternating 1 and 0 (pattern 101010...). This provides a divide by 2 test clock of the current Tx clock.

The Clock Test mode can be canceled by any of the following methods:

- Software programs TMCR[CLKTST] = 0.
- For LFAST Slave: Reception of ICLC frame with payload 0x38 (Test mode off) from LFAST Master.
- For LFAST Slave: Deassertion of the LFAST interface enable (lfast_sysclk_en).

In clock test mode the Tx Controller does not output any synchronization pattern or header, just a pattern of alternating 1's and 0's.

This mode does not affect the functionality of the Rx Interface.

80.8 Packet memory

The LFAST stores packet frames for transmission, and reads packet frames after reception. The transmitter has its own dedicated buffer and the receiver has it own dedicated buffer, and they are not shared between each other.

Table 2008. Frames supported by LFAST interfaces

Frame Type	Tx Buffer (in bits)	Rx Buffer (in bits)	Memory Type
Data Frame	6 × 32 max 2 packets	8 × 32 max 2 packets	FIFO
Unsolicited Frame	9 × 32 max 1 packet	9 × 32 max 1 packet	Registers UNSTD[8-0], UNSRDR[8-0]
ICLC Frame	N/A	1 × 8 max 1 packet(1)	Registers PICR[PNGPYLD]
CTS Frame	N/A	N/A	N/A

1. Ping response packet

80.9 Resets

The various blocks in LFAST are reset as described in *Table 2009*.

Table 2009. Recommended receive exception handling mechanism

Reset	Blocks reset
Asynchronous Hardware reset	Polarity: Active Low – Clock control module (CCM) – LFAST Domain Logic (Rx and Tx block) – System Side Module Interface FIFOs – LFAST Register Space
DRFRST bit of *Mode Configuration Register (MCR)*	Polarity: Active High – Clock control module (CCM) – LFAST Domain Logic (Rx and Tx block) – System Side Module Interface FIFOs – LFAST Register Space

Table 2009. Recommended receive exception handling mechanism (continued)

Reset	Blocks reset
DRFEN bit of *Mode Configuration Register (MCR)*	Polarity: Active Low – Clock control module (CCM) – LFAST Domain Logic (Rx and Tx block) – System Side Module Interface FIFOs – LFAST Status Registers – SCR[RDR] and SCR[TDR] – TMCR[CLKTST] and TMCR[LPON] – ICR[ICLCSEQ] and ICR[SNDICLC] – PICR[PNGREQ] – UNSTCR[USNDRQ]
LFAST interface enable (lfast_sysclk_en)	Polarity: Active Low – Clock control module (CCM) – LFAST Domain Logic (Rx and Tx block) – System Side Module Interface FIFOs – LFAST Status Registers except the following: TISR, RISR, RIISR, GSR[LPTXDN] and GSR[LPFPDV] These bits remain reset while the LFAST interface enable (lfast_sysclk_en) is Low.
	Polarity: Negedge only – SCR[RDR] and SCR[TDR] – TMCR[CLKTST] and TMCR[LPON] – ICR[ICLCSEQ] and ICR[SNDICLC] – PICR[PNGREQ] – UNSTCR[USNDRQ] These bits do not remain reset while the LFAST interface enable (lfast_sysclk_en) is Low.
	Polarity: positive edge only – TISR, – RISR, – RIISR, – GSR[LPTXDN] and GSR[LPFPDV] These bits do not remain reset while the LFAST interface enable (lfast_sysclk_en) is High.

80.10 Clocks

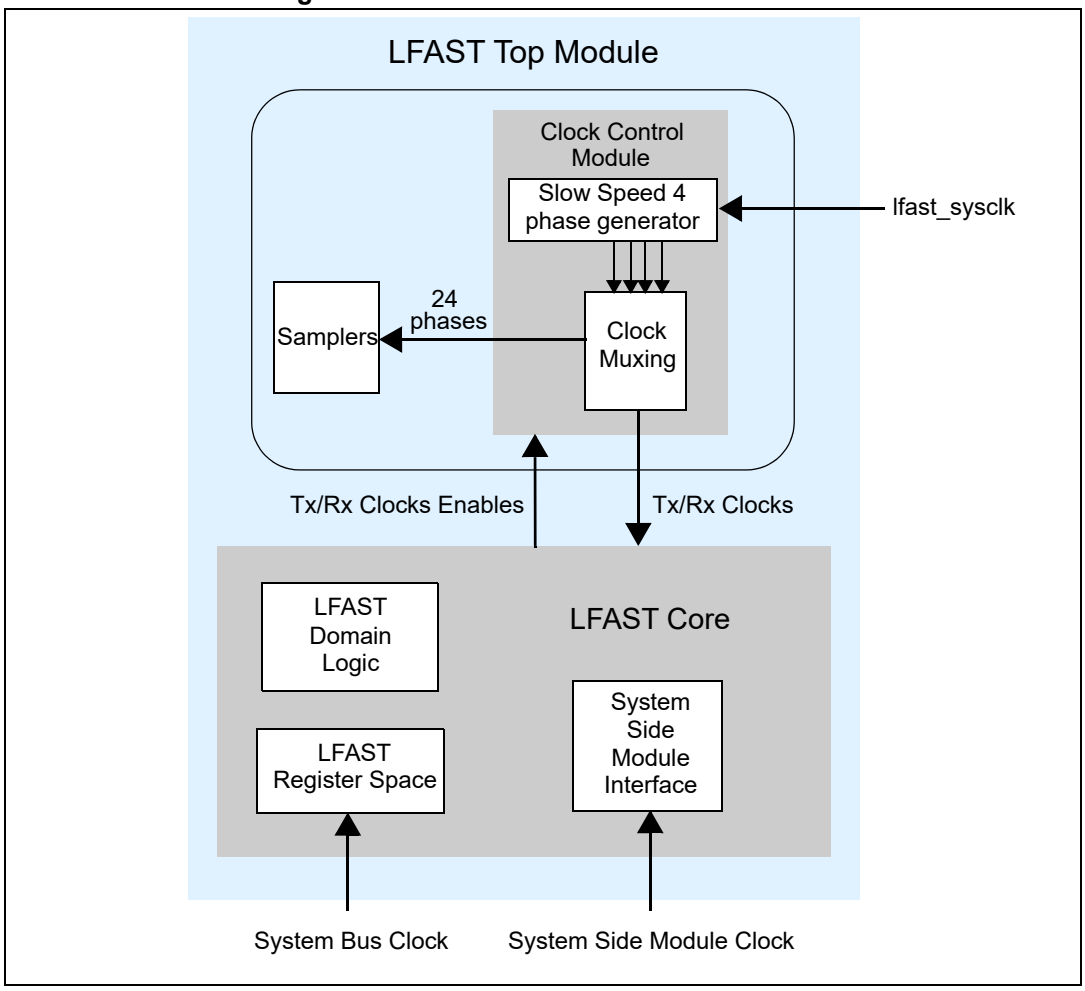
The LFAST mainly works on two clocks:

- System Bus Clock, used to program the registers.
- System Side Module clock, which is synchronous to the System Bus clock.

80.10.1 Clocking strategy

Figure 2113 shows the clock domain in which each module functions. The Clock Module will generate four phases of slow speed clock using both the edges of lfast_sysclk, muxes high speed and low speed clocks and provides a muxed clock to the Sampler Module.

Figure 2113. LFAST module clock domains

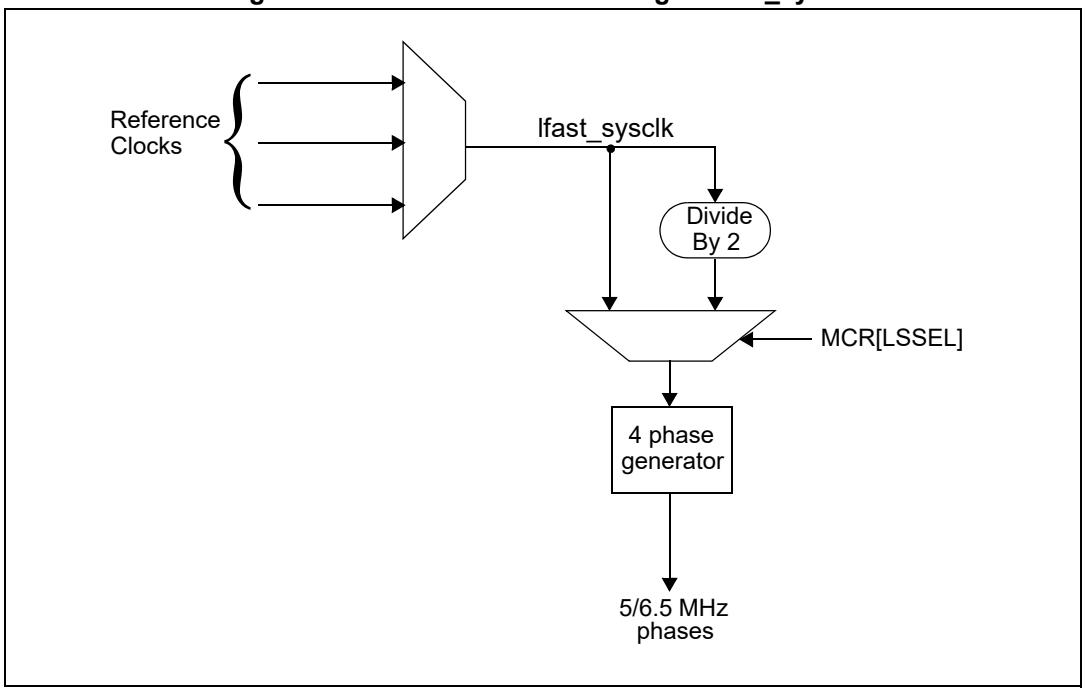


80.10.2 Slow speed clock

80.10.2.1 External muxing

The Clock Control Module will receive lfast_sysclk from the clocking subsystem as shown in *Figure 2114*, *Figure 2115* and *Figure 2116*.

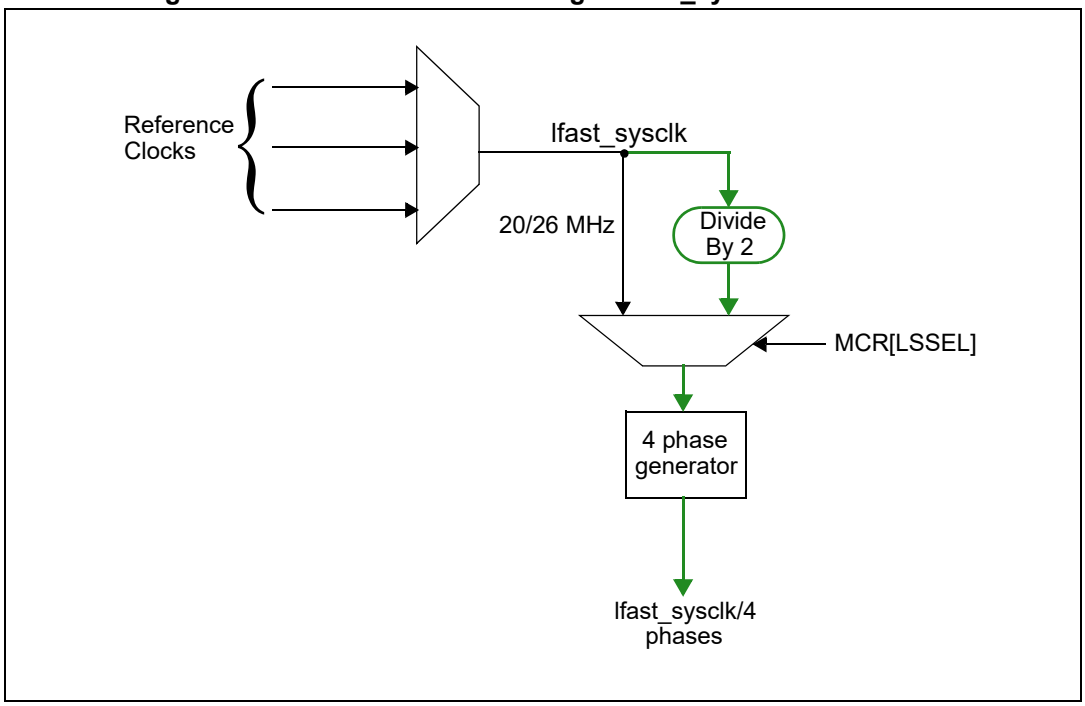
Figure 2114. External clock muxing of lfast_sysclk



All the reference clocks will be muxed first to provide lfast_sysclk to the module and then either div/2 or direct muxed clock will be used to generate 4 slow speed phases in the Clock Control Module of LFAST as shown in *Figure 2114*.

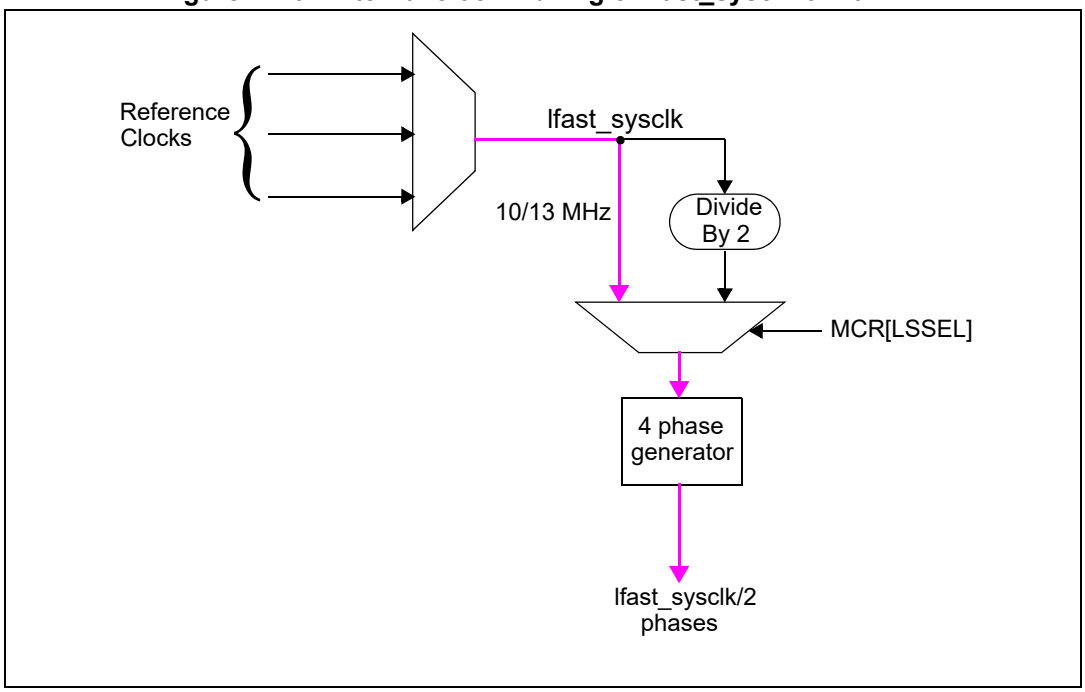
The lfast_sysclk could be either 20/26 MHz or 10/13 MHz. When lfast_sysclk is 20/26 MHz frequency, the clock control module will generate 4 phases of slow speed clock of frequency lfast_sysclk/4 when MCR[LSSEL] = 1. If lfast_sysclk is 20/26 MHz it needs to be first divided by 2 before using it for 4 phase generation in LFAST Clock Module, which generates 4 phases of half of the input frequency as shown in *Figure 2115* (selected clock path shown in green).

Figure 2115. External clock muxing of lfast_sysclk of 20/26 MHz



In this case, lfast_sysclk is 10/13 MHz frequency, the clock control module will generate 4 phases of slow speed clock of frequency lfast_sysclk/2 when MCR[LSSEL] = 0. If lfast_sysclk is 10/13 MHz then it does not need to be first divided by 2 before using it for 4 phase generation in LFAST Clock Module, which generates 4 phases of half of the input frequency as shown in *Figure 2116* (selected clock path selected shown in magenta).

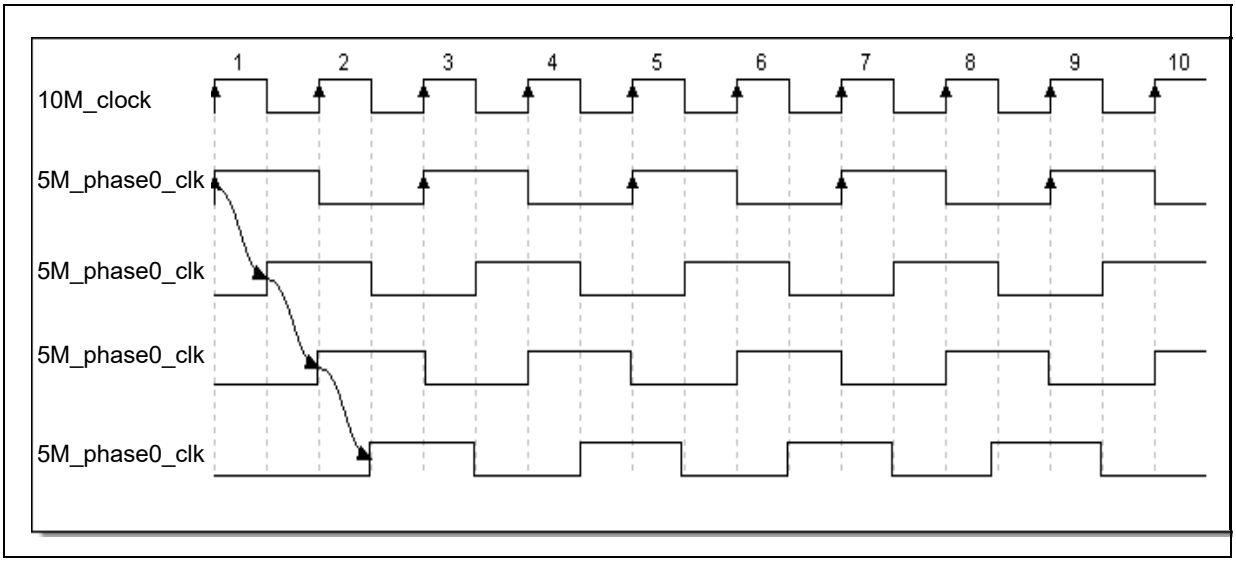
Figure 2116. External clock muxing of lfast_sysclk of 10 MHz



80.10.2.2 Slow speed 4-phase generator

Clock control module will generate 4 phases of slow speed clock of frequency of 10/13 MHz. For instance, if lfast_sysclk is 10 MHz then 4 phases of 5 MHz will be generated. *Figure 2117* Shows 4 phases getting generated using lfast_sysclk.

Figure 2117. Slow speed 4 phases generation

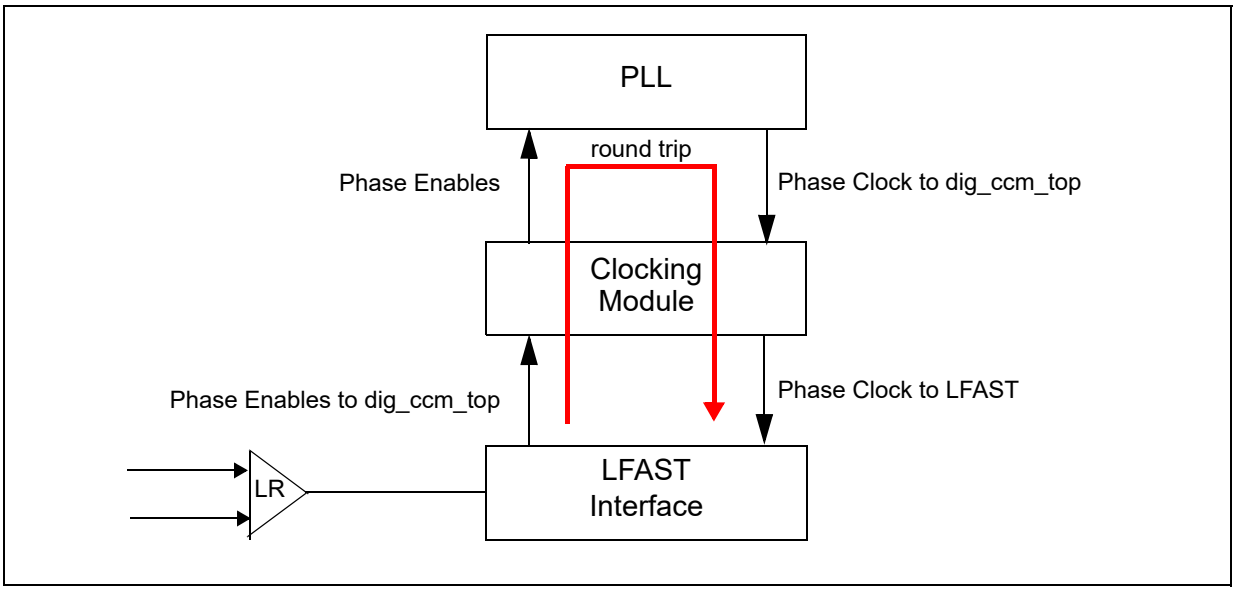


80.10.3 Rx controller clocks

80.10.3.1 Clocking module requirements for high speed phases

The high speed Phases are obtained from the PLL via the Clocking Module as shown in *Figure 2118*.

Figure 2118. Clock enables and clock paths



The LFAST block will know which of the phases will be required for the different modes of operations. Therefore, the LFAST block will provide enables and speed mode switches to the Clocking Module. The Clocking Module will use these signals to control the enables to the clock phases to reduce the power consumption.

The 8 phases of clocks signal are fed to the Clocking Module and muxed with the low speed phases. Depending on enables and data rate speed modes, the selected clocks are passed to the Sampler block and interface block.

COCR[PHSSEL] is connected to the Clocking Module, which selects whether 8 or 4 phases are selected for High Speed mode. It is only valid for high speed.

The routing of the 8 high speed phases to the interface is critical. Each clock phase will need to have the same routing track length from the PLL to the interface of Clocking Module. Each of the 8 high speed phases are separated by 45 degrees. This separation needs to be maintained from the phase generator to the interface.

80.10.3.2 Clock module requirements for low speed phases

80.10.3.2.1 Low speed

The low speed clock phases are generated in Clocking Module. These four phases are required to sample the data correctly at Low Speed mode. The LFAST block will provide the enable for the phases. The clock source for the low speed phases is SYSCLK. The Clocking Module block will need to generate the 4 phases of low speed clock 90 degrees apart.

Using the Low Speed mode on the interface allows the polyphase generation logic to be turned off and the requirement of high-speed-freq × 8 MHz clock from the PLL is not needed.

Table 2010. Rx clocks summary

Rx Speed mode	Source
Low Speed	lfast_sysclk
High Speed	PLL

80.10.4 Tx controller clocks

80.10.4.1 Clocking module requirements for speed phases

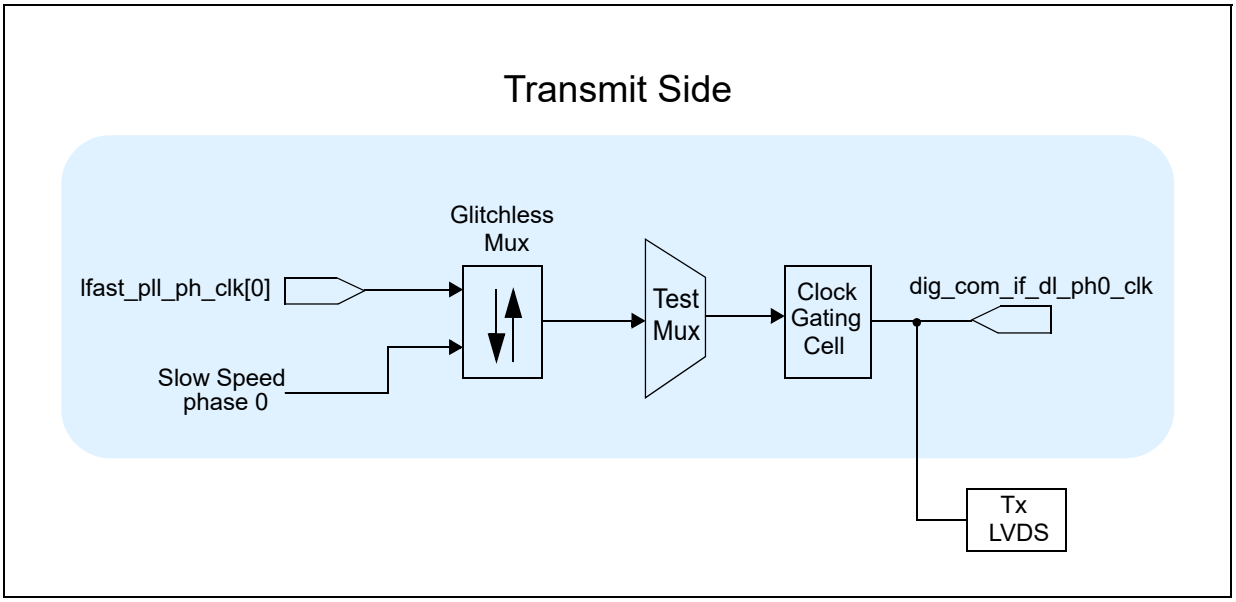
The low speed phase 0 clock is sourced from the lfast_sysclk and the high speed phase 0 clock is sourced from the PLL. See *Table 2011*.

Table 2011. Tx clocks summary

Tx Speed mode	Source
Low Speed	lfast_sysclk
High Speed	PLL

80.10.4.2 Transmit clock muxing

Figure 2119. Transmit clocks muxing



80.10.4.3 Tx request clock control

The Tx phase 0 clock is enabled when all the resets are negated.

81 Development Trigger Semaphore (DTS)

81.1 Introduction

The Development Trigger Semaphore (DTS) module enables software to signal an external tool by driving a persistent (affected only by reset or an external tool) signal on an external device pin. There are a variety of ways this module can be used, including as a component of an external real-time data acquisition system.

Note: When used as a component of a triggered data acquisition system, Nexus read/write access is different than the data acquisition protocol defined in the IEEE-ISTO 5001-2003 or IEEE-ISTO 5001-2011 Nexus standards, which use the Nexus Auxiliary port.

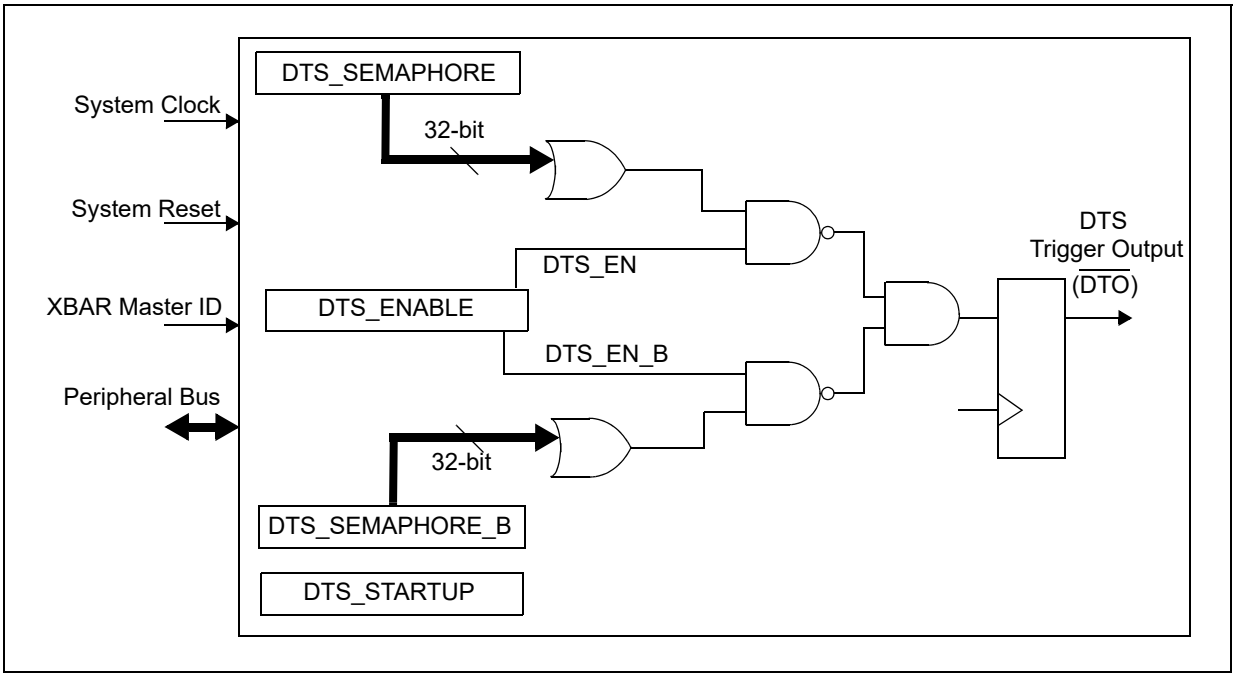
81.2 Overview

The Development Trigger Semaphore (DTS) module consists of registers and a small amount of combinational logic to generate an output signal, DTS Trigger Output (DTO). The registers are as follows:

- DTS_SEMAPHORE register: Any bit in this 32-bit register, when set to a value of logic ‘1’, causes the DTS module output signal to be asserted if the corresponding DTS_EN bit is set in the DTS_ENABLE register. This enables an external tool to detect up to 32 signals from the application software. In an application, each bit is generally associated with a specific data set. Only the processor core and DMA module can set bits in this register. The bits can only be cleared by a tool access via Nexus Read/Write Access or debug Zipwire.
- DTS_SEMAPHORE_B register: Any bit in this 32-bit register, when set to a value of logic ‘1’, causes the DTS module output signal to be asserted if the corresponding DTS_EN_B bit is set in the DTS_ENABLE register. This enables an external tool to detect an additional 32 signals from the application software. In an application, each bit is generally associated with a specific data set. Only the processor core and DMA module can set bits in this register. The bits can only be cleared by a tool access via Nexus Read/Write Access or debug Zipwire.
- DTS_STARTUP register: This register provides a mechanism for the external tool to notify software running on the CPU that the tool is connected and can provide information about either the type of tool or options that can be used by the software.
- DTS_ENABLE register: This register provides an enable/disable capability for the DTS feature.

The architecture is shown in *Figure 2120*.

Figure 2120. DTS block diagram

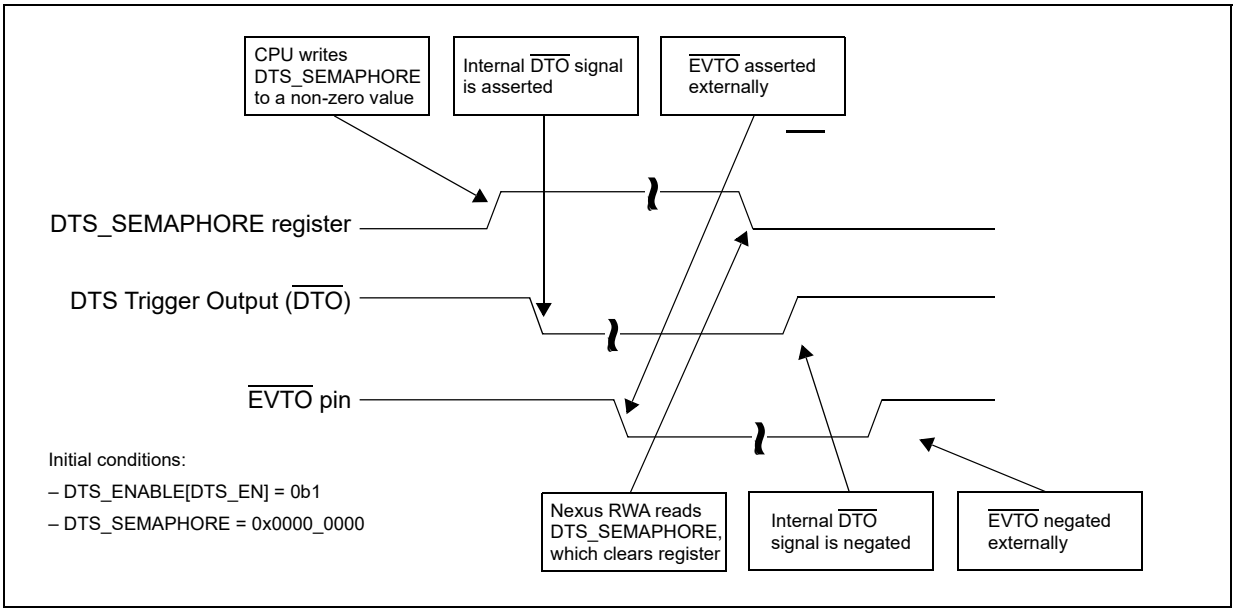


The DTS Trigger Output ($\overline{\text{DTO}}$) signal is connected to one of the $\overline{\text{EVTO}}$ inputs of the Debug and Calibration Interface (DCI). The other $\overline{\text{EVTO}}$ inputs are connected to the other Nexus modules in the device. $\overline{\text{DTO}}$ is asserted when any bit in a semaphore register is set and the DTS function of that register is enabled.

Note: When the DTS module is enabled (DTS_ENABLE[DTS_EN] = 0b1 or DTS_ENABLE[DTS_EN_B] = 0b1), all other Nexus $\overline{\text{EVTO}}$ functions associated should be disabled by the tool and $\overline{\text{EVTO}}$ becomes the $\overline{\text{DTO}}$. Unlike the $\overline{\text{EVTO}}$ function that only asserts for one clock, the $\overline{\text{DTO}}$ function remains asserted until the tool reads the semaphore register that is the source of the assertion, clearing the register's contents.

Figure 2121 shows an example of the chain of events that begins with setting of a bit in the DTS_SEMAPHORE register and the clearing of the register caused by a Nexus read.

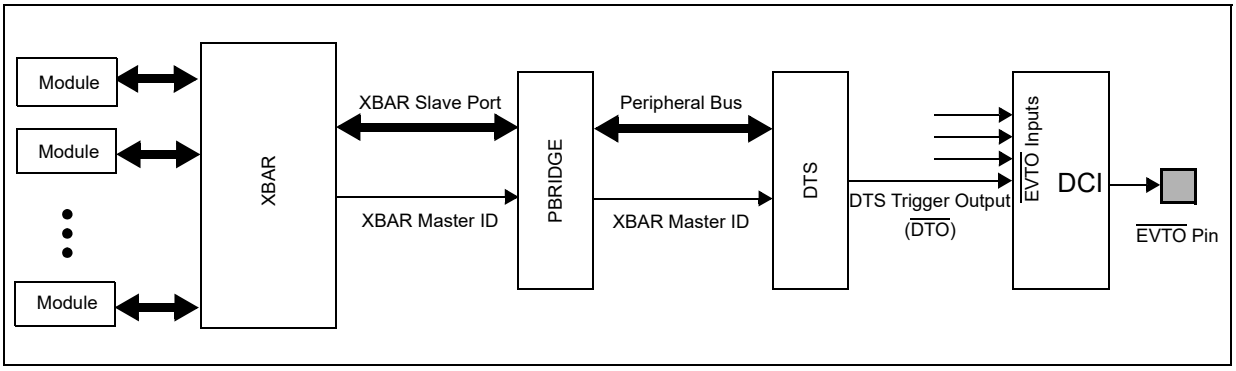
Figure 2121. $\overline{\text{DTO}}$ event sequence example



81.3 DTS device connections

Figure 2122 shows the DTS device connections. The DTS module connects to the Peripheral Bridge (PBRIDGE) for access to the registers. The PBRIDGE is connected to the device modules through a slave port of the Crossbar bus interface (XBAR).

Figure 2122. DTS device connections



The registers have limited access as described in *Section 81.3.1: DTS register access*. Access is based on the XBAR Master ID of the accessing module. Writing to the semaphore registers is limited to the cores and the eDMA module and is restricted to only setting bits. Only an access via a Nexus Read/Write Access from an external tool or debug Zipwire can clear bits in the semaphore registers (semaphore register bits are cleared automatically when read via Nexus/Write Access). Similarly, the DTS_ENABLE and DTS_STARTUP registers can only be written via a Nexus Read/Write Access or debug Zipwire.

Note: Nexus Read/Write Accesses use the core load/store bus to perform accesses, but Nexus accesses have a different Master ID than normal core load/stores.

81.3.1 DTS register access

A summary of accesses to all DTS registers by bus masters is provided in *Table 2012*. Only proper 32-bit accesses are valid. The effects of write accesses that are not 32 bits are not defined.

Table 2012. DTS register access effects

Register	32-bit read				32-bit write			
	RWA(1)	Core	eDMA	Other crossbar masters	RWA(1)	Core	eDMA	Other crossbar masters
DTS_ENABLE	Data	Data	Data	Data	Data	No effect	No effect	No effect
DTS_STARTUP	Data	Data	Data	Data	Data	No effect	No effect	No effect
DTS_SEMAPHORE	Data and Clear(2)	Data	Data	Data	No effect	Bit OR	Bit OR	No effect
DTS_SEMAPHORE_B	Data and Clear(2)	Data	Data	Data	No effect	Bit OR	Bit OR	No effect

1. Nexus Read/Write access via an external tool
2. A read of the semaphore registers by Nexus Read/Write Access module is destructive and clears all bits in the register

Access to DTS module registers is controlled based on the XBAR Master ID of the accessing module.

Note: The XBAR Master ID should not be confused with the Master Port number of the XBAR. See Chapter 18: Crossbar switch (XBAR), for details.

Tools must access the DTS registers through the Nexus Read/Write Access mechanism of a core or debug Zipwire. External tool accesses through either core appear as if the access is via the core and therefore do not have the same level of access as a Nexus Read/Write Access.

81.4 Memory map and register definition

Table 2013 shows the memory map of the DTS module registers.

Table 2013. DTS memory map

Offset addesss	Register name	Access	Reset Value	Location
0x0000	DTS Output Enable register (DTS_ENABLE)	Restricted R/W(1)	0x0000_0000	*Section 81.4.1.1*
0x0004	DTS Startup register (DTS_STARTUP)	Restricted R/W(1)	0x0000_0000	*Section 81.4.1.2*
0x0008	DTS Semaphore Register (DTS_SEMAPHORE)	Restricted R/W(1)	0xFFFF_FFFF	*Section 81.4.1.3*
0x000C	DTS Semaphore Extension (DTS_SEMAPHORE_B)	Restricted R/W(1)	0xFFFF_FFFF	*Section 81.4.1.4*

1. Only certain types of accesses are allowed. See the register figure note of each register for a description of the accesses that are allowed.

81.4.1 Register descriptions

81.4.1.1 DTS output enable register (DTS_ENABLE)

The DTS_ENABLE register controls the DTS Trigger Output ($\overline{\text{DTO}}$) and whether $\overline{\text{DTO}}$ is active on the $\overline{\text{EVTO}}$ output pin of the device. *Figure 2123* shows the format of the DTS_ENABLE register. The DTS_ENABLE register can be read by the core, but can only be written by a Nexus Read Write Access (RWA) or debug Zipwire.

Note: Access to the DTS_SEMAPHORE, DTS_SEMAPHORE_B, and DTS_STARTUP registers are unaffected by the state of this register.

Offset 0x0000 Access: Restricted R/W(1)

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	DTS_EN_B	DTS_EN
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. The DTS_ENABLE register can be read by the core, but can only be written by a Nexus Read Write Access (RWA).

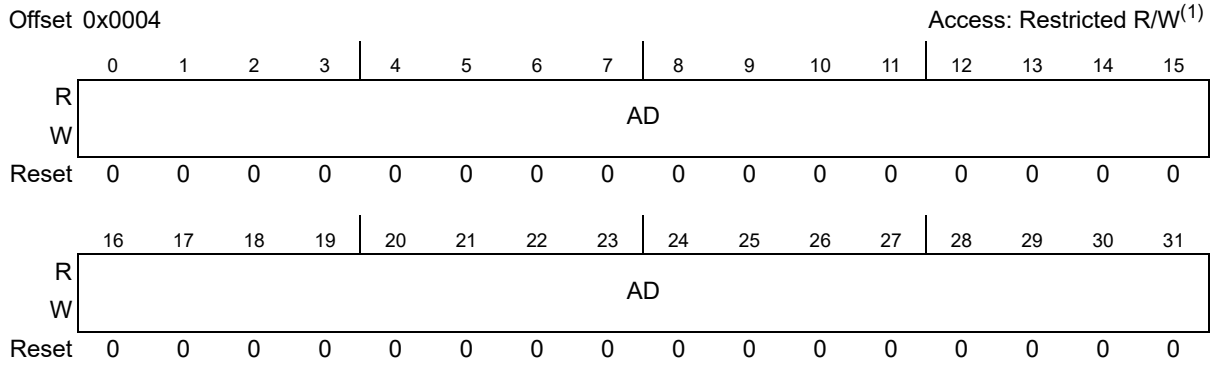
Figure 2123. DTS output enable register (DTS_ENABLE)

Table 2014. DTS_ENABLE field descriptions

Field	Description
30 DTS_EN_B	DTS Enable B Controls whether the $\overline{\text{DTO}}$ signal is routed to the $\overline{\text{EVTO}}$ pin. The DTS Enable bit is cleared by a device reset (either the assertion of the external $\overline{\text{RESET}}$ or by an internally generated reset). A Nexus reset does not change the state of this register. 0 DTS output disabled. Any bit set in the DTS_SEMAPHORE_B register does not assert the DTS trigger output signal. 1 DTS output enabled. Any bit set in the DTS_SEMAPHORE_B register asserts the DTS trigger output signal ($\overline{\text{DTO}}$).
31 DTS_EN	DTS Enable Controls whether the $\overline{\text{DTO}}$ signal is routed to the $\overline{\text{EVTO}}$ pin. The DTS Enable bit is cleared by a device reset (either the assertion of the external $\overline{\text{RESET}}$ or by an internally generated reset). A Nexus reset does not change the state of this register. 0 DTS output disabled. Any bit set in the DTS_SEMAPHORE register does not assert the DTS trigger output signal. 1 DTS output enabled. Any bit set in the DTS_SEMAPHORE register asserts the DTS trigger output signal ($\overline{\text{DTO}}$).

81.4.1.2 DTS startup register (DTS_STARTUP)

Figure 2124 shows the format of the DTS_STARTUP register, used for tool detection and startup information exchange between external data acquisition tool and the embedded controller. DTS_STARTUP register can be read by the core, the eDMA modules and other crossbar masters but can only be updated by a Nexus Read Write Access (RWA) or debug Zipwire.



1. The DTS_STARTUP register can be read by the cores, the eDMA module and Nexusbut can only be updated by a Nexus Read Write Access (RWA).

Figure 2124. DTS startup register (DTS_STARTUP)

Table 2015. DTS_STARTUP field descriptions

Field	Description
0:31 AD	Application Dependent register bits. The bits have no defined meaning to the microcontroller. They are used to by an external tool to pass information (for example, application options and status) to application software running on target microcontroller at startup time. Use a Nexus RWA 32-bit write access or debug Zipwire to update the contents of this register. A device reset (either from the $\overline{\text{RESET}}$ pin or an internally generated reset) clears all bits in the register. A Nexus reset does not change the contents of the register.

81.4.1.3 DTS semaphore register (DTS_SEMAPHORE)

Figure 2125 shows the format of the DTS_SEMAPHORE register. The DTS_SEMAPHORE register is used by software to assert the $\overline{\text{DTO}}$ signal on the device $\overline{\text{EVTO}}$ pin. A one in any bit of this register causes the $\overline{\text{DTO}}$ signal on the $\overline{\text{EVTO}}$ pin to be driven low. The intended use of this register is for the $\overline{\text{DTO}}$ signal to notify tools that data is available. Individual bits are used to identify the specific data.

Offset 0x08 Access: Restricted R/W

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	ST															
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	ST															
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Figure 2125. DTS semaphore register (DTS_SEMAPHORE)

Table 2016. DTS_SEMAPHORE field descriptions

Field	Description
0:31 ST	Semaphore Trigger. When a core or eDMA writes a logical ‘1’ to a bit, the bit is set. A write of ‘0’ by the core or DMA does not change the state of the bit. – All register bits are set to ‘1’ by a device reset. – A Nexus reset does not change the state of this register. – The register can be accessed, with restrictions, by any core, DMA or any Nexus RWA or debug Zipwire. – For the core or DMA, only 32-bit write or read accesses are valid. – A core or DMA valid read access returns the current value of the register and leaves the register unchanged. 0 No flag 1 Flag is set

81.4.1.4 DTS semaphore extension (DTS_SEMAPHORE_B)

Figure 2126 shows the format of the DTS_SEMAPHORE_B register. The DTS_SEMAPHORE_B register is used by software to assert the DTO signal on the device EVTO pin. A one in any bit of this register causes the DTO signal on the EVTO pin to be driven low. The intended use of this register is for the DTO signal to notify tools that data is available. Individual bits are used to identify the specific data.

Offset 0x0C Access: Restricted R/W

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	ST_B															
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	ST_B															
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Figure 2126. DTS Semaphore extension register (DTS_SEMAPHORE_B)

Table 2017. SEMAPHORE_B field descriptions

Field	Description
0:31 ST_B	Semaphore Trigger Extension. When a core or eDMA writes a logical ‘1’ to a bit, the bit is set. A write of ‘0’ by the core or DMA does not change the state of the bit. – All register bits are set to ‘1’ by a device reset. – A Nexus reset does not change the state of this register. – The register can be accessed, with restrictions, by any core, DMA or any Nexus RWA or debug Zipwire. – For the core or DMA, only 32-bit write or read accesses are valid. – A core or DMA valid read access returns the current value of the register and leaves the register unchanged. 0 No flag 1 Flag is set

81.5 Example application

The calibration process of a new engine requires real-time access to calibration tables and the ability to update the tables in real-time The DTS module enables this capability by enabling software to assert a signal to an external device pin to notify an external tool that data is available. The tool can then retrieve the data.

In this type of application the DTS_SEMAPHORE register and DTS Trigger Output (DTO) signal provide a mechanism to notify the calibration tool that the calibration variable or variables (or sets of measurements), up to 32, have been updated with new values and are available for the tool to access.

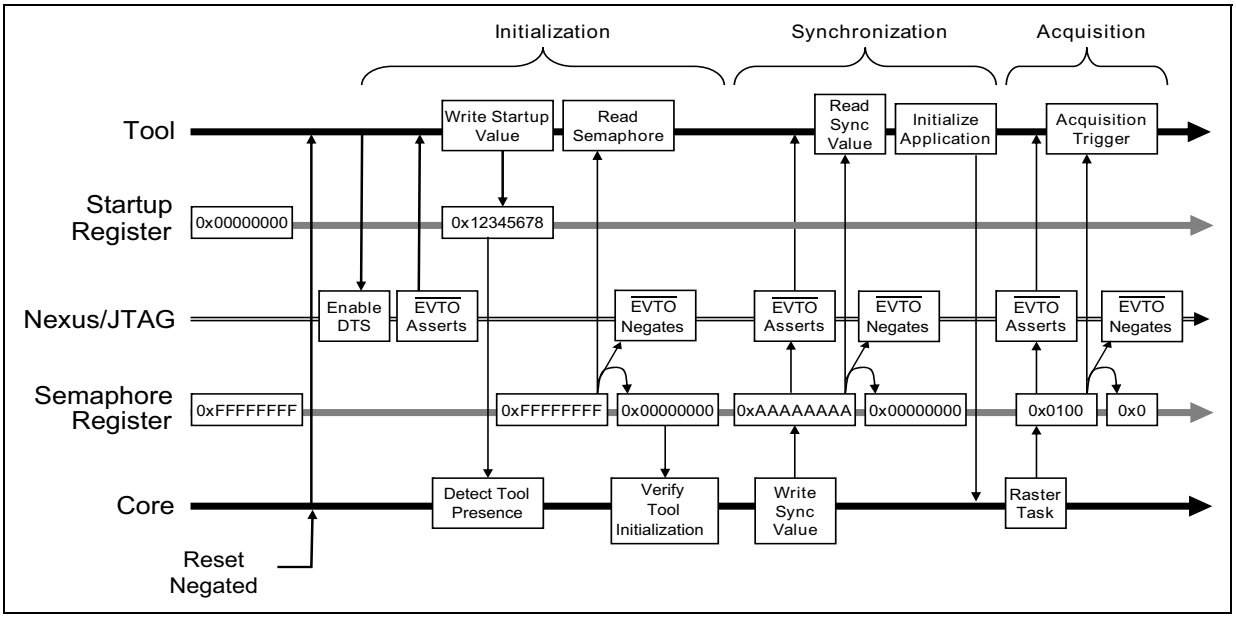
Note: It is the user’s responsibility to ensure that the tool has time to retrieve the data prior to that particular trigger being set a second time. It is also permissible to have multiple triggers active at the same time or for a second trigger to be set before a previous trigger has been serviced, as long as it is not the same trigger (unless it is acceptable to the tool to not receive every data set).

Figure 2127 shows an example DTS startup sequence for an external real-time data acquisition system. The startup and synchronization sequence can be as simple or as complicated as the need requires. However, a typical startup sequence is as follows:

1. The DTS_STARTUP register is cleared by a power on reset or any CPU reset.
2. The tool writes a non-zero value to the DTS_STARTUP register.
3. The CPU (user application software) then reads the value of the DTS_STARTUP register. Based on this value, different initialization options can be selected. The bits can be used for any application specific definitions.
4. Since the DTS_SEMAPHORE register is cleared when the tool reads the current value. The tool should perform all necessary initialization before reading this register. The application software can then check that the DTS_SEMAPHORE register was cleared by the tool, to determine that it is safe to start using it for its intended raster trigger semaphore function.
5. An optional hand shake from the CPU can be used to inform the tool that the user software has detected that the tool is attached and the CPU has performed the proper initialization for the tool by writing a predefined value to the DTS_SEMAPHORE register (the example shown in the figure above uses 0xAAAA_AAAA, all A’s was used

since it is unrealistic that 16 channels could be enabled very quickly after start up after a reset).

Figure 2127. DTS startup sequence example



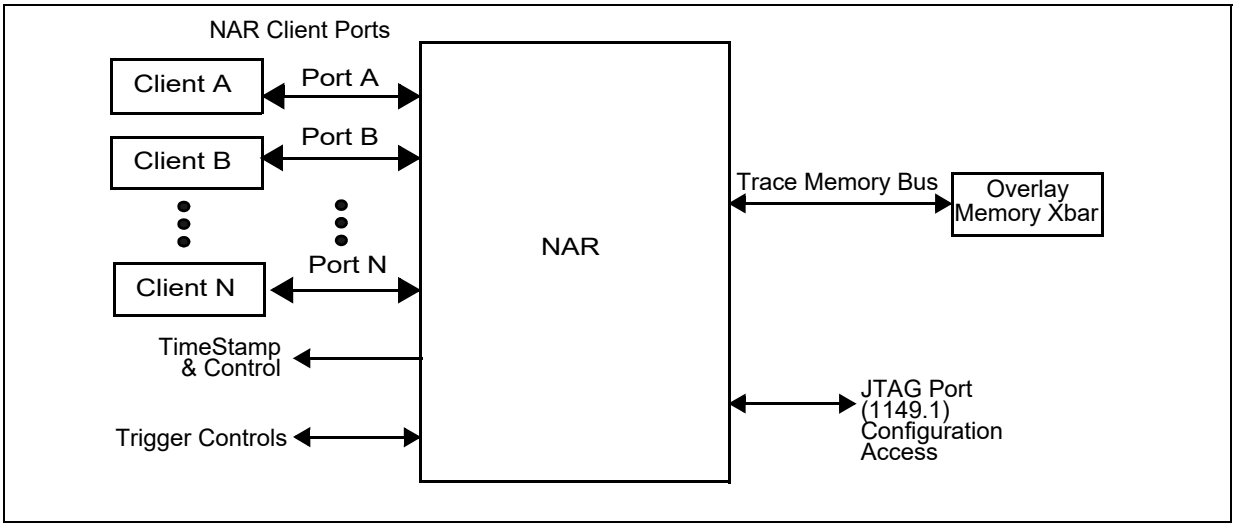
82 Nexus Aurora Router (NAR)

82.1 Introduction

On the device, there are multiple blocks that require development interface support. The blocks that are Nexus-compliant (based on the IEEE-ISTO 5001 standard) are expected to interface with the development tool through a dedicated high-bandwidth debug port. The Nexus Aurora Router (NAR) controls the usage of the output port in a manner that allows all the individual development interface blocks to share the port, and appear to the development tool to be a single block.

Top-level architecture is shown in *Figure 2128*. Multiple clients send messages by sharing the high-bandwidth port through the NAR. For a list of the specific clients that are connected to the NAR, see the Calibration and Debug configuration chapter which describes how the modules are configured.

Figure 2128. NAR top level diagram



The trace memory bus port provides added flexibility to the channel trace data to memory on the Nexus Aurora crossbar. Setup and configuration of the NAR is performed by accessing the configuration registers via a JTAG (1149.1) port.

Each Nexus client has a dedicated receive queue to allow seamless switching between sources at Nexus message boundaries. Incoming Nexus data is parsed into the receive queues so that end-of-message can be determined. Messages are then transferred into the main queue. An arbitration block determines the order in which clients are serviced, using an optional client request level bus to prioritize the data (lacking request level data, a simple round-robin algorithm is used). The NAR has two output ports: the high bandwidth debug port (Aurora out) and the trace memory bus. If both output ports are enabled at the same time, the main queue is logically partitioned inside NAR. The Production Device (PD) does not support partitioning of the main queue; therefore, only one of the ports can be active at a time. Data going through the Aurora out port or the Trace port goes through an additional translation block to put the data in the correct format. The actual Aurora formatting is done in the NAL. Additional blocks for message generation and timestamp generation are described in more detail in the following sections.

The NAR supports the following client interfaces:

- Legacy interface to support existing client interface protocol through a conversion gasket logic with clock domain transfer
- New client interface with clock domain transfer

82.2 Overview

The NAR serves as an arbiter for the high-bandwidth debug port (HBDP). The NAR pulls Nexus messages from multiple clients, queues the data, and forwards it (based on the user-controlled configuration) to the high-speed Aurora Out port or the Trace port, or both. The trace memory bus controller allows the NAR to transmit data via the normal system interface. NAR control and configuration registers are accessed via JTAG (1149.1) port.

82.3 Features

The NAR supports the following features:

- Connects 10 client(s) with dedicated receive queue(s). Clients can transmit data as long as there is space in its associated receive queue, but only the active port receive queue empties into the main queue.
- Legacy client interface for backward compatibility
- Aurora output data format for dedicated high-bandwidth serial transmission
- Aurora output format support at client interface
- Trace Memory port (64-bit address and data) can be used to write Nexus messages directly to external overlay memory. Block transfer mode provides highest-bandwidth option.
- Configurable Start Address and Range Selection for Trace Memory port write operation provides flexibility to choose Overlay memory Section and range to store Trace data.
- Time-stamping provides a central synchronization resource for message ordering. Gray coded timer output for clients working in different clock domians.
- Re-allocatable receive queues allow queue-space of an inactive client to be reassigned to an active client
- Stall detection and handling allows NAR to continue when a client stops sending data in the middle of a message
- Event generation on External Trace Memory (overlay) programmable full
- Suppress mode can be used to shut down message delivery and relieve back-pressure to clients without turning off Nexus messaging in the clients.

82.4 Register definition

The NAR configuration registers provide configuration and control for all NAR operations. They are accessible through the JTAG (1149.1) port. Addressing (within the local NAR address space) for the NAR registers is shown in the following table. If the NAR clock is turned off, the NAR registers cannot be accessed.

Table 2018. NAR registers

Register name	Read/ Write	JTAG index (NAR relative)	Location
NAR_CR—NAR Control Register	R/W	0	*on page 3487*
NAR_ST—NAR Status Register	R	1	*on page 3489*
NAR_SFR—Source Filter Register	R/W	2	*on page 3491*
NAR_TFR—Type Filter Register	R/W	3	*on page 3492*
NAR_TBALO—Trace memory port base address, low	R/W	4	*on page 3493*
NAR_TBAHI—Trace memory port base address, high	R/W	5	*on page 3493*
NAR_TCR—Trace Memory bus control register	R/W	6	*on page 3494*
NAR_STCR—Suppress Trigger Control Registers 1–4	R/W	7 (Trigger 1) 8 (Trigger 2) 9 (Trigger 3) A (Trigger 4)	*on page 3495*
NAR_CSSR–Client Suppress Status Register	R	B	*on page 3497*
NAR_CDR—Client Disable Register	R/W	C	*on page 3497*
NAR_AHFPAR—Trace memory fill level and partition configuration	R/W	E	*on page 3499*

82.4.1 Register descriptions

82.4.1.1 NAR control register (NAR_CR)

The NAR control register provides all the main control and enable bits for the NAR and its output ports. The following figure shows the format of the NAR_CR.

Register index: 0

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	NEN	NCD	TEN	TRS	TSG		SED	SRD	TBW	AQP		0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	WPE			0	0	0	0	0	ALT		0	AFA	0	0	PUS	NSR
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 2129. NAR control register (NAR_CR)

The NAR_CR is described in the following table.

Table 2019. NAR_CR field descriptions

Field	Description
31 NEN	NAR Enable. 0 NAR is disabled unless HBDP is independently enabled. See *Section 82.5.2.2: Disabled* for more information. 1 NAR is enabled.
30 NCD	NAR Communication Disable. 0 NAR monitors clients for activity 1 NAR communication with clients is disabled
29 TEN	Timestamp Enable. 0 Timestamp packet is not included in messages 1 Timestamp packet is included in messages from all client including NAR generated messages
28 TRS	Timestamp Reset. 0 to 1 transition: reset timestamp counter in NAR
27–26 TSG	Timestamp Granularity (Mode). *Note: Some clients may only support timestamps on every message.* 00 Add timestamp to every message 01 Add timestamp to every 4th message 10 Add timestamp to every 16th message 11 Add timestamp to every 32nd message
25 SED	Stall Error Disable. 0 NAR injects dummy complete message when active client stalls 1 NAR does not inject dummy complete message
24 SRD	Stall Recovery Disable. 0 Stalled receive queue resumes operation when its client comes back on line 1 Stall error recovery is disabled
23 TBW	Tracebuffer Wrap. 0 NAR does not capture new data when main queue is full 1 NAR begins overwriting main queue from the beginning upon queue overflow
22–21 AQP	Alternate Queue Partition. 00 Queue partitioning is disabled 01 Reserved 10 Reserved 11 Reserved
20	Reserved and must be written with 0
19–16	Reserved and must be written with 0s
15–13 WPE	Watchpoint Enable 000 NAR watchpoint messaging is disabled xx1 NAR generates a watchpoint message upon a sync event x1x NAR generates a watchpoint message upon a trace memory bus or tracebuffer block full/wrap event 1xx NAR generates a watchpoint message upon a suppress-mode event
12	Reserved
11	Reserved. This bit must be written with 0.
10–9	Reserved and must be written with 0s.

Table 2019. NAR_CR field descriptions (continued)

Field	Description
8	Reserved
7–6 ALT	Alternate Port. Defines which port is the target for non-filtered messages; main port in single-port applications. 00 No alternate port. Running in trace-buffer mode. 01 Aurora out is the ALT 10 Trace memory bus is the ALT 11 Reserved
5	Reserved
4 AFA	Automatic Flush Accumulators. 0 Output queue accumulators do not flush stranded beats 1 Accumulators automatically flush stranded beats
3	Reserved. This bit must be written with 0.
2	Reserved and must be written with 0.
1 PUS	Power-Up Suppressed. 0 NAR powers up active 1 NAR powers up in global suppressed mode
0 NSR	NAR Soft Reset. 1 Resets all configuration registers and sets queue address pointers to the top of their respective queues. Also resets itself after a single-cycle delay. (This does not affect other blocks)

82.4.1.2 NAR status register (NAR_ST)

The NAR_ST register provides information on NAR activity and queue status, as well as HBDP status and errors. It is a read-only register. The following figure shows the format of the NAR_ST.

Register index: 1

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	NEN	NUM			NCE				0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	HPR	HPE	MPE	MOV	0	TOV	QWA									
W		w1c	w1c	w1c		w1c										
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

Figure 2130. NAR status register (NAR_ST)

The NAR_ST register is described in the following table.

Table 2020. NAR_ST field descriptions

Field	Description
31 NEN	NAR Enabled. 0 NAR is disabled and inactive 1 NAR is enabled with clocks on
30–28 NUM	NAR Mode. 000 NAR is in disabled mode 001 NAR is in global suppressed mode 010 NAR is in communication-disabled mode 011 NAR is in tracebuffer mode 100 NAR is in active transmit mode 101 Reserved 110 Reserved 111 NAR is in run-control mode
27–24 NCE	NAR Config Error. See *Table 2034* for coding
23–16	Reserved
15 HPR	HBDP Ready. 0 HBDP (Aurora Out port) is not ready 1 HBDP is ready and accepts data for transmission
14 HPE	HBDP Error. 0 No HBDP error 1 Error condition exists on HBDP
13 MPE	Trace memory bus Error. 0 No Trace memory bus error 1 Error condition detected on trace transaction
12 MOV	Trace memory bus block overflow. 0 No Trace memory bus overflow 1 Targeted external memory block is full. Trace memory bus writes are halted (or wrapped).
11	Reserved
10 TOV	Tracebuffer Overflow. 0 No tracebuffer overflow 1 The tracebuffer has overflowed/wrapped at least once
9–0 QWA	Queue Write Address. Current value of the output queue write pointer (last valid write + 1)

82.4.2 Message filtering registers

The two output ports (HBDP and trace memory bus) are independently controllable. In the BD NAR, two ports can be active simultaneously. When multiple ports are active, user-programmable filters are determine which port receives a message, based on the message source or type. Filters apply to the designated Message Select Port (MSP), and are used to select messages for that port. All other messages go to the other (alternate) port. If multiple ports are used, an MSP must be designated; otherwise a configuration error results. See *Section 82.5.6: Output arbitration*, for details on how the NAR arbitrates between multiple output ports. These registers should only be set in the BD NAR because partitioning is not supported in the PD NAR.

82.4.2.1 Source filter register (NAR_SFR)

The MSP can be configured to accept or exclude messages from up to four sources. The sources which are to be filtered are programmed in the source filter register (NAR_SFR) as shown in the following figure. This register should only be set in the BD NAR because partitioning is not supported in the PD NAR.

Register index: 2

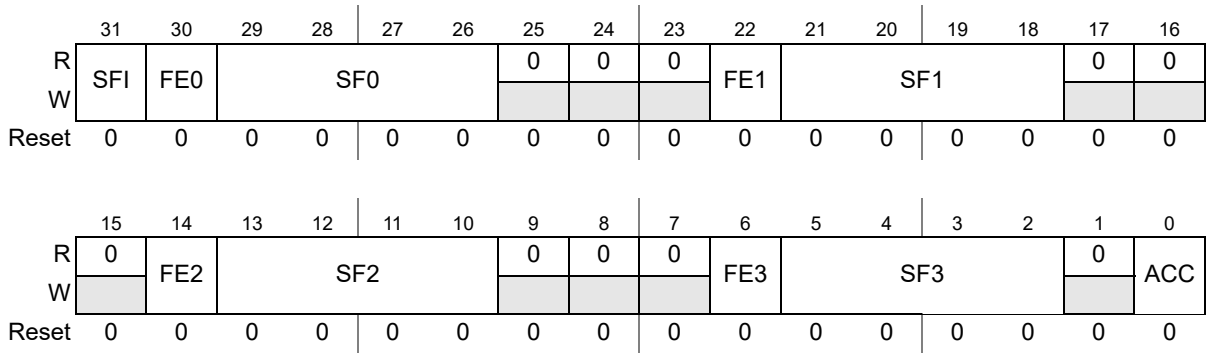


Figure 2131. Source filter register (NAR_SFR)

The NAR_SFR is described in the following table. The source trace IDs, that are compared to the source filter fields (SFn), are device specific. For a list of the clients that are connected to the NAR and their corresponding Nexus source trace IDs, see the Calibration and Debug configuration chapter which describes how the modules are configured.

Table 2021. NAR_SFR field descriptions

Field	Description
31 SFI	Source Filter Invert. 0 Match if message source equals any of the enabled filters 1 Match if message source equals none of the enabled filters
30 FE0	Enable Filter 0. 0 Filter 0 disabled. SF0 ignored 1 Filter 0 enabled and contained in SF0
29–26 SF0	Source Filter 0. The message source trace ID is compared with this field. See *Section 82.5.6: Output arbitration* for an example.
25–23	Reserved.
22 FE1	Enable Filter 1. 0 Filter 1 disabled. SF1 ignored 1 Filter 1 enabled and contained in SF1
21–18 SF1	Source Filter 1. The message source trace ID is compared with this field. See *Section 82.5.6: Output arbitration* for an example.
17–15	Reserved
14 FE2	Enable Filter 2. 0 Filter 2 disabled. SF2 ignored 1 Filter 2 enabled and contained in SF2

Table 2021. NAR_SFR field descriptions (continued)

13–10 SF2	Source Filter 2. The message source trace ID is compared with this field. See *Section 82.5.6: Output arbitration* for an example.
9–7	Reserved
6 FE3	Enable Filter 3. 0 Filter 3 disabled. SF3 ignored 1 Filter 3 enabled and contained in SF3
5–2 SF3	Source Filter 3. The message source trace ID is compared with this field. See *Section 82.5.6: Output arbitration* for an example.
1	Reserved
0 ACC	OR source and type filters. 0 Match if source AND type filters match 1 Match if source OR type filters match

82.4.2.2 Message type filter register (NAR_TFR)

The MSP can be configured to accept or exclude messages of up to four types, as encoded in the message TCODE field. The TCODEs which are to be filtered are programmed in the message type filter register (NAR_TFR) as shown in the following figure. This register should only be set in the BD NAR because partitioning is not supported in the PD NAR.

Register index: 3

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	TFI	FE0	TF0						0	FE1	TF1					
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	FE2	TF2						0	FE3	TF3					
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 2132. Message type filter register (NAR_TFR)

The NAR_TFR is described in the following table.

Table 2022. NAR_TFR field descriptions

Field	Description
31 TFI	Type Filter Invert. 0 Match if message TCODE equals any of the enabled filters 1 Match if message TCODE equals none of the enabled filters
30 FE0	Enable Filter 0. 0 Filter 0 disabled. TF0 ignored 1 Filter 0 enabled and contained in TF0
29–24 TF0	Type Filter 0. The message TCODE is compared with this field. See *Section 82.5.6: Output arbitration* for an example.

Table 2022. NAR_TFR field descriptions (continued)

23	Reserved
22 FE1	Enable Filter 1. 0 Filter 1 disabled. TF1 ignored 1 Filter 1 enabled and contained in TF1
21–16 TF1	Type Filter 1. The message TCODE is compared with this field. See *Section 82.5.6: Output arbitration* for an example.
15	Reserved
16 FE2	Enable Filter 2. 0 Filter 2 disabled. TF2 ignored 1 Filter 2 enabled and contained in TF2
13–8 TF2	Type Filter 2. The message TCODE is compared with this field. See *Section 82.5.6: Output arbitration* for an example.
7	Reserved
6 FE3	Enable Filter 3. 0 Filter 3 disabled. TF3 ignored 1 Filter 3 enabled and contained in TF3
5–0 SF3	Type Filter 3. The message TCODE is compared with this field. See *Section 82.5.6: Output arbitration* for an example.

82.4.3 Trace memory bus configuration registers

The trace memory bus can be used to transfer debug messages out to a dedicated block of external memory space. The trace memory bus configuration registers are used to specify the location and size of the external debug space.

82.4.3.1 Trace memory bus base address registers (NAR_TBALO and NAR_TBAHI)

The trace memory bus base address registers determine where Nexus data sent out through the trace memory bus goes. The NAR sends out blocks of data on the trace memory bus, writing to an address block defined by the specified base address. Data can transmitted either on an as-available, word-wide basis, or in larger blocks. In block transfer mode, the size of the data block is determined by the BTM field of the Control Register; the default is 64 bytes. Even if block transfer mode is not used, the 5, 6, 7, or 8 (depending on the BTS field in NAR_TCR register) LSBs of the base address are not accessible. This forces the debug memory space to be aligned to a block boundary.

When read, the address registers return the current address, allowing the user to monitor the progress of a trace memory transfer. The following figures show the format of the trace memory base address registers (NAR_TBALO and NAR_TBAHI). For a description of the allowed address for these registers, see the Calibration and Debug configuration chapter which describes how the modules are configured.

Register index: 5

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

Figure 2133. Trace memory base address register NAR_TBAHI

Register index: 4

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	TBALO																						ADR LO[9]	ADR LO[8]	ADR LO[7]	ADR LO[6]	0	0	0	0	0	0
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 2134. Trace memory base address register NAR_TBALO

The NAR_TBALO and NAR_TBAHI registers are described in the following tables.

Table 2023. NAR_TBAHI field descriptions

Field	Description
31–0 TBAHI	Trace Debug Memory Base Address on AHB-EW port, upper part. For 32-bit address applications this field is all zeros.

Table 2024. NAR_TBALO field descriptions

Field	Description
31–10 TBALO	Trace Debug Memory Base Address on AHB-EW port, lower part
9 ADRLO[9]	Base Address LSB for BTS = 11 (384-byte block)
8 ADRLO[8]	Base Address LSB for BTS = 10 (256-byte block)
7 ADRLO[7]	Base Address LSB for BTS = 01 (128-byte block)
6 ADRLO[6]	Base Address LSB for BTS = 00 (64-byte block)
5–0	Reserved

82.4.3.2 Trace memory bus control register (NAR_TCR)

The trace memory bus control register is used to control the attributes of the trace memory bus port, such as block transfer size and message priority. Since the trace memory bus transfers are expected to go to a static block of external memory, a maximum transfer size must be specified to prevent the NAR from overwriting memory it is not authorized to access. This is done via the MXFR max transfer size register field. The memory allocation is

specified in units of the trace memory bus block transfer size (64, 128, 256, or 384 bytes), so the actual size of the reserved memory block depends on the BTS setting. The following figure shows the format of the NAR_TCR.

Register index: 6

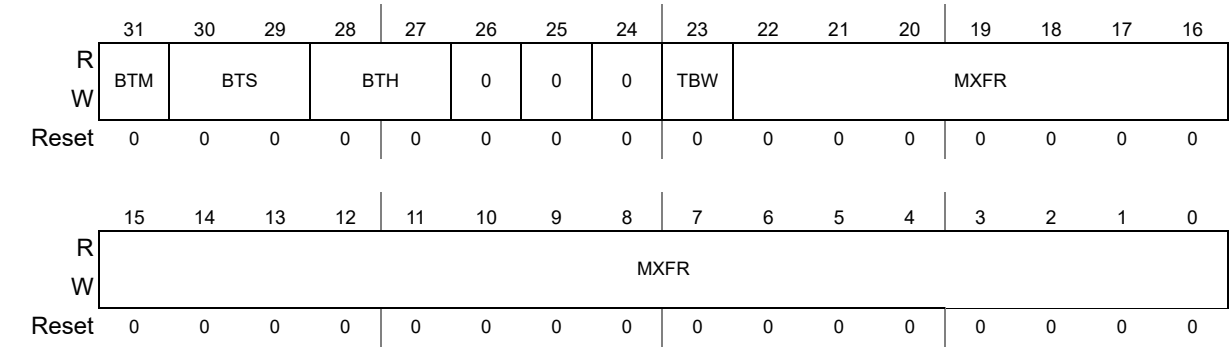


Figure 2135. Trace memory bus control register (NAR_TCR)

The NAR_TCR is described in the following table.

Table 2025. NAR_TCR field description

Field	Description
31 BTM	Block Transfer Mode. 0 Data for the Trace Memory bus port is transmitted one word at a time, upon first availability 1 Data to the Trace Memory bus port is bundled into blocks before transmitting
30–29 BTS	Block Transfer Size. 00 Block transfer size = 64 bytes 01 Block transfer size = 128 bytes 10 Block transfer size = 256 bytes 11 Block transfer size = 384 bytes
28–27 BTH	Block Transfer Threshold. 00 Full threshold. Number of entries in partition must equal block transfer size before block transfer starts 01 Begin transfer when block is 25% full 10 Begin transfer when block is 50% full 11 Begin transfer when block is 75% full
26–24	Reserved. These bits must be written with 0.
32 TBW	Trace Memory Bus Block Wrap. 0 NAR stops writing to the Trace Memory bus debug memory space once Max Size has been reached 1 NAR overwrites Trace Memory bus debug memory from the beginning upon Trace Memory bus block overflow.
22–0 MXFR	Trace Memory Bus Max Transfer Size. Total size of the dedicated debug memory space, in Trace Memory bus data transfer blocks.

82.4.4 Suppress mode triggers register (NAR_STCR)

Four suppress mode triggers are provided for precise control of the NAR suppress mode feature described in *Section 82.5.4: Suppress mode*. The suppress mode triggers are only available on the PD NAR and do not apply to the BD NAR. The registers are present in the BD NAR, but input suppress triggers are not connected. Each trigger can be programmed to

suppress or wake clients individually or the entire NAR. All receive queues can be suppressed on the occurrence of a selected trigger, or the full NAR can be brought out of suppress mode upon the occurrence of the appropriate trigger. A trigger can also be programmed to toggle one or more clients between the suppressed and active states. Each Suppress Trigger Control Register, shown in the following figure, is associated with one of the triggers, and are used to determine the action taken, if any, upon the assertion of that trigger. The suppress triggers are connected to the Sequence Processing Unit (SPU) action unit.

Register index: 7–A

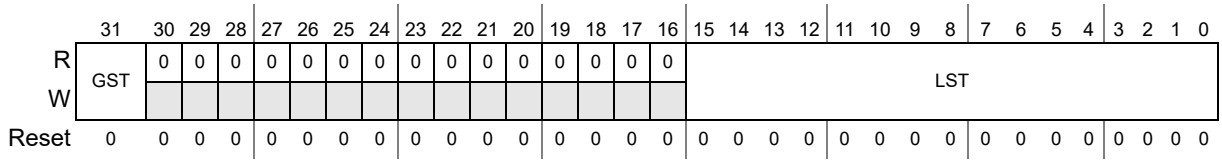


Figure 2136. Suppress trigger control registers (NAR_STCR1–4)

The NAR_STCR are described in the following table. The set of clients connected to the NAR is device specific. For a list of the clients that are connected to the NAR and their corresponding NAR client index for the local suppress trigger, see the Calibration and Debug configuration chapter which describes how the modules are configured.

Table 2026. NAR_STCR field descriptions

Field	Description
31 GST	Global Suppress Trigger. 0 The associated trigger cannot cause NAR to enter global suppress mode 1 The associated trigger is enabled for global suppress mode
29–16	Reserved
15–0 LST	Local Suppress Trigger. 0000000000000000The associated trigger is disabled for local suppress mode for all clients xxxxxxxxxxxxxxx1Trigger is enabled for Client 1 local suppress xxxxxxxxxxxxxx1xTrigger is enabled for Client 2 local suppress xxxxxxxxxxxxx1xxTrigger is enabled for Client 3 local suppress xxxxxxxxxxxx1xxxTrigger is enabled for Client 4 local suppress xxxxxxxxxxx1xxxxTrigger is enabled for Client 5 local suppress xxxxxxxxxx1xxxxxTrigger is enabled for Client 6 local suppress xxxxxxxxx1xxxxxxTrigger is enabled for Client 7 local suppress xxxxxxxx1xxxxxxxTrigger is enabled for Client 8 local suppress xxxxxxx1xxxxxxxxTrigger is enabled for Client 9 local suppress xxxxxx1xxxxxxxxxTrigger is enabled for Client 10 local suppress xxxxx1xxxxxxxxxxTrigger is enabled for Client 11 local suppress xxxx1xxxxxxxxxxxTrigger is enabled for Client 12 local suppress xxx1xxxxxxxxxxxxTrigger is enabled for Client 13 local suppress xx1xxxxxxxxxxxxxTrigger is enabled for Client 14 local suppress x1xxxxxxxxxxxxxxTrigger is enabled for Client 15 local suppress 1xxxxxxxxxxxxxxxTrigger is enabled for Client 16 local suppress All other options Reserved

82.4.4.1 Client suppress status register (NAR_CSSR)

Clients can be individually suppressed, leaving the rest of the NAR unaffected. When enabled for local suppress, assertion of one of the suppress triggers causes one or more client receive queues to stop queueing messages. A subsequent pulse of the same trigger causes the client(s) to awaken. Local suppress mode is enabled by setting the appropriate bit in one of the NAR_STCRs. An additional status register, Client Suppress Status (NAR_CSSR), shown in the following figure, can then be read to determine which clients are suppressed, and which are active.

Register index: B

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

Figure 2137. Client suppress status register (NAR_CSSR)

The NAR_CSSR is described in the following table. The set of clients connected to the NAR is device specific. For a list of the clients that are connected to the NAR and their corresponding NAR client index for the local suppress status, see the Calibration and Debug configuration chapter which describes how the modules are configured.

Table 2027. NAR_CCSR field descriptions

Field	Description
31–16 LSS	Local Suppress Status. 0000000000000000No client is in local suppress mode xxxxxxxxxxxxxxx1Client 1 is suppressed xxxxxxxxxxxxxx1xClient 2 is suppressed xxxxxxxxxxxxx1xxClient 3 is suppressed xxxxxxxxxxxx1xxxClient 4 is suppressed xxxxxxxxxxx1xxxxClient 5 is suppressed xxxxxxxxxx1xxxxxClient 6 is suppressed xxxxxxxxx1xxxxxxClient 7 is suppressed xxxxxxxx1xxxxxxxClient 8 is suppressed xxxxxxx1xxxxxxxxClient 9 is suppressed xxxxxx1xxxxxxxxxClient 10 is suppressed xxxxx1xxxxxxxxxxClient 11 is suppressed xxxx1xxxxxxxxxxxClient 12 is suppressed xxx1xxxxxxxxxxxxClient 13 is suppressed xx1xxxxxxxxxxxxxClient 14 is suppressed x1xxxxxxxxxxxxxxClient 15 is suppressed 1xxxxxxxxxxxxxxxClient 16 is suppressed All other optionsReserved
15–0	Reserved

82.4.5 Receive queue client disable register (NAR_CDR)

Disabling a client reallocates its receive queue to the next active client, allowing that memory space to be put to use. The Client Disable Register, shown in the following figure, allows the user to individually disable or enable clients as needed. The status bits also inform the user when the client has new data available.

Register index: C

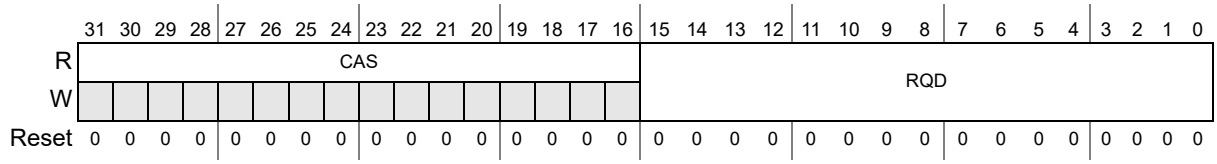


Figure 2138. Client disable register (NAR_CDR)

The NAR_CDR is described in the following table. The set of clients connected to the NAR is device specific. For a list of the clients that are connected to the NAR and their corresponding NAR client index for the client activity status and receive queue disable, see the Calibration and Debug configuration chapter which describes how the modules are configured.

Table 2028. NAR_CDR field descriptions

Field	Description
31–16 CAS	Client Activity Status. xxxxxxxxxxxxxxx1 Client 1 is active with new data xxxxxxxxxxxxxx1x Client 2 is active with new data xxxxxxxxxxxxx1xx Client 3 is active with new data xxxxxxxxxxxx1xxx Client 4 is active with new data xxxxxxxxxxx1xxxx Client 5 is active with new data xxxxxxxxxx1xxxxx Client 6 is active with new data xxxxxxxxx1xxxxxx Client 7 is active with new data xxxxxxxx1xxxxxxx Client 8 is active with new data xxxxxxx1xxxxxxxx Client 9 is active with new data xxxxxx1xxxxxxxxx Client 10 is active with new data xxxxx1xxxxxxxxxx Client 11 is active with new data xxxx1xxxxxxxxxxx Client 12 is active with new data xxx1xxxxxxxxxxxx Client 13 is active with new data xx1xxxxxxxxxxxxx Client 14 is active with new data x1xxxxxxxxxxxxxx Client 15 is active with new data 1xxxxxxxxxxxxxxx Client 16 is active with new data All other optionsReserved
15–0 RQD	Receive Queue Disable. xxxxxxxxxxxxxxx1 Disable receive queue for Client 1 xxxxxxxxxxxxxx1x Disable receive queue for Client 2 xxxxxxxxxxxxx1xx Disable receive queue for Client 3 xxxxxxxxxxxx1xxx Disable receive queue for Client 4 xxxxxxxxxxx1xxxx Disable receive queue for Client 5 xxxxxxxxxx1xxxxx Disable receive queue for Client 6 xxxxxxxxx1xxxxxx Disable receive queue for Client 7 xxxxxxxx1xxxxxxx Disable receive queue for Client 8 xxxxxxx1xxxxxxxx Disable receive queue for Client 9 xxxxxx1xxxxxxxxx Disable receive queue for Client 10 xxxxx1xxxxxxxxxx Disable receive queue for Client 11 xxxx1xxxxxxxxxxx Disable receive queue for Client 12 xxx1xxxxxxxxxxxx Disable receive queue for Client 13 xx1xxxxxxxxxxxxx Disable receive queue for Client 14 x1xxxxxxxxxxxxxx Disable receive queue for Client 15 1xxxxxxxxxxxxxxx Disable receive queue for Client 16 All other optionsReserved

82.4.6 Trace Memory fill level and partition configuration register (NAR_AHFPAR)

This register provides the configuration to set the fill level marker for debug trace memory, connected at the trace memory bus, to generate the partial full event indication and the partition related configuration for main queue (internal central queue). The following figure shows the format of the NAR_AHFPAR.

Register index: E

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

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	AHB01kEN	0	0	0	0	0	0	AHB_01K_Marker			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 2139. Trace memory fill level and partition configuration register (NAR_AHFPAR)

The NAR_AHFPAR is described in the following table.

Table 2029. NAR_AHFPAR field description

Field	Description
31-11	Reserved and must be written with 0s
10 AHB01kEN	Enable 1K Marker Event
9-4	Reserved and must be written with 0s
3–0 AHB_01K_Marker	Value at this field is used to compare a 4-bit slice of internal Trace Memory address bus which works at boundary of 1K. A match with this value generates an event to SPU to indicate the fill status of External Overlay Trace memory. See *Section 82.5.12: Overlay and internal trace memory full status generation* **Note:** This is used in PD as it has memory ranges from 1K to 15K

82.5 Functional description

High-level descriptions of the operation of the NAR and its main functional blocks are given in the following sections.

82.5.1 Reset/Startup

Assertion of global reset causes the NAR operation to immediately cease. All queues and configuration registers are reset. Upon exiting reset, the Nexus Port Controller (NPC) block inside the NAR is disabled. Holding TMS high for five consecutive rising edges of TCK puts TAP controller into the Test-Logic-Reset state regardless of the current state. This applies reset to only TAP controller and JTAG related logic. Applying soft reset through the NAR_CR resets the entire NAR including the TAP controller and configuration registers.

The typical startup procedure for running a trace through the NPC is as follows:

1. Reset the block
2. Upon exiting reset, the Aurora Phy configuration is determined by sampling the configuration pin values from the Nexus Aurora Phy (NAP). If the NAP is enabled, then the NAR is enabled. The Aurora communication is started only when the NAP is ready to accept data. (If any debug lanes are enabled, the Aurora Link begins training at this time.)
3. Debugger/external user sets the NAR enable (NEN) bit in the NAR_CR and configures the NAR. The configuration includes setting up message selection/filtering when two output ports are enabled, enabling timestamp and so on.
4. User programs additional device level debug facilities, such as event-processing, performance-monitoring, and so on.
5. User sets up trace filtering.
6. Enable Nexus transmission in clients.

To enable NAR operation, either the Aurora Link must be enabled at reset, or the user must set the NEN bit in the NAR control register.

Note: The HBDP may already be active by the time the NAR comes up. The NAR only tries to initialize the HBDP if it is currently inactive.

The NAR can be enabled to begin immediate operation, or to start in suppress mode, see *Section 82.5.4: Suppress mode*. Once the NAR and desired ports are enabled, the user should wait until the NAR-ready and HBDP port-ready status bits are set in the NAR status register before beginning Nexus messaging in the clients.

82.5.2 Operation modes

The NAR operating mode is determined by the state of the main control signals: reset and Phy enable. In its enabled state, operation is controlled by its configuration registers. A truth table for all the NAR states is given in this table.

Table 2030. NAR operational modes truth table

Reset	NAR enable	NAR comm disable	Global suppress	HBDP enable	ALT partition	MSP partition	Mode
0	x	x	x	x	x	x	RESET
1	0	x	x	0	x	x	Disabled
1	1	1	x	0	x	x	Trace-Disable
1	x	1	x	1	x	x	Trace-Disable
1	0	x	x	1	x	x	Trace-Disable
1	1	0	1	x	x	x	Suppressed
1	1	0	0	x	Trace Memory bus	x	NAR Transmit
1	1	0	0	1	AUR	x	NAR Transmit
1	1	0	0	1	Trace Memory bus	AUR	NAR Transmit
1	1	0	0	1	AUR	Trace Memory bus	NAR Transmit

There can be overlap between modes as modes are not mutually exclusive.

82.5.2.1 Reset

During reset all internal resources and queues are reset and all output ports are disabled. All inputs are ignored. Upon exiting reset mode, the NAR is in a fully disabled state.

82.5.2.2 Disabled

In the disabled state, all message traffic is ignored, and all output ports are disabled. The NAR is disabled if and only if the NAR enable bit of the NAR control register is not set and the Aurora Link is inactive.

82.5.2.3 Suppressed

The suppressed state differs from the disabled state in that the NAR is on. The NAR pulls out any messages that come up at the clients, keeping the Message Start-End Out (MSEO) bits (so that it can keep track of message boundaries) and discarding the rest.

In the suppressed state, the NAR is waiting for a trigger to begin normal operation. Once that trigger is received, the NAR begins queueing messages that started after the trigger arrived. Incomplete messages that were in transit to the NAR when the trigger arrived are discarded. The NAR can be put into suppressed mode by enabling one of the global suppress states.

82.5.2.4 Trace disable

In trace disable mode, the NAR is powered up but all client-trace is disabled, and no messages are pulled. This mode is also useful for port initialization (to prevent any activity from occurring before the outputs are ready) or for debug/trace-buffer mode, to ensure that, during data retrieval, the NARs queues are stable.

The NAR goes into trace disable (communication-disable) mode when the NAR-enable and NAR communication-disable bits are set or when the HBDP channel is active while the NAR-enable bit is not set.

82.5.2.5 NAR transmit

This is the standard operational mode of the NAR when it is transferring Nexus messages from multiple clients to one or two enabled output port(s). The NAR performs four main operations as described in this table on the incoming messages before transmitting to one of the output ports.

Table 2031. NAR transmit operations

Operation	Description
Data reception	NAR accepts data from each client based on the status of the client receive queue status. The maximum data rate is 8×32-bit data words (where 8 is the number of Nexus message beats in a full entry and 32 is the number of bits in the Nexus word) per cycle per client. MSEO bits of incoming data are monitored to identify end-of-message boundaries.
Input Arbitration	Active receive queue unloads into main queue(s), 8×32 bits per cycle (where 8 is the number of Nexus message beats in a full entry and 32 is the number of bits in the Nexus word). If end-of-message detected, new active receive queue is determined based on client request level or round-robin algorithm.

Table 2031. NAR transmit operations (continued)

Operation	Description
Output Arbitration	If multiple output ports are enabled, route messages from top of queue to appropriate output port based on message type or source. If split queue is used, outputs can run simultaneously, with each sub-queue communicating directly with a port.
Translation	Data going out the Aurora or trace memory bus ports must be translated into the correct format prior to transmission.

Enabling the NAR and one or more output ports without setting the communication-disable bit puts the NAR into transmit mode.

82.5.3 Data reception and arbitration

The NAR supports 10 clients. Each client has a dedicated input port on the NAR and communicates with its associated input port via a dedicated data bus. A data transmission must consist of an integer number of 32-bit Nexus words (32-bit word is called a beat). The number of beats varies from client to client. The number of beats (Bn) for each client are described in the Calibration and Debug configuration chapter which describes how the modules are configured.

There are two types of clients supported at NAR client interface. The client types have their own protocols to send Nexus messages. The only common parameter is the Nexus word size (beat). One Nexus word consists of 30 Message Data Out (MDO) bits plus two Message Start/End Out (MSEO) control bits. Internally, the NAR converts all client interfaces to a common interface. To achieve this, the NAR instantiates interface level gaskets that handle the protocol conversion and clock domain transfer.

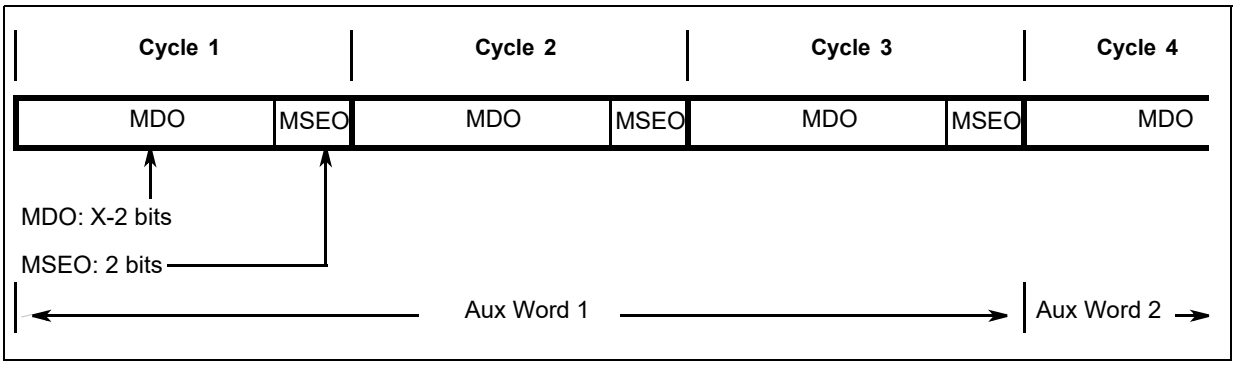
When new data is available in the core, it is transferred over to the asynchronous gasket in the internal NAR clock domain. The gasket then asserts enough data valids to fill internal receive queue, using a receive queue available signal to track available space in the associated queue, under the assumption that the receive queue available signal is initialized with the size of the queue. If the internal arbiter consumes any of its queued entries, it responds with a data accept signal, which increments the receive queue available signal and allows the gasket to send the next piece of data. Under certain circumstances, like receive queue reallocation, it is possible for the internal arbiter to send out surplus data accept signals such that the receive queue available signal can become larger than the size of the queue. If the receive queues are sized correctly, when NAR is actively servicing a particular client, then asserted data accept signals allow the client to continually stream data until the NAR switches to a new client. The data tracking at the gasket assures that it never sends more data than the receive queue can accept and inactive queues are always full and ready for when they are switched to.

The client gaskets must always be in sync with the internal logic of NAR in terms of the number of available entries in each receive queue. For this reason, the internal NPC and its client gaskets need to be reset by the same signal. Any soft-reset of the NAR (via the SRESET configuration bit) is communicated to all gaskets and used to reset every thing inside it.

Each input port has a dedicated receive queue with a minimum of two entries, where an entry is wide enough to store an entire Bn $\times$ 32 data word (where 32 is the number of bits in the Nexus word or the beat). The receive queue can be made larger to allow the NAR to continue transmitting during switches from one client to another in a pipelined environment.

Every cycle, if there is space in a particular receive queue, it latches in one full AUX word from its client. Because the client may be running at a much higher frequency than the NAR, several cycles worth of Nexus messages are contained in a single transaction, as shown in the following figure for a 3-beat per word transaction.

Figure 2140. AUX bus parsing



Each AUX word consists of Bn cycles of Nexus data, along with an optional request-level sideband signal consisting of a priority field (bit 0) and a weight field (bits 1 and 2). Request level priority indicates to the NAR the clients queue status, allowing it to give preferential treatment to clients that are closer to queue overflow. Clients with a set priority bit are always chosen ahead of clients without the bit. For this reason, priority should be used sparingly, only when a client is in danger of overflowing. Request-level weighting also gives a client preferential treatment, but without entirely shutting out the other clients. Normally, the NAR attempts to switch to a new client every time it sees an end-of-message boundary.

As the AUX word is pulled into the receive queue, the MSEO bits are monitored to determine the current Nexus state of the data flow; since Nexus is a variable-length protocol, messages can end in any cycle. If the start message or end message state is detected on any data beat, the beat is marked as containing an end-of-message boundary.

Any received beat which is in the idle state, indicating null data content, is discarded (see the IEEE-ISTO 5001-2003 standard for details of the Nexus protocol). At the same time, if two outputs are enabled, then the source and TCODE fields are extracted from each new message and compared with the user-programmed filters. Messages that generate a match are tagged with an MSP bit, which allows the NAR to route it to the correct port.

Every cycle, the NAR transfers 8 beats of data from the receive queues into the main output queue. Only one port (the active port) is allowed to empty into the main queue at a time: the active port unloads its receive queue into the main queue until the end of the current Nexus message is detected. At that point, a new active port is selected based on the latest request level received at each port. In the case where multiple ports have the same request level (or when request level is not used), then the active port is rotated amongst those ports. If the end of message occurs in the middle of an AUX word, then all the beats up to the end of message are transferred from the old active port, while the rest of the beats come from the new active port. For example, if the message ended at beat two of a three-beat AUX word, then the first two beats into the main queue would come from the original active port, while the last beat would come from the new active port.

82.5.4 Suppress mode

The suppress mode feature enables the user to quickly turn Nexus messaging through the NAR on or off, based on the four dedicated suppress-mode triggers. Suppress mode is only

supported in the PD NAR, it is not supported in the BD NAR. A suppress can either be global, meaning messaging is turned off for all clients, or local, which only applies to selected clients. In suppress mode, the NAR pulls messages from the suppressed client(s), but does not queue or deliver them. After extracting the MSEO bits to keep track of message boundaries, the rest of the message is discarded. This allows the NAR to better keep up with fast-running Nexus clients without having to turn off messaging in the client.

Typically, a client would be suppressed during less interesting periods, and would be awakened when some preset condition is detected. This is communicated to the NAR via the one of the suppress triggers, which can be sourced by a client or an external Event Processing block. When an enter suppress trigger is received, the NAR output queue finishes servicing the current active ports until the end of the message. If the end of the message occurs in a middle beat of a Nexus word, then the remainder of the word is padded with null messages. Once the last message has been loaded into the output queue, the receive queues are all emptied. New messages are pulled from clients as they appear, go through their respective receive queues and then are discarded. In this way, the NAR can naturally monitor the MSEO bits and keep track of each client's Nexus state. In suppress mode, the output ports remain active until the output queue is emptied, meaning there is a delay between the assertion of the suppress trigger and the end of messaging.

When the NAR is in suppress mode and receives a wake-up trigger, it continues discarding any partial messages still in the receive queues. Any message that begins after the trigger is queued normally, insuring that only complete messages are sent to the output queue.

Global suppress mode applies to all clients. Any of the suppress triggers can be programmed to cause the NAR to toggle between suppressed and awake by setting the GST bit of the associated Suppress Trigger Control Register (NAR_STCR). Additionally, the NAR starts up in global suppress mode if the PUS bit of the Control Register is set when the NAR is enabled. All clients are then suppressed until a wake-up trigger is received. If the PUS bit is set without enabling any of the triggers for global suppress, a configuration error results and the NAR powers up in active mode.

Clients can go also into local suppress mode without putting the entire NAR to sleep. Each NAR_STCR has a 16-bit LST field. Setting one or more of these bits causes the trigger assigned to that NAR_STCR to toggle the associated clients between suppressed and awake. All other clients are unaffected. Thus, suppress trigger one can be assigned to one set of clients, while suppress trigger two can be assigned to another set of clients, and different parts of the device can be ignored at different times, all based on different, programmable events. Global suppress always has higher priority than local suppress. That is, if the NAR is in global-suppressed mode, any local suppress wake-up triggers are ignored. Also, if some clients are in local suppress when a global suppress trigger hits, then all go into suppress mode; however, upon the subsequent global wake-up, the clients that were previously in local suppress remain so.

82.5.5 Stall detection

Since the NAR can only switch between clients on an end-of-message boundary, every message that goes into the NAR must complete. If a client happens to die or stall in the middle of transmitting a message, this uncompleted message could cause the NAR to stall, at a time when other clients might still be able to deliver useful information. To prevent this from occurring, the NAR monitors each client for a stall condition, which is defined as occurring when data is not valid in the middle of a Nexus message for a set number of cycles.

When a stall condition is detected, the NAR can do one thing. If the client stalls, then the NAR must complete the current stalled message to continue operating. This scenario is termed a stall error. On a stall error, the NAR injects a dummy message beat into the message stream after the last valid beat of data received from the stalled client. The dummy message is a single beat injected at the output of the receive queue into the main queue. The MSEO bits of the data payload are set to 2b11 regardless of the previous Nexus state. This forces the dummy message into the End Message state. When the NAR injects a dummy message, it also generates an error message to notify the user what has happened.

Injecting a dummy message is an intrusive act, which alters the data stream from the client. If a stalled client is thought likely to recover cleanly, it might be advisable to suppress stall errors to prevent the data stream from being corrupted. This can be done by setting the SED bit of the configuration register, which carries the risk of a truly dead client causing the NAR to hang. Otherwise, the default operation is for the NAR to inject the dummy message and then monitor the client for signs of recovery. If the client does come back online, the NAR attempts to wait until it sees an end-of-message boundary before capturing new data. At best, if the client recovers smoothly, only the interrupted message is lost; if the client does not recover smoothly, it might not be possible for the client and NAR to get back in sync in terms of their Nexus states, which would lead to nothing but garbled data coming out. For this reason, it is also possible to prevent the NAR from trying to recover from a stall error, by setting the SRD bit of the NAR_CR. If the SRD bit is set, a stalled client is not allowed to recover until the SRD bit is unset or the NAR is reset.

82.5.6 Output arbitration

The NAR allows two output ports to be active at the same time. This feature is only available on the BD NAR. Typically, this feature is only useful if one of the ports is used to service messages of a specific type or messages from a particular client, or both, while the other port supports all the rest of the message traffic. For example, the NAR could be set up to send in-circuit trace (ICT) traffic out the HBDP and to reserve the trace memory bus port for all other Nexus messages.

If two output ports are enabled, one of them must be designated as the Message Select Port (MSP). The MSP is the destination for those messages that satisfy a set of user-defined requirements as to the source and type of message. All other traffic is sent to the other enabled port. Ports are enabled by setting the port enable bits in the BD NAR_CR. The MSP is designated by setting bits [10–9] and ALT is designated by setting bits [7–6]. Once the MSP has been specified, other bits in the NAR_CR, along with the Source Filter Register (NAR_SFR) and Type Filter Register (NAR_TFR), are used to determine which messages go to the MSP and which do not. These facilities can be used to build a logical expression that can filter messages based on up to four types or four sources, or both.

The NAR_SFR can be programmed with up to four source codes, while the NAR_TFR can hold up to four message-type codes. Each filter also includes an enable bit to turn it on or off. A source match occurs if a messages source code matches one of the source filters or (if the SFI bit is set) if it does not match any of the source filters. Similarly, a type match occurs if a messages type code matches one of the type filters or (if the TFI bit is set) if it does not match any of the type filters. Finally, the ACC bit of the NAR_SFR determines when both source and type filters are used, whether the two are ANDed together (ACC = 0)

or ORed together (ACC = 1). For example, to send ICT messages out the HBDP and all other traffic to the trace memory bus port:

- Set ALT = 10 in the NAR_CR—sets trace memory bus port as the alternate port
- Set MSP = 01 in the NAR_CR—sets HBDP as the message select port
- Set FE0 = 1 & TF0 = 100010 (0x22) in the NAR_TFR—enables type filter 0 to match ICT messages

To send everything but data trace from source 13 to the trace memory bus port, while all other traffic goes to the HBDP:

- Set ALT = 01 in the NAR_CR—sets HBDP as the alternate port
- Set MSP = 10 in the NAR_CR—sets trace memory bus port as the message select port
- Set TFI = 1—inverts type-filter logic
- Set FE0 = 1 & TF0 = 000101 (0x05) in the NAR_TFR to select data trace of size 8 bits
- Set FE1 = 1 & TF1 = 000110 (0x06) in the NAR_TFR to select data trace of size 16 bits
- Set FE2 = 1 & TF2 = 001101 (0x0d) in the NAR_TFR to select data trace of size 32 bits
- Set FE3 = 1 & TF3 = 001110 (0x0e) in the NAR_TFR to select data trace size of 64 bits
- Set FE0 = 1 & SF0 = 001101 (0x0d) in the NAR_SFR to select source trace ID = 13
- Set ACC = 0 in NAR_SFR so the final filter is ~(data_trace) AND (source = 13)HBDP

Data is written into the output queue at a maximum rate of one entry per cycle. If the current messages source and type pass the filters, then the message is marked for the MSP. When the entry is later pulled from the queue, it is then sent to the MSP. Unmarked entries are sent to the alternate port. Message destination must be recalculated on every end-of-message boundary, using the fixed-width SRC and TYPE fields at the beginning of each Nexus message.

82.5.7 Message translation and formatting

The NAR performs the formatting on Nexus messages before sending them to an output port. The exact translation performed depends on the destination port.

82.5.7.1 Aurora format

Nexus messages are pre-translated before sending to the HBDP using the Aurora protocol. The NAR output is a stream of 16 $\times$ 4-bit words (where 4 is the width of the NAR Aurora out port) that are further encoded by an external Aurora block before being sent for transmission on the Aurora (HBDP) port.

Entries from the output queue are chopped into chunks of 4 $\times$ 16-bit words. The amount of data that can be transmitted per Aurora cycle depends on the number of lanes available in the HBDP (transmitted HBDP word = 16 $\times$ number of lanes).

82.5.7.2 Trace memory bus format

The trace memory bus works on AHB-EW protocol. Data destined for the trace memory bus port is split back out into 32-bit beats (Nexus word). Depending on the relative sizes of the beat and the width of the trace memory bus port, one or more Nexus words are packed into a trace memory bus data word, with any remaining bits padded to 0.

For example, if the size of the beat = 32 and the width of the trace memory bus = 64, then each trace memory bus data packet contains two beats of Nexus data. Since the data is stored in a regular format (MSEO bits in same place at every location), this makes data

interpretation straightforward, regardless of where the trace memory bus port is sending the data. The trace memory bus translation block performs the necessary accumulation and resizing in the likely case where the trace memory bus width does not equal the queue entry size.

82.5.8 Aurora interface

The external Nexus Aurora Link (NAL) contains all of the link-layer, PCS, and control logic for the HBDP channel. The Aurora data is transmitted from the NAR to the NAL for sending to the HBDP. Typically, the NAL powers up coming out of reset, but it can also be enabled by the NAR by turning on the NAR with the HBDP enabled. The reverse is true as well. If the NAL is active while the NAR is disabled, then the NAR powers up to communication-disabled mode.

Data flow between the NAR and the NAL uses a simple, synchronous interface. If the NAL is active, it indicates this by a ready signal from the NAL. If Aurora is configured as output it is indicated by a signal from the NAR.

The Aurora interface can either be operated in framing or streaming mode. Streaming mode provides the highest performance and lowest communication overhead through the link. Whereas framing mode provides greater robustness and improved error-handling at the receive side of the link. When framing mode is enabled, the NAR encapsulates the Aurora data in frames consisting of 256 Aurora chunks; that is, after every 256 Aurora transmissions, the NAR asserts end of frame and start of frame to mark the end of the last frame and the beginning of the next frame. In either mode, the NAR always asserts a start-of-frame when beginning transmission, and an end-of-frame when its output queue empties, to notify the channel partner of the end of valid data.

82.5.9 Trace memory bus output port usage

The NAR can communicate directly to external memory by sending Nexus data out onto the trace memory bus port, using the AHB-EW (Early Write) protocol. This provides a convenient, secondary area for data in applications or in situations where the HBDP is not available. The trace memory bus access are generated internally by trace memory bus controller.

Because in many systems it is desirable to have the trace memory bus interface run at a divided clock rate relative to the main platform clock, the trace memory bus port operates off of a separate AHB clock. This clock is assumed to be free-running (that is unaffected by NAR-enable and the status of the Aurora Link), synchronous to the platform clock used by the rest of the NAR, and either running at the same rate as the platform clock or an integer divide of that rate.

The trace memory bus interface protocol is a slightly modified version of AMBA AHB specification. The write data comes in the address phase instead of data phase and it is termed as Early Write (EW). When NAR uses the trace memory bus port as a destination for Nexus data, it performs a series of writes to a predefined block of memory space.

This memory space is specified through the trace memory bus base address and trace memory bus max transfer configuration registers. The size of the block is set by the max transaction count (MXFR field in NAR_TCR) times the size of a transaction (trace memory

bus-width, 64-byte, 128-byte, 256-byte, or 384-byte, depending on block transfer mode and block transfer size). For example, if:

- Base address = 0x0D00_0000 (that is TBAHI = 0x0, TBALO = 0x0D00_0000)
- NAR_TCR[BTM] = 1 (Block-transfer mode enabled)
- NAR_TCR[BTS] = 00 (Block-transfer size = 64-byte)
- NAR_TCR[MXFR] = 0x100 (max transaction count)

then the total block size would be 0x100 $\times$ 0x40 = 0x4000 bytes. The dedicated debug address range would then be 0x0D00_0000 – 0x0D00_3FFF.

The NAR can either write to the debug space one (trace memory bus-width) word at a time, or for improved performance, it can use a block transfer. If the BTM bit of the trace memory bus Control Register (NAR_TCR) is set, then the NAR uses block transactions to send data through the trace memory bus port, where the size of the block is determined by the setting of the BTS field of the NAR_TCR.

In block transfer mode, the NAR waits until the main queue partition dedicated to the trace memory bus port reaches a specified threshold point, determined by the BTH field of the NAR_TCR, before initiating a block transfer. The default threshold is the size of the block transfer. That is, a block transfer does not start until there is enough data in the associated partition to fill the block. But it might be desirable, for bandwidth reasons or because the associated partition is smaller than the block transfer size, to begin the transfer sooner. Hence, based on BTH, a transfer can begin when the partition is at 25%, 50%, or 75% of the block transfer size. This capability should be used with caution. If the Nexus client does not generate enough new data during the transfer to fill the block, then a block transfer error results.

The NAR generates Nexus trace write transactions for the trace memory bus port using the address from the base address register for the first transaction and then incrementing through the memory space. Once the programmed max number of transactions have been sent, the NAR sets the trace memory bus overflow status bit and, if desired, asserts the NAR event-out. At this point, the NAR can either shut down the trace memory bus port for trace, or, if the TBW bit of the NAR_TCR is set, reset the address to the beginning of the debug space and continue sending trace data. This allows the debugger to view the most recent data, in cases where the buffer overflows.

When TBW bit of NAR_TCR is not set then NAR shuts down the trace memory bus port. To restart the trace memory write again from trace memory base address, the user should write either the NPC trace memory bus base address (TBALO/TBAHI) or the transaction count (MXFR). The user does not have to actually change the value, but can write the same target base address again. The NAR sees any write to one of these registers as a reconfiguration of the trace memory bus port, and resets its internal address pointer to the base address along with clearing the target-space-full flag.

82.5.10 Internal message generation

Generally, the NAR simply conveys Nexus messages from client to port, and does not generate messages of its own. There are, however, certain situations when the NAR needs to communicate directly with the debugger. Internally generated messages are added to the queue the same as external messages. If the queue is partitioned, then internal messages are always channeled to whichever partition is dedicated to the HBDP.

The NAR generates the following message types:

- Device_ID—The first message transmitted when the NAR comes up. It consists of a 6-bit TCODE (TCODE = 1) and the 32-bit device ID. The first beat of the message is always an IDLE (MSEO = 2b11, MDO = all 0s) to allow the debugger Nexus state machine to correctly initialize (by moving to end message or idle state). Device_ID is also generated if the queue is repartitioned (by changing either the ALT or MSP setting in the NAR_CR) while it is not empty. This serves the dual purpose of insuring that the user always gets at least one device_ID message and providing a message boundary in the case where there might have been corruption due to the lost data.
- Watchpoint—The NAR generates its own watchpoint match message under the following circumstances as shown in *Table 2035*.
- NAR_Error (TCODE = 8)—If an error condition is detected inside the NAR, it generates its own error message. An NAR_Error message consists of the 6-bit TCODE, the 6-bit NAR source ID, and a 32-bit data field which corresponds to the NAR status register. An NAR error can be triggered by:
  - HBDP error
  - Illegal configuration
  - Stall Error
  - Trace memory bus read/write access error

This table describes the NAR message formats.

Table 2032. NAR message formats

Message name	Min packet size (bits)	Max packet size (bits)	Packet type	Packet name	Packet description
Device ID	6	6	fixed	TCODE	TCODE value = 1
	32	32	fixed	DID	DID register value
	0	24	variable	TSTAMP	Optional, globally synchronized timestamp
Error Message(1)	6	6	fixed	TCODE	TCODE value = 8
	6	6	fixed	SRC(2)	NAR source ID
	4	4	fixed	ETYPE	See *Table 2033*
	12	12	fixed	ECODE	See *Table 2034*
	0	24	variable	TSTAMP	Optional, globally synchronized timestamp
Watchpoint Hit Message(3)	6	6	fixed	TCODE	Value = 15
	6	6	fixed	SRC(2)	NAR source ID
	6	6	fixed	WPHIT	See *Table 2035*
	0	24	variable	TSTAMP	Optional, globally synchronized timestamp

1. The Error message conforms to the IEEE-ISTO 5001-2011 standard.
2. Tools should treat the NAR SRC field as a 4-bit SRC field and a 2-bit unused field. The extra 2-bit field (the two most significant bits of the 6-bit TCODE) is always 0b00. This operation is not compliant to the IEEE-ISTO 5001 Nexus standard.
3. The Watchpoint Hit message does not conform to the IEEE-ISTO 5001-2011 standard, it conforms to the IEEE-ISTO 5001-2003 standard. The WPHIT should be a variable length packet, but is implemented as a fixed length packet. In most cases, tools will not see a difference, however, a difference will be seen if a timestamp field is appended to the message.

82.5.10.1 Error conditions

NAR errors can occur upon events such as illegal memory accesses, block-transfer overloads, communication errors, or when the NAR is configured incorrectly. If the configuration registers are programmed incorrectly, it is possible to put the NAR into an illegal state. The NARs response varies depending on the error. For a minor error, the NAR reverts to a default mode and continues operation, while major errors result in complete shutdown.

In either case, a Nexus error message is sent to the Aurora out port, if its enabled, otherwise to the trace memory bus port. The Nexus error message contains both ETYPE and ECODE fields:

- ETYPE is used to denote the basic type of error that has occurred
- ECODE carries specific additional information about the error

All NAR error types are defined in this table.

Table 2033. NAR error types

ETYPE	Description	Classification	NAR action	Comment
0000 (h'0)	Receive queue overrun	Minor	Dropped message	If a client sends a data word that the NAR cannot accept (due to a full receive queue), the new data word is dropped.
0001 (h'1)	Internal message contention	Minor	Dropped message	Internal errors/watchpoints occurred faster than NAR could generate messages for them, causing lower-priority message(s) to be dropped. The type of message dropped is conveyed in the ECODE field, see *Table 2034*.
0011 (h'3)	Trace memory bus port read/write access error	Minor	Access fails	Unable to send an Aurora-to-trace memory bus request. Typical causes would be writes larger than the data buffer or sending in a new request before the previous one completes.(1)
0101 (h'5)	Invalid NRR	Minor	Access fails	Tried to access an un-implemented NRR.
1001 (h'9)	Config error	Major	See *Table 2034*	The specific configuration error is conveyed in bits 0–3 of the ECODE field (*Table 2034*), as well as the NCE field of Status Register.
1100 (h'C)	HBDP error	Minor	N/A	A hard error is automatically reset the Aurora Link and causes it to re-initialize. Although the Link is down, the error message is queued and released as soon as the Link comes back up.
1110 (h'E)	Block transfer error	Minor	Dummy fill	If the Trace memory bus port becomes data-starved during a block transfer, it fills the rest of the block with IDLE messages.
1111 (h'F)	Stall error	Minor	—	—

1. In the case where a request was successfully translated to a Trace memory bus transaction which itself returned an error, that is conveyed in the STATUS field of the Nexus Target Response message subsequently returned to the debugger.

Typically, the ECODE field is used to communicate which type of message was dropped in queue overflow or message contention situation, but for the NAR it is also used to denote the type of configuration error that occurred. If a configuration error is detected, the NAR generates a Config Error Nexus message and transmits it, with the actual type of error encoded in the four MSBs of ECODE, as well as the NCE field of the Status Register. The ECODE field encodings are listed in this table.

Table 2034. Configuration error codes

ECODE	ETYPE	Description	NAR action	Comment
0000xxxxxxx1	0001	Watchpoint message dropped	N/A	Occurs with internal message overflow error.
0000xxxxxx1x	0001	Data trace message dropped	N/A	Occurs with internal message overflow error; MMA response message was dropped.
00001xxxxxxx	0001	Response message dropped	N/A	Occurs with internal message overflow error; target response (memory access or NRR) message was dropped.
0000xxx1xxxx	0001	Status message dropped	N/A	Occurs with internal message overflow error.
000100000000	1001	Config error: multiply defined port	Communication disable mode	Both MSP and ALT were assigned to the same port. NPC waits until the configuration is fixed before proceeding.
001000000000	1001	Config error: no ALT defined	Tracebuffer	An MSP was defined without defining the ALT. The NPC reverts to tracebuffer mode.
001100000000	1001	Config error: cannot enable port	Communication disable mode	User has attempted to enable a port that does not exist in this implementation of the NPC.
010000000000	1001	Config error: cannot support message filtering	Message filtering and/or queue partitioning disabled	User has attempted to enable message filtering or queue partitioning with only one output enabled. Filtering/partitioning is turned off.
100100000000	1001	Config error: no wakeup trigger	Power up in active mode	Occurs if none of the defined clients in the system can cause a wakeup (PUS is set in NAR_CR, but no NAR_STCRs have WUT set).

82.5.10.2 Watchpoint messages

The NAR generates its own watchpoint match message under the circumstances shown in the following table. The value of the Watchpoint Hit field of the Watchpoint Message is also shown in this table.

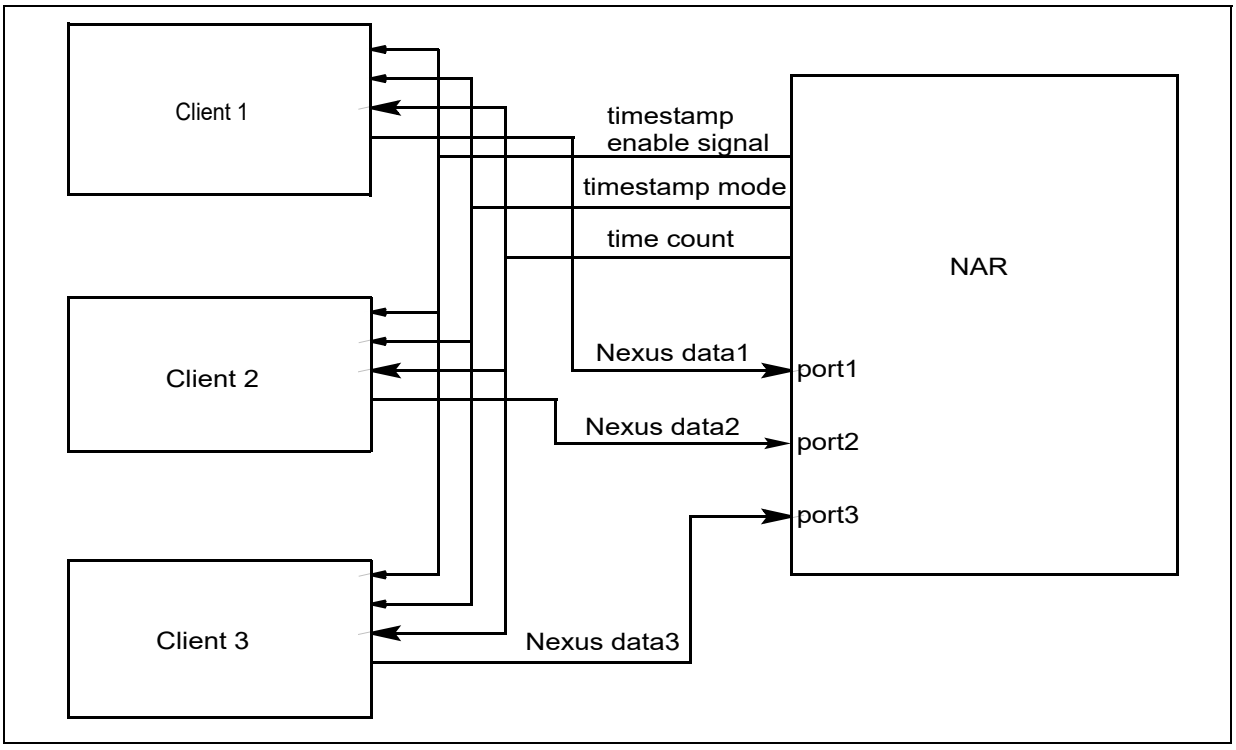
Table 2035. Watchpoint Hit values

WPHIT	Description	Comment
b010000	Entering Suppress Mode	If (local or global) suppress mode is enabled and the appropriate trigger is received, the NAR generates a watchpoint message. The type of event (in this case, a suppress event) is encoded in the WPHIT field of the watchpoint message (see the Nexus standard, sec. 5.3.21).
b100000	Client Sync	If sync messaging is enabled, the NAR generates a sync watchpoint message whenever a client sync event occurs.
b000100	Trace Memory bus Block Overflow (TBO)	If the trace memory bus block is filled, causing it to either disable trace memory bus data trace or wrap back to the beginning of the dedicated debug memory block and if trace memory bus overflow messaging is enabled, then an TBO message is generated.
b001000	Tracebuffer Wrap	If the NAR is being used in tracebuffer mode with the wrap-around feature, then a watchpoint message is generated every time the tracebuffer wraps back to address 0. This would be useful in informing the user that the buffer had wrapped and that older data had been overwritten.

82.5.11 Timestamp function

The NAR and all of its clients are required to transmit timestamps, to allow precise ordering of events. All clients synchronous or asynchronous uses NAR timer value broadcasted over a bus, this ensures that all timestamps are in sync as shown in *Figure 2141*. The NAR uses enable signal to centrally control the enabling of time stamping, which informs the clients to start adding the timestamp field to their message traffic.

Figure 2141. Timestamp architecture



Timestamping is enabled by setting the TEN bit of the NAR_CR. This causes the NAR to assert the time stamp enable signal to clients. Typically, the user would also set the TRS bit at this time to reset the NAR Time stamp counter. Internally the NAR has a 30-bit counter external to the NPC block. This 30-bit counter is broadcast to all clients after encoding its value in Gray code. This minimizes the error in synchronization by the clients. The NPC uses an additional internal 24-bit counter to send the timestamp inside its messages. The client converts the gray coded timestamp to binary to append to the Nexus messages.

The Time stamp mode, a 2-bit field, is broadcasted to all client which tells the granularity of sending time stamps into message. This is controlled by the TSG bits of the NAR_CR.

Asynchronous clients, those on a separate clock domain from the NAR, synchronize the enable, mode and timestamp value in their domain. The error introduced by this is assumed to be within tolerable range by application.

A timestamp counter reset can be forced by causing a 0 to 1 transition in the TRS bits of the NAR_CR. The NAR deasserts the enable signal to clients while the reset occurs, and then the counter is restarted (assuming the TEN bit is set). Data throughput can be increased (at the loss of timestamp granularity) by controlling the TSG bits of the NAR_CR. Rather than adding a timestamp to every message, clients can be told to only stamp every 4th, 16th, or 32nd message, easing the bandwidth requirements on the output ports.

The timestamp block maintains the timestamp counter for NAR. It is the central time source for all synchronous/asynchronous clients.

82.5.12 Overlay and internal trace memory full status generation

In certain events, such as main memory (internal trace memory) full or external overlay trace memory full, it is required to control the clients to stop sending data, to avoid any loss of messages from the clients.

The NAR provides three outputs (triggers) to indicate three different situations as described below:

- Internal trace memory (the main queue) is full
- External debug memory on trace memory bus is full
- External debug memory on trace memory bus is full up to a programmable watermark set by the configuration register

The last two triggers are used by SPU to generate stall and suppress triggers. Typically the stall request should be generated whenever space in the external memory connected to trace memory bus has been exhausted. The NAR starts writing from the first location (wrap) as specified in trace memory bus address register in this condition if overwrite trace memory bus debug memory (TBW bit) is set in the NAR_TCR.

The user should configure the partial full watermark register such that it can give around 90% full indication of external debug memory on the trace memory bus. The emulator tool memory full indication is typically generated from the DCI block which actually communicates with the tool.

For example, the following steps can be used to generate event at around 90% fill level for trace memory connected in AHB-EW and to set the watermark in the configuration register.

- From the configuration, the programmer knows the following:
  - AHB start (base) address—Base address = 0x0D00_0000 (that is TBAHI = 0x0, TBALO = 0x0D00_0000
  - Block transfer size—NAR_TCR[BTM] = 1 (Block-transfer mode enabled)
  - Max transfer count—NAR_TCR[MXFR] = 0xC0 (maximum transaction count)
- Thus, the total block size is (0xC0 $\times$ 0x40) = 0x3000 bytes. The dedicated debug address range is then 0x0D00_0000 – 0x0D00_2FFF.
- To generate an event at nearly 90% fill level of the trace memory at 1 KB boundary, the value to be programmed in the register NAR_AHFPAR[3:0] is calculated as:
  - Number of 1K blocks in 0x3000 bytes = (3000 right shift by 10 bits) = C = 12
  - So the value to be programmed should be 11 as 90% of 12 is almost 11.
- NAR_AHFPAR[3:0] = (NAR_TBALO[13:10] + 0xB) = (0x7+0xB) 0x2 (upper bits truncated due to width) and enable this with NAR_AHFPAR[10] = 1.

This assumes that NAR starts writing from base address and stops at maximum address and no reading is done in overlay memory trace data area during this operation. Whenever the address bus bits [13:10] match with this value, the partial event is generated (toggled).

82.5.13 Configuration/debug interface JTAG (1149.1)

The configuration and status registers of the NAR are accessed through the standard IEEE 1149.1 JTAG interface. The JTAG frequency (TCK) must be 1/2 of the NAR operating clock. The access mechanism through JTAG is described in this section.

The NAR block uses the IEEE 1149.1-2001 TAP for accessing registers. TAP signals include TCK, TDI, TMS, and TDO. The NAR implements a TAP controller state machine that transitions based on the state of the IEEE 1149.1-2001 16-state state machine. It also implements Nexus controller state machine as defined by the IEEE-ISTO 5001-2001 standard.

The instructions implemented by the NAR TAP controller are listed in *Table 2036*. Each unimplemented instruction acts like the BYPASS instruction. The size of the NAR instruction register is 4 bits.

Figure 2142. Nexus controller state machine

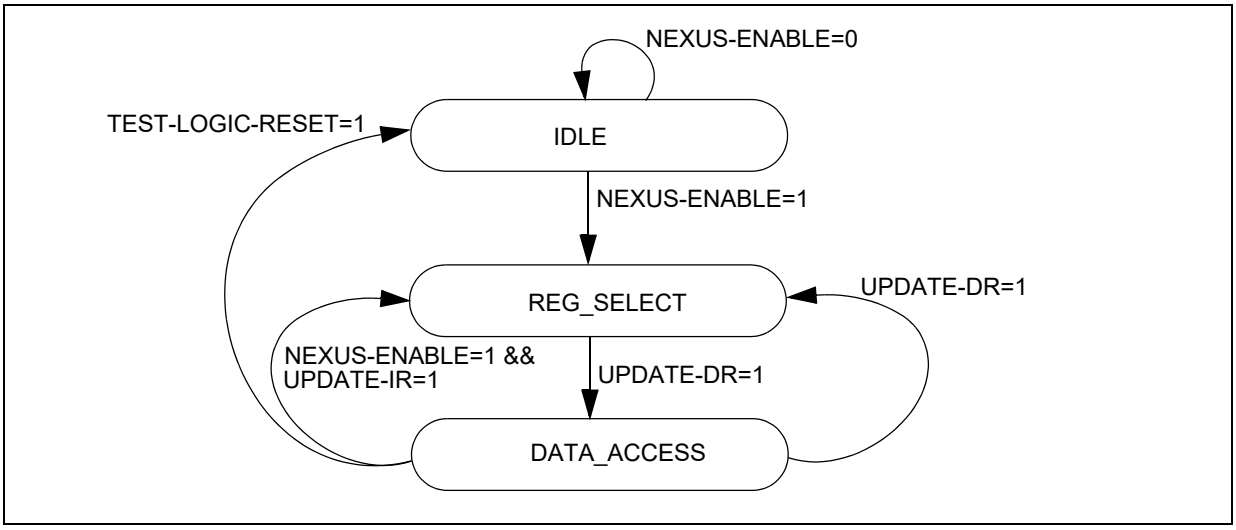
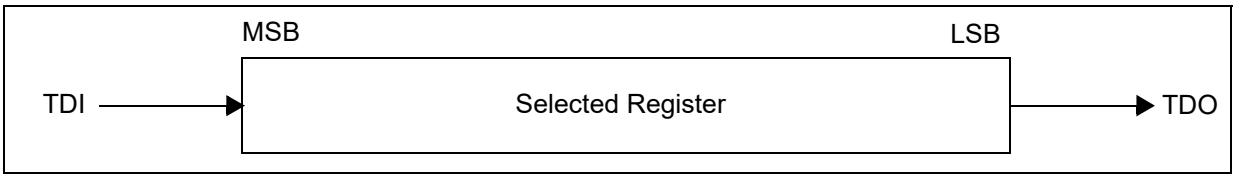


Table 2036. Implemented instructions

Instruction Name	Private/ Public	Opcode	Description
NEXUS_ENABLE	Public	0x0	Activate Nexus controller state machine to read and write NAR registers.
BYPASS	Private	0x0F	NAR BYPASS instruction. Also the value loaded into the NAR IR upon exit of reset.

Data is shifted between TDI and TDO starting with the least significant bit as illustrated in the following figure. This applies for the instruction register and all Nexus tool-mapped registers.

Figure 2143. Shifting data into register



82.5.13.1 Enabling the NAR TAP controller

Assertion of the power-on reset signal resets the NAR TAP controller. When not in power-on reset the NAR TAP controller is enabled by driving JCOMP with the NAR enable value and exiting the Test-Logic-Reset state. Loading the NEXUS-ENABLE instruction then grants access to Nexus debug.

82.5.13.2 Loading NEXUS-ENABLE instruction

Access to the NAR registers is enabled when the TAP controller instruction register is loaded with the NEXUS-ENABLE instruction. This instruction is shifted in via the SELECT-IR-SCAN path and loaded in the UPDATE-IR state. At this point, the Nexus controller state machine, shown in *Figure 2142*, transitions to the REG_SELECT state. The Nexus

controller has three states: idle, register select, and data access. The following table illustrates the IEEE 1149.1 sequence to load the NEXUS-ENABLE instruction.

Table 2037. Loading NEXUS-ENABLE instruction

Clock	TMS	IEEE 1149.1 state	Nexus state	Description
0	0	RUN-TEST/IDLE	IDLE	IEEE 1149.1-2001 TAP controller in idle state
1	1	SELECT-DR-SCAN	IDLE	Transitional state
2	1	SELECT-IR-SCAN	IDLE	Transitional state
3	0	CAPTURE-IR	IDLE	Internal shifter loaded with current instruction
4	0	SHIFT-IR	IDLE	TDO becomes active, and the IEEE 1149.1-2001 shifter is ready. Shift in all but the last bit of the NEXUS-ENABLE instruction.
5 TCKS				
12	1	EXIT1-IR	IDLE	Last bit of instruction shifted in
13	1	UPDATE-IR	IDLE	NEXUS-ENABLE loaded into instruction register
14	0	RUN-TEST/IDLE	REG_SELECT	Ready to be read/write Nexus registers

82.5.13.3 Selecting a JTAG register to create a bus (memory or peripheral) access

When the NEXUS-ENABLE instruction is decoded by the TAP controller, the input port allows the development tool access to all NAR registers. Each register has a 7-bit address index.

All register access is performed via the SELECT-DR-SCAN path. The Nexus controller defaults to the REG_SELECT state when enabled. Accessing a register requires two passes through the SELECT-DR-SCAN path: one pass to select the register and the second pass to read/write the register.

The first pass through the SELECT-DR-SCAN path is used to enter an 8-bit Nexus command consisting of a read/write control bit in the LSB followed by a 7-bit register address index, as illustrated in *Figure 2144*. The read/write control bit is set to 1 for writes and 0 for reads. The index for registers are shown in *Table 2018*.

Figure 2144. Nexus command format for register selection

MSB	LSB
7-bit register index	R/W

The second pass through the SELECT-DR-SCAN path is used to read or write the register data by shifting in the data (LSB first) during the SHIFT-DR state.When reading a register, the register value is loaded into the IEEE 1149.1-2001 shifter during the CAPTURE-DR state.When writing a register, the value is loaded from the IEEE 1149.1-2001 shifter to the register during the UPDATE-DR state. When reading a register, there is no requirement to shift out the entire register contents. Shifting may be terminated once the required number of bits have been acquired.

This table illustrates a sequence which writes a 32-bit value to a register.

Table 2038. Writing a 32-bit value to a register

Clock	TMS	IEEE 1149.1 State	Nexus State	Description
0	0	RUN-TEST/IDLE	REG_SELECT	IEEE 1149.1-2001 TAP controller in idle state
1	1	SELECT-DR-SCAN	REG_SELECT	First pass through SELECT-DR-SCAN path
2	0	CAPTURE-DR	REG_SELECT	Internal shifter loaded with current value of controller command input.
3	0	SHIFT-DR	REG_SELECT	TDO becomes active, and write bit and 4 bits of register index shifted in.
7 TCKs				
11	1	EXIT1-DR	REG_SELECT	Last bit of register index shifted into TDI
12	1	UPDATE-DR	REG_SELECT	Controller decodes and selects register
13	1	SELECT-DR-SCAN	DATA_ACCESS	Second pass through SELECT-DR-SCAN path
14	0	CAPTURE-DR	DATA_ACCESS	Internal shifter loaded with current value of register
15	0	SHIFT-DR	DATA_ACCESS	TDO becomes active, and outputs current value of register while new value is shifted in through TDI
31 TCKs				
47	1	EXIT1-DR	DATA_ACCESS	Last bit of current value shifted out TDO. Last bit of new value shifted in TDI.
48	1	UPDATE-DR	DATA_ACCESS	Value written to register
49	0	RUN-TEST/IDLE	REG_SELECT	Controller returned to idle state. It could also return to SELECT-DR-SCAN to write another register.

82.5.14 Trace memory bus controller

The purpose of the trace memory bus is to transfer trace data from main queue to external overlay memory. This port works on a modified AHB protocol that supports early write. It has AHB interface signals as defined in AMBA specification (Rev 2.0). In addition, there are side-band signals for added functionality.

Key features of the module include:

- AHB-EW Interface
  - Early Write (for higher BW in special module)
  - Configurable 32/64-bit address bus width
  - 64-bit data bus width
  - Transfer Types
  - IDLE, NONSEQ, SEQ
  - Single Transfers
  - Burst Transfers
  - Configurable support for INCR
  - Configurable support for INCR4, INCR8, INCR16, WRAP4
  - Response Type
  - OKAY, ERROR
  - Transfer Size
  - Byte, Half Word, Word, Double-Word
  - Protection Control
  - Cacheable/ Non Cacheable Access Control
  - User/Privilege Access
- Endianess Support
  - Configurable Little Endian/Big Endian support on AMBA-AHB interface
- Pipeline Support
  - Total of two transactions (One plus one ongoing)

82.5.14.1 Operation

The trace memory bus controller allows all read/write operations to be cut through. The controller has a pipeline depth of two. While the data phase of one transaction is ongoing, it can accept one more read/write request from the internal bus master and initiate the address phase on the AHB interface during the last data cycle of the ongoing transaction.

82.5.14.2 Write operation

When the trace memory bus initiator asserts a request for a write operation, it can find the acknowledge signal parked asserted if the controller is ready to accept the transaction. If the pipeline is full or in case of an ongoing read transaction, the controller inserts wait states by keeping the acknowledge low. The trace memory bus controller initiates the address phase on AHB interface once it accepts the transaction by asserting the acknowledge on the trace memory bus interface. The write data from the trace memory bus initiator is accepted by the gasket by asserting the acknowledge only on completion of address phase at AHB interface, which is signified by receiving the ready signal asserted. Simultaneously, the data is sent to the data bus. Once the requisite number of bytes have been accepted from the trace memory bus initiator, the gasket generates end of data and transfer according to the Trace memory bus protocol.

The AHB protocol supports data transactions of byte, half-word, word and double-word (when both trace memory bus and AHB data bus width is 64 bits). A write transaction from the trace memory bus initiator take place as single/multiple transactions on the AHB side.

82.5.14.3 Read operation

When the trace memory bus initiator asserts a request signal to request for a read operation, it can find the acknowledge signal parked asserted if the gasket is ready to accept the transaction. If the pipeline is full, the controller inserts wait states by keeping the acknowledge low. The trace memory bus controller initiates the address phase on AHB-EW interface once it accepts the transaction by asserting the acknowledge on trace memory bus interface. The controller reads the data from the bus when the ready signal is sensed asserted. The data is transferred on the trace memory bus data bus with the data acknowledge asserted. The gasket asserts end of data and transfer at the end of transaction as per trace memory bus protocol.

The AHB protocol supports data transactions of byte, word, half-word and double-word (when both trace memory bus and AHB data bus width is 64-bits). A read transaction from trace memory bus initiator takes place as single/multiple transaction on the AHB interface before the read data is merged and sent on the trace memory bus read data bus.

In case of variable data bus width on trace memory bus and AHB interface (32-bit data bus on one interface and 64-bit data bus on the other interface), a maximum of 64 bits is read on each beat.

82.5.15 Legacy client interface controller

This block converts the legacy client protocol into internal client protocol that is accepted by the internal NPC block. The Nexus beat size must be same for both interfaces. The Nexus beat is composed of MDO bits along with two MSEO bits. The trace data width from the legacy client is fixed at one beat. The architecture of this block is shown in *Figure 2145*.

The legacy client sends each message after arbitrating the bus. It requests the gasket with a 2-bit request signal whenever it has data to send. A 00 means no request, 01 a low-priority request, 10 a medium-priority request, and 11 a high-priority request. The legacy client can send one maximum message sized data on receiving a grant from the block. There is no backpressure mechanism during transfer of a message from the client to this block. The block asserts a grant if it has at least one max message size space in its FIFO. Seeing the grant the client asserts busy signal and holds it until it wants to send data, it may hold it up to max message size data is transferred.

The following architecture describes the legacy client to internal client protocol conversion block. There is a small FIFO of paramaterizeable depth used to store the data from client. The block asserts a grant only when at least one max message sized space is available in FIFO. The write happens in the client's clock domain. The write pointer is passed to other domain which is NPC clock domain.

The logic on other side compares read and write pointer to check the data availability in the block's FIFO and space availability in NPC's receive queue. If space and data both are available it tries to send message to the NPC.

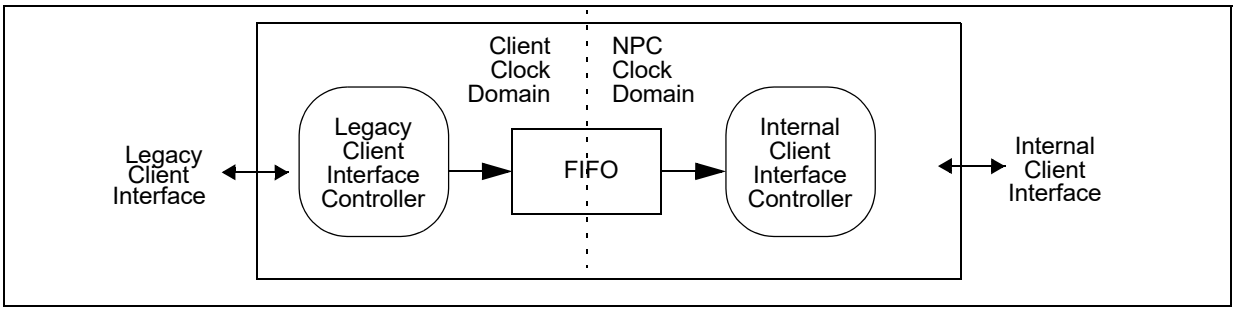


Figure 2145. Legacy client Interface controller

The size of the receive queue of the corresponding client at NPC must be known to the block. The block drives a not empty signal whenever it has data in FIFO and looks for a ready signal. It can send as many Nexus beats as receive queue can store. It operates on decrements a counter after sending data to the NPC so that it can track how much data it can send. The counter is incremented by the value of acknowledge from NPC which tells how may spaces are freed in receive queue.

82.6 Initialization information

The NAR is initialized by asserting the block reset signal for a minimum of one cycle of the slowest clock. Reset clears all latches to 0, with the only exception being the output queue, whose initialization state depends on the reset behavior of the compiled memory used. Upon exiting reset, the NAR is in the disabled state, as described in *Section 82.5.2.1: Reset*.

Pointers and state machines are initialized by the starting of NAR functional clocks (which occurs either upon the setting of the NAR enable bit or upon the assertion of the Phy enable signal) or by an NAR soft reset.

82.7 Application information

The base application for the NAR is collecting and collating Nexus trace data. Programming examples for how to set up the NAR for trace are given in *Section 82.5.6: Output arbitration*.

83 e200z420n3 Nexus 3 Module

The e200z420n3 Nexus 3 module provides real-time development capabilities for corresponding core processors in compliance with the IEEE-ISTO 5001 standard. This module provides development support capabilities without requiring the use of address and data pins for internal visibility.

A portion of the pin interface (the JTAG port) is also shared with the OnCE / Nexus 1 unit. The IEEE-ISTO 5001 standard defines an extensible auxiliary port that is used in conjunction with the JTAG port in these processors.

83.1 Introduction

83.1.1 General Description

This chapter defines the auxiliary pin functions, transfer protocols and standard development features of a Class 3 device in compliance with the IEEE-ISTO 5001 standard. The development features supported are Program Trace, Data Trace, Watchpoint Messaging, Ownership Trace, Data Acquisition Messaging, and Read/Write Access via the JTAG interface. The Nexus 3 module also supports two Class 4 features: Watchpoint Triggering, and Processor Overrun Control.

83.1.2 Terms and Definitions

Table 2039 contains a set of terms and definitions associated with the Nexus 3 module.

Table 2039. Terms and Definitions

Term	Description
IEEE-ISTO 5001	Consortium & standard for real-time embedded system design.
Auxiliary Port	Refers to Nexus auxiliary port. Used as auxiliary port to the IEEE 1149.1 JTAG interface.
Branch Trace Messaging (BTM)	Visibility of addresses for taken branches and exceptions, and the number of sequential instructions executed between each taken branch.
Data Read Message (DRM)	External visibility of data reads to memory-mapped resources.
Data Write Message (DWM)	External visibility of data writes to memory-mapped resources.
Data Trace Messaging (DTM)	External visibility of how data flows through the embedded system. This may include DRM and/or DWM.
Data Acquisition Messaging (DQM)	Data Acquisition Messaging (DQM) allows code to be instrumented to export customized information to the Nexus Auxiliary Output Port.
JTAG Compliant	Device complying to IEEE 1149.1 JTAG standard
JTAG IR & DR Sequence	JTAG Instruction Register (IR) scan to load an opcode value for selecting a development register. The JTAG IR corresponds to the OnCE command register (OCMD). The selected development register is then accessed via a JTAG Data Register (DR) scan.

Table 2039. Terms and Definitions (continued)

Term	Description
Nexus1	The (OnCE) debug module. This module integrated with each processor provides all static (core halted) debug functionality. This module is compliant with Class1 of the IEEE-ISTO 5001 standard.
Ownership Trace Message (OTM)	Visibility of process/function that is currently executing.
Public Messages	Messages on the auxiliary pins for accomplishing common visibility and controllability requirements
SoC	“System-on-a-Chip”. SoC signifies all of the modules on a single die. This generally includes one or more processors with associated peripherals, interfaces & memory modules.
Standard	The phrase “according to the standard” is used to indicate according to the IEEE-ISTO 5001 standard.
Transfer Code (TCODE)	Message header that identifies the number and/or size of packets to be transferred, and how to interpret each of the packets.
Watchpoint	A Data or Instruction Breakpoint or other debug event that does not cause the processor to halt. Instead, a pin is used to signal that the condition occurred. A Watchpoint Message may also be generated.

83.1.3 Feature List

The Nexus 3 module is compliant with Class 3 of the IEEE-ISTO 5001 standard, with additional Class 4 features available. The following features are implemented:

- Program Trace via Branch Trace Messaging (BTM). Branch trace messaging displays program flow discontinuities (such as direct and indirect branches, exceptions), allowing the development tool to interpolate what transpires between the discontinuities. Thus static code may be traced.
- Data Trace via Data Write Messaging (DWM) and Data Read Messaging (DRM). This provides the capability for the development tool to trace reads and/or writes to selected internal memory resources.
- Ownership Trace via Ownership Trace Messaging (OTM). OTM facilitates ownership trace by providing visibility of which process ID or operating system task is activated. An Ownership Trace Message is transmitted when a new process/task is activated, allowing the development tool to trace ownership flow.

- Run-time access to embedded processor memory map via the JTAG port. This allows for enhanced download/upload capabilities.
- Watchpoint Messaging via the auxiliary pins
- Watchpoint Trigger enable of Program and/or Data Trace Messaging
- External hardware trigger inputs to independently enable/disable Program Trace, Data Trace, Ownership Trace, and Watchpoint Trace Messaging
- Auxiliary interface for higher data input/output
  - Configurable (min/max) Message Data Out pins (**nex_mdo[n:0]**)
  - One (1) or two (2) Message Start/End Out pins (**nex_mseo_b[1:0]**)
  - One (1) Read/Write Ready pin (**nex_rdy_b**) pin
  - One (1) Read/Write Error pin (**nex_err_b**) pin
  - One (1) Watchpoint Event output pin (**nex_evto_b**)
  - Four (4) additional Watchpoint Event output pins (**nex_wevto[3:0]**) for SoC use
  - One (1) Event In pin (**nex_evti_b**)
  - One (1) MCKO (Message Clock Out) pin
- Registers for Program Trace, Data Trace, Ownership Trace and Watchpoint Trigger.
- All features controllable and configurable via the JTAG port
- Conditional software control of the module via SoC signaling input (**nex_sfwcntl_en**)

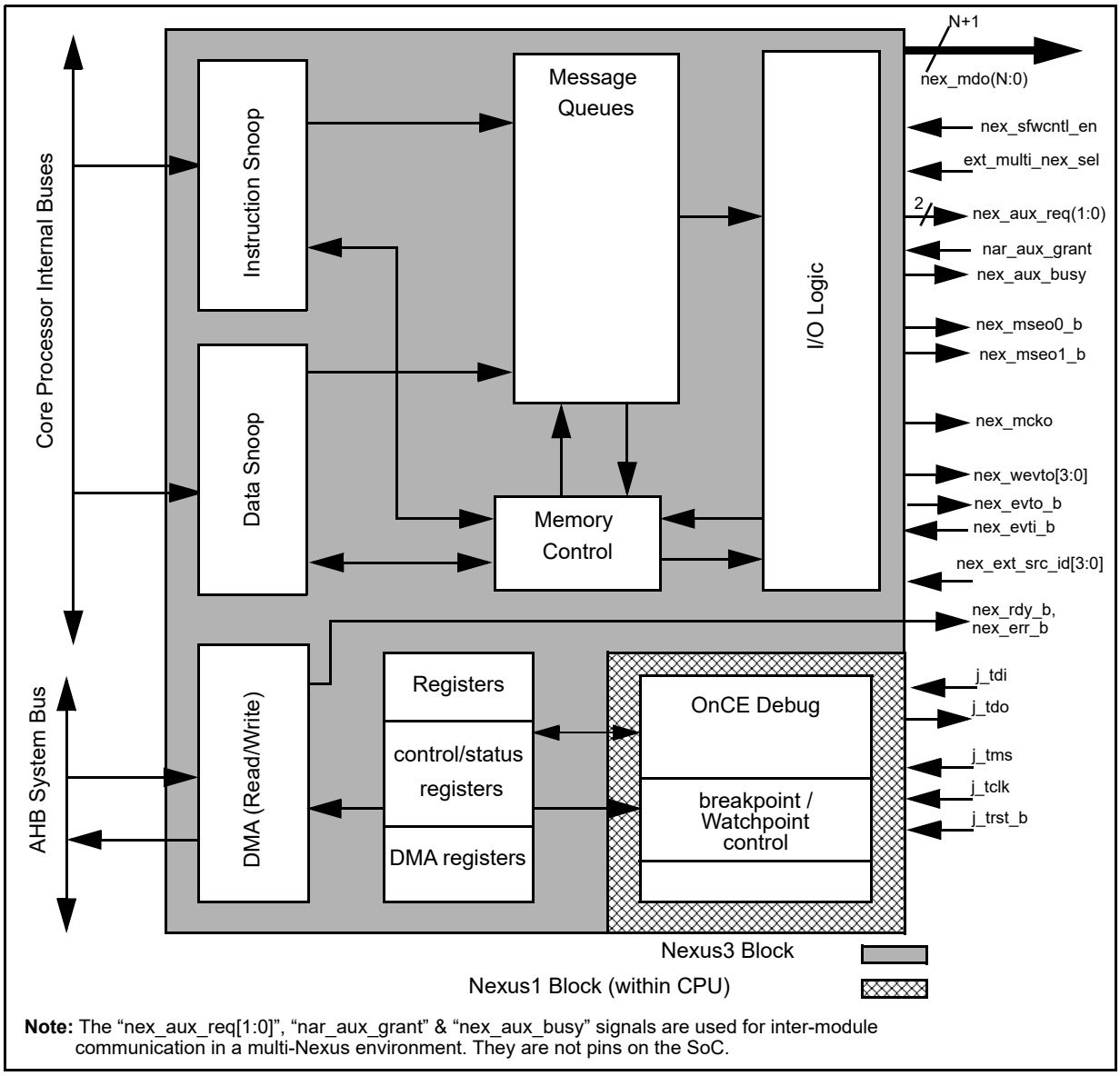
Note: For multi-Nexus implementations, the configuration of the Message Data Out pins is controlled by the Port Control Register (@ the SoC level). For single Nexus implementations (not normally implemented on an SoC), this configuration is controlled by Development Control Register 1 (DC1) within the Nexus 3 module.

In either implementation, Full Port Mode (FPM — maximum number of MDO pins) or Reduced Port Mode (RPM — minimum number of MDO pins) are supported. This setting should not be changed while the system is running.

Note: The configuration of the Message Start/End Out pins (1 or 2) is determined at the SOC integration level. This option will be hard-wired based on SOC bandwidth requirements.

83.1.4 Functional Block Diagram

Figure 2146. Nexus 3 functional block diagram



Note: The “nex_aux_req[1:0]”, “nar_aux_grant” & “nex_aux_busy” signals are used for inter-module communication in a multi-Nexus environment. They are not pins on the SoC.

83.2 Enabling Nexus 3 Operation

The Nexus module is enabled by loading a single instruction (NEXUS3-ACCESS) into the JTAG Instruction Register (IR) (OnCE OCMD register). For the Nexus3 module, the OCMD value is 0b0001111100. The Nexus 3 module may alternately be enabled if the nex_evti_b input signal is asserted at the time that j_trst_b is initially negated, or is asserted during the Test-Logic-Reset TAP state. Once enabled, the module will be ready to accept control input via the JTAG/OnCE pins.

Enabling the Nexus 3 module automatically enables the generation of Debug Status Messages.

The Nexus 3 module is disabled when the JTAG state machine initially reaches the Test-Logic-Reset state from any other state. This state can be reached by the assertion of the **j_trst_b** pin or by cycling through the state machine using the **j_tms** pin. The Nexus module will also be disabled if a Power-on-Reset (POR) event occurs. If the Nexus 3 module is disabled, no trace output will be provided, and the module will disable (drive inactive) auxiliary port output pins (**nex_mdo[n:0]**, **nex_mseo[1:0]**, **nex_mcko**). Nexus registers will not be available for reads or writes.

In order to support software control of the Nexus 3 module when no external development tool is present, the Nexus 3 module is not forced to be disabled when the JTAG state remains in the Test-Logic-Reset state. Software is allowed to control the Nexus 3 module when the input signal **nex_sfwcntl_en** is asserted by the SoC. This signal is intended to provide a mechanism for allowing software to use the Nexus 3 module to fill on-chip trace buffers or other visibility mechanisms. It is up to the SoC to determine whether a top-level Nexus 3 controller has been enabled by a hardware debugger, or whether appropriate security mechanisms have granted the capability for software to use these resources, and to drive the appropriate value to the **nex_sfwcntl_en** input. Software can enable the module by a write to any of the module's DCRs when **nex_sfwcntl_en** is asserted.

Reset of the Nexus 3 module is accomplished by a transition on **j_trst_b**, on initial entry into the Test_Logic_Reset state from another state, or if a Power-on-Reset (POR) event occurs. The module is not reset by the CPU's **p_reset_b** signal, even when software has control of the module.

83.2.1 Interaction with Low Power Modes

The Nexus 3 module will continue to operate in the Waiting and Halted states, as long as **nex_clk** remains active. In the Stopped state, **nex_clk** is gated off internally to the Nexus 3 logic, therefore watchpoint or hardware triggering recognition, message generation, and Nexus 3 read-write access to memory is suspended. The Nexus 3 logic will wait to enter the Stopped state until the message FIFOs are empty and any in-progress Nexus R/W transfer has completed. If a block transfer has been requested, the remainder of the block transfer is not completed, and the RWCS AC bit is cleared, and the ERR bit is set. Once the Stopped state has been entered, no further messages are queued and no triggering conditions are monitored. Also, Nexus R/W accesses are no longer available. Upon exiting the Stopped state, normal functions will resume assuming **nex_clk** is active.

83.3 TCODEs supported

The Nexus 3 pins allow for flexible transfer operations via Public Messages. A TCODE defines the transfer format, the number and/or size of the packets to be transferred, and the purpose of each packet. The IEEE-ISTO 5001-2003 standard defines a set of public messages and allocates additional TCODEs for vendor-specific features outside the scope of the public messages. The Nexus 3 block supports the TCODEs shown in *Table 2040*.

Table 2040. Supported TCODEs

Message Name	Min Field Size (bits)	Max Field Size (bits)	Field Name	Field Type	Field Description
Debug Status	6	6	TCODE	fixed	TCODE number = 0
	4	4	SRC	fixed	source processor identifier
	8	8	STATUS	fixed	Development Status Register (DS[31:24])
Ownership Trace Message	6	6	TCODE	fixed	TCODE number = 2
	4	4	SRC	fixed	source processor identifier
	1	32	PROCESS	variable	Task/Process ID tag
Program Trace — Direct Branch Message	6	6	TCODE	fixed	TCODE number = 3
	4	4	SRC	fixed	source processor identifier
	1	8	ICNT	variable	# sequential instructions completed since last predicate instruction, transmitted instruction count, or taken change of flow
Program Trace — Indirect Branch Message	6	6	TCODE	fixed	TCODE number = 4
	4	4	SRC	fixed	source processor identifier
	1	1	MAP	fixed	(always set to 0)
	1	8	ICNT	variable	# sequential instructions completed since last predicate instruction, transmitted instruction count, or taken change of flow
	1	32	U-ADDR	variable	unique part of target address for taken branches/exceptions
	6	6	TCODE	fixed	TCODE number = 5
Data Trace — Data Write Message	4	4	SRC	fixed	source processor identifier
	1	1	MAP	fixed	(always set to 0)
	4	4	DSZ	fixed	data size (Refer to *Table 2045*)
	1	32	U-ADDR	variable	unique portion of the data write address
	1	64	DATA	variable	data write value(s) (see Data Trace section for details)
Data Trace — Data Read Message	6	6	TCODE	fixed	TCODE number = 6
	4	4	SRC	fixed	source processor identifier
	1	1	MAP	fixed	(always set to 0)
	4	4	DSZ	fixed	data size (Refer to *Table 2045*)
	1	32	U-ADDR	variable	unique portion of the data read address
	1	64	DATA	variable	data read value(s) (see Data Trace section for details)

Table 2040. Supported TCODEs (continued)

Message Name	Min Field Size (bits)	Max Field Size (bits)	Field Name	Field Type	Field Description
Data Acquisition Message	6	6	TCODE	fixed	TCODE number = 7
	4	4	SRC	fixed	source processor identifier
	8	8	DQTAG	fixed	identification tag taken from $DEVENT_{DQTAG}$ register field
	1	32	DQDATA	variable	exported data taken from DDAM register
Error Message	6	6	TCODE	fixed	TCODE number = 8
	4	4	SRC	fixed	source processor identifier
	4	4	ETYPE	fixed	error type
	8	8	ECODE	fixed	error code
Program Trace — Synchronization Message	6	6	TCODE	fixed	TCODE number = 9
	4	4	SRC	fixed	source processor identifier
	1	1	MAP	fixed	(always set to 0)
	1	8	I-CNT	variable	# sequential instructions completed since last predicate instruction, transmitted instruction count, or taken change of flow. Cleared for most sync conditions.
	1	32	F-ADDR	variable	full target address (leading zero (0) truncated)
Program Trace — Direct Branch Message w/ Sync	6	6	TCODE	fixed	TCODE number = 11
	4	4	SRC	fixed	source processor identifier
	1	8	ICNT	variable	# sequential instructions completed since last predicate instruction, transmitted instruction count, or taken change of flow
	1	32	F-ADDR	variable	full target address (leading zeros truncated)
Program Trace — Indirect Branch Message w/ Sync	6	6	TCODE	fixed	TCODE number = 12
	4	4	SRC	fixed	source processor identifier
	1	1	MAP	fixed	(always set to 0)
	1	8	ICNT	variable	# sequential instructions completed since last predicate instruction, transmitted instruction count, or taken change of flow
	1	32	F-ADDR	variable	full target address (leading zeros truncated)
	6	6	TCODE	fixed	TCODE number = 13
Data Trace — Data Write Message w/ Sync	4	4	SRC	fixed	source processor identifier
	1	1	MAP	fixed	(always set to 0)
	4	4	DSZ	fixed	data size (Refer to *Table 2045*)
	1	32	F-ADDR	variable	full access address (leading zeros truncated)
	1	64	DATA	variable	data write value(s) (see Data Trace section for details)

Table 2040. Supported TCODEs (continued)

Message Name	Min Field Size (bits)	Max Field Size (bits)	Field Name	Field Type	Field Description
Data Trace — Data Read Message w/ Sync	6	6	TCODE	fixed	TCODE number = 14
	4	4	SRC	fixed	source processor identifier
	1	1	MAP	fixed	(always set to 0)
	4	4	DSZ	fixed	data size (Refer to *Table 2045*)
	1	32	F-ADDR	variable	full access address (leading zeros truncated)
	1	64	DATA	variable	data read value(s) (see Data Trace section for details)
Watchpoint Message	6	6	TCODE	fixed	TCODE number = 15
	4	4	SRC	fixed	source processor identifier
	1	32	WPHIT	variable	Field indicating watchpoint source(s) (leading zeros truncated)
Resource Full Message	6	6	TCODE	fixed	TCODE number = 27
	4	4	SRC	fixed	source processor identifier
	4	4	RCODE	fixed	resource code (Refer to *Table 2043*) — indicates which resource is the cause of this message
	1	32	RDATA	variable	branch / predicate instruction history (See *Section 83.12.4: Resource Full Messages*.)
Program Trace — Indirect Branch History Message	6	6	TCODE	fixed	TCODE number = 28 (see Note below)
	4	4	SRC	fixed	source processor identifier
	1	1	MAP	fixed	(always set to 0)
	1	8	I-CNT	variable	# sequential instructions completed since last predicate instruction, transmitted instruction count, or taken change of flow
	1	32	U-ADDR	variable	unique part of target address for taken branches/exceptions
	1	32	HIST	variable	branch / predicate instruction history (see *Section 83.12.1: Branch Trace Messaging types*)
Program Trace — Indirect Branch History Message w/ Sync	6	6	TCODE	fixed	TCODE number = 29 (see Note below)
	4	4	SRC	fixed	source processor identifier
	1	1	MAP	fixed	(always set to 0)
	1	8	I-CNT	variable	# sequential instructions completed since last predicate instruction, transmitted instruction count, or taken change of flow
	1	32	F-ADDR	variable	full target address (leading zero (0) truncated)
	1	32	HIST	variable	branch / predicate instruction history (See *Section 83.12.1: Branch Trace Messaging types*.)

Table 2040. Supported TCODEs (continued)

Message Name	Min Field Size (bits)	Max Field Size (bits)	Field Name	Field Type	Field Description
Program Trace — Program Correlation Message	6	6	TCODE	fixed	TCODE number = 33
	4	4	SRC	fixed	source processor identifier
	4	4	EVCODE	fixed	event correlated w/ program flow (Refer to *Table 2044*)
	2	2	CDF	fixed	# fields of information in CDATA. 00 — reserved, 01 — one field (CDATA1) (reserved), 10 — two fields (CDATA1 + CDATA2), 11 — three fields (reserved)
	1	8	I-CNT	variable	# sequential instructions completed since last predicate instruction, transmitted instruction count, or taken change of flow
	1	32	CDATA1	variable	correlation data field 1 — [branch / predicate instruction history] (See *Section 83.12.5: Program Correlation Messages*.)
	0	32	CDATA2	variable	correlation data field 2— PID info (See *Section 83.12.5: Program Correlation Messages*.)

Note: Program Trace can be implemented using either Branch History/Predicate Instruction Messages, or traditional Direct/Indirect Branch Messages. The user can select between the two types of Program Trace. The advantages for each are discussed in Section 83.12.1: Branch Trace Messaging types. If the Branch History method is selected, the shaded TCODES above will not be messaged out.

Table 2041 shows the error code encodings used when reporting an error via the Nexus 3 Error Message.

Table 2041. Error Code (ECODE) Encoding (TCODE = 8)

Error Code	Description
xxxxxxx1	Watchpoint Trace Message(s) Lost
xxxxxx1x	Data Trace Message(s) Lost
xxxxx1xx	Program Trace Message(s) Lost
xxxx1xxx	Ownership Trace Message(s) Lost
xxx1xxxx	Status Message(s) Lost (such as Debug Status messages)
xx1xxxxx	Data Acquisition Message(s) Lost
x1xxxxxx	Reserved
1xxxxxxx	Reserved

Table 2042 shows the error type encodings used when reporting an error via the Nexus 3 Error Message.

Table 2042. Error Type (ETYPE) Encoding (TCODE = 8)

Error Type	Description
0000	Message Queue Overrun caused one or more messages to be lost
0001	Contention with higher priority messages caused one or more messages to be lost
0010	Reserved
0011	Reserved
0100	Reserved
0101	Invalid access opcode (Nexus Register unimplemented)
0110 – 1111	Reserved

Table 2043 shows the encodings used for resource codes for certain messages.

Table 2043. Resource Code (RCODE) values (TCODE = 27)

Resource Code	Description
0000	Program Trace Instruction counter reached 255 and was reset.
0001	Program Trace, Branch / Predicate Instruction History full. This type of packet is terminated by a stop bit set to 1 after the last history bit.

Table 2044 shows the event code encodings used for certain messages.

Table 2044. Event Code (EVCODE) Encoding (TCODE = 33)

Event Code	Description
0000	Entry into Debug Mode
0001	Entry into Low Power Mode (CPU only)
0010–0011	Reserved for future functionality
0100	Disabling Program Trace
0101	Process ID value is established in PID0/NPIDR via **mtspr PID0/NPIDR**
0110–1000	Reserved for future functionality
1001	Begin masking of program trace messages due to MSR_{PMM} = 0 and $DC4_{PTMARK}$= 1
1010	Branch and link occurrence (direct branch function call)
1011–1111	Reserved for future functionality

Table 2045 shows the data trace size encodings used for certain messages.

Table 2045. Data Trace Size (DSZ) Encodings (TCODE = 5,6,13,14)

DTM Size Encoding	Transfer Size
0000	0 — no data
0001	Byte

Table 2045. Data Trace Size (DSZ) Encodings (TCODE = 5,6,13,14) (continued)

DTM Size Encoding	Transfer Size
0010	Halfword (2 bytes)
0011	Three bytes
0100	Word (4 bytes)
0101	Five bytes
0110	Six bytes
0111	Seven bytes
1000	Doubleword (8 bytes)
1001–1111	Reserved

83.4 Nexus 3 Programmer's Model

This section describes the Nexus 3 programmers model. Nexus 3 registers are accessed using the JTAG/OnCE port in compliance with IEEE 1149.1, or by software via DCRs. See *Section 83.5: Nexus 3 Register Access via JTAG/OnCE* and *Section 83.6: Nexus 3 Register Access via Software*for details on Nexus 3 register access.

Note: Nexus 3 registers and output signals are numbered using bit 0 as the least significant bit. This bit ordering is consistent with the ordering defined by the IEEE-ISTO 5001 standard.

Table 2046 details the register map for the Nexus 3 module.

Table 2046. Nexus 3 Register Map

Nexus Register	Nexus Access Opcode	Read/ Write	Read Address	Write Address	DCR #(1)
Client Select Control (CSC)(2)	0x1	R	0x02	—	
Port Configuration Register (PCR)(3)	PCR_INDEX(2)	R/W	—	—	
Development Control 1 (DC1)	0x2	R/W	0x04	0x05	368
Development Control 2 (DC2)	0x3	R/W	0x06	0x07	369
Development Control 3 (DC3)	0x4	R/W	0x08	0x09	370
Development Control 4 (DC4)	0x5	R/W	0x0A	0x0B	371
Reserved	0x6	R/W	0x18	0x19	—
Read/Write Access Control/Status (RWCS)	0x7	R/W	0x0E	0x0F	—
Reserved	0x8	R/W	0x18	0x19	—
Read/Write Access Address (RWA)	0x9	R/W	0x12	0x13	—
Read/Write Access Data (RWD)	0xA	R/W	0x14	0x15	—
Watchpoint Trigger (WT)	0xB	R/W	0x16	0x17	375
Reserved	0xC	R/W	0x18	0x19	—
Data Trace Control (DTC)	0xD	R/W	0x1A	0x1B	376
Data Trace Start Address 1 (DTSA1)	0xE	R/W	0x1C	0x1D	377

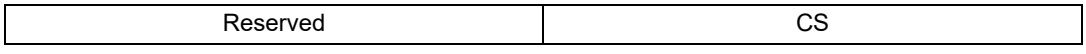
Table 2046. Nexus 3 Register Map (continued)

Nexus Register	Nexus Access Opcode	Read/ Write	Read Address	Write Address	DCR #(1)
Data Trace Start Address 2 (DTSA2)	0xF	R/W	0x1E	0x1F	378
Data Trace Start Address 3 (DTSA3)	0x10	R/W	0x20	0x21	379
Data Trace Start Address 4 (DTSA4)	0x11	R/W	0x22	0x23	380
Data Trace End Address 1 (DTEA1)	0x12	R/W	0x24	0x25	381
Data Trace End Address 2 (DTEA2)	0x13	R/W	0x26	0x27	382
Data Trace End Address 3 (DTEA3)	0x14	R/W	0x28	0x29	383
Data Trace End Address 4 (DTEA4)	0x15	R/W	0x2A	0x2B	408
Reserved	0x16 → 0x2F	—	0x28→0x5E	0x29→0x5F	—
Development Status (DS)	0x30	R	0x60	—	409
Reserved	0x31	R/W	0x62	0x63	
Overrun Control (OVCR)	0x32	R/W	0x64	0x65	410
Watchpoint Mask (WMSK)	0x33	R/W	0x66	0x67	411
Reserved	0x34	—	0x68	0x69	—
Program Trace Start Trigger Control (PTSTC)	0x35	R/W	0x6A	0x6B	412
Program Trace End Trigger Control (PTETC)	0x36	R/W	0x6C	0x6D	413
Data Trace Start Trigger Control (DTSTC)	0x37	R/W	0x6E	0x6F	414
Data Trace End Trigger Control (DTETC)	0x38	R/W	0x70	0x71	415
Reserved	0x39 → 0x3F	—	0x72→0x7E	0x73→0x7F	—

1. Software access via the **mfdcr** and **mtdcr** instructions use these values for the DCR number. Software writes to these registers via **mtdcr** when **nex_sfwcntl_en** is negated are ignored.
2. The CSC and PCR registers are shown in this table as part of the Nexus programmer's model. They are only present at the top level SoC Nexus controller in a multi-Nexus implementation, not in the Nexus 3 module. The SoC's CSC Register is readable through Nexus, but the PCR is shown for reference only here.
3. The "PCR_INDEX" is a parameter determined by the SoC.

83.4.1 Client Select Control (CSC) register

The CSC Register determines which Nexus client is under development. This register is present at the top-level SOC Nexus 3 controller to select one of multiple on-chip Nexus 3 units.



Nexus Reg# 0x1; Read-only; Reset: 0x0

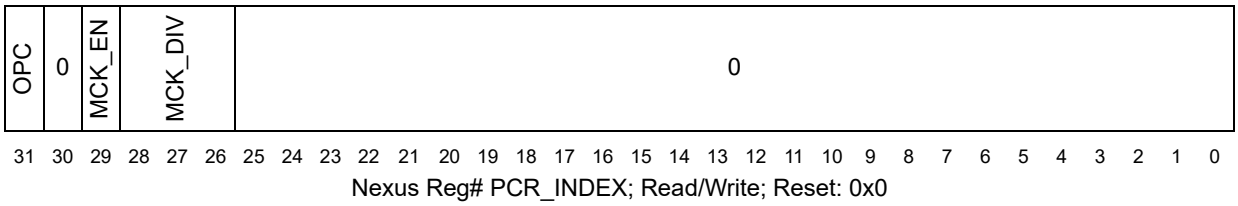
Figure 2147. Client Select Control (CSC) register

Table 2047. CSC field descriptions

Bits	Description
CSC[7:4]	Reserved for future Nexus Clients (read as 0)
CSC[3:0]	Client Select Control 0*xx* Nexus client (SoC level)

83.4.2 Port Configuration Register (PCR) — reference only

The Port Configuration Register (PCR) controls the basic port functions for all Nexus modules in a multi-Nexus environment. This includes clock control and auxiliary port width. All bits in this register are writable only once after system reset.



Nexus Reg# PCR_INDEX; Read/Write; Reset: 0x0

Figure 2148. Port Configuration Register (PCR)

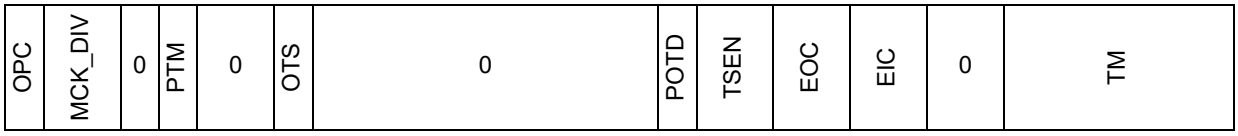
Table 2048. PCR field descriptions

Bit	Name	Description
31	OPC	Output Port Mode Control (SoC Level) 0 Reduced Port Mode configuration (min# **nex_mdo[n:0]** pins defined by SOC) 1 Full Port Mode configuration (max# **nex_mdo[n:0]** pins defined by SOC)
30	—	Reserved for future functionality
29	MCK_EN	MCKO Clock Enable (SoC Level) 0 **nex_mcko** is disabled 1 **nex_mcko** is enabled
28:26	MCK_DIV	MCKO Clock Divide Ratio (see note below) (SoC Level) 000 **nex_mcko** is 1x processor clock freq. 001 **nex_mcko** is 1/2x processor clock freq. 010 Reserved (default to 1/2x processor clock freq.) 011 **nex_mcko** is 1/4x processor clock freq. 100 Reserved (default to 1/2x processor clock freq.) 101 Reserved (default to 1/2x processor clock freq.) 110 Reserved (default to 1/2x processor clock freq.) 111 **nex_mcko** is 1/8x processor clock freq.
25:0	—	Reserved for future functionality

Note: The CSC and PCR Registers exist in a separate module at the SoC level in a multi-Nexus environment. If the core Nexus 3 module is the only Nexus module, these registers are not implemented and the Nexus 3 defined Development Control Register 1 (DC1) is used to control the SoC-level Nexus port functionality.

83.4.3 Nexus Development Control Register 1 (DC1)

Nexus Development Control Register 1 is used to control the basic development features of the Nexus 3 module. Development Control Register 1 is shown in *Figure 2149* and its fields are described in *Table 2049*.



Nexus Reg# 0x2; DCR: 368; Read/Write; Reset: 0x0

Figure 2149. Development Control 1 (DC1) register

Table 2049. DC1 field descriptions

Bits	Name	Description
31	OPC	Output Port Mode Control 0 — Reduced Port Mode configuration (min# **nex_mdo[n:0]** pins defined 1 — Full Port Mode configuration (max# **nex_mdo[n:0]** pins defined
30:29	MCK_DIV	MCKO Clock Divide Ratio (see note below) 00 — **nex_mcko** is 1x processor clock freq. 01 — **nex_mcko** is 1/2x processor clock freq. 10 — **nex_mcko** is 1/4x processor clock freq. 11 — **nex_mcko** is 1/8x processor clock freq.
28	—	Reserved for future functionality
27	PTM	Program Trace Method 0 — Program Trace uses traditional Branch Messages 1 — Program Trace uses Branch History Messages
26:25	—	Reserved for future functionality
24	OTS	Ownership Trace PID Select 0 — PID0 data is transmitted within Ownership Trace Messages 1 — Nexus PID Register (NPIDR) data is transmitted within Ownership Trace Messages
26:15	—	Reserved for future functionality
14	POTD	Periodic Ownership Trace Disable 0 — Periodic Ownership Trace message events are enabled 1 — Periodic Ownership Trace message events are disabled
13:12	TSEN	Timestamp Enable — (not implemented, write to 00) 00 — Timestamp is disabled
11:10	EOC	EVTO Control 00 — **nex_evto_b** upon occurrence of Watchpoints (configured in DC2 and DC3) 01 — **nex_evto_b** upon entry into Debug Mode 1*x* — Reserved
9:8	EIC	EVTI Control 00 — **nex_evti_b** is used for synchronization (Program Trace Sync msg is generated) 01 — **nex_evti_b** is used for Debug request 10 — **nex_evti_b** is disabled 1*x* — Reserved

Table 2049. DC1 field descriptions (continued)

Bits	Name	Description
7:6	—	Reserved for future functionality
5:0	TM	Trace Mode(1) 000000 — All Trace Disabled *xxxxx*1 — Ownership Trace enabled *xxxx*1*x* — Data Trace enabled *xxx*1*xx* — Program Trace enabled *xx*1*xxx* — Watchpoint Trace enabled *x*1*xxxx* — Reserved 1*xxxxx* — Data Acquisition Trace enabled

1. This field may be updated by hardware in response to watchpoint triggering or external hardware trigger inputs if not already enable for tracing by a previous direct write to DC1. Direct writes to this field to enable one or more trace types take precedence over hardware updates. Refer to *Section 83.4.7: Watchpoint Trigger (WT, PTSTC, PTETC, DTSTC, DTETC) registers* for more information on watchpoint triggering. Refer to *Section 83.16: External Hardware Trigger Controls* for more information on external hardware triggering.

Note: The Output Port Mode Control bit (OPC) and MCKO Clock Divide Ratio bits (MCK_DIV) MUST ONLY be modified during system reset or debug mode to insure correct output port and output clock functionality. It is also recommended that all other bits of the DC1 also only be modified in one of these two modes.

83.4.4 Nexus Development Control Registers 2 & 3 (DC2, DC3)

Nexus Development Control Registers 2 and 3 are used to control output signaling on the Nexus 3 module. A table of watchpoints can be found in the Core (e200z420n3) Core Debug Support chapter.

Development Control Register 2 is shown in *Figure 2150* and its fields are described in *Table 2050*.

WEVTO[3]C	WEVTO[2]C	WEVTO[1]C	WEVTO[0]C	EWC

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

Nexus Reg# 0x3; DCR: 369; Read/Write; Reset: 0x0

Figure 2150. Development Control 2 (DC2) register

Table 2050. DC2 field descriptions

Bits	Name	Description
31:28	WEVTO[3]C	Watchpoint Event Out 3 Configuration 0000 — No Watchpoints #0–14 trigger **nex_wevto[3]** 0001 — Watchpoint #0 triggers **nex_wevto[3]** 0010 — Watchpoint #1 triggers **nex_wevto[3]** 0011 — Watchpoint #2 triggers **nex_wevto[3]** 0100 — Watchpoint #3 triggers **nex_wevto[3]** 0101 — Watchpoint #4 triggers **nex_wevto[3]** 0110 — Watchpoint #5 triggers **nex_wevto[3]** 0111 — Watchpoint #6 triggers **nex_wevto[3]** 1000 — Watchpoint #7 triggers **nex_wevto[3]** 1001 — Watchpoint #8 triggers **nex_wevto[3]** 1010 — Watchpoint #9 triggers **nex_wevto[3]** 1011 — Watchpoint #10 triggers **nex_wevto[3]** 1100 — Watchpoint #11 triggers **nex_wevto[3]** 1101 — Watchpoint #12 triggers **nex_wevto[3]** 1110 — Watchpoint #13 triggers **nex_wevto[3]** 1111 — Watchpoint #14 triggers **nex_wevto[3]**
27:24	WEVTO[2]C	Watchpoint Event Out 2 Configuration 0000 — No Watchpoints #0–14 trigger **nex_wevto[2]** 0001 — Watchpoint #0 triggers **nex_wevto[2]** 0010 — Watchpoint #1 triggers **nex_wevto[2]** 0011 — Watchpoint #2 triggers **nex_wevto[2]** 0100 — Watchpoint #3 triggers **nex_wevto[2]** 0101 — Watchpoint #4 triggers **nex_wevto[2]** 0110 — Watchpoint #5 triggers **nex_wevto[2]** 0111 — Watchpoint #6 triggers **nex_wevto[2]** 1000 — Watchpoint #7 triggers **nex_wevto[2]** 1001 — Watchpoint #8 triggers **nex_wevto[2]** 1010 — Watchpoint #9 triggers **nex_wevto[2]** 1011 — Watchpoint #10 triggers **nex_wevto[2]** 1100 — Watchpoint #11 triggers **nex_wevto[2]** 1101 — Watchpoint #12 triggers **nex_wevto[2]** 1110 — Watchpoint #13 triggers **nex_wevto[2]** 1111 — Watchpoint #14 triggers **nex_wevto[2]**
23:20	WEVTO[1]C	Watchpoint Event Out 1 Configuration 0000 — No Watchpoints #0–14 trigger **nex_wevto[1]** 0001 — Watchpoint #0 triggers **nex_wevto[1]** 0010 — Watchpoint #1 triggers **nex_wevto[1]** 0011 — Watchpoint #2 triggers **nex_wevto[1]** 0100 — Watchpoint #3 triggers **nex_wevto[1]** 0101 — Watchpoint #4 triggers **nex_wevto[1]** 0110 — Watchpoint #5 triggers **nex_wevto[1]** 0111 — Watchpoint #6 triggers **nex_wevto[1]** 1000 — Watchpoint #7 triggers **nex_wevto[1]** 1001 — Watchpoint #8 triggers **nex_wevto[1]** 1010 — Watchpoint #9 triggers **nex_wevto[1]** 1011 — Watchpoint #10 triggers **nex_wevto[1]** 1100 — Watchpoint #11 triggers **nex_wevto[1]** 1101 — Watchpoint #12 triggers **nex_wevto[1]** 1110 — Watchpoint #13 triggers **nex_wevto[1]** 1111 — Watchpoint #14 triggers **nex_wevto[1]**

Table 2050. DC2 field descriptions (continued)

Bits	Name	Description
19:16	WEVTO[0]C	Watchpoint Event Out 0 Configuration 0000 — No Watchpoints #0–14 trigger **nex_wevto[0]** 0001 — Watchpoint #0 triggers **nex_wevto[0]** 0010 — Watchpoint #1 triggers **nex_wevto[0]** 0011 — Watchpoint #2 triggers **nex_wevto[0]** 0100 — Watchpoint #3 triggers **nex_wevto[0]** 0101 — Watchpoint #4 triggers **nex_wevto[0]** 0110 — Watchpoint #5 triggers **nex_wevto[0]** 0111 — Watchpoint #6 triggers **nex_wevto[0]** 1000 — Watchpoint #7 triggers **nex_wevto[0]** 1001 — Watchpoint #8 triggers **nex_wevto[0]** 1010 — Watchpoint #9 triggers **nex_wevto[0]** 1011 — Watchpoint #10 triggers **nex_wevto[0]** 1100 — Watchpoint #11 triggers **nex_wevto[0]** 1101 — Watchpoint #12 triggers **nex_wevto[0]** 1110 — Watchpoint #13 triggers **nex_wevto[0]** 1111 — Watchpoint #14 triggers **nex_wevto[0]**
15:0	EWC	EVTO Watchpoint Configuration(1) 0000000000000000 No Watchpoints #0–15 trigger **nex_evto_b** *xxxxxxxxxxxxxxx*1 Watchpoint #0 triggers **nex_evto_b** *xxxxxxxxxxxxxx*1*x* Watchpoint #1 triggers **nex_evto_b** *xxxxxxxxxxxxx*1*xx* Watchpoint #2 triggers **nex_evto_b** *xxxxxxxxxxxx*1*xxx* Watchpoint #3 triggers **nex_evto_b** *xxxxxxxxxxx*1*xxxx* Watchpoint #4 triggers **nex_evto_b** *xxxxxxxxxx*1*xxxxx* Watchpoint #5 triggers **nex_evto_b** *xxxxxxxxx*1*xxxxxx* Watchpoint #6 triggers **nex_evto_b** *xxxxxxxx*1*xxxxxxx* Watchpoint #7 triggers **nex_evto_b** *xxxxxxx*1*xxxxxxxx* Watchpoint #8 triggers **nex_evto_b** *xxxxxx*1*xxxxxxxxx* Watchpoint #9 triggers **nex_evto_b** *xxxxx*1*xxxxxxxxxx* Watchpoint #10 triggers **nex_evto_b** *xxxx*1*xxxxxxxxxxx* Watchpoint #11 triggers **nex_evto_b** *xxx*1*xxxxxxxxxxxx* Watchpoint #12 triggers **nex_evto_b** *xx*1*xxxxxxxxxxxxx* Watchpoint #13 triggers **nex_evto_b** *x*1*xxxxxxxxxxxxxx* Watchpoint #14 triggers **nex_evto_b** 1*xxxxxxxxxxxxxxx* Watchpoint #15 triggers **nex_evto_b**

1. EOC bits in DC1 must be programmed to trigger $\overline{\text{EVTO}}$ on Watchpoint occurrence for EWC bits to have any effect.

Development Control Register 3 is shown in *Figure 2151* and its fields are described in *Table 2051*.

WEVTO[3]C	WEVTO[2]C	WEVTO[1]C	WEVTO[0]C	0	EWC

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

Nexus Reg# 0x4; DCR: 370; Read/Write; Reset: 0x0

Figure 2151. Development Control 3 (DC3) register

Table 2051. DC3 field descriptions

Bits	Name	Description
31:28	WEVTO[3]C	Watchpoint Event Out 3 Configuration 0000 No Watchpoints #15–#26 trigger **nex_wevto[3]** 0001 Watchpoint #15 triggers **nex_wevto[3]** 0010 Watchpoint #16 triggers **nex_wevto[3]** 0011 Watchpoint #17 triggers **nex_wevto[3]** 0100 Watchpoint #18 triggers **nex_wevto[3]** 0101 Watchpoint #19 triggers **nex_wevto[3]** 0110 Watchpoint #20 triggers **nex_wevto[3]** 0111 Watchpoint #21 triggers **nex_wevto[3]** 1000 Watchpoint #22 triggers **nex_wevto[3]** 1001 Watchpoint #23 triggers **nex_wevto[3]** 1010 Watchpoint #24 triggers **nex_wevto[3]** 1011 Watchpoint #25 triggers **nex_wevto[3]** 1100 Watchpoint #26 triggers **nex_wevto[3]** 11*x*1 Reserved
27:24	WEVTO[2]C	Watchpoint Event Out 2 Configuration 0000 No Watchpoints #15–#26 trigger **nex_wevto[2]** 0001 Watchpoint #15 triggers **nex_wevto[2]** 0010 Watchpoint #16 triggers **nex_wevto[2]** 0011 Watchpoint #17 triggers **nex_wevto[2]** 0100 Watchpoint #18 triggers **nex_wevto[2]** 0101 Watchpoint #19 triggers **nex_wevto[2]** 0110 Watchpoint #20 triggers **nex_wevto[2]** 0111 Watchpoint #21 triggers **nex_wevto[2]** 1000 Watchpoint #22 triggers **nex_wevto[2]** 1001 Watchpoint #23 triggers **nex_wevto[2]** 1010 Watchpoint #24 triggers **nex_wevto[2]** 1011 Watchpoint #25 triggers **nex_wevto[2]** 1100 Watchpoint #26 triggers **nex_wevto[2]** 11*x*1 Reserved
23:20	WEVTO[1]C	Watchpoint Event Out 1 Configuration 0000 No Watchpoints #15–#29 trigger **nex_wevto[1]** 0001 Watchpoint #15 triggers **nex_wevto[1]** 0010 Watchpoint #16 triggers **nex_wevto[1]** 0011 Watchpoint #17 triggers **nex_wevto[1]** 0100 Watchpoint #18 triggers **nex_wevto[1]** 0101 Watchpoint #19 triggers **nex_wevto[1]** 0110 Watchpoint #20 triggers **nex_wevto[1]** 0111 Watchpoint #21 triggers **nex_wevto[1]** 1000 Watchpoint #22 triggers **nex_wevto[1]** 1001 Watchpoint #23 triggers **nex_wevto[1]** 1010 Watchpoint #24 triggers **nex_wevto[1]** 1011 Watchpoint #25 triggers **nex_wevto[1]** 1100 Watchpoint #26 triggers **nex_wevto[1]** 1101 Watchpoint #27 triggers **nex_wevto[1]** 1110 Watchpoint #28 triggers **nex_wevto[1]** 1111 Watchpoint #29 triggers **nex_wevto[1]**

Table 2051. DC3 field descriptions (continued)

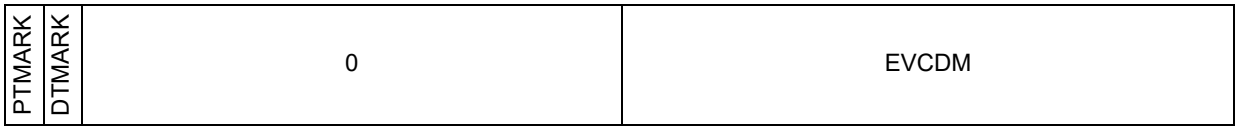
Bits	Name	Description
19:16	WEVTO[0]C	Watchpoint Event Out 0 Configuration 0000 No Watchpoints #15–#29 trigger **nex_wevto[0]** 0001 Watchpoint #15 triggers **nex_wevto[0]** 0010 Watchpoint #16 triggers **nex_wevto[0]** 0011 Watchpoint #17 triggers **nex_wevto[0]** 0100 Watchpoint #18 triggers **nex_wevto[0]** 0101 Watchpoint #19 triggers **nex_wevto[0]** 0110 Watchpoint #20 triggers **nex_wevto[0]** 0111 Watchpoint #21 triggers **nex_wevto[0]** 1000 Watchpoint #22 triggers **nex_wevto[0]** 1001 Watchpoint #23 triggers **nex_wevto[0]** 1010 Watchpoint #24 triggers **nex_wevto[0]** 1011 Watchpoint #25 triggers **nex_wevto[0]** 1100 Watchpoint #26 triggers **nex_wevto[0]** 1101 Watchpoint #27 triggers **nex_wevto[0]** 1110 Watchpoint #28 triggers **nex_wevto[0]** 1111 Watchpoint #29 triggers **nex_wevto[0]**
15:14	—	Reserved for watchpoint expansion
13:0	EWC	EVTO Watchpoint Configuration(1) 00000000000000 No Watchpoints #16–#29 trigger **nex_evto_b** *xxxxxxxxxxxxx*1 Watchpoint #16 triggers **nex_evto_b** *xxxxxxxxxxxx*1*x* Watchpoint #17 triggers **nex_evto_b** *xxxxxxxxxxx*1*xx* Watchpoint #18 triggers **nex_evto_b** *xxxxxxxxxx*1*xxx* Watchpoint #19 triggers **nex_evto_b** *xxxxxxxxx*1*xxxx* Watchpoint #20 triggers **nex_evto_b** *xxxxxxxx*1*xxxxx* Watchpoint #21 triggers **nex_evto_b** *xxxxxxx*1*xxxxxx* Watchpoint #22 triggers **nex_evto_b** *xxxxxx*1*xxxxxxx* Watchpoint #23 triggers **nex_evto_b** *xxxxx*1*xxxxxxxx* Watchpoint #24 triggers **nex_evto_b** *xxxx*1*xxxxxxxxx* Watchpoint #25 triggers **nex_evto_b** *xxx*1*xxxxxxxxxx* Watchpoint #26 triggers **nex_evto_b** *xx*1*xxxxxxxxxxx* Watchpoint #27 triggers **nex_evto_b** *x*1*xxxxxxxxxxxx* Watchpoint #28 triggers **nex_evto_b** 1*xxxxxxxxxxxxx* Watchpoint #29 triggers **nex_evto_b**

1. EOC bits in DC1 must be programmed to trigger $\overline{\text{EVTO}}$ on Watchpoint occurrence for EWC bits to have any effect.

83.4.5 Nexus Development Control Register 4 (DC4)

Nexus Development Control Register 4 is used to control mark selection for Program and Data Trace Messaging, as well as masking of events that initiate Program Correlation Messages on the Nexus 3 module.

Development Control Register 4 is shown in *Figure 2152* and its fields are described in *Table 2052*.



Nexus Reg# 0x5; DCR: 371; Read/Write; Reset: 0x0

Figure 2152. Development Control 4 (DC4) register

Table 2052. DC4 field descriptions

Bits	Name	Description
31	PTMARK	Program Trace Mark 0 Ignore MSR_{PMM} for masking program trace messages 1 Mask program trace messages when MSR_{PMM}='0', unmask program trace messages when MSR_{PMM}='1'
30	DTMARK	Data Trace Mark 0 Ignore MSR_{PMM} for masking data trace messages 1 Mask data trace messages when MSR_{PMM}='0', unmask data trace messages when MSR_{PMM}='1'
29:16	—	Reserved
15:0	EVCDM	Event Code (EVCODE) Mask(1) 0000000000000000 No EVCODEs masked for Program Correlation Messages *xxxxxxxxxxxxxxx*1 EVCODE #0 is masked for Program Correlation Messages *xxxxxxxxxxxxxx*1*x* EVCODE #1 is masked for Program Correlation Messages *xxxxxxxxxxxxx*1*xx* EVCODE #2 is masked for Program Correlation Messages *xxxxxxxxxxxx*1*xxx* EVCODE #3 is masked for Program Correlation Messages *xxxxxxxxxxx*1*xxxx* EVCODE #4 is masked for Program Correlation Messages *xxxxxxxxxx*1*xxxxx* EVCODE #5 is masked for Program Correlation Messages *xxxxxxxxx*1*xxxxxx* EVCODE #6 is masked for Program Correlation Messages *xxxxxxxx*1*xxxxxxx* EVCODE #7 is masked for Program Correlation Messages *xxxxxxx*1*xxxxxxxx* EVCODE #8 is masked for Program Correlation Messages *xxxxxx*1*xxxxxxxxx* EVCODE #9 is masked for Program Correlation Messages *xxxxx*1*xxxxxxxxxx* EVCODE #10 is masked for Program Correlation Messages *xxxx*1*xxxxxxxxxxx* EVCODE #11 is masked for Program Correlation Messages *xxx*1*xxxxxxxxxxxx* EVCODE #12 is masked for Program Correlation Messages *xx*1*xxxxxxxxxxxxx* EVCODE #13 is masked for Program Correlation Messages *x*1*xxxxxxxxxxxxxx* EVCODE #14 is masked for Program Correlation Messages 1*xxxxxxxxxxxxxxx* EVCODE #15 is masked for Program Correlation Messages

1. Refer to *Table 2044* for implemented EVCODEs.

83.4.6 Development Status Register (DS)

The Development Status Register is used to report system debug status. When Debug Mode is entered or exited, or an SoC- or core-defined Low Power Mode is entered (see note below), a Debug Status Message is transmitted with DS[31:24]. The external tool can read this register at any time.



Nexus Reg# 0x4; DCR: 409; Read-only; Reset: 0x0

Figure 2153. Development Status (DS) register

Table 2053. DS field descriptions

Bits	Name	Description
31	DBG	CPU Debug Mode Status 0 CPU not in Debug mode 1 CPU in Debug mode (**jd_debug_b** signal asserted)
30:28	LPS	System Low Power Mode Status 000 Normal (Run) mode 001 Waiting State (**p_waiting** signal asserted 010 Halted State (**p_halted** signal asserted 011 Reserved 1*xx* Reserved
27:26	LPC	CPU Low Power Mode Status 00 Normal (Run) mode 01 CPU in Halted state (**p_halted** signal asserted) 10 CPU in Stopped state (**p_stopped** signal asserted) 11 CPU in Waiting state (**p_waiting** signal asserted)
25:0	—	Reserved for future functionality (read as 0)

83.4.7 Watchpoint Trigger (WT, PTSTC, PTETC, DTSTC, DTETC) registers

The Watchpoint Trigger Registers allows the watchpoints defined within the Nexus1 logic to trigger actions. These watchpoints can control Program and/or Data Trace enable and disable. The control bits can be used to produce a related “window” for triggering Trace Messages. Watchpoint trigger register WT is used to control triggering by a single selected watchpoint. The Program Trace Start Trigger Control (PTSTC), Program Trace End Trigger Control (PTETC), Data Trace Start Trigger Control (DTSTC), and Data Trace End Trigger Control (DTETC) are used for extended trigger controls for the respective function. If multiple watchpoints are desired for triggering, or a watchpoint beyond watchpoint #13 is required, then one or more of the extended watchpoint trigger registers may be used. A field encoding of 4’b1111 in one of the WT register fields enables the corresponding extended trigger register. For all other WT field encodings, the corresponding extended trigger register is disabled and the contents are ignored. Note that direct writes to enable program trace and/or data trace in DC1 will override these controls, and trace will remain enabled until another direct write to DC1 to disable program and/or data trace occurs.

When a start trigger is detected, the designated trace features become enabled, and the corresponding enable bits of the DC1 register are set. Whenever a stop trigger is detected, the designated trace features become disabled, and the corresponding enable bits of the DC1 register are cleared. If the same trigger condition is used for both start and stop triggering, then the designated trace features will toggle between being enabled and disabled at each occurrence of the trigger condition. Similarly, if start and stop triggers for a trace feature occur simultaneously, then the designated trace feature will toggle between

enabled and disabled depending on the enable state at the time of the trigger events. For example, if tracing is enabled, and a start and stop trigger occur simultaneously, then tracing will be disabled. Direct writes of the DC1 register take precedence over any trace feature enable state that is derived from watchpoint triggering. A table of watchpoints can be found in the Core (e200z420n3) Core Debug Support chapter.

PTS	PTE	DTS	DTE	0

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

Nexus Reg# 0xB; DCR: 375; Read/Write; Reset: 0x0

Figure 2154. Watchpoint Trigger (WT) register

Table 2054 details the Watchpoint Trigger register fields.

Table 2054. WT field descriptions

Bits	Name	Description
31:28	PTS	Program Trace Start Control 0000 Trigger disabled 0001 Use Watchpoint #0 0010 Use Watchpoint #1 . . 1110 Use Watchpoint #13 1111 Use control settings in the PTSTC register
27:24	PTE	Program Trace End Control 0000 Trigger disabled 0001 Use Watchpoint #0 0010 Use Watchpoint #1 . . 1110 Use Watchpoint #13 1111 Use control settings in the PTETC register
23:20	DTS	Data Trace Start Control 0000 Trigger disabled 0001 Use Watchpoint #0 0010 Use Watchpoint #1 . . 1110 Use Watchpoint #13 1111 Use control settings in the DTSTC register
19:16	DTE	Data Trace End Control 0000 Trigger disabled 0001 Use Watchpoint #0 0010 Use Watchpoint #1 . . 1110 Use Watchpoint #13 1111 Use control settings in the DTETC register
15:0	—	Reserved for future functionality (read as 0)

For extended Program Trace start trigger control, the PTSTC register is used.

PTST

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

Nexus Reg# 0x35; DCR: 412; Read/Write; Reset: 0x0

Figure 2155. Program Trace Start Trigger Control (PTSTC) register

Table 2055 details the PTSTC register fields.

Table 2055. PTSTC field descriptions

Bits	Name	Description
31:0	PTST	Program Trace Start Trigger Control 00000000000000000000000000000000 Trigger disabled *xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx*1 Use Watchpoint #0 *xxxxxxxxxxxxxxxxxxxxxxxxxxxxxx*1*x* Use Watchpoint #1 *xxxxxxxxxxxxxxxxxxxxxxxxxxxxx*1*xx* Use Watchpoint #2 *xxxxxxxxxxxxxxxxxxxxxxxxxxxx*1*xxx* Use Watchpoint #3 *xxxxxxxxxxxxxxxxxxxxxxxxxxx*1*xxxx* Use Watchpoint #4 *xxxxxxxxxxxxxxxxxxxxxxxxxx*1*xxxxx* Use Watchpoint #5 *xxxxxxxxxxxxxxxxxxxxxxxxx*1*xxxxxx* Use Watchpoint #6 *xxxxxxxxxxxxxxxxxxxxxxxx*1*xxxxxxx* Use Watchpoint #7 *xxxxxxxxxxxxxxxxxxxxxxx*1*xxxxxxxx* Use Watchpoint #8 *xxxxxxxxxxxxxxxxxxxxxx*1*xxxxxxxxx* Use Watchpoint #9 *xxxxxxxxxxxxxxxxxxxxx*1*xxxxxxxxxx* Use Watchpoint #10 *xxxxxxxxxxxxxxxxxxxx*1*xxxxxxxxxxx* Use Watchpoint #11 *xxxxxxxxxxxxxxxxxxx*1*xxxxxxxxxxxx* Use Watchpoint #12 *xxxxxxxxxxxxxxxxxx*1*xxxxxxxxxxxxx* Use Watchpoint #13 *xxxxxxxxxxxxxxxxx*1*xxxxxxxxxxxxxx* Use Watchpoint #14 *xxxxxxxxxxxxxxxx*1*xxxxxxxxxxxxxxx* Use Watchpoint #15 *xxxxxxxxxxxxxxx*1*xxxxxxxxxxxxxxxx* Use Watchpoint #16 *xxxxxxxxxxxxxx*1*xxxxxxxxxxxxxxxxx* Use Watchpoint #17 *xxxxxxxxxxxxx*1*xxxxxxxxxxxxxxxxxx* Use Watchpoint #18 *xxxxxxxxxxxx*1*xxxxxxxxxxxxxxxxxxx* Use Watchpoint #19 *xxxxxxxxxxx*1*xxxxxxxxxxxxxxxxxxxx* Use Watchpoint #20 *xxxxxxxxxx*1*xxxxxxxxxxxxxxxxxxxxx* Use Watchpoint #21 *xxxxxxxxx*1*xxxxxxxxxxxxxxxxxxxxxx* Use Watchpoint #22 *xxxxxxxx*1*xxxxxxxxxxxxxxxxxxxxxxx* Use Watchpoint #23 *xxxxxxx*1*xxxxxxxxxxxxxxxxxxxxxxxx* Use Watchpoint #24 *xxxxxx*1*xxxxxxxxxxxxxxxxxxxxxxxxx* Use Watchpoint #25 *xxxxx*1*xxxxxxxxxxxxxxxxxxxxxxxxxx* Use Watchpoint #26 *xxxx*1*xxxxxxxxxxxxxxxxxxxxxxxxxxx* Use Watchpoint #27 *xxx*1*xxxxxxxxxxxxxxxxxxxxxxxxxxxx* Use Watchpoint #28 *xx*1*xxxxxxxxxxxxxxxxxxxxxxxxxxxxx* Use Watchpoint #29 *x*1*xxxxxxxxxxxxxxxxxxxxxxxxxxxxxx* Use Watchpoint #30 1*xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx* Use Watchpoint #31

For extended Program Trace end trigger control, the PTETC register is used.

PTET

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

Nexus Reg# 0x36; DCR: 413; Read/Write; Reset: 0x0

Figure 2156. Program Trace End Trigger Control (PTETC) register

Table 2056 details the PTETC register fields.

Table 2056. PTETC field descriptions

Bits	Name	Description
31:0	PTET	Program Trace End Trigger Control 00000000000000000000000000000000 Trigger disabled *xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx*1 Use Watchpoint #0 *xxxxxxxxxxxxxxxxxxxxxxxxxxxxxx*1*x* Use Watchpoint #1 *xxxxxxxxxxxxxxxxxxxxxxxxxxxxx*1*xx* Use Watchpoint #2 *xxxxxxxxxxxxxxxxxxxxxxxxxxxx*1*xxx* Use Watchpoint #3 *xxxxxxxxxxxxxxxxxxxxxxxxxxx*1*xxxx* Use Watchpoint #4 *xxxxxxxxxxxxxxxxxxxxxxxxxx*1*xxxxx* Use Watchpoint #5 *xxxxxxxxxxxxxxxxxxxxxxxxx*1*xxxxxx* Use Watchpoint #6 *xxxxxxxxxxxxxxxxxxxxxxxx*1*xxxxxxx* Use Watchpoint #7 *xxxxxxxxxxxxxxxxxxxxxxx*1*xxxxxxxx* Use Watchpoint #8 *xxxxxxxxxxxxxxxxxxxxxx*1*xxxxxxxxx* Use Watchpoint #9 *xxxxxxxxxxxxxxxxxxxxx*1*xxxxxxxxxx* Use Watchpoint #10 *xxxxxxxxxxxxxxxxxxxx*1*xxxxxxxxxxx* Use Watchpoint #11 *xxxxxxxxxxxxxxxxxxx*1*xxxxxxxxxxxx* Use Watchpoint #12 *xxxxxxxxxxxxxxxxxx*1*xxxxxxxxxxxxx* Use Watchpoint #13 *xxxxxxxxxxxxxxxxx*1*xxxxxxxxxxxxxx* Use Watchpoint #14 *xxxxxxxxxxxxxxxx*1*xxxxxxxxxxxxxxx* Use Watchpoint #15 *xxxxxxxxxxxxxxx*1*xxxxxxxxxxxxxxxx* Use Watchpoint #16 *xxxxxxxxxxxxxx*1*xxxxxxxxxxxxxxxx* Use Watchpoint #17 *xxxxxxxxxxxxx*1*xxxxxxxxxxxxxxxx* Use Watchpoint #18 *xxxxxxxxxxxx*1*xxxxxxxxxxxxxxxx* Use Watchpoint #19 *xxxxxxxxxxx*1*xxxxxxxxxxxxxxxx* Use Watchpoint #20 *xxxxxxxxxx*1*xxxxxxxxxxxxxxxx* Use Watchpoint #21 *xxxxxxxxx*1*xxxxxxxxxxxxxxxx* Use Watchpoint #22 *xxxxxxxx*1*xxxxxxxxxxxxxxxx* Use Watchpoint #23 *xxxxxxx*1*xxxxxxxxxxxxxxxx* Use Watchpoint #24 *xxxxxx*1*xxxxxxxxxxxxxxxx* Use Watchpoint #25 *xxxxx*1*xxxxxxxxxxxxxxxx* Use Watchpoint #26 *xxxx*1*xxxxxxxxxxxxxxxx* Use Watchpoint #27 *xxx*1*xxxxxxxxxxxxxxxx* Use Watchpoint #28 *xx*1*xxxxxxxxxxxxxxxx* Use Watchpoint #29 *x*1*xxxxxxxxxxxxxxxx* Use Watchpoint #30 1*xxxxxxxxxxxxxxxx* Use Watchpoint #31

For extended Data Trace start trigger control, the DTSTC register is used.

0	DTST

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

Nexus Reg# 0x37; DCR: 414; Read/Write; Reset: 0x0

Figure 2157. Data Trace Start Trigger Control (DTSTC) register

The following table details the DTSTC register fields.

Table 2057. DTSTC field descriptions

Bits	Name	Description
31:30	—	Reserved for future functionality (read as 0)
29:0	DTST	Data Trace Start Trigger Control 00000000000000000000000000000000 Trigger disabled *xxxxxxxxxxxxxxxxxxxxxxxxxxxxxx*1 Use Watchpoint #0 *xxxxxxxxxxxxxxxxxxxxxxxxxxxxx*1*x* Use Watchpoint #1 *xxxxxxxxxxxxxxxxxxxxxxxxxxxx*1*xx* Use Watchpoint #2 *xxxxxxxxxxxxxxxxxxxxxxxxxxx*1*xxx* Use Watchpoint #3 *xxxxxxxxxxxxxxxxxxxxxxxxxx*1*xxxx* Use Watchpoint #4 *xxxxxxxxxxxxxxxxxxxxxxxxx*1*xxxxx* Use Watchpoint #5 *xxxxxxxxxxxxxxxxxxxxxxxx*1*xxxxxx* Use Watchpoint #6 *xxxxxxxxxxxxxxxxxxxxxxx*1*xxxxxxx* Use Watchpoint #7 *xxxxxxxxxxxxxxxxxxxxxx*1*xxxxxxxx* Use Watchpoint #8 *xxxxxxxxxxxxxxxxxxxxx*1*xxxxxxxxx* Use Watchpoint #9 *xxxxxxxxxxxxxxxxxxxx*1*xxxxxxxxxx* Use Watchpoint #10 *xxxxxxxxxxxxxxxxxxx*1*xxxxxxxxxxx* Use Watchpoint #11 *xxxxxxxxxxxxxxxxxx*1*xxxxxxxxxxxx* Use Watchpoint #12 *xxxxxxxxxxxxxxxxx*1*xxxxxxxxxxxxx* Use Watchpoint #13 *xxxxxxxxxxxxxxxx*1*xxxxxxxxxxxxxx* Use Watchpoint #14 *xxxxxxxxxxxxxxx*1*xxxxxxxxxxxxxxx* Use Watchpoint #15 *xxxxxxxxxxxxxx*1*xxxxxxxxxxxxxxx* Use Watchpoint #16 *xxxxxxxxxxxxx*1*xxxxxxxxxxxxxxxx* Use Watchpoint #17 *xxxxxxxxxxxx*1*xxxxxxxxxxxxxxxx* Use Watchpoint #18 *xxxxxxxxxxx*1*xxxxxxxxxxxxxxxx* Use Watchpoint #19 *xxxxxxxxxx*1*xxxxxxxxxxxxxxxx* Use Watchpoint #20 *xxxxxxxxx*1*xxxxxxxxxxxxxxxx* Use Watchpoint #21 *xxxxxxxx*1*xxxxxxxxxxxxxxxx* Use Watchpoint #22 *xxxxxxx*1*xxxxxxxxxxxxxxxx* Use Watchpoint #23 *xxxxxx*1*xxxxxxxxxxxxxxxx* Use Watchpoint #24 *xxxxx*1*xxxxxxxxxxxxxxxx* Use Watchpoint #25 *xxxx*1*xxxxxxxxxxxxxxxx* Use Watchpoint #26 *xxx*1*xxxxxxxxxxxxxxxx* Use Watchpoint #27 *xx*1*xxxxxxxxxxxxxxxx* Use Watchpoint #28 *x*1*xxxxxxxxxxxxxxxx* Use Watchpoint #29 1*xxxxxxxxxxxxxxxx* Use Watchpoint #29

For extended Data Trace end trigger control, the DTETC register is used.

0	DTET

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

Nexus Reg# 038; DCR: 415; Read/Write; Reset: 0x0

Figure 2158. Data Trace End Trigger Control (DTETC) register

The following table details the DTETC register fields.

Table 2058. DTETC field descriptions

Bits	Name	Description
31:30	—	Reserved for future functionality (read as 0)
29:0	DTET	Data Trace End Trigger Control 00000000000000000000000000000000 Trigger disabled *xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx*1 Use Watchpoint #0 *xxxxxxxxxxxxxxxxxxxxxxxxxxxxxx*1*x* Use Watchpoint #1 *xxxxxxxxxxxxxxxxxxxxxxxxxxxxx*1*xx* Use Watchpoint #2 *xxxxxxxxxxxxxxxxxxxxxxxxxxxx*1*xxx* Use Watchpoint #3 *xxxxxxxxxxxxxxxxxxxxxxxxxxx*1*xxxx* Use Watchpoint #4 *xxxxxxxxxxxxxxxxxxxxxxxxxx*1*xxxxx* Use Watchpoint #5 *xxxxxxxxxxxxxxxxxxxxxxxxx*1*xxxxxx* Use Watchpoint #6 *xxxxxxxxxxxxxxxxxxxxxxxx*1*xxxxxxx* Use Watchpoint #7 *xxxxxxxxxxxxxxxxxxxxxxx*1*xxxxxxxx* Use Watchpoint #8 *xxxxxxxxxxxxxxxxxxxxxx*1*xxxxxxxxx* Use Watchpoint #9 *xxxxxxxxxxxxxxxxxxxxx*1*xxxxxxxxxx* Use Watchpoint #10 *xxxxxxxxxxxxxxxxxxxx*1*xxxxxxxxxxx* Use Watchpoint #11 *xxxxxxxxxxxxxxxxxxx*1*xxxxxxxxxxxx* Use Watchpoint #12 *xxxxxxxxxxxxxxxxxx*1*xxxxxxxxxxxxx* Use Watchpoint #13 *xxxxxxxxxxxxxxxxx*1*xxxxxxxxxxxxxx* Use Watchpoint #14 *xxxxxxxxxxxxxxxx*1*xxxxxxxxxxxxxxx* Use Watchpoint #15 *xxxxxxxxxxxxxxx*1*xxxxxxxxxxxxxxxx* Use Watchpoint #16 *xxxxxxxxxxxxxx*1*xxxxxxxxxxxxxxxx* Use Watchpoint #17 *xxxxxxxxxxxxx*1*xxxxxxxxxxxxxxxx* Use Watchpoint #18 *xxxxxxxxxxxx*1*xxxxxxxxxxxxxxxx* Use Watchpoint #19 *xxxxxxxxxxx*1*xxxxxxxxxxxxxxxx* Use Watchpoint #20 *xxxxxxxxxx*1*xxxxxxxxxxxxxxxx* Use Watchpoint #21 *xxxxxxxxx*1*xxxxxxxxxxxxxxxx* Use Watchpoint #22 *xxxxxxxx*1*xxxxxxxxxxxxxxxx* Use Watchpoint #23 *xxxxxxx*1*xxxxxxxxxxxxxxxx* Use Watchpoint #24 *xxxxxx*1*xxxxxxxxxxxxxxxx* Use Watchpoint #25 *xxxxx*1*xxxxxxxxxxxxxxxx* Use Watchpoint #26 *xxxx*1*xxxxxxxxxxxxxxxx* Use Watchpoint #27 *xxx*1*xxxxxxxxxxxxxxxx* Use Watchpoint #28 *xx*1*xxxxxxxxxxxxxxxx* Use Watchpoint #29 *x*1*xxxxxxxxxxxxxxxx* Use Watchpoint #30 1*xxxxxxxxxxxxxxxx* Use Watchpoint #31

83.4.8 Nexus Watchpoint Mask (WMSK) register

The Nexus Watchpoint Mask register controls which watchpoint events are enabled to produce Watchpoint Trace Messages ($DC1_{TM}$ must also be programmed to generate Watchpoint Trace Messages).

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

Nexus Reg# 0x33; DCR: 411; Read/Write; Reset: 0x0

Figure 2159. Watchpoint Mask (WMSK) register

Table 2059 details the Watchpoint Trigger register fields.

Table 2059. WMSK field descriptions

Bits	Name	Description
31:0	WEM	Watchpoint Enable for Messaging 00000000000000000000000000000000 No Watchpoints enabled for Watchpoint Trace Messaging *xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx*1 Watchpoint #0 enabled for WTM *xxxxxxxxxxxxxxxxxxxxxxxxxxxxxx*1*x* Watchpoint #1 enabled for WTM *xxxxxxxxxxxxxxxxxxxxxxxxxxxxx*1*xx* Watchpoint #2 enabled for WTM *xxxxxxxxxxxxxxxxxxxxxxxxxxxx*1*xxx* Watchpoint #3 enabled for WTM *xxxxxxxxxxxxxxxxxxxxxxxxxxx*1*xxxx* Watchpoint #4 enabled for WTM *xxxxxxxxxxxxxxxxxxxxxxxxxx*1*xxxxx* Watchpoint #5 enabled for WTM *xxxxxxxxxxxxxxxxxxxxxxxxx*1*xxxxxx* Watchpoint #6 enabled for WTM *xxxxxxxxxxxxxxxxxxxxxxxx*1*xxxxxxx* Watchpoint #7 enabled for WTM *xxxxxxxxxxxxxxxxxxxxxxx*1*xxxxxxxx* Watchpoint #8 enabled for WTM *xxxxxxxxxxxxxxxxxxxxxx*1*xxxxxxxxx* Watchpoint #9 enabled for WTM *xxxxxxxxxxxxxxxxxxxxx*1*xxxxxxxxxx* Watchpoint #10 enabled for WTM *xxxxxxxxxxxxxxxxxxxx*1*xxxxxxxxxxx* Watchpoint #11 enabled for WTM *xxxxxxxxxxxxxxxxxxx*1*xxxxxxxxxxxx* Watchpoint #12 enabled for WTM *xxxxxxxxxxxxxxxxxx*1*xxxxxxxxxxxxx* Watchpoint #13 enabled for WTM *xxxxxxxxxxxxxxxxx*1*xxxxxxxxxxxxxx* Watchpoint #14 enabled for WTM *xxxxxxxxxxxxxxxx*1*xxxxxxxxxxxxxxx* Watchpoint #15 enabled for WTM *xxxxxxxxxxxxxxx*1*xxxxxxxxxxxxxxxx* Watchpoint #16 enabled for WTM *xxxxxxxxxxxxxx*1*xxxxxxxxxxxxxxxxx* Watchpoint #17 enabled for WTM *xxxxxxxxxxxxx*1*xxxxxxxxxxxxxxxxxx* Watchpoint #18 enabled for WTM *xxxxxxxxxxxx*1*xxxxxxxxxxxxxxxxxxx* Watchpoint #19 enabled for WTM *xxxxxxxxxxx*1*xxxxxxxxxxxxxxxxxxxx* Watchpoint #20 enabled for WTM *xxxxxxxxxx*1*xxxxxxxxxxxxxxxxxxxxx* Watchpoint #21 enabled for WTM *xxxxxxxxx*1*xxxxxxxxxxxxxxxxxxxxxx* Watchpoint #22 enabled for WTM *xxxxxxxx*1*xxxxxxxxxxxxxxxxxxxxxxx* Watchpoint #23 enabled for WTM *xxxxxxx*1*xxxxxxxxxxxxxxxxxxxxxxxx* Watchpoint #24 enabled for WTM *xxxxxx*1*xxxxxxxxxxxxxxxxxxxxxxxxx* Watchpoint #25 enabled for WTM *xxxxx*1*xxxxxxxxxxxxxxxxxxxxxxxxxx* Watchpoint #26 enabled for WTM *xxxx*1*xxxxxxxxxxxxxxxxxxxxxxxxxxx* Watchpoint #27 enabled for WTM *xxx*1*xxxxxxxxxxxxxxxxxxxxxxxxxxxx* Watchpoint #28 enabled for WTM *xx*1*xxxxxxxxxxxxxxxxxxxxxxxxxxxxx* Watchpoint #29 enabled for WTM *x*1*xxxxxxxxxxxxxxxxxxxxxxxxxxxxxx* Watchpoint #30 enabled for WTM 1*xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx* Watchpoint #31 enabled for WTM

83.4.9 Nexus Overrun Control (OVCR) register

The Nexus Overrun Control register controls Nexus behavior as the internal message queues fill up. Response options include suppressing selected message types, or stalling processor instruction execution.

0	SPTHOLD	0	SPEN	0	STTHOLD	0	STEN

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

Nexus Reg# 0x32; DCR: 410; Read/Write; Reset: 0x0

Figure 2160. Nexus Overrun Control (OVCR) register

Table 2060. OVCR field descriptions

Bits	Name	Description
31:30	—	Reserved, should be cleared
29:28	SPTHOLD	Suppression Threshold 00 Suppression threshold is when message queues are 1/4 full 01 Suppression threshold is when message queues are 1/2 full 10 Suppression threshold is when message queues are 3/4 full 11 Reserved
27:22	—	Reserved, should be cleared
21:16	SPEN	Suppression Enable 000000 Suppression is disabled *xxxxx*1 Ownership Trace message suppression is enabled *xxxx*1*x* Data Trace message suppression is enabled *xxx*1*xx* Program Trace message suppression is enabled *xx*1*xxx* Watchpoint Trace message suppression is enabled *x*1*xxxx* Reserved 1*xxxxx* Data Acquisition message suppression is enabled
15:14	—	Reserved, should be cleared
13:12	STTHOLD	Stall Threshold 00 Stall threshold is when message queues are 1/4 full 01 Stall threshold is when message queues are 1/2 full 10 Stall threshold is when message queues are 3/4 full 11 Reserved
11:1	—	Reserved, should be cleared
0	STEN	Stall Enable 0 Stalling is disabled 1 Stalling is enabled

83.4.10 Data Trace Control (DTC) register

The Data Trace Control Register controls whether DTM Messages are restricted to reads, writes, or both for a user programmable address range. In addition, control is provided to restrict data trace message generation to only those load or store accesses that are not stack-related, in order to minimize required data trace message bandwidth.

There are four Data Trace channels controlled by the DTC for the Nexus 3 module. Channels can be programmed to trace data accesses or instruction accesses, but not independently.

RWT1	RWT2	RWT3	RWT4	0	STDC1	STDC2	STDC3	STDC4	0	RC1	RC2	RC3	RC4	DI	0
31 30	29 28	27 26	25 24	23 22 21 20 19 18 17 16	15	14	13	12	11 10 9 8	7	6	5	4	3	2 1 0

Nexus Reg# 0xD; DCR: 376; Read/Write; Reset: 0x0

Figure 2161. Data Trace Control (DTC) register

The following table details the Data Trace Control register fields.

Table 2061. DTC field descriptions

Bits	Name	Description
31:30	RWT1	Read/Write Trace 1 00 No trace enabled *x*1 Enable Data Read Trace 1*x* Enable Data Write Trace
29:28	RWT2	Read/Write Trace 2 00 No trace enabled *x*1 Enable Data Read Trace 1*x* Enable Data Write Trace
27:26	RWT3	Read/Write Trace 3 00 No trace enabled *x*1 Enable Data Read Trace 1*x* Enable Data Write Trace
25:24	RWT4	Read/Write Trace 4 00 No trace enabled *x*1 Enable Data Read Trace 1*x* Enable Data Write Trace
23:16	—	Reserved for future functionality (read as 0)
15	STDC1	Stack Trace Disable Control 1 0 Tracing of stack accesses are not disabled for range 1 1 Tracing of stack accesses are disabled for range 1; data accesses using GPR R1 in effective address computations are not traced and do not generate data range watchpoint outputs
14	STDC2	Stack Trace Disable Control 2 0 Tracing of stack accesses are not disabled for range 2 1 Tracing of stack accesses are disabled for range 2; data accesses using GPR R1 in effective address computations are not traced and do not generate data range watchpoint outputs
13	STDC3	Stack Trace Disable Control 3 0 Tracing of stack accesses are not disabled for range 3 1 Tracing of stack accesses are disabled for range 3; data accesses using GPR R1 in effective address computations are not traced and do not generate data range watchpoint outputs
12	STDC4	Stack Trace Disable Control 4 0 Tracing of stack accesses are not disabled for range 4 1 Tracing of stack accesses are disabled for range 4; data accesses using GPR R1 in effective address computations are not traced and do not generate data range watchpoint outputs

Table 2061. DTC field descriptions (continued)

Bits	Name	Description
11:8	—	Reserved for future functionality (read as 0)
7	RC1	Range Control 1 0 Condition trace on address within range 1 Condition trace on address outside of range
6	RC2	Range Control 2 0 Condition trace on address within range 1 Condition trace on address outside of range
5	RC3	Range Control 3 0 Condition trace on address within range 1 Condition trace on address outside of range
4	RC4	Range Control 4 0 Condition trace on address within range 1 Condition trace on address outside of range
3	DI	Data Access / Instruction Access Trace 0 Condition trace on data accesses 1 Condition trace on instruction accesses **Note:** The support for monitoring instruction accesses is not guaranteed to be present in all versions of the Nexus 3 module. **Note:** The setting of the DI bit also affects the information provided on the data trace port outputs (See the “Data Trace Port Signals” section in the Core (e200z420n3) Core Debug Support chapter.)
2:0	—	Reserved for future functionality (read as 0)

83.4.11 Data Trace Start Address Registers (DTSA1–4)

The Data Trace Start Address Registers define the start addresses for each trace channel.

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

Nexus Reg# 0xE; DCR: 377; Read/Write; Reset: 0x0

Figure 2162. Data Trace Start Address 1 (DTSA1) register

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

Nexus Reg# 0xF; DCR: 378; Read/Write; Reset: 0x0

Figure 2163. Data Trace Start Address 2 (DTSA2) register

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

Nexus Reg# 0x10; DCR: 379; Read/Write; Reset: 0x0

Figure 2164. Data Trace Start Address 3 (DTSA3) register

Data Trace Start Address

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

Nexus Reg# 0x11; DCR: 380; Read/Write; Reset: 0x0

Figure 2165. Data Trace Start Address 4 (DTSA4) register

83.4.12 Data Trace End Address 1–4 (DTEA1–4) registers

The Data Trace End Address Registers define the end addresses for each trace channel.

Data Trace End Address

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

Nexus Reg# 0x12; DCR: 381; Read/Write; Reset: 0x0

Figure 2166. Data Trace End Address 1 register

Data Trace End Address

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

Nexus Reg# 0x13; DCR: 382; Read/Write; Reset: 0x0

Figure 2167. Data Trace End Address 2 register

Data Trace End Address

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

Nexus Reg# 0x14; DCR: 383; Read/Write; Reset: 0x0

Figure 2168. Data Trace End Address 3 register

Data Trace End Address

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

Nexus Reg# 0x15; DCR: 408; Read/Write; Reset: 0x0

Figure 2169. Data Trace End Address 4 register

Table 2062 illustrates the range that will be selected for Data Trace for various cases of DTSA being less than, greater than, or equal to DTEA.

Table 2062. Data trace — address range options

Programmed values	Range Control Bit Value	Range Selected
DTSA < DTEA	0	DTSA → ← DTEA
DTSA < DTEA	1	← DTSA DTEA →
DTSA > DTEA	N/A	Invalid Range — no trace
DTSA = DTEA	N/A	Invalid Range — no trace

Note: DTSA must be less than DTEA in order to guarantee correct Data Write/Read Traces. Data Trace ranges are inclusive of the DTSA and DTEA addresses for Range Control settings

indicating "within range," and are exclusive of the DTSA and DTEA addresses for Range Control settings indicating "outside of range."

Accesses that meet the range and access type qualifiers will cause assertion of a watchpoint output for Ranges 1, 2, and 3. There are three dedicated watchpoint outputs, one for each DTSA/DTEA sets 1, 2, and 3. Range 4 does not provide a watchpoint output. Note that when DTC_{DI} = 1, all instruction fetches and prefetches (including discarded prefetches) are monitored, and thus these range watchpoints differ from the IACx watchpoint outputs, which are not asserted for instructions that are not executed (that is when the instruction prefetch is discarded).

83.4.13 Read/Write Access Control/Status (RWCS) register

The Read Write Access Control/Status Register provides control for Read/Write Access. Read/Write access provides DMA-like access to memory-mapped resources on the AHB System bus either while the processor is halted, or during runtime. Control is provided over access type, size, count, and certain bus attributes. Note that the internal CPU interfaces to the caches and local memory are not accessible or utilized by this mechanism. Since the external interface is used, base address port settings are used to access local memory, and not the settings of local memory control register base address values. The RWCS Register also provides Read/Write Access Status information per *Table 2064*.

AC	RW	SZ	MAP	PR	ATTR	0	CNT	ERR	DV
31	30	29 28 27	26 25 24	23 22	21 20 19 18	17 16	15 14 13 12 11 10 9 8 7 6 5 4 3 2	1	0

Nexus Reg# 0x7; Read/Write(1); Reset: 0x0

Figure 2170. Read/Write Access Control/Status (RWCS) register

1. ERR and DV are read-only

Table 2063. RWCS field descriptions

Bits	Name	Description
RWCS[31]	AC	Access Control 0 End access/ Access has completed 1 Start access
RWCS[30]	RW	Read/Write Select 0 Read access 1 Write access
RWCS[29:27]	SZ	Word Size 000 8-bit (byte) 001 16-bit (half-word) 010 32-bit (word) 011 64-bit (doubleword, requires two passes through RWD) 1*xx* Reserved (default to word)
RWCS[26:24]	MAP	MAP Select 000 Primary memory map All other values are reserved

Table 2063. RWCS field descriptions (continued)

Bits	Name	Description
RWCS[23:22]	PR(1)	Read/Write Access Priority 00 Reserved (default to highest priority) 01 Reserved (default to highest priority) 10 Reserved (default to highest priority) 11 Highest access priority
RWCS[21:18]	ATTR	Access Attributes 0xxx Reserved 1xxx Reserved x0xx **p_d_hprot[4]** driven to 0 for accesses x1xx **p_d_hprot[4]** driven to 1 for accesses xx0x **p_d_hprot[3]** driven to 0 for accesses xx1x **p_d_hprot[3]** driven to 1 for accesses xxx0 **p_d_hprot[2]** driven to 0 for accesses xxx1 **p_d_hprot[2]** driven to 1 for accesses
RWCS[17:16]	—	Reserved for future functionality
RWCS[15:2]	CNT	Access Control Count hhhh Number of accesses of word size SZ
RWCS[1]	ERR(2)	Read/Write Access Error (see *Table 2064*)
RWCS[0]	DV(2)	Read/Write Access Data Valid (see *Table 2064*)

1. The priority functionality is not currently implemented.
2. ERR and DV are read-only.

Table 2064. Read/Write Access Status Bit Encoding

Read Action	Write Action	ERR	DV
Read Access has not completed	Write Access completed without error	0	0
Read Access error has occurred	Write Access error has occurred	1	0
Read Access completed without error	Write Access has not completed	0	1
Not Allowed	Not allowed	1	1

83.4.14 Read/Write Access Data (RWD) register

The Read/Write Access Data (RWD) register provides the data to/from system bus memory-mapped locations when initiating a read or a write access.

Read/Write Data

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

Nexus Reg# 0xA; Read/Write; Reset: 0x0

Figure 2171. Read/Write Access Data (RWD) register

Read/Write accesses to the AHB require that the debug firmware properly retrieve/place the data in the RWD. *Table 2065* shows the proper placement of data into the RWD. Note that doubleword transfers require two passes through RWD.

Table 2065. RWD data placement for transfers

Transfer Size and byte offset	RWA(2:0)	RWCS[SZ]	RWD			
			31:24	23:16	15:8	7:0
Byte	*x x x*	0 0 0	—	—	—	X
Half	*x x* 0	0 0 1	—	—	X	X
Word	*x* 0 0	0 1 0	X	X	X	X
Doubleword first RWD pass (low order data)	0 0 0	0 1 1	X	X	X	X
second RWD pass (high order data)			X	X	X	X

Table Notes:
"X" indicates byte lanes with valid data
"—" indicates byte lanes that will contain unused data.

Table 2066 shows the mapping of RWD bytes to byte lanes of the AHB read and write data buses.

Table 2066. RWD byte lane mapping

Transfer Size and byte offset	RWA(2:0)	RWD			
		31:24	23:16	15:8	7:0
Byte @000	0 0 0	—	—	—	AHB[7:0]
Byte @001	0 0 1	—	—	—	AHB[15:8]
Byte @010	0 1 0	—	—	—	AHB[23:16]
Byte @011	0 1 1	—	—	—	AHB[31:24]
Byte @100	1 0 0	—	—	—	AHB[39:32]
Byte @101	1 0 1	—	—	—	AHB[47:40]
Byte @110	1 1 0	—	—	—	AHB[55:48]
Byte @111	1 1 1	—	—	—	AHB[63:56]
Half @000	0 0 0	—	—	AHB[15:8]	AHB[7:0]
Half @010	0 1 0	—	—	AHB[31:24]	AHB[23:16]
Half @100	1 0 0	—	—	AHB[47:40]	AHB[39:32]
Half @110	1 1 0	—	—	AHB[63:56]	AHB[55:48]
Word @000	0 0 0	AHB[31:24]	AHB[23:16]	AHB[15:8]	AHB[7:0]
Word @100	1 0 0	AHB[63:56]	AHB[55:48]	AHB[47:40]	AHB[39:32]

Table 2066. RWD byte lane mapping (continued)

Transfer Size and byte offset	RWA(2:0)	RWD			
		31:24	23:16	15:8	7:0
Doubleword @000	0 0 0				
first RWD pass		AHB[31:24]	AHB[23:16]	AHB[15:8]	AHB[7:0]
second RWD pass		AHB[63:56]	AHB[55:48]	AHB[47:40]	AHB[39:32]

Table Notes:
"—" indicates byte lanes that will contain unused data.

83.4.15 Read/Write Access Address (RWA)

The Read/Write Access Address Register provides the system bus address to be accessed when initiating a read or a write access.

Read/Write Address

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

Nexus Reg# 0x9; Read/Write; Reset: 0x0

Figure 2172. Read/Write Access Address (RWA) register

83.5 Nexus 3 Register Access via JTAG/OnCE

Access to Nexus 3 register resources is enabled by loading a single instruction ("*NEXUS3-ACCESS*") into the JTAG Instruction Register (IR) (OnCE OCMD register). For the Nexus 3 block, the OCMD value is 0b0001111100.

Once the NEXUS3-ACCESS instruction has been loaded, the JTAG/OnCE port allows tool/target communications with all Nexus 3 registers according to the register map in *Table 2046*.

Reading/writing of a Nexus 3 register then requires two (2) passes through the Data-Scan (DR) path of the JTAG state machine. (See *Section 83.23: IEEE 1149.1 (JTAG) RD/WR sequences*.)

1. The first pass through the DR selects the Nexus 3 register to be accessed by providing an index (see *Table 2046*), and the direction (read/write). This is achieved by loading an 8-bit value into the JTAG Data Register (DR). This register has the following format:

Figure 2173. JTAG DR format

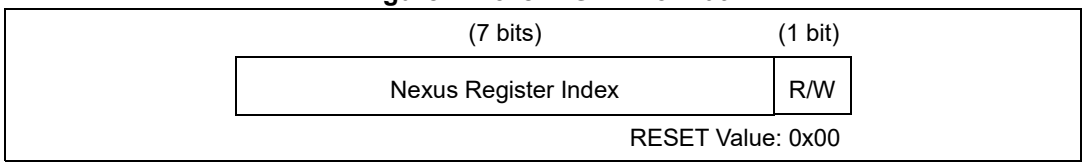


Table 2067. JTAG DR field descriptions

Nexus Register Index	Selected from values in *Table 2046*
Read/Write (R/W)	0 Read 1 Write

2. The second pass through the DR then shifts the data in or out of the JTAG port, LSB first.
   a) During a read access, data is latched from the selected Nexus register when the JTAG state machine passes through the “Capture-DR” state.
   b) During a write access, data is latched into the selected Nexus register when the JTAG state machine passes through the “Update-DR” state.

83.6 Nexus 3 Register Access via Software

Access to most Nexus 3 register resources by software is supported by mapping the Nexus 3 registers to privileged DCRs (device control registers) that are accessible via the **mfdcr** and **mtdcr** instructions. It is intended that these access only occur when no possible conflicts with hardware-based accesses are possible. A conflict between hardware and software accesses will produce undefined results. *Table 2046* shows the mappings of Nexus 3 registers to DCR numbers. Software writes to Nexus 3 register resources are blocked when the **nex_sfwcntl_en** input signal is negated, without signaling an exception. Note that the Nexus 3 Read/Write Access functionality is not available to software, since software can use load and store instructions to access memory.

83.7 Nexus Message Fields

Nexus messages are comprised of fields. Each field contains a distinct piece of information within a message, and each message contains multiple fields. Messages are transferred in packets over the Auxiliary Output protocol. A packet is a collection of fields. A packet may contain any number of fixed length fields, but may contain at most one variable length field. The variable length field must be the last field in a packet. The following sub-sections describe a subset of the message field types.

83.7.1 TCODE Field

The TCODE field is a 6-bit fixed length field that identifies the type of message and its format. The field encodings are assigned by IEEE-ISTO 5001.

83.7.2 Source ID Field (SRC)

Each Nexus module in a device is identified by a unique Client Source Identification Number. The number assigned to each Nexus module is determined by the SoC integrator, and is provided on the **nex3_ext_src_id[0:3]** input signals. Multi-threaded processors may assign additional source ID information to indicate which thread a message is associated with. The Nexus 3 module implements a 4-bit fixed length Source ID field consisting of a Client Source ID.

83.7.3 Relative Address Field (U-ADDR)

The non-sync forms of the Program and Data Trace messages include addresses that are relative to the address that was transmitted in the previous Program or Data Trace message, respectively. The relative address format is compliant with IEEE-ISTO 5001 and is designed to reduce the number of bits transmitted for address fields.

The relative address is generated by XORing the new address with the previous, and then using only the results up to the most significant '1'. To recreate the original address, the relative address is XORed with the previously decoded address.

The relative address of a Program Trace message is calculated with respect to the previous Program Trace message, regardless of any address information that may have been sent in any other trace messages in the interim between the two Program Trace messages.

The relative address of a Data Trace message is calculated with respect to the previous Data Trace message, regardless of any address information that may have been sent in any other trace messages in the interim between the two Data Trace messages.

Program trace messages also provide the execution mode status in the least significant bit of the **reconstructed** address field. A value of '0' indicates that preceding instruction count and history information should be interpreted in a non-VLE context. A value of '1' indicates that the preceding instruction count and history information should be interpreted in a VLE context.

Previous Address (A1) =0x0003FC01, New Address (A2) = 0x0003F365

Figure 2174. Relative address generation and recreation

Message Generation:

A1 = 0000 0000 0000 0011 1111 1100 0000 0001
A2 = 0000 0000 0000 0011 1111 0011 0110 0101

A1 ⊕ A2 = 0000 0000 0000 0000 0000 1111 0110 0100

Address Message (M1) = 1111 0110 0100

Address Re-creation:

A1 ⊕ M1 = A2
A1 = 0000 0000 0000 0011 1111 1100 0000 0001
M1 = 0000 0000 0000 0000 0000 1111 0110 0100

A2 = 0000 0000 0000 0011 1111 0011 0110 0101

83.7.4 Full Address Field (F-ADDR)

Program Trace synchronization messages provide the full address associated with the trace event (leading zeroes may be truncated) with the intent of providing a reference point for development tools to operate from when reconstructing relative addresses. Synchronization

messages are generated at significant mode switches and are also generated periodically to ensure that development tools are guaranteed to have a reference address given a sufficiently large sample of trace messages. Program trace messages also provide the execution mode status in the least significant bit of the **reconstructed** address field. A value of '0' indicates that preceding instruction count and history information should be interpreted in a non-VLE context. A value of '1' indicates that the preceding instruction count and history information should be interpreted in a VLE context.

83.8 Nexus Message Queues

The Nexus 3 module implements internal message queues capable of storing up to three messages per cycle into a small initial queue, which then fills a larger queue at up to two messages per cycle. Messages that enter the queues are transmitted in the order in which they are received.

If more than three messages attempt to enter the queue in the same cycle, the highest priority messages are stored and the remaining message(s) will be dropped due to a collision. Collision events are expected to be rare.

The Overrun Control register (OVCR) controls the Nexus behavior as the message queue fills. The Nexus block may be programmed to:

- Allow the queue to overflow, drain the contents, queue an overrun error message and resume tracing.
- Stall the processor when the queue utilization reaches the selected threshold.
- Suppress selected message types when the queue utilization reaches the selected threshold.

83.8.1 Message Queue Overrun

In this mode, the message queue will stop accepting messages when an overrun condition is detected. The contents of the queues will be allowed to drain until empty. Incoming messages are discarded until the queue is emptied. Once empty, an overrun error message is enqueued that contains information about the types of messages that were discarded due to the overrun condition.

83.8.2 CPU Stall

In this mode, processor instruction issue is stalled when the queue utilization reaches the selected threshold. The processor is stalled long enough to drop one threshold level below the level that triggered the stall. For example, if stalling the processor is triggered at 1/4 full, the stall will stay in effect until the queue utilization drops to empty. There may be significant skid from the time that the stall request is made until the processor is able to stop completing instructions. This skid should be taken into consideration when programming the threshold. Refer to *Section 83.4.9: Nexus Overrun Control (OVCR) register*, for complete programming options.

83.8.3 Message Suppression

In this mode, the message queue will disable selected message types when the queue utilization reaches the selected threshold. This allows lower bandwidth tracing to continue and possibly avoid an overrun condition. If an overrun condition occurs despite this message suppression, the queue will respond according to the behavior described in

Section 83.8.1: Message Queue Overrun. Suppressed message types are reported in the error message if they occur after overrun in addition to other discarded message types. Once triggered, message suppression will remain in effect until queue utilization drops to the threshold below the level selected to trigger suppression.

83.8.4 Nexus Message Priority

Nexus messages may be lost due to contention with other message types under the following circumstance:

- More than three messages are generated in the same cycle

Table 2068 lists the various message types and their relative priority from highest to lowest.

Up to three message requests can be queued into the message buffer in a given cycle. If more than three message requests exist in a given cycle, the three highest priority message classes are queued into the message buffer. The remaining messages that did not successfully queue into the message buffer in that cycle will generate subsequent responses, as detailed in *Table 2068*.

The CPU is capable of completing two instructions per cycle. If multiple trace messages need to be queued at the same time, they will be queued with the following priority: Instruction 0 (oldest instruction) (WPM → DQM → PCM_{PIDMSG} → OTM → BTM → DTM) → Instruction1 (newer instruction) (WPM → DQM → OTM → BTM → DTM). As many as three messages may be simultaneously queued. Note that for the cycle following a dropped PTM, non-periodic OTM, or DQM message, only two other messages may be queued in addition to the dropped Error Message.

Watchpoint Messages from instructions that complete at the same time or events that occur during the same cycle will be combined.

Table 2068. Message Type Priority and Message Dropped Responses

Message Type	Message	Priority	Message Dropped Response
Error	Error	0 (highest)	N/A(1)
WP (Watchpoint Trace)	WPM (Watchpoint Message)	1	N/A(1)
DQ (Data Acquisition)	DQM (Data Acquisition Message)	2	DQM Error Message
Program Trace (PID MSG)	PCM — PID/NPIDR update (Program Correlation Message)	2	OTM Error Message
OT (Ownership)	OTM — PID/NPIDR update (Ownership Trace Message)	2	OTM Error Message(2)

Table 2068. Message Type Priority and Message Dropped Responses (continued)

Message Type	Message	Priority	Message Dropped Response
Program Trace	BTM (Branch Trace Message)	2	BTM Error Message, Sync upgrade next BTM
	Sync (Program Sync Message)	3	Sync upgrade next BTM
	RFM (Resource Full for Instruction counter or history buffer)	4	BTM Error Message Sync upgrade next BTM
	DS (Debug Status Message)	5	Sync upgrade next BTM
	PCM (Program Correlation Message)	6	BTM Error Message Sync upgrade next BTM
DT (Data Trace)	DTM (Data Trace Message)	7	Sync upgrade next DTM
OT (Ownership)	OTM — Periodic update (Ownership Trace Message)	8 (lowest)	none

1. Error and Watchpoint messages are not dropped due to collisions, due to their priority.
2. Message will always be dropped if program trace is enabled, and program correlation messages for PID messages are not masked (Event Code = 0101). No error message is sent for this case since the PID value is contained in the higher priority message.

83.8.5 Data Acquisition Message Priority Loss Response

If a Data Acquisition Message (DQM) loses arbitration due to contention with higher priority messages, an error message will be generated to indicate that a DQM has been lost due to contention.

83.8.6 Ownership Trace Message Priority Loss Response

If an Ownership Trace message (OTM) due to software updates to the Process ID state loses arbitration due to contention with higher priority messages other than a program correlation message with EVCODE = 0101 (PID update), an error message will be generated to indicate that a OTM has been lost due to contention. If the pending OTM is a periodic update, the event is dropped without generating an error message.

83.8.7 Program Trace Message Priority Loss Response

If a Program Trace message (PTM) loses arbitration due to contention with higher priority messages, and the discarded PTM is a Program Correlation message, a Resource Full message for instruction count or history buffer, or a Branch Trace message, then an Error message is generated to indicate that branch trace information has been lost, and the next Branch Trace message will be upgraded to a sync-type message.

If the discarded PTM is a Program Correlation message with PID information (EVCODE=0101), the Error message will indicate a dropped OTM and a dropped Program Trace (Error code = xxxx11xx).

83.8.8 Program Trace Sync Message Priority Loss Response

If a Program Trace Sync message (SYNC) loses arbitration due to contention with higher priority messages, the Sync event is discarded, and the next Branch Trace message will be upgraded to a sync-type message.

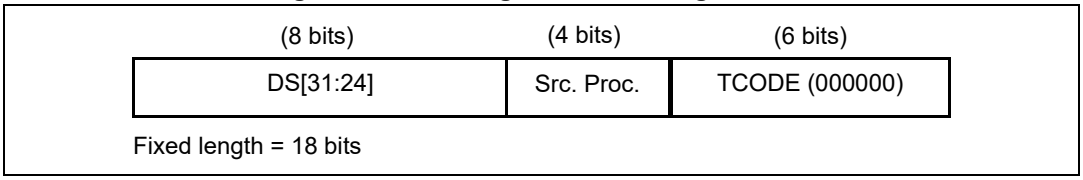
83.8.9 Data Trace Message Priority Loss Response

If a Data Trace message (DTM) loses arbitration due to contention with higher priority messages, the DTM event is discarded, and the next DTM will be upgraded to a sync-type message.

83.9 Debug Status messages

Debug Status Messages report low power mode and debug status. Debug Status messages are enabled when Nexus 3 is enabled. Entering/exiting Debug Mode as well as entering, exiting, or changing Low Power Mode(s) will trigger a Debug Status message, indicating the value of the most significant byte in the Development Status register. Debug status information is sent out in the following format:

Figure 2175. Debug Status message format



83.10 Error messages

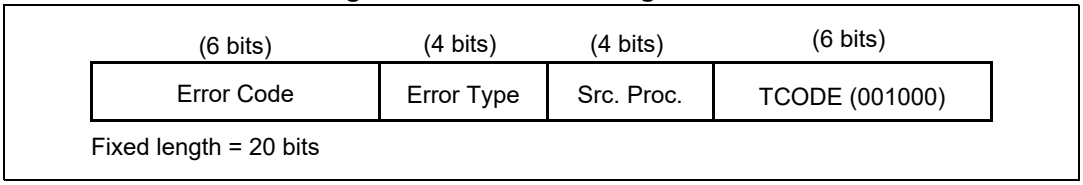
Error messages are enabled whenever the debug logic is enabled. There are two conditions that will produce an error message, each receiving a separate error type designation:

- A message is discarded due to contention with other (higher priority) message types. These errors will have an Error Type value of 1.
- The message queue overruns. After the queue is drained, an error message is enqueued with an error code that indicates what types of messages were discarded during the interim. These errors will have an Error Type value of 0.

Note: The OVCR Register can be used in order to alleviate potential overrun situations.

Error information is messaged out in the following format (also see *Table 2041* and *Table 2042*):

Figure 2176. Error message format



83.11 Ownership trace

This section details the ownership trace features of the Nexus 3 module.

83.11.1 Overview

Ownership trace provides a macroscopic view, such as task flow reconstruction, when debugging software written in a high level (or object-oriented) language. It offers the highest level of abstraction for tracking operating system software execution. This is especially useful when the developer is not interested in debugging at lower levels.

83.11.2 Ownership Trace Messaging (OTM)

Ownership trace information is messaged via the auxiliary port using an Ownership Trace Message (OTM). Core (e200z420n3) processors contain a *PowerISA 2.06* defined “Process ID” register within the CPU. It is updated by the operating system software to provide task/process ID information. The contents of this register are replicated on the pins of the processor and connected to Nexus. The Process ID (PID) register value can be accessed using the **mfspr**/**mtspr** instructions.

Note: The CPU includes a Process ID register (PID0), thus the Nexus UBA functionality is not implemented.

In addition, to decouple trace information from the PID0 register and to provide a full independent 32-bit process ID for debug use, the Nexus PID Register (NPIDR) may be used instead of PID0 to provide OTM process ID information. It is updated by the operating system software to provide task/process ID information. The Nexus Process ID (NPIDR) register value can be accessed using the **mfspr**/**mtspr** instructions. This register is accessible in user or supervisor mode.

Nexus 3 Process ID

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

SPR — 517; Read/Write; Reset: 0

Figure 2177. Nexus 3 Process ID Register (NPIDR)

Note: OS updates to NPIDR should be performed in addition to normal PID0 updates when a process switch occurs, in order to properly generate OTM messages with new process ID information when NPIDR is selected for OTM use.

The process ID source (PID0 or NPIDR) is selected by the setting of the $DC1_{OTS}$ control bit.

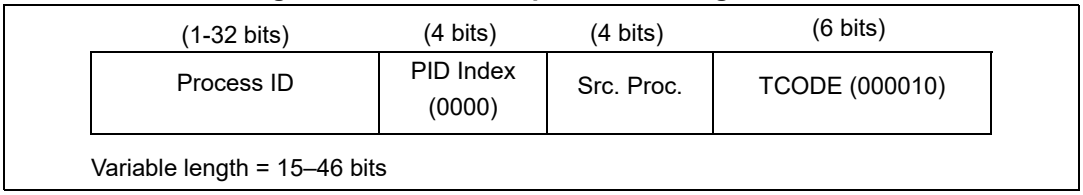
There are two conditions that will cause an Ownership Trace Message when Ownership Trace is enabled:

- When the PID0 register is written to by the processor and $DC1_{OTS}$ indicates PID0 should be used, or when the NPIDR register is written to by the processor and $DC1_{OTS}$ indicates NPIDR should be used, the data is latched within Nexus, and is messaged out via the auxiliary port, allowing development tools to trace ownership flow. However, if Program Trace is enabled, and program correlation messages for PID updates are not masked (Event Code = 0101), then an OTM will not be generated for an update to the selected process ID register, since the program correlation message will provide this process ID update information.
- Periodically, at least once every 256 messages, the most recent state of the selected process ID register is messaged out. The resulting Ownership Trace message will

indicate in the PID Index sub-field that PID0/NPIDR status is being reported and the most recent value of the PID0/NPIDR register will be conveyed in the Process ID value sub-field. These periodic Ownership Trace message events can be disabled by setting $DC1_{POTD}$.

Ownership trace information is messaged out in the following format:

Figure 2178. Ownership Trace Message format



83.12 Program Trace

This section details the program trace mechanism supported by Nexus3 for the e200z420n3 processor. Program trace is implemented via Branch Trace Messaging (BTM) as per the **IEEE-ISTO 5001** standard definition. Branch Trace Messaging for Core (e200z420n3) processors is accomplished by snooping the internal address bus (between the CPU and Cache), attribute signals, and CPU Status (**p_mode[0:3]**, **p_pstat_pipe{0,1}[0:5]**).

83.12.1 Branch Trace Messaging types

Traditional Branch Trace Messaging facilitates program trace by providing the following types of information:

- Messaging for taken direct branches includes how many sequential instructions were executed since the last taken branch or exception, including the taken direct branch. Branch instructions are included in the count of sequential instructions.
- Messaging for taken indirect branches and exceptions includes how many sequential instructions were executed since the last taken branch or exception and the unique portion of the branch target address or exception vector address. Branch instructions are included in the count of sequential instructions. For taken indirect branches that trigger generation of a message, the branch is also included in the count.

Branch History Messaging facilitates program trace by providing the following information.

- Messaging for taken indirect branches and exceptions includes a) how many sequential instructions (I-CNT) were executed since the last predicate instruction, taken/not taken direct branch, taken/not-taken indirect branch, or exception, b) the unique portion of the branch target address or exception vector address, and c) a branch/predicate instruction history field. Each bit in the history field represents a direct branch or predicated instruction where a value of one (1) indicates taken, and a value of zero (0) indicates not taken. (evsel)Not-taken indirect branches will generate a history bit with a value of zero (0). Instructions that generate history bits are not included in instruction counts. For taken indirect branches that trigger generation of this message type, the branch is included in the count, but not in the history field.

83.12.1.1 Indirect Branch Message instructions

Table 2069 shows the types of instructions and events that cause Indirect Branch Messages or Branch History Messages to be encoded.

Table 2069. Indirect Branch Message sources

Source of Indirect Branch Message	Instructions / Detail
Taken branch relative to a register value	**bcctr**, **bcctrl**, **bclr**, **bclrl, se_bctr, se_bctrl, se_blr, se_blrl**
System Call / Trap exceptions taken	se_sc, tw
Return from interrupts / exceptions	se_rfi, se_rfci, se_rfdi
Exit from reset with Program Trace Enabled	Indirect branch with Sync, target address is initial instruction, count = 1

83.12.1.2 Direct Branch Message Instructions

Table 2070 shows the types of instructions that will cause Direct Branch Messages or will toggle a bit in the instruction history buffer to be messaged out in a Resource Full Message or Branch History Message.

Table 2070. Direct Branch Message sources

Source of Direct Branch Message	Instructions
Taken direct branch instructions Instruction Synchronize	**b**, **ba**, **bl**, **bla**, **bc**, **bca**, **bcl**, **bcla, se_b. se_bc, se_bl, e_b, e_bc, e_bl, e_bcl, se_isync**

83.12.1.3 BTM using Branch History Messages

Traditional BTM Messaging can accurately track the number of sequential instructions between branches, but cannot accurately indicate which instructions were conditionally executed, and which were not.

Branch History Messaging solves this problem by providing a predicated instruction history field in each Indirect Branch Message. Each bit in the history represents a predicated instruction or direct branch, or a not-taken indirect branch. A value of one (1) indicates the conditional instruction was executed or the direct branch was taken. A value of zero (0) indicates the conditional instruction was not executed or the branch was not taken. (**evsel**)

Branch History Messages solve predicated instruction tracking and save bandwidth since only indirect branches cause messages to be queued.

83.12.1.4 BTM using Traditional Program Trace Messages

Based on the PTM bit in the DC1 Register, Program Tracing can utilize either Branch History Messages (PTM = 1) or traditional Direct/Indirect Branch Messages (PTM = 0).

Branch History will save bandwidth and keep consistency between methods of Program Trace, yet may lose temporal order between BTM messages and other types of messages. Since direct branches are not messaged, but are instead included in the history field of the Indirect Branch History Message, other types of messages may enter the FIFO between Branch History Messages. The development tool cannot determine the ordering of “events” that occurred with respect to direct branches simply by the order in which messages are sent out.

Traditional BTM messages maintain their temporal ordering because each event that can cause a message to be queued will enter the FIFO in the order it occurred and will be messaged out maintaining that order.

83.12.2 BTM Message For mats

The Nexus 3 block supports three types of traditional BTM Messages — Direct, Indirect, and Synchronization Messages. It supports two types of branch history BTM Messages — Indirect Branch History, and Indirect Branch History with Synchronization Messages.

83.12.2.1 Indirect branch messages (history)

Indirect branches include all taken branches whose destination is determined at run time, interrupts, and exceptions. If $DC1_{PTM}$ is set to '1', indirect branch information is messaged out in the following format:

Figure 2179. Indirect branch message (history) format

(1-32 bits)	(1–32 bits)	(1–8 bits)	(1 bit)	(4 bits)	(6 bits)
Branch History	Relative Address	Sequence Count	(0)	Source Proc.	TCODE (011100)

Max length = 83 bits; Min length = 14 bits

83.12.2.2 Indirect branch messages (traditional)

If $DC1_{PTM}$ is cleared to '0', indirect branch information is messaged out in the following format:

Figure 2180. Indirect branch message format

(1–32 bits)	(1–8 bits)	(1 bit)	(4 bits)	(6 bits)
Relative Address	Sequence Count	(0)	Source Proc.	TCODE (000100)

Max length = 51 bits; Min length = 13 bits

83.12.2.3 Direct branch messages (traditional)

Direct branches (conditional or unconditional) are all taken branches whose destination is fixed in the instruction opcode. Direct branch information is messaged out in the following format:

Figure 2181. Direct branch message format

(1–8 bits)	(4 bits)	(6 bits)
Sequence Count	Src. Proc.	TCODE (000011)

Max length = 18 bits; Min length = 11 bits

Note: When $DC1_{PTM}$ is set, Direct Branch Messages will not be transmitted. Instead, each direct branch, not-taken indirect branch, or predicated instruction will be recorded in the history buffer.

83.12.3 Program Trace Message Fields

The following subsections describe specific fields used for Program Trace messages.

83.12.3.1 Sequential Instruction Count Field (ICNT)

Most of the program trace messages include an instruction count field. For traditional Branch messages, ICNT represents the number of sequential instructions including non-taken branches since the last Direct/Indirect Branch messages. Branch instructions that trigger message generation are included in the ICNT.

For Branch History messages, ICNT represents the number of instructions executed since the last taken/non-taken direct branch, predicate instruction, last taken/not-taken indirect branch, or exception. Branch instructions that trigger message generation are included in the ICNT. Instructions that generate history bits are not included in the ICNT.

The sequential instruction counter overflows after its value reaches 255 and is reset to 0. In addition, the next BTM Message (corresponding to the 256th or later instruction) will be converted to a w/sync type message.

The instruction counter is reset every time the instruction count is transmitted in a message or whenever there is a branch/predicate history event, as well as on exiting from debug mode.

83.12.3.2 Branch/Predicate Instruction History (HIST)

If $DC1_{PTM}$ is set, BTM messaging will use the Branch History format. The branch history (HIST) field in these messages provides a history of branch execution used for reconstructing the program flow. The branch/predicate history buffer stores information about branch and predicate instruction execution. The buffer is implemented as a left-shifting register. The buffer is preloaded with a one (1), which acts as a stop bit (the most significant 1 in the history field is a termination bit for the field). The pre-loaded bit itself is not part of the history, but is transmitted with the packet.

A value of one (1) is shifted into the history buffer for each taken direct branch (program counter relative branch) or predicate instruction whose condition evaluates to true. A value of zero (0) is shifted into the history buffer for each not-taken branch (including indirect branch instructions) or predicate instruction whose condition evaluates to false. For the **evsel** instruction, two bits are shifted in, corresponding to the low element (shifted in first) and the high element (shifted in second) conditions.

This history buffer information is transmitted as part of an Indirect Branch with History message, as part of a Program Correlation message, or as part of a Resource Full message if the history buffer becomes full. The history buffer is reset every time the history information is transmitted in a message, as well as on exiting from debug mode.

Table 2071. Branch/Predicate history events

Branch/Predicate History Event	History Bit(s)	Relevant Instructions
Not taken register indirect branches	0	**bcctr, bcctrl, bclr, bclrl**
Not taken direct branches	0	**b, ba, bc, bca, bla, bcla, bl, bcl**
Taken direct branches	1	**b, ba, bc, bca, bla, bcla, bl, bcl**(1)
evsel instruction	00,01,10, or 11	evsel

1. If the EVCODE for direct branch function calls is not masked in DC4, taken bl and bcl instructions will generate Program Correlation Messages and will not be logged in the history buffer.

83.12.3.3 Execution Mode Indication

In order for a development tool to properly interpret instruction count and history information, it must be aware of the execution mode context of that information. VLE instructions will be interpreted differently from non-VLE instructions.

Program trace messages provide the execution mode status in the least significant bit of the **reconstructed** address field. A value of ‘0’ indicates that preceding instruction count and history information should be interpreted in a non-VLE context. A value of ‘1’ indicates that the preceding instruction count and history information should be interpreted in a VLE context.

83.12.4 Resource Full Messages

The Resource Full Message is used in conjunction with Branch Trace and Branch History Messages. The Resource Full Message is generated when either the internal branch/predicate history buffer is full, or if the BTM Instruction sequence counter (I-CNT) overflows. If synchronization is needed at the time this message is generated, the synchronization is delayed until the next Branch Trace Message that is not a Resource Full Message.

For history buffer overflow, the Resource Full Message transmits a Resource Code (RCODE) of 0b0001 and the current contents of the history buffer, including the stop bit, are transmitted in the Resource Data (RDATA) field. This history information can be concatenated by the development tool with the branch/predicate history information from subsequent messages to obtain the complete branch/predicate history between indirect changes of flow.

For instruction counter overflow, the Resource Full Message transmits an RCODE of 0b0000 and a value of 0xFF is transmitted in the RDATA field, indicating that 255 sequential instructions have been executed since the last change of flow; or, if program trace is in history mode, since the last instruction that recorded history information.

Figure 2182. Resource Full Message format

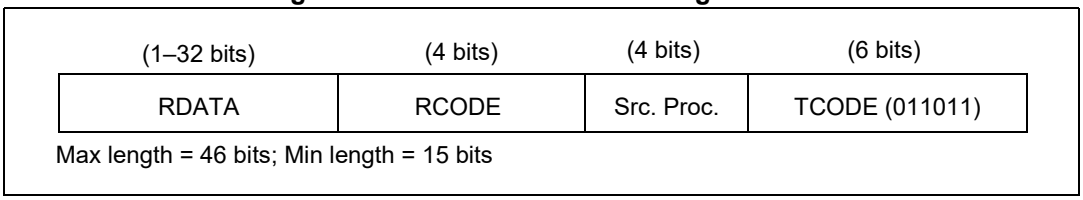


Table 2072 shows the RCODE encodings and RDATA information used for Resource Full Messages.

Table 2072. RCODE encoding

RCODE	Description	RDATA field
0000	Program Trace Instruction counter reached 255 and was reset.	0xFF
0001	Program Trace, Branch / Predicate Instruction History full.	Branch HIstory. This type of packet is terminated by a stop bit set to 1 after the last history bit.

83.12.5 Program Correlation Messages

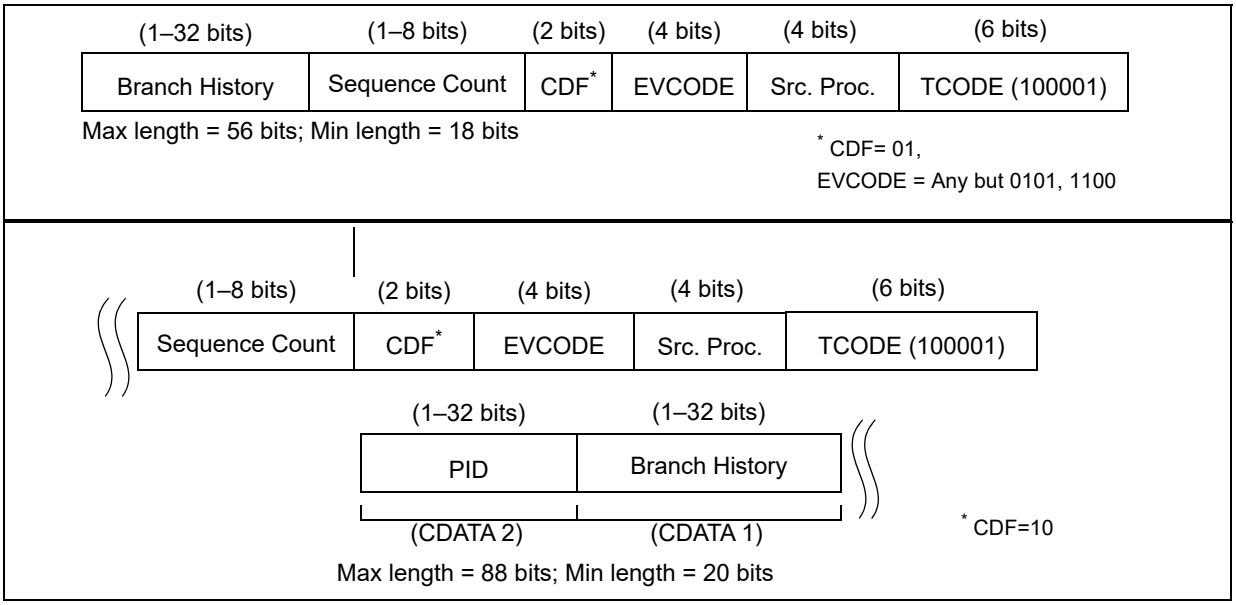
Program Correlation Messages (PCMs) are used to correlate events to the program flow that may or may not be associated with the instruction stream. The following events will result in a PCM when program trace is enabled:

- When the CPU enters debug mode, a PCM is generated. The instruction count and history information provided by the PCM can be used to determine the last sequence of instructions executed prior to debug mode entry.
- When the CPU first enters a low power mode in which instructions are no longer executed, a PCM is generated. The instruction count and history information provided by the PCM can be used to determine the last sequence of instructions executed prior to low power mode entry.
- Whenever program trace is disabled by any means, a PCM is generated. The instruction count and history information provided by the PCM can be used to determine the last sequence of instructions executed prior to disabling program trace.
- When a “Branch and Link” instruction executes (direct branch function call — **bl**/**bcl**/**bla**/**bcla**-type instructions)
- When program trace becomes masked due to MSR_{PMM}=‘0’ and $DC4_{PTMARK}$=‘1’.
- When a write to the process ID register selected by $DC1_{OTS}$ (PID0 or NPIDR) is made via a **mtspr** PID0 or **mtspr** NPIDR.

Refer to *Table 2044* for the event codes that are supported in this implementation. Event code masking is available via the EVCDM field of the DC4 register to allow for control over generation of Program Correlation messages for each event type.

Program Correlation is messaged out in the following formats:

Figure 2183. Program Correlation message formats



83.12.5.1 Program Correlation message generation for PID updates

When a (potentially) new value is established in the selected process ID register via a **mtspr** PID0/NPIDR, a Program Correlation message is generated containing the information regarding the new process ID value. This PCM also contains the current history

and instruction count. The message is provided so that address translation information can be processed by the development tool in the proper program flow. The **mtspr** PID0/NPIDR is included in the instruction count information. Note that Ownership Trace Messages (other than the periodic OTM) are redundant with the information provided, and may be disabled to avoid unnecessary message bandwidth or collisions.

83.12.6 Program Trace Overflow Error messages

an Error Message occurs when a new message cannot be queued due to the message queue being full. The FIFO will discard incoming messages until it has completely emptied the queue. Once emptied, an Error Message will be queued. The error encoding will indicate which type(s) of messages attempted to be queued while the FIFO was being emptied.

83.12.7 Program Trace Synchronization messages

By default, program trace messages will perform XOR compression on the branch target address to produce the address field for the message. This compression is consistent with the specification in IEEE-ISTO 5001.

Under some conditions an uncompressed address is sent to provide development tools with a baseline reference address. A Program Trace Synchronization message is messaged via the auxiliary port (provided Program Trace is enabled) for the following conditions (see *Table 2073*):

- Initial Program Trace message after exit from system reset or whenever program trace is enabled.
- Upon exiting from a CPU Low Power state.
- Upon exiting from Debug Mode.
- Upon occurrence of queue overrun (can be caused by any trace message), provided Program Trace is enabled.
- Upon assertion of the Event In (**nex_evti_b**) pin if the EIC bits within the DC1 Register have enabled this feature.
- When program trace becomes unmasked due to $MSR_{PMM} \rightarrow$ ‘1’ with $DC4_{PTMARK}$=‘1’.

Note that the ICNT information for these messages is driven to zero, thus will not always be meaningful for some of these cases.

The format for Program Trace Synchronization messages is as follows:

Figure 2184. .Program Trace Synchronization message format

(1–32 bits)	(1–8 bits)	(1 bit)	(4 bits)	(6 bits)
Full Target Address	Seq. Count (0)	(0)	Source Proc.	TCODE (001001)

Max length = 51 bits; Min length = 13 bits

Exception conditions that result in Program Trace Synchronization are summarized in *Table 2073*.

Table 2073. Program Trace exception summary

Exception Condition	Exception Handling
System Reset Negation	At the negation of JTAG reset (**j_trst_b**), queue pointers, counters, state machines, and registers within the Nexus 3 module are reset. Upon exiting system reset (**p_reset_b**), if Program Trace is already enabled, a Program Trace Sync Message is sent.
Program Trace Enabled	The first Program Trace Message (after Program Trace has been enabled or re-enabled) is a synchronization message.
Exit from Low Power/Debug	Upon exit from a Low Power mode or Debug mode, a Program Trace Sync Message is sent.
Queue Overrun	An Error Message occurs when a new message cannot be queued due to the message queue being full. The FIFO will discard messages until it has completely emptied the queue. Once emptied, an Error Message will be queued. The error encoding will indicate which type(s) of messages attempted to be queued while the FIFO was being emptied. The next BTM message in the queue will be a Program Trace Sync Message.
Event In	If the Nexus module is enabled, a **nex_evti_b** assertion initiates a Program Trace Sync Message if Program Trace is enabled and the EIC bits of the DC1 Register have enabled this feature.

Note that for cases where program trace sync messages are generated due to program trace being enabled, the address contained in the sync message may either be the address of the instruction that caused program trace to be enabled, or may be the address of the first instruction of an exception handler that is executed in response to unsuccessful completion of that instruction.

83.12.8 Program Trace Direct/Indirect Branch with Sync messages

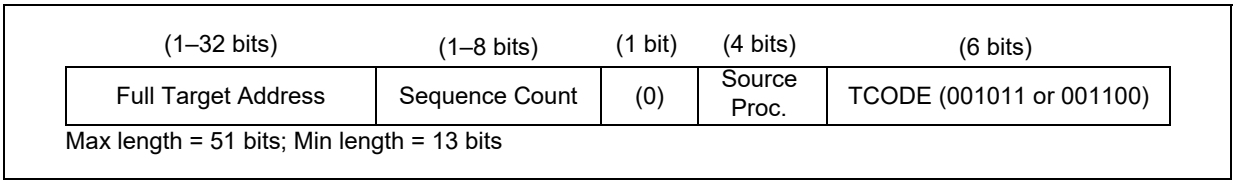
Under some conditions an uncompressed address is sent to provide development tools with a baseline reference address. A Program Trace Synchronization or a Direct/Indirect Branch with Sync message is messaged via the auxiliary port (provided Program Trace is enabled) for the following conditions (see *Table 2073*):

- Upon direct/indirect branch after a Program Sync Message was lost due to a collision while attempting to enter the message queue.
- Upon direct/indirect branch after the sequential instruction counter has expired indicating 255 instructions have occurred since the last change of flow.
- When the periodic program trace counter has expired indicating 255 *without-sync* Program Trace messages have occurred since the last *with-sync* message occurred.

Note that the ICNT and History information for the first message will not always be meaningful, since the temporary masking of program trace may result in ambiguous values. Subsequent w/sync messages will not have this issue.

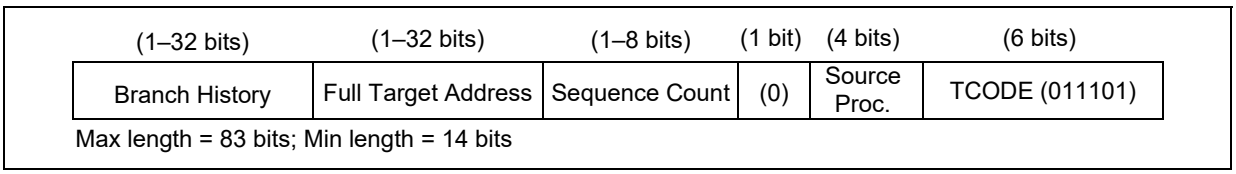
The format for Program Trace Direct/Indirect Branch with Sync messages is as follows:

Figure 2185. Direct/Indirect Branch with Sync message format



The format for Program Trace Indirect Branch History with Sync. Messages is as follows:

Figure 2186. Indirect Branch History w/ Sync message format



Exception conditions that result in Program Trace Synchronization using a w/Sync message type are summarized in *Table 2074*.

Table 2074. Program Trace Exception Summary for w/Sync BTM messages

Exception Condition	Exception Handling
Periodic Program Trace Sync.	A forced synchronization occurs periodically after 255 non-sync Program Trace Messages have been queued. A Direct/Indirect Branch w/ Sync. Message is queued. The periodic program trace message counter then resets.
Sequential Instruction Count Overflow	After the sequential instruction counter reaches its maximum count (up to 255 sequential instructions may be executed), a forced synchronization occurs. The sequential counter then resets. A Program Trace Direct/Indirect Branch w/ Sync.Message is queued upon execution of the next branch. A Resource Full Message is Queued on the overflow event. If a branch instruction is the 255th instruction to occur, and causes a Program Trace message to be queued, then no Resource Full Message is queued, and the w/Sync message will be queued for the *next* Program Trace Direct/Indirect Branch Message.
Collision Priority	All Messages have the following priority: Instruction 0 (WPM → DQM → PCM_{PIDMSG}→OTM → BTM → DTM)→ Instruction1 (WPM → DQM → OTM → BTM → DTM), where instruction0 is the oldest instruction. A BTM Message from Instruction1 that attempts to enter the queue at the same time as three higher priority messages from either instruction will be lost. An Error Message will be sent indicating the BTM was lost. The following direct/indirect branch will queue a Direct/Indirect Branch w/ Sync. Message. The count value within this message will reflect the number of sequential instructions executed after the last successful BTM Message was generated. This count will include the branch that did not generate a message due to the collision.

83.12.9 Enabling Program Trace

Program Trace Messaging can be enabled in one of three ways:

- Setting the TM field of the DC1 Register to enable Program Trace
- Using the PTS field of the WT Register to enable Program Trace on Watchpoint hits (watchpoints are configured within the CPU)
- Using the external hardware trace enable input signal (nex_ptm_starttr)
- Filtering of Program Trace messages may be performed using the MSR_{PMM} bit and the setting of $DC4_{PTMARK}$

83.12.10 Program Trace Timing Diagrams (2 MDO / 1 MSEO configuration)

Figure 2187. Program Trace — Indirect Branch message (Traditional)

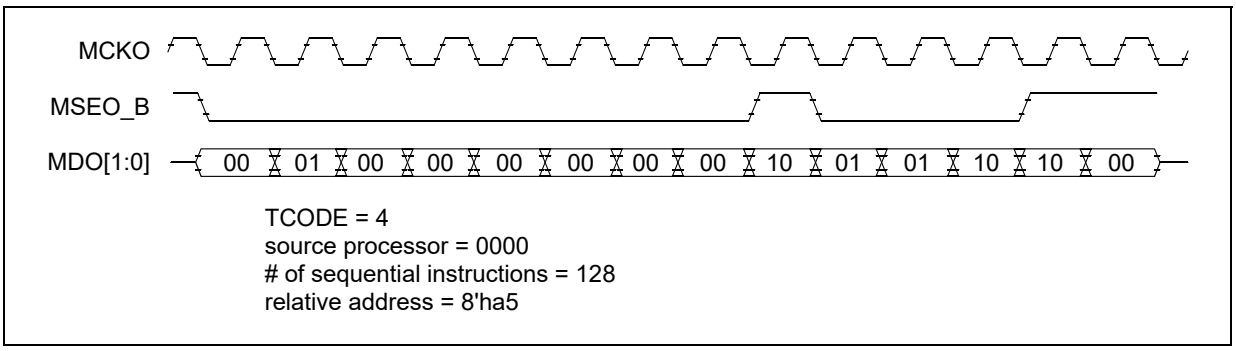


Figure 2188. Program Trace — Indirect Branch message (history)

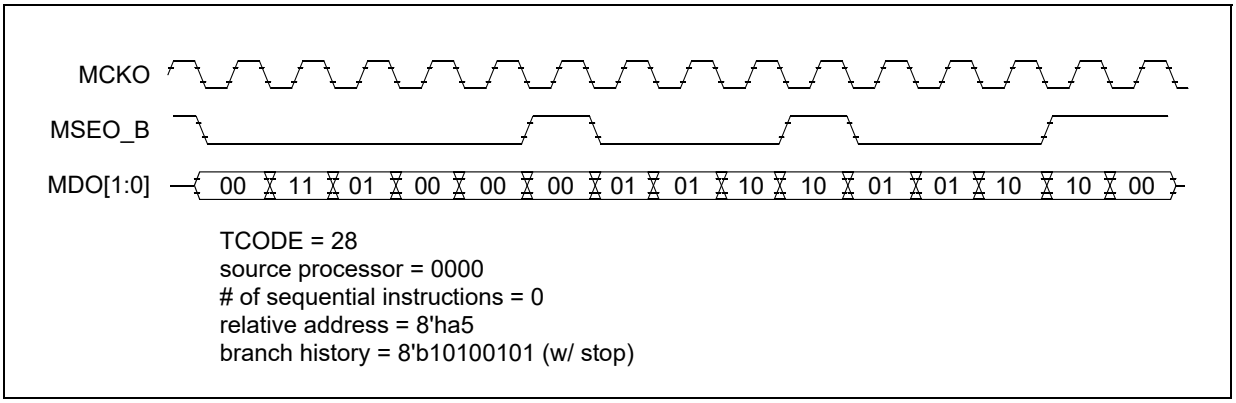


Figure 2189. Program Trace — Direct Branch (traditional) & Error messages

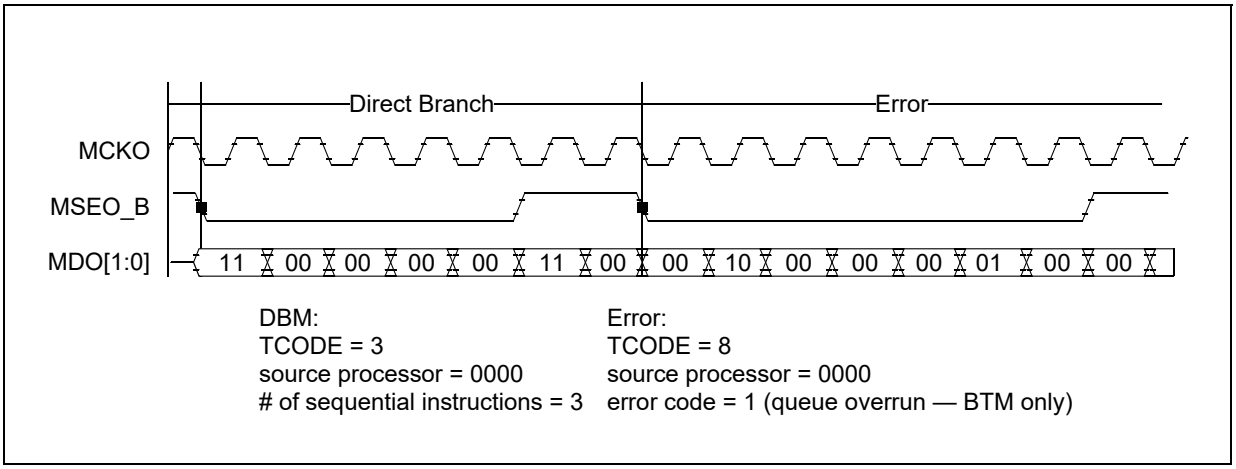
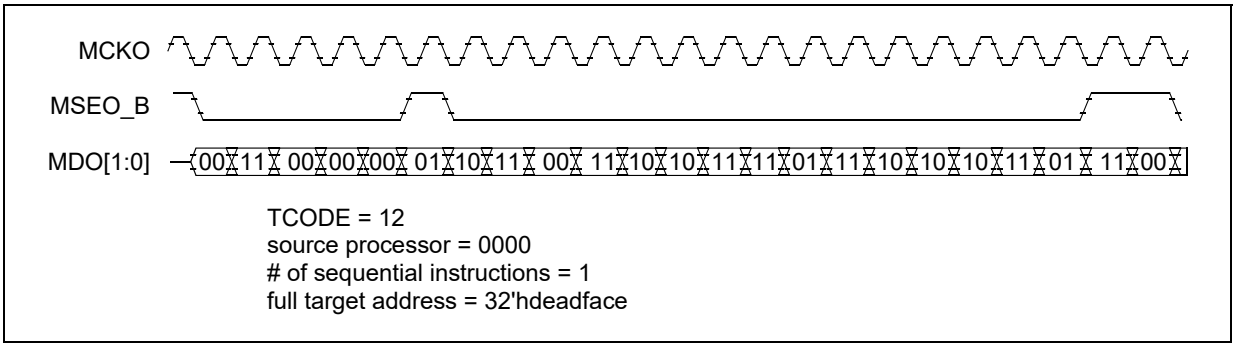


Figure 2190. Program Trace — Indirect Branch w/ Sync message



83.13 Data Trace

This section deals with the Data Trace mechanism supported by the Nexus 3 module. Data Trace is implemented via Data Write Messaging (DWM) and Data Read Messaging (DRM), as per the **IEEE-ISTO 5001** standard.

83.13.1 Data Trace Messaging (DTM)

Data Trace Messaging is accomplished by snooping the internal address and data buses, and storing the information for qualifying accesses (based on enabled features and matching target addresses). The Nexus 3 module traces all data access that meet the selected range and attributes.

Note: Data Trace is only performed on the internal data buses. This allows for data visibility for Core (e200z420n3) processors that incorporate a data cache. Only CPU initiated accesses will be traced. No DMA accesses to the AHB system bus will be traced.

Data Trace Messaging can be enabled in one of three ways.

- Setting the TM field of the DC1 Register to enable Data Trace.
- Using the DTS field of the WT Register to enable Data Trace on Watchpoint hits (watchpoints are configured within the Nexus1 module).
- Using the external hardware trace enable signal (**nex_dtm_starttr**)

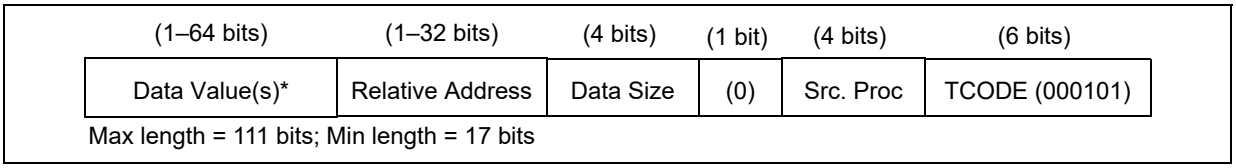
83.13.2 DTM Message formats

The Nexus 3 block supports five types of DTM Messages — Data Write, Data Read, Data Write Synchronization, Data Read Synchronization, and Error Messages.

83.13.2.1 Data Write messages

The Data Write message contains the data write value and the address of the write access, relative to the previous Data Trace message. Data Write message information is messaged out in the following format:

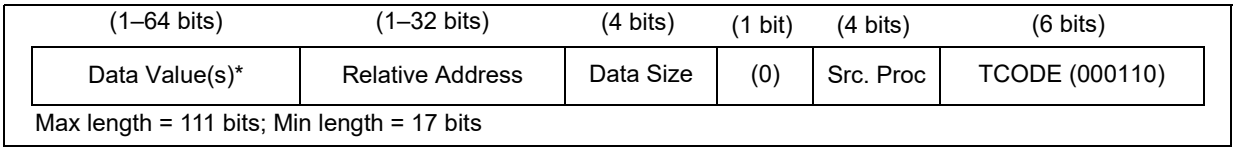
Figure 2191. Data Write message format



83.13.2.2 Data Read messages

The Data Read message contains the data read value and the address of the read access, relative to the previous Data Trace message. Data Read message information is messaged out in the following format:

Figure 2192. Data Read message format



*Note: Core (e200z420n3)-based CPUs are capable of generating two (2) reads or writes per clock cycle in cases where multiple registers are accessed with a single instruction (lmw/stmw). These will have a double word pair size encoding (**p_tsiz** = 0b000). In these cases, the Nexus 3 module will send one (1) Data Trace Message with the two 32-bit data values as one combined 64-bit value for each message.*

*For Core (e200z420n3)-based CPUs, the doubleword encoding (**p_tsiz** = 0b000) may also indicate a doubleword access and will be sent out as a single Data Trace Message with a single 64-bit data value.*

The debug/development tool will need to distinguish the two cases based on the family of processor.

83.13.2.3 Data Trace Synchronization messages

A Data Trace Write/Read with Sync. message is messaged via the auxiliary port (provided Data Trace is enabled) for the following conditions (see *Table 2075*):

- Initial Data Trace Message after exit from system reset or whenever Data Trace is enabled.
- Upon returning from a CPU Low Power state.
- Upon returning from Debug Mode.
- After occurrence of queue overrun (can be caused by any trace message), provided Data Trace is enabled.
- After the periodic data trace counter has expired indicating 255 *without-sync* Data Trace messages have occurred since the last *with-sync* message occurred.
- Upon assertion of the Event In (**nex_evti_b**) pin, the first Data Trace Message will be a synchronization message if the EIC bits of the DC1 Register have enabled this feature.
- Upon Data Trace Write/Read after the previous DTM message was lost due to an attempted access to a secure memory location (for SOC's w/ security).
- Upon Data Trace Write/Read after the previous DTM message was lost due to a collision entering the FIFO between the DTM message and any two of the following: Watchpoint message, Ownership Trace message, or Program Trace message.

Data Trace Synchronization Messages provide the full address (without leading zeros) and insure that development tools fully synchronize with Data Trace regularly. Synchronization messages provide a reference address for subsequent DTMs, in which only the unique portion of the Data Trace address is transmitted. The format for Data Trace Write/Read with Sync. Messages is as follows:

Figure 2193. Data Write/Read with Sync message format

(1–64 bits)	(1–32 bits)	(4 bits)	(1 bit)	(4 bits)	(6 bits)
Data Value	Full Address	Data Size	(0)	Source Proc.	TCODE (001101 or 001110)

Max length = 111 bits; Min length = 17 bits

Exception conditions that result in Data Trace Synchronization are summarized in *Table 2075*.

Table 2075. Data Trace Exception summary

Exception Condition	Exception Handling
System Reset Negation	At the negation of JTAG reset (**j_trst_b**), queue pointers, counters, state machines, and registers within the Nexus 3 module are reset. If Data Trace is enabled, the first Data Trace Message is a Data Write/Read w/ Sync. Message.
Data Trace Enabled	The first Data Trace Message (after Data Trace has been enabled) is a synchronization message.
Exit from Low Power/Debug	Upon exit from a Low Power mode or Debug mode the next Data Trace Message will be converted to a Data Write/Read with Sync. Message.

Table 2075. Data Trace Exception summary (continued)

Exception Condition	Exception Handling
Queue Overrun	An Error Message occurs when a new message cannot be queued due to the message queue being full. The FIFO will discard messages until it has completely emptied the queue. Once emptied, an Error Message will be queued. The error encoding will indicate which type(s) of messages attempted to be queued while the FIFO was being emptied. The next DTM message in the queue will be a Data Write/Read w/ Sync. Message.
Periodic Data Trace Sync.	A forced synchronization occurs periodically after 255 Data Trace Messages have been queued. A Data Write/Read w/ Sync. Message is queued. The periodic data trace message counter then resets.
Event In	If the Nexus module is enabled, a **nex_evti_b** assertion initiates a Data Trace Write/Read w/ Sync. Message upon the next data write/read (if Data Trace is enabled and the EIC bits of the DC1 Register have enabled this feature).
Collision Priority	All Messages have the following priority: Instruction 0 (WPM → DQM → PCM_{PIDMSG} → OTM → BTM → DTM)→ Instruction1 (WPM → DQM → OTM → BTM → DTM), where instruction0 is the oldest instruction. A DTM Message that attempts to enter the queue at the same time as three other higher priority messages will be lost. A subsequent read/write will queue a Data Trace Read/Write w/ Sync. Message.

83.13.3 DTM operation

83.13.3.1 Data Trace windowing

Data Write/Read Messages are enabled via the RWT field in the Data Trace Control Register (DTC) for each DTM channel. Data Trace windowing is achieved via the address range defined by the DTEA and DTSA Registers and by the RC field in the DTC register. All CPU initiated read/write accesses that fall inside or outside these address ranges, as programmed, are candidates to be traced.

83.13.3.2 Data Access / Instruction Access Data Tracing

The Nexus3 module is capable of tracing either instruction access data or data access data and can be configured for either type of data trace by setting the DI1 field within the Data Trace Control Register. This setting applies to all DTM channels.

83.13.3.3 Data Trace filtering

Data Trace filtering is available base on the settings of MSR_{PMM} and $DC4_{DTMARK}$.

83.13.3.4 Bus cycle special cases

Table 2076. Bus cycle special cases

Special case	Action
Bus cycle aborted	Cycle ignored
Bus cycle with data error ($\overline{TEA}$)(1)	Data Trace Message discarded

Table 2076. Bus cycle special cases (continued)

Special case	Action
Bus cycle completed without error(1)	Cycle captured & transmitted
AHB bus cycle initiated by Nexus 3	Cycle ignored
Bus cycle is an instruction fetch	Cycle selectively ignored based on DTC_{DI} setting
Bus cycle accesses misaligned data (across 64-bit boundary) — both 1st & 2nd transactions within data trace range	1st & 2nd cycle captured & a single or a pair of DTM(s) is (are) transmitted (see Note)
Bus cycle accesses misaligned data (across 64-bit boundary) — 1st transaction within data trace range — 2nd transaction out of data trace range	1st & 2nd cycle captured & a single or a pair of DTM(s) is (are) transmitted (see Note)
Bus cycle accesses misaligned data (across 64-bit boundary) — 1st transaction within data trace range — 2nd transaction (regardless of within range or not) receives a bus error	Data Trace Message discarded
Bus cycle accesses misaligned data (across 64-bit boundary) — 1st transaction out of data trace range — 2nd transaction within data trace range	1st & 2nd cycle captured & a single or a pair of DTM(s) is (are) transmitted (see Note)
Bus cycle accesses misaligned data (across 64-bit boundary) — 1st transaction out of data trace range — 2nd transaction within range, receives a bus error	Data Trace Message discarded

1. Buffering of stores in the CPU store buffer may generate a DTM prior to the actual memory access, regardless of an error termination condition from memory.

Note: For misaligned accesses (crossing 64-bit boundary), the access is broken into two accesses by the CPU. If either access is within the data trace range, a single DTM will be sent with a size encoding indicating the size of the original access (that is word), and the address indicating the original misaligned accesses, unless the misaligned access wraps into the doubleword at address 0. In this case, since the two portions of the misaligned access are not contiguous, two DTMs will be sent, one for each portion. The size encodings and the addresses of the DTMs will indicate the accessed bytes of data. The data trace port handles these cases in the same manner. (See the Data Trace Port section in the Core (e200z420n3) Core Debug Support chapter.)

Note: A store to the cache's store buffer within the data trace range may initiate a DTM message prior to completion of the actual memory access.

83.13.4 Data Trace Timing Diagrams (8 MDO / 2 MSEO configuration)

Figure 2194. Data Trace — Data Write message

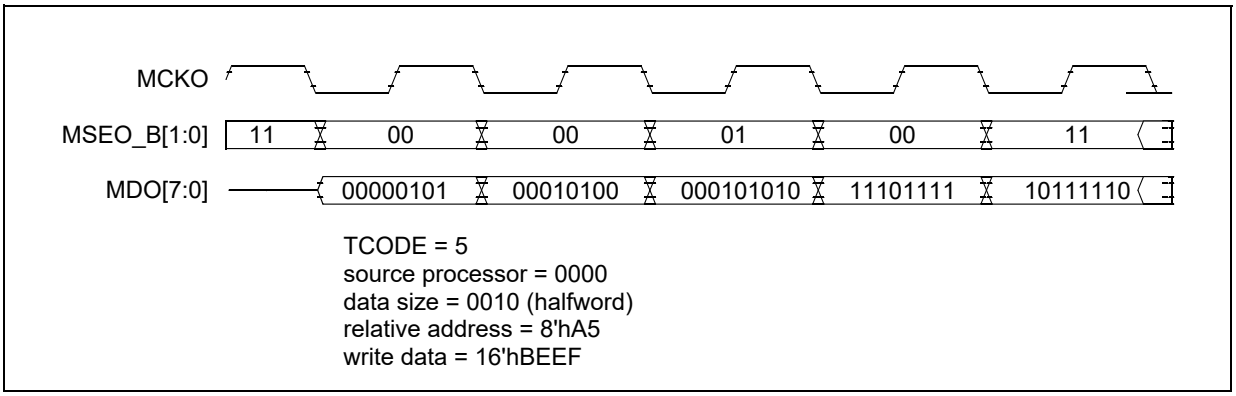
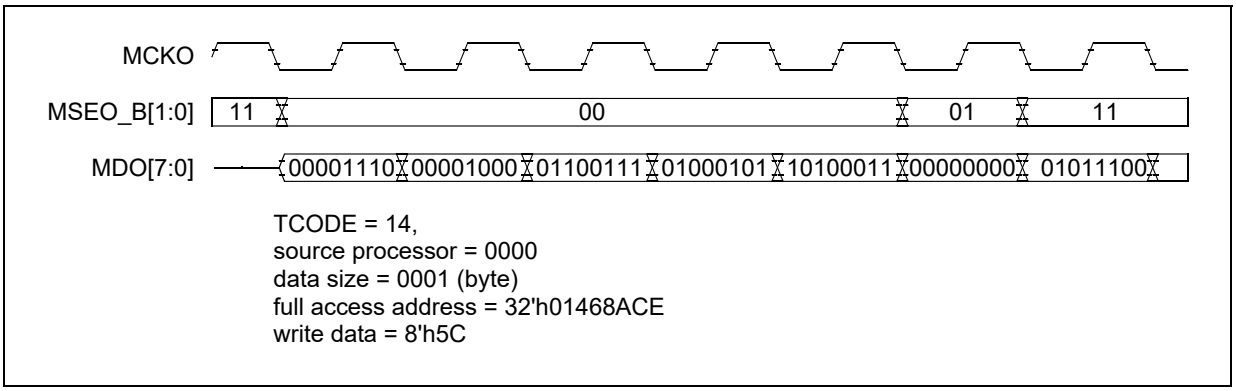


Figure 2195. Data Trace — Data Read w/ Sync message



83.14 Data Acquisition messaging

this section details the Data Acquisition mechanisms supported by the Nexus 3 module. Data Acquisition Trace is implemented using Data Acquisition Trace Messages in accordance with IEEE-ISTO 5001 definitions. The control mechanism to export the data is different from the recommendations of the standard, however.

Data Acquisition Trace provides a convenient and flexible mechanism for the debugger to observe the architectural state of the machine through software instrumentation.

83.14.1 Data Acquisition ID Tag field

The DQTAG Tag field (DQTAG) is an 8-bit value specifying control or attribute information for the data included in the Data Acquisition message. DQTAG is sampled from $DEVENT_{DQTAG}$ when a write to DDAM is performed via **mtspr** operations. The usage of the DQTAG is left to the discretion of the development tool to be used in whatever manner is deemed appropriate for the application.

83.14.2 Data Acquisition Data field

The Data Acquisition Data field (DQDATA) is the data captured from the DDAM write operation via **mtspr** operations. Leading zeros are omitted from the message.

83.14.3 Data Acquisition Trace event

For DQM, a dedicated SPR has been allocated (DDAM). It is expected that the general use case is to instrument the software and use **mtspr** operations to generate Data Acquisition messages.

There is no explicit error response for failed accesses as a result of contention between an internal and external debugger. Software may be blocked or given ownership of DDAM and the DQTAG field of the DEVENT register via control in EDBRAC0 while in External Debug Mode. Hardware always has access to these registers. Refer to the “External Debug Resource Allocation Control Register (EDBRAC0)” section in the Core (e200z420n3) Core Debug Support chapter for more detail on EDBRAC0.

Reads from the Data Acquisition channel do not generate a Data Acquisition event and will return zeroes for the read data.

Table 2077. .Data Acquisition message format

(1–32 bits)	(8 bits)	(4 bits)	(6 bits)
DQDATA	DQTAG	Src. Proc.	TCODE (011011)

Max length = 50 bits; Min length = 19 bits

83.15 Watchpoint Trace messaging

Enabling Watchpoint messaging is done by setting the Watchpoint Trace Enable bit in the DC1 Register. Setting the individual Watchpoint sources is supported through the Nexus1 module and the Performance Monitor unit. The Nexus1 module is capable of setting multiple types of watchpoints. Please refer to the Core (e200z420n3) Core Debug Support chapter for details on watchpoint initialization.

When watchpoints occur due to one or more asserted watchpoint event signals and Watchpoint Trace Messaging is enabled, a Watchpoint Trace message will be sent to the message queue to be messaged out. This message includes the watchpoint number indicating which watchpoint(s) caused the message. If more than one enabled watchpoint occurs in a single cycle, only one Watchpoint Trace message is generated and multiple bits of the watchpoint hit field will be set. The settings of the $WMSK_{WEM}$ field control which watchpoints are enabled to generate watchpoint trace messages.

The occurrence of any of the defined watchpoints can also be programmed to assert the Event Out (**nex_evto_b**) pin for one (1) period of the output clock (**nex_mcko**) based on settings in the DC2 and DC3 registers. See *Table 2081* for details on **nex_evto_b**.

Watchpoint information is messaged out in the following format:

Figure 2196. Watchpoint message format

(1–30 bits)	(4 bits)	(6 bits)
Watchpoint Source	Src. Proc.	TCODE (001111)

Variable length = 11–40 bits

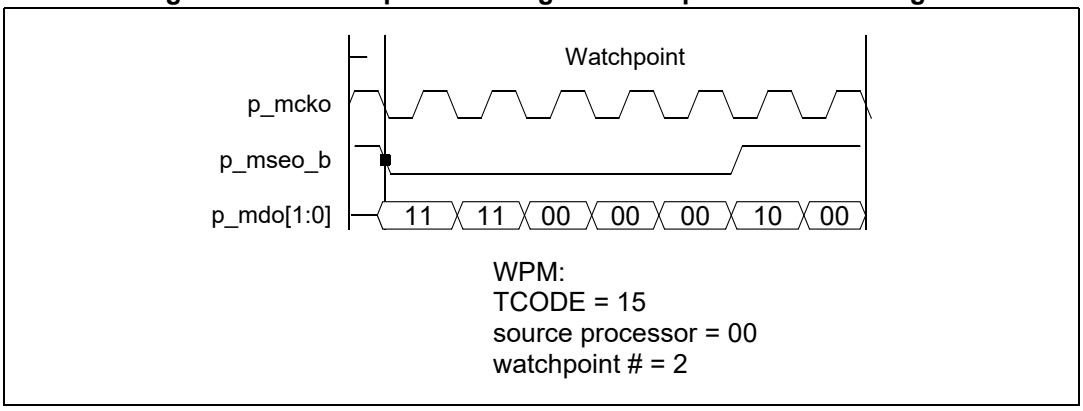
The Watchpoint Source message field will contain a ‘1’ for each asserted watchpoint. Leading zeros are truncated.

Table 2078. Watchpoint source encoding

Watchpoint source (1–30 bits)	Watchpoint description
00000000000000000000000000000000	No Watchpoints enabled for Watchpoint Trace Messaging
*xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx*1	Watchpoint #0 enabled for WTM
*xxxxxxxxxxxxxxxxxxxxxxxxxxxxxx*1*x*	Watchpoint #1 enabled for WTM
*xxxxxxxxxxxxxxxxxxxxxxxxxxxxx*1*xx*	Watchpoint #2 enabled for WTM
*xxxxxxxxxxxxxxxxxxxxxxxxxxxx*1*xxx*	Watchpoint #3 enabled for WTM
*xxxxxxxxxxxxxxxxxxxxxxxxxxx*1*xxxx*	Watchpoint #4 enabled for WTM
*xxxxxxxxxxxxxxxxxxxxxxxxxx*1*xxxxx*	Watchpoint #5 enabled for WTM
*xxxxxxxxxxxxxxxxxxxxxxxxx*1*xxxxxx*	Watchpoint #6 enabled for WTM
*xxxxxxxxxxxxxxxxxxxxxxxx*1*xxxxxxx*	Watchpoint #7 enabled for WTM
*xxxxxxxxxxxxxxxxxxxxxxx*1*xxxxxxxx*	Watchpoint #8 enabled for WTM
*xxxxxxxxxxxxxxxxxxxxxx*1*xxxxxxxxx*	Watchpoint #9 enabled for WTM
*xxxxxxxxxxxxxxxxxxxxx*1*xxxxxxxxxx*	Watchpoint #10 enabled for WTM
*xxxxxxxxxxxxxxxxxxxx*1*xxxxxxxxxxx*	Watchpoint #11 enabled for WTM
*xxxxxxxxxxxxxxxxxxx*1*xxxxxxxxxxxx*	Watchpoint #12 enabled for WTM
*xxxxxxxxxxxxxxxxxx*1*xxxxxxxxxxxxx*	Watchpoint #13 enabled for WTM
*xxxxxxxxxxxxxxxxx*1*xxxxxxxxxxxxxx*	Watchpoint #14 enabled for WTM
*xxxxxxxxxxxxxxxx*1*xxxxxxxxxxxxxxx*	Watchpoint #15 enabled for WTM
*xxxxxxxxxxxxxxx*1*xxxxxxxxxxxxxxxx*	Watchpoint #16 enabled for WTM
*xxxxxxxxxxxxxx*1*xxxxxxxxxxxxxxxx*	Watchpoint #17 enabled for WTM
*xxxxxxxxxxxxx*1*xxxxxxxxxxxxxxxx*	Watchpoint #18 enabled for WTM
*xxxxxxxxxxxx*1*xxxxxxxxxxxxxxxx*	Watchpoint #19 enabled for WTM
*xxxxxxxxxxx*1*xxxxxxxxxxxxxxxx*	Watchpoint #20 enabled for WTM
*xxxxxxxxxx*1*xxxxxxxxxxxxxxxx*	Watchpoint #21 enabled for WTM
*xxxxxxxxx*1*xxxxxxxxxxxxxxxx*	Watchpoint #22 enabled for WTM
*xxxxxxxx*1*xxxxxxxxxxxxxxxx*	Watchpoint #23 enabled for WTM
*xxxxxxx*1*xxxxxxxxxxxxxxxx*	Watchpoint #24 enabled for WTM
*xxxxxx*1*xxxxxxxxxxxxxxxx*	Watchpoint #25 enabled for WTM
*xxxxx*1*xxxxxxxxxxxxxxxx*	Watchpoint #26 enabled for WTM
*xxxx*1*xxxxxxxxxxxxxxxx*	Watchpoint #27 enabled for WTM
*xxx*1*xxxxxxxxxxxxxxxx*	Watchpoint #28 enabled for WTM
*xx*1*xxxxxxxxxxxxxxxx*	Watchpoint #29 enabled for WTM

83.15.1 Watchpoint Timing Diagram (2 MDO / 1 MSEO configuration)

Figure 2197. Watchpoint Message & Watchpoint Error message



83.16 External Hardware Trigger Controls

The Nexus 3 module provides a set of hardware trigger inputs to allow for external hardware to also independently control enabling/disabling of Program Trace, Data Trace, Ownership Trace, and Watchpoint Trace messaging. These signals can be used to produce a “window” for trace messaging for each type.

When a start trigger condition is detected for one of the trace types, and the corresponding trace type has not already been enabled by a software write to the DC1 register, the designated trace features become enabled, and the corresponding enable bit(s) of the DC1 register is set. When a stop trigger condition is detected and the corresponding trace type has not already been enabled by a software write to the DC1 register, the designated trace feature becomes disabled, and the corresponding enable bit of the DC1 register is cleared. If a simultaneous trigger condition is received for both start and stop triggering, then the designated trace feature will toggle between enabled and disabled depending on the enable state at the time of the trigger events. For example, if tracing is enabled, and a start and stop trigger condition occur simultaneously, then tracing will be disabled. Direct writes of the DC1 register take precedence over any trace feature control action that is derived from external hardware triggering, and if the corresponding trace feature has been enabled by a direct write to DC1, then the corresponding hardware trigger signals will have no effect. *Table 2079* details the external hardware trigger signals.

Table 2079. External Hardware Trigger Signals

Name	Description
nex_dtm_starttr	Data Trace Messaging start trigger input. A transition from 0→1 on this input signals a start trace condition.
nex_dtm_stoptr	Data Trace Messaging stop trigger input. A transition from 0→1 on this input signals a stop trace condition.
nex_otm_starttr	Ownership Trace Messaging start trigger input. A transition from 0→1 on this input signals a start trace condition.
nex_otm_stoptr	Ownership Trace Messaging stop trigger input. A transition from 0→1 on this input signals a stop trace condition.

Table 2079. External Hardware Trigger Signals (continued)

Name	Description
nex_ptm_starttr	Program Trace Messaging start trigger input. A transition from 0→1 on this input signals a start trace condition.
nex_ptm_stoptr	Program Trace Messaging stop trigger input. A transition from 0→1 on this input signals a stop trace condition.
nex_wtm_starttr	Watchpoint Trace Messaging start trigger input. A transition from 0→1 on this input signals a start trace condition.
nex_wtm_stoptr	Watchpoint Trace Messaging stop trigger input. A transition from 0→1 on this input signals a stop trace condition.

These signals are active-high inputs and are transition sensitive. A 0→1 transition on the signal indicates the corresponding condition is requested. These signals are not synchronized, and must meet setup and hold requirements relative to **nex_clk**.

83.17 Nexus 3 Read/Write access to memory-mapped resources

The Read/Write access feature allows access to memory-mapped resources via the JTAG/OnCE port. The Read/Write mechanism supports single as well as block reads and writes to AHB resources.

The Nexus 3 module is capable of accessing resources on the system bus (AHB). Memory-mapped registers and other non-cached memory can be accessed via the standard memory map settings.

All accesses are setup and initiated by the Read/Write Access Control/Status Register (RWCS), as well as the Read/Write Access Address (RWA) and Read/Write Access Data Registers (RWD). Nexus 3 read/write accesses are run as privileged data non-cacheable accesses by default, and drive the **p_d_hprot[5:0]** bus access attributes to 6‘b000011 accordingly. The $RWCS_{ATTR}$ field is provided to allow a portion of these default values to be modified when performing read or write accesses using the Nexus 3 Read/Write access mechanism.

Using the Read/Write Access Registers (RWCS/RWA/RWD), memory mapped AHB resources can be accessed through Nexus 3. The following subsections describe the steps required to access memory-mapped resources.

Note: Read/Write Access can only access memory mapped resources when system reset is de-asserted and clocks are running.

Misaligned accesses are NOT supported in the Nexus 3 module.

Uncorrectable ECC errors on Nexus 3 read access will result in the RWD register being updated with the raw data received.

83.17.1 Single write access

1. Initialize the Read/Write Access Address Register (RWA) through the access method outlined in *Section 83.5: Nexus 3 Register Access via JTAG/OnCE*. Configure as follows:
   a) Write Address → 32h'xxxxxxxx (write address)
2. Initialize the Read/Write Access Control/Status Register (RWCS) through the access method outlined in *Section 83.5: Nexus 3 Register Access via JTAG/OnCE*. Configure the bits as follows:
   a) Access Control (AC) → 1b'1 (to indicate start access)
   b) Map Select (MAP) → 3b'000 (primary memory map)
   c) Access Priority (PR) → 2b'11 (highest priority)
   d) Read/Write (RW) → 1b'1 (write access)
   e) Word Size (SZ) → 3b'0xx (32-bit, 16-bit, 8-bit)
   f) Access Count (CNT) → 14h'0000 or 14h'0001(single access)

Note: Access Count (CNT) of 14'h0000 or 14'h0001 will perform a single access.

3. Initialize the Read/Write Access Data Register (RWD) through the access method outlined in *Section 83.5: Nexus 3 Register Access via JTAG/OnCE*. Configure as follows:
   a) Write Data → 32h'xxxxxxxx (write data)
4. The Nexus block will then arbitrate for the AHB system bus and transfer the data value from the data buffer RWD Register to the memory mapped address in the Read/Write Access Address Register (RWA). The **nex_ahb_start** output will be asserted during the first clock of the address phase of the transfer. When the access has completed, Nexus clears the AC and DV bits in the RWCS Register. If the access has completed without error, (ERR=1'b0), Nexus asserts the **nex_rdy_b** pin (see *Table 2081* for details) and clears the ERR bit in the RWCS Register. Otherwise, if the access completes with an error, the **nex_err_b** pin will be asserted and the ERR bit will be set to indicate an error has occurred, and the **nex_rdy_b** pin will not be asserted. Once DV has been cleared, this indicates that the device is ready for the next access, or that an error has occurred.

Note: Only the **nex_ahb_start, nex_rdy_b, and nex_err_b** *pins as well as the AC, DV, and ERR bits within the RWCS provide Read/Write Access status to the external development tool.*

83.17.2 Block write access

1. For a block write access, follow Steps 1, 2, and 3 outlined in *Section 83.17.1: Single write access* to initialize the registers, but using a value greater than one (14'h0001) for the CNT field in the RWCS Register.
2. The Nexus block will then arbitrate for the AHB system bus and transfer the first data value from the RWD Register to the memory mapped address in the Read/Write Access Address Register (RWA). The **nex_ahb_start** output will be asserted during the first clock of the address phase of the transfer. When the transfer has completed without error, the address from the RWA Register is incremented to the next word size (specified in the SZ field), the number from the CNT field is decremented, and the **nex_rdy_b** pin is asserted. If the access has completed without error, Nexus clears the ERR and DV bits in the RWCS Register, otherwise the ERR bit will be set and the DV bit will be cleared to indicate an error has occurred, the **nex_err_b** pin is asserted, the block transfer is aborted, and the AC bit in the RWCS register is cleared. Once DV has

been cleared, this indicates that the device is ready for the next access in the block transfer, or an error has occurred.

3. If AC has not been cleared due to an error, repeat Step 3 in *Section 83.17.1: Single write access* until the internal CNT value is zero (0). When this occurs, the AC bit within the RWCS will be cleared to indicate the end of the block write access.

Note: The actual RWA value as well as the CNT field within the RWCS are not changed when executing a block write access. The original values can be read by the external development tool at any time.

83.17.3 Single read access

1. Initialize the Read/Write Access Address Register (RWA) through the access method outlined in *Section 83.5: Nexus 3 Register Access via JTAG/OnCE*. Configure as follows:
   a) Read Address → 32h'xxxxxxxx (read address)
2. Initialize the Read/Write Access Control/Status Register (RWCS) through the access method outlined in *Section 83.5: Nexus 3 Register Access via JTAG/OnCE*. Configure the bits as follows:
   a) Access Control (AC) → 1b'1 (to indicate start access)
   b) Map Select (MAP) → 3b'000 (primary memory map)
   c) Access Priority (PR) → 2b'11 (highest priority)
   d) Read/Write (RW) → 1b'0 (read access)
   e) Word Size (SZ) → 3b'0xx (32-bit, 16-bit, 8-bit)
   f) Access Count (CNT) → 14h'0000 or 14h'0001(single access)

Note: Access Count (CNT) of 14'h0000 or 14'h0001 will perform a single access.

3. The Nexus block will then arbitrate for the AHB system bus and the read data will be transferred from the AHB to the RWD Register. The **nex_ahb_start** output will be asserted during the first clock of the address phase of the transfer. When the access has completed without error, Nexus clears the ERR bit and sets the DV bit in the RWCS Register and asserts the **nex_rdy_b** pin (see *Table 2081* for detail on **nex_rdy_b**). Otherwise, if the access has completed with an error, the **nex_err_b** pin will be asserted, the ERR bit will be set and the DV bit will be cleared to indicate an error has occurred, and the **nex_rdy_b** pin will not be asserted. Once ERR or DV has been set, this indicates that the device is ready for the next access, or that an error has occurred. The AC bit will be cleared in either case.
4. The data can then be read from the Read/Write Access Data Register (RWD) through the access method outlined in *Section 83.5: Nexus 3 Register Access via JTAG/OnCE*.

Note: Only the **nex_ahb_start, nex_rdy_b,** *and* **nex_err_b** *pins as well as the AC, DV and ERR bits within the RWCS provide Read/Write Access status to the external development tool.*

83.17.4 Block read access

1. For a block read access, follow Steps 1 and 2 outlined in *Section 83.17.3: Single read access* to initialize the registers, but using a value greater than one (14'h0001) for the CNT field in the RWCS Register.
2. The Nexus block will then arbitrate for the AHB system bus and the read data will be transferred from the AHB to the RWD Register. When the access has completed without error, Nexus clears the ERR bit and sets the DV bit in the RWCS Register and

asserts the **nex_rdy_b** pin (see *Table 2081* for detail on **nex_rdy_b**). Otherwise, if the access has completed with an error, the **nex_err_b** pin will be asserted, the ERR bit will be set and the DV bit will be cleared to indicate an error has occurred, and the **nex_rdy_b** pin will not be asserted. Once ERR or DV has been set, this indicates that the device is ready for the next access, or that an error has occurred. The AC bit will be cleared in either case.

3. When the transfer has completed without error, the address from the RWA Register is incremented to the next word size (specified in the SZ field), the number from the CNT field is decremented, Nexus clears the ERR bit and sets the DV bit in the RWCS Register, and asserts the **nex_rdy_b** pin (see *Table 2081* for detail on **nex_rdy_b**). Otherwise, if the access has completed with an error, the **nex_err_b** pin will be asserted, the ERR bit will be set and the DV bit will be cleared to indicate an error has occurred, the AC bit is cleared and the block transfer is aborted, and the **nex_rdy_b** pin will not be asserted. Once ERR or DV has been set, this indicates that the device is ready for the next access, or that an error has occurred. Once DV has been set, this indicates that the device is ready for the next access in the block transfer, or if ERR is set (AC will be cleared), the block transfer has been aborted.
4. The data can then be read from the Read/Write Access Data Register (RWD) through the access method outlined in *Section 83.5: Nexus 3 Register Access via JTAG/OnCE*.
5. If AC has not been cleared due to an error, repeat Steps 3 and 4 in *Section 83.17.3: Single read access* until the CNT value is zero (0). When this occurs, the AC bit within the RWCS is cleared to indicate the end of the block read access.

Note: The data values must be shifted out 32 bits at a time LSB first (that is doubleword read = two word reads from the RWD).

Note: The actual RWA value as well as the CNT field within the RWCS are not changed when executing a block read access. The original values can be read by the external development tool at any time.

83.17.5 Error handling

The Nexus 3 module handles various error conditions as follows:

83.17.5.1 AHB Read/Write error

All address and data errors that occur on read/write accesses to the AHB system bus will return a transfer error encoding on the **p_hresp[1:0]** signals. If this occurs:

1. The access is terminated without retrying (AC bit is cleared).
2. The ERR bit in the RWCS Register is set.
3. The Error Message is sent (TCODE = 8) indicating Read/Write error.

83.17.5.2 Access termination

The following cases are defined for sequences of the Read/Write protocol that differ from those described in the above sections.

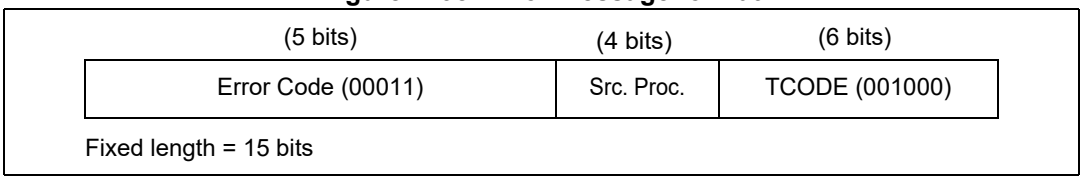
1. If the AC bit in the RWCS Register is set to start Read/Write accesses and invalid values are loaded into the RWD and/or RWA, then an AHB access error may occur. This is handled as described above.
2. If a block access is in progress (all cycles not completed), and the RWCS Register is written, then the original block access is terminated at the boundary of the nearest completed access.
   a) If the RWCS is written with the AC bit set, the next Read/Write access will begin and the RWD can be written to/ read from.
   b) If the RWCS is written with the AC bit cleared, the Read/Write access is terminated at the nearest completed access. This method can be used to break (early terminate) block accesses.

83.17.6 Read/Write access error message

The Read/Write Access Error Message is sent out when an AHB system bus access error (read or write) has occurred.

Error information is messaged out in the following format:

Figure 2198. Error message format



83.18 Nexus 3 pin interface

This section details information regarding the Nexus 3 pins and pin protocol.

The Nexus 3 pin interface provides the function of transmitting messages from the messages queues to the external tools. It is also responsible for handshaking with the message queues.

83.18.1 Pins implemented

The Nexus 3 module implements an auxiliary port consisting of one (1) **nex_evti_b** and one (1) **nex_mseo_b** or two (2) **nex_mseo_b[1:0]**. It also implements a configurable number of **nex_mdo[n:0]** pins, (1) **nex_rdy_b** pin, (1) **nex_err_b** pin, (1) **nex_evto_b** pin, (4) **nex_wevto[3:0]** pins, and one (1) clock output pin (**nex_mcko**), as well as additional configuration pins described in *Table 2081*. The output pins are synchronized to the Nexus 3 output clock (**nex_mcko**).

All Nexus 3 input functionality may be controlled through the JTAG/OnCE port in compliance with IEEE 1149.1 (see *Section 83.5: Nexus 3 Register Access via JTAG/OnCE* for details). The JTAG pins are incorporated as I/O to the processor, and are further described in the “JTAG/OnCE Pins” section of the Core (e200z420n3) Core Debug Support chapter.

Table 2080. JTAG pins for Nexus 3

JTAG pins	Input/ Output	Description of JTAG pins (included in Nexus 1)
j_tdo	O	The Test Data Output (**j_tdo**) pin is the serial output for test instructions and data. **j_tdo** is three-stateable and is actively driven in the “Shift-IR” and “Shift-DR” controller states. **j_tdo** changes on the falling edge of **j_tclk**.
j_tdi	I	The Test Data Input (**j_tdi**) pin receives serial test instruction and data. TDI is sampled on the rising edge of **j_tclk**.
j_tms	I	The Test Mode Select (**j_tms**) input pin is used to sequence the OnCE controller state machine. **j_tms** is sampled on the rising edge of **j_tclk**.
j_tclk	I	The Test Clock (**j_tclk**) input pin is used to synchronize the test logic, and control register access through the JTAG/OnCE port.
j_trst_b	I	The Test Reset (**j_trst_b**) input pin is used to asynchronously initialize the JTAG/OnCE controller.

The auxiliary pins are used to send and receive messages and are described in *Table 2081*.

Table 2081. Nexus 3 auxiliary pins

Auxiliary pins	Input/ Output	Description of auxiliary pins
nex_mcko	O	Message Clock Out (**nex_mcko**) is a free running output clock to development tools for timing of **nex_mdo[n:0]** & **nex_mseo_b[1:0]** pin functions. **nex_mcko** is programmable through the DC1 Register.
nex_mdo[n:0]	O	Message Data Out (**nex_mdo[n:0])** are output pins used for OTM, BTM, and DTM. External latching of **nex_mdo[n:0]** shall occur on the rising edge of the Nexus3 clock (**nex_mcko**).
nex_mseo_b[1:0]	O	Message Start/End Out (**nex_mseo_b[1:0]**) are output pins that indicate when a message on the **nex_mdo[n:0]** pins has started, when a variable length packet has ended, and when the message has ended. External latching of **nex_mseo_b[1:0]** shall occur on the rising edge of the Nexus3 clock (**nex_mcko**). One or two pin MSEO functionality is determined at integration time per SOC implementation
nex_rdy_b	O	Ready (**nex_rdy_b**) is an output pin used to indicate to the external tool that the Nexus block is ready for the next Read/Write Access. If Nexus is enabled, this signal is asserted upon successful (without error) completion of an AHB system bus transfer (Nexus read or write) & is held asserted until the JTAG/OnCE state machine reaches the “Capture_DR” state. Upon exit from system reset or if Nexus is disabled, **nex_rdy_b** remains deasserted
nex_err_b	O	Error (**nex_err_b**) is an output pin used to indicate to the external tool that the Nexus block is ready for the next Read/Write Access and an error has occurred on the previous access. If Nexus is enabled, this signal is asserted upon unsuccessful (with error) completion of an AHB system bus transfer (Nexus read or write) & is held asserted until the JTAG/OnCE state machine reaches the “Capture_DR” state. Upon exit from system reset or if Nexus is disabled, **nex_err_b** remains deasserted

Table 2081. Nexus 3 auxiliary pins (continued)

Auxiliary pins	Input/ Output	Description of auxiliary pins
nex_evto_b	O	Event Out (**nex_evto_b**) is an output that, when asserted, indicates one of two events has occurred based on the EOC bits in the DC1 Register. **nex_evto_b** is held asserted for one (1) cycle of **nex_mcko**: 1) One (or more) watchpoints has occurred (from Nexus1) & EOC = 2'b00 2) Debug mode was entered (jd_debug_b asserted from Nexus1) & EOC = 2'b01
nex_evti_b	I	Event In (**nex_evti_b**) is an input that, when asserted, will initiate one of two events based on the EIC bits in the DC1 Register (if the Nexus module is enabled at reset): 1) Program Trace & Data Trace synchronization messages (provided Program Trace & Data Trace are enabled & EIC = 2'b00). 2) Debug request to Nexus1 module (provided EIC = 2'b01 and this feature is implemented).
nex_wevto[3:0]	O	Watchpoint Event Out 3:0 (**nex_wevto[3:0]**) are outputs that, when asserted, indicates one or more watchpoint events has occurred based on the settings in the DC2 and DC3 registers. **nex_wevto[3:0]** is held asserted for one (1) cycle of **nex_mcko**.
nex_ext_src_id[0:3]	I	nex_ext_src_id[0:3] is used to provide the SRC field value used in each message. These pins are tied to a predetermined value at SoC integration time
nex_sfwcntl_en]	I	nex_sfwcntl_en is used to allow software to control the module resources via the DCR register mappings. SoC logic should drive this signal appropriately in a semi-static manner based on the presence of an external hardware debugger and appropriate security precautions.

The Nexus auxiliary port arbitration pins are used when the Nexus 3 module is implemented in a multi-Nexus SoC that shares a single auxiliary output port. The arbitration is controlled by an SoC level Nexus Aurora Router (NAR). Refer to *Section 83.20: Auxiliary port arbitration*, for detail on Nexus port arbitration.

Table 2082. Nexus port arbitration signals

Nexus port arbitration pins	Input/ Output	Description of arbitration pins
nex_aux_req[1:0]	O	Nexus Auxiliary Request (**nex_aux_req[1:0]**) output signals indicate to an SoC level Nexus arbiter a request for access to the shared Nexus auxiliary port in a multi-Nexus implementation. The priority encodings are determined by how many messages are currently in the message queues (See *Table 2084*.)
nex_aux_busy	O	Nexus Auxiliary Busy (**nex_aux_busy**) is an output signal to an SoC level Nexus arbiter indicating that the Nexus 3 module is currently transmitting its message after being granted the Nexus auxiliary port.
nar_aux_grant	I	Nexus Auxiliary Grant (**nar_aux_grant**) is an input from the SoC level NAR that the auxiliary port has been granted to the Nexus 3 module to transmit its message.
ext_multi_nex_sel	I	Multi-Nexus Select (**ext_multi_nex_sel**) is a static signal indicating that the Nexus 3 module is implemented within a multi-Nexus environment. If set, port control and arbitration is controlled by the SoC level arbitration module (NAR).

83.18.2 Pin protocol

The protocol for the processor transmitting messages via the auxiliary pins is accomplished with the MSEO pin function outlined in *Table 2083*. Both single and dual pin cases are shown.

nex_mseo_b[1:0] is used to signal the end of variable-length packets, and not fixed length packets. **nex_mseo_b[1:0]** is sampled on the rising edge of the Nexus 3 clock (**nex_mcko**).

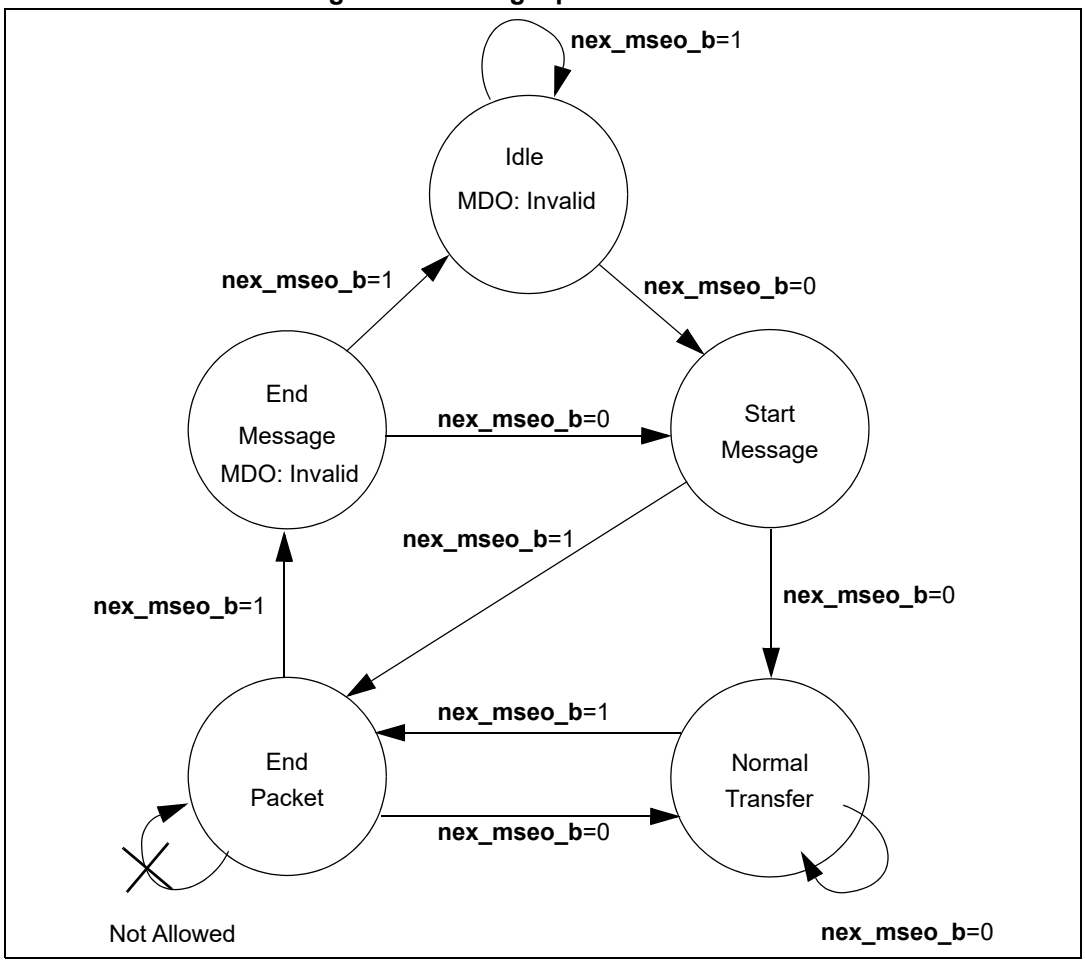
Single pin MSEO is not supported on the SPC58xEx/SPC58xGx.

Table 2083. MSEO pin(s) protocol

nex_mseo_b function	Single nex_mseo_b data (serial)	Dual nex_mseo_b[1:0] data
Start of message	1-1-0	11-00
End of message	0-1-1-(more 1's)	00 (or 01)-11-(more 1's)
End of variable length packet	0-1-0	00-01
Message transmission	0's	00's
Idle (no message)	1's	11's

Figure 2199 illustrates the state diagram for single pin MSEO transfers.

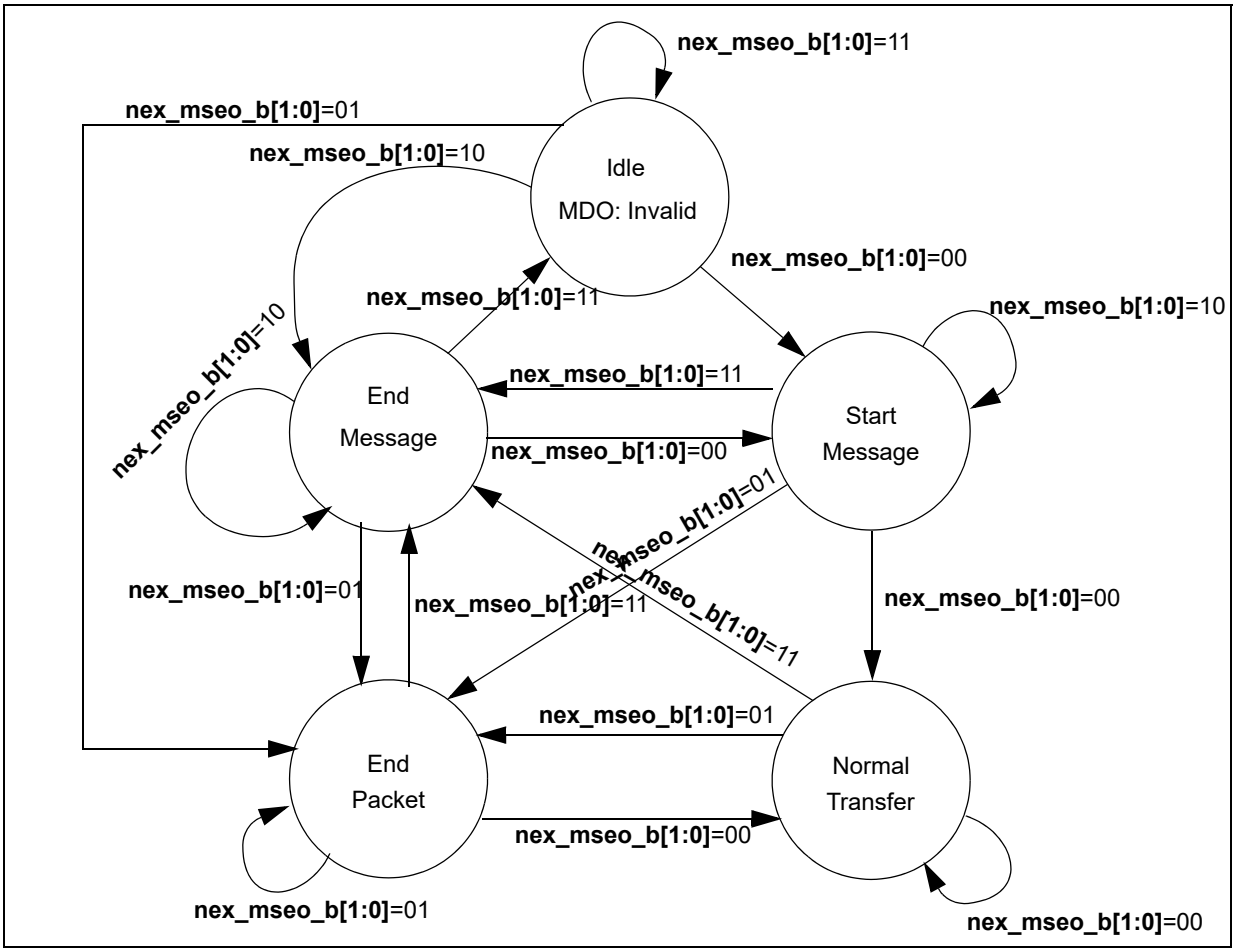
Figure 2199. Single pin MSE transfers



Note that the “End Message” state does not contain valid data on the **nex_mdo[n:0]** pins. Also, It is not possible to have two consecutive “End Packet” messages. This implies the minimum packet size for a variable length packet is 2x the number of **nex_mdo[n:0]** pins. This ensures that a false end of message state is not entered by emitting two consecutive ‘1’s on the **nex_mseo_b** pin before the actual end of message.

Figure 2200 illustrates the state diagram for dual pin MSEO transfers.

Figure 2200. Dual pin MSEO transfers



The dual pin MSEO option is more robust that the single pin option. Termination of the current message may immediately be followed by the start of the next message on the consecutive clocks. An extra clock to end the message is not necessary as with the one MSEO pin option. The dual pin option also allows for consecutive “End Packet” states. This can be an advantage when small, variable sized packets are transferred.

Note: The “End Message” state may also indicate the end of a variable-length packet as well as the end of the message when using the dual pin option.

83.19 Rules for output messages

Class 3 compliant embedded processors must provide messages via the auxiliary port in a consistent manner as described below:

- A variable-sized packet within a message must end on a port boundary.
- A variable-sized packet may start within a port boundary only when following a fixed length packet. (If two variable-sized packets end and start on the same clock, it is

impossible to know which bit is from the last packet and which bit is from the next packet.)

- Whenever a variable-length packet is sized such that it does not end on a port boundary, it is necessary to extend and zero fill the remaining bits after the highest-order bit so that it can end on a port boundary.

  For example, if the **nex_mdo[n:0]** port is 2 bits wide, and the unique portion of an indirect address TCODE is 5 bits, then the remaining 1 bit of **nex_mdo[n:0]** must be packed with a 0.

83.20 Auxiliary port arbitration

In a multi-Nexus environment, the Nexus 3 module must arbitrate for the shared Nexus port at the SoC level.The request scheme is implemented as a 2-bit request with various levels of priority. The priority levels are defined in *Table 2084* below. The Nexus 3 module will receive a 1-bit grant signal (**nar_aux_grant**) from the SoC level arbiter. When a grant is received, the Nexus 3 module will begin transmitting its message following the protocol outlined in *Section 83.18.2: Pin protocol*. The Nexus 3 module will maintain control of the port, by asserting the **nex_aux_busy** signal, until the MSEO state machine reaches the “End Message” state.

Table 2084. MDO request encodings

Request level	MDO request encoding (nex_aux_req[1:0])	Condition of queue
No Request	00	No message to send
Low Priority	01	Message queue less than 1/2 full
—	10	Reserved
High Priority	11	Message queue 1/2 full or more

83.21 Examples

The following are examples of Program Trace and Data Trace Messages.

Table 2085 illustrates an example Indirect Branch Message with 2 MDO / 1MSEO configuration. *Table 2086* illustrates the same example with an 8 MDO / 2 MSEO configuration.

Note that T0 and S0 are the least significant bits where:

- Tx = TCODE number (fixed)
- Sx = Source processor (fixed)
- MAP = (always set to 0)
- Ix = Number of instructions (variable)
- Ax = Unique portion of the address (variable)

Note that during clock 13, the **nex_mdo[n:0]** pins are ignored in the single MSEO case.

Table 2085. Indirect branch message example (2 MDO / 1 MSEO)

Clock	nex_mdo[1:0]		nex_mseo_b	State
0	X	X	1	Idle (or end of last message)
1	T1	T0	0	Start Message
2	T3	T2	0	Normal Transfer
3	T5	T4	0	Normal Transfer
4	S1	S0	0	Normal Transfer
5	S3	S2	0	Normal Transfer
6	I0	MAP	0	Normal Transfer
7	I2	I1	0	Normal Transfer
8	I4	I3	1	End Packet
9	A1	A0	0	Normal Transfer
10	A3	A2	0	Normal Transfer
11	A5	A4	0	Normal Transfer
12	A7	A6	1	End Packet
13	0	0	1	End Message
14	T1	T0	0	Start Message

Table 2086. Indirect branch message example (8 MDO / 2 MSEO)

Clock	nex_mdo[7:0]								nex_mseo_b [1:0]		State
0	X	X	X	X	X	X	X	X	1	1	Idle (or end of last message)
1	S1	S0	T5	T4	T3	T2	T1	T0	0	0	Start Message
2	I4	I3	I2	I1	I0	MAP	S3	S2	0	1	End Packet
3	A7	A6	A5	A4	A3	A2	A1	A0	1	1	End Packet/ End Message
4	S1	S0	T5	T4	T3	T2	T1	T0	0	0	Start Message

Table 2087 and *Table 2088* illustrate examples of Direct Branch Messages: one with 2 MDO / 1 MSEO, and one with 8 MDO / 2 MSEO.

Note that T0 and I0 are the least significant bits where:

- Tx = TCODE number (fixed)
- Sx = Source processor (fixed)
- Ix = Number of Instructions (variable)

Table 2087. Direct branch message example (2 MDO / 1 MSEO)

Clock	nex_mdo[1:0]		nex_mseo_b	State
0	X	X	1	Idle (or end of last message)
1	T1	T0	0	Start Message
2	T3	T2	0	Normal Transfer
3	T5	T4	0	Normal Transfer
4	S1	S0	0	Normal Transfer
5	S3	S2	0	Normal Transfer
6	I1	I0	1	End Packet
7	0	0	1	End Message

Table 2088. Direct branch message example (8 MDO / 2 MSEO)

Clock	nex_mdo[7:0]								nex_mseo_b[1:0]		State
0	X	X	X	X	X	X	X	X	1	1	Idle (or end of last message)
1	S1	S0	T5	T4	T3	T2	T1	T0	0	0	Start Message
2	0	0	0	0	I1	I0	S3	S2	1	1	End Packet/End Message
3	S1	S0	T5	T4	T3	T2	T1	T0	0	0	Start Message

Table 2089 illustrates an example Data Write Message with 8 MDO / 1 MSEO configuration, and *Table 2090* illustrates the same DWM with 8 MDO / 2 MSEO configuration

Note that T0, A0, D0 are the least significant bits where:

- Tx = TCODE number (fixed)
- Sx = Source processor (fixed)
- MAP = (always set to 0)
- Zx = Data size (fixed)
- Ax = Unique portion of the address (variable)
- Dx = Write data (variable 8-, 16- or 32-bit)

Table 2089. Data write message example (8 MDO / 1 MSEO)

Clock	nex_mdo[7:0]								nex_mseo_b	State
0	X	X	X	X	X	X	X	X	1	Idle (or end of last message)

Table 2089. Data write message example (8 MDO / 1 MSEO) (continued)

Clock	nex_mdo[7:0]								nex_mseo_b	State
1	S1	S0	T5	T4	T3	T2	T1	T0	0	Start Message
2	A0	Z3	Z2	Z1	Z0	0	S3	S2	1	End Packet
3	D7	D6	D5	D4	D3	D2	D1	D0	0	Normal Transfer
4	0	0	0	0	0	0	0	0	1	End Packet
5	0	0	0	0	0	0	0	0	1	End Message

Table 2090. Data write message example (8 MDO / 2 MSEO)

Clock	nex_mdo[7:0]								nex_mseo_b[1:0]		State
0	X	X	X	X	X	X	X	X	1	1	Idle (or end of last message)
1	S1	S0	T5	T4	T3	T2	T1	T0	0	0	Start Message
2	A0	Z3	Z2	Z1	Z0	0	S3	S2	0	1	End Packet
3	D7	D6	D5	D4	D3	D2	D1	D0	1	1	End Packet/ End Message

83.22 Electrical characteristics

For all electrical characteristics related to core processor and Nexus 3 operation, please refer to the SPC58xEx/SPC58xGx Data Sheet.

83.23 IEEE 1149.1 (JTAG) RD/WR sequences

This section contains example JTAG/OnCE sequences used to access resources.

83.23.1 JTAG sequence for accessing internal Nexus registers

Table 2091. Accessing internal Nexus 3 registers via JTAG/OnCE

Step #	TMS Pin	Description
1	1	IDLE → SELECT-DR_SCAN
2	0	SELECT-DR_SCAN → CAPTURE-DR (Nexus Command Register value loaded in shifter)
3	0	CAPTURE-DR → SHIFT-DR
4	0	(7) TCK clocks issued to shift in direction (rd/wr) bit and first 6 bits of Nexus reg. addr.
5	1	SHIFT-DR → EXIT1-DR (7th bit of Nexus reg. shifted in)
6	1	EXIT1-DR → UPDATE-DR (Nexus shifter is transferred to Nexus Command Register)
7	1	UPDATE-DR → SELECT-DR_SCAN
8	0	SELECT-DR_SCAN → CAPTURE-DR (Register value is transferred to Nexus shifter)
9	0	CAPTURE-DR → SHIFT-DR

Table 2091. Accessing internal Nexus 3 registers via JTAG/OnCE (continued)

Step #	TMS Pin	Description
10	0	(31) TCK clocks issued to transfer register value to TDO pin while shifting in TDI value
11	1	SHIFT-DR → EXIT1-DR (MSB of value is shifted in/out of shifter)
12	1	EXIT1-DR → UPDATE -DR (if access is write, shifter is transferred to register)
13	0	UPDATE-DR → RUN-TEST/IDLE (transfer complete — Nexus controller to Reg. Select state)

83.23.2 JTAG sequence for read access of memory-mapped resources

Table 2092. Accessing memory-mapped resources (reads)

Step #	TCLK clocks	Description
1	13	Nexus Command = write to Read/Write Access Address Register (RWA)
2	37	Write RWA (initialize starting read address — data input on TDI)
3	13	Nexus Command = write to Read/Write Control/Status Register (RWCS)
4	37	Write RWCS (initialize read access mode and CNT value — data input on TDI)
5	—	Wait for falling edge of **nex_rdy_b** or **nex_err_b** pin
6	13	Nexus Command = read Read/Write Access Data Register (RWD)
7	37	Read RWD (data output on TDO)
8	—	If CNT > 0, go back to Step #5

83.23.3 JTAG sequence for write access of memory-mapped resources

Table 2093. Accessing memory-mapped resources (writes)

Step #	TCLK clocks	Description
1	13	Nexus Command = write to Read/Write Access Control/Status Register (RWCS)
2	37	Write RWCS (initialize write access mode and CNT value — data input on TDI)
3	13	Nexus Command = write to Read/Write Address Register (RWA)
4	37	Write RWA (initialize starting write address — data input on TDI)
5	13	Nexus Command = read Read/Write Access Data Register (RWD)
6	37	Write RWD (data output on TDO)
7	—	Wait for falling edge of **nex_rdy_b** or **nex_err_b** pin
8	—	If CNT > 0, go back to Step #5

84 e200z4256 Nexus 3 module

The e200z4256 Nexus 3 module provides real-time development capabilities for corresponding core processors in compliance with the IEEE-ISTO 5001 standard. This module provides development support capabilities without requiring the use of address and data pins for internal visibility.

A portion of the pin interface (the JTAG port) is also shared with the OnCE / Nexus 1 unit. The IEEE-ISTO 5001 standard defines an extensible auxiliary port that is used in conjunction with the JTAG port in these processors.

84.1 Introduction

84.1.1 General description

This chapter defines the auxiliary pin functions, transfer protocols and standard development features of a Class 3 device in compliance with the IEEE-ISTO 5001 standard. The development features supported are Program Trace, Data Trace, Watchpoint Messaging, Ownership Trace, Data Acquisition Messaging, and Read/Write Access via the JTAG interface. The Nexus 3 module also supports two Class 4 features: Watchpoint Triggering, and Processor Overrun Control.

84.1.2 Terms and definitions

Table 2094 contains a set of terms and definitions associated with the Nexus 3 module.

Table 2094. Terms and definitions

Term	Description
IEEE-ISTO 5001	Consortium & standard for real-time embedded system design.
Auxiliary Port	Refers to Nexus auxiliary port. Used as auxiliary port to the IEEE 1149.1 JTAG interface.
Branch Trace Messaging (BTM)	Visibility of addresses for taken branches and exceptions, and the number of sequential instructions executed between each taken branch.
Data Read Message (DRM)	External visibility of data reads to memory-mapped resources.
Data Write Message (DWM)	External visibility of data writes to memory-mapped resources.
Data Trace Messaging (DTM)	External visibility of how data flows through the embedded system. This may include DRM or DWM, or both.
Data Acquisition Messaging (DQM)	Data Acquisition Messaging (DQM) allows code to be instrumented to export customized information to the Nexus Auxiliary Output Port.
JTAG Compliant	Device complying to IEEE 1149.1 JTAG standard
JTAG IR & DR Sequence	JTAG Instruction Register (IR) scan to load an opcode value for selecting a development register. The JTAG IR corresponds to the OnCE command register (OCMD). The selected development register is then accessed via a JTAG Data Register (DR) scan.

Table 2094. Terms and definitions (continued)

Term	Description
Nexus1	The (OnCE) debug module. This module integrated with each processor provides all static (core halted) debug functionality. This module is compliant with Class1 of the IEEE-ISTO 5001 standard.
Ownership Trace Message (OTM)	Visibility of process/function that is currently executing.
Public Messages	Messages on the auxiliary pins for accomplishing common visibility and controllability requirements
SoC	“System-on-a-Chip”. SoC signifies all of the modules on a single die. This generally includes one or more processors with associated peripherals, interfaces & memory modules.
Standard	The phrase “according to the standard” is used to indicate according to the IEEE-ISTO 5001 standard.
Transfer Code (TCODE)	Message header that identifies the number or size of packets, or both, to be transferred, and how to interpret each of the packets.
Watchpoint	A Data or Instruction Breakpoint or other debug event that does not cause the processor to halt. Instead, a pin is used to signal that the condition occurred. A Watchpoint Message may also be generated.

84.1.3 Feature list

The Nexus 3 module is compliant with Class 3 of the IEEE-ISTO 5001 standard, with additional Class 4 features available. The following features are implemented:

- Program Trace via Branch Trace Messaging (BTM). Branch trace messaging displays program flow discontinuities (such as direct and indirect branches, exceptions), allowing the development tool to interpolate what transpires between the discontinuities. Thus static code may be traced.
- Data Trace via Data Write Messaging (DWM) and Data Read Messaging (DRM). This provides the capability for the development tool to trace reads or writes, or both, to selected internal memory resources.
- Ownership Trace via Ownership Trace Messaging (OTM). OTM facilitates ownership trace by providing visibility of which process ID or operating system task is activated. An Ownership Trace Message is transmitted when a new process/task is activated, allowing the development tool to trace ownership flow.

- Run-time access to embedded processor memory map via the JTAG port. This allows for enhanced download/upload capabilities.
- Watchpoint Messaging via the auxiliary pins.
- Watchpoint Trigger enable of Program or Data Trace Messaging, or both.
- External hardware trigger inputs to independently enable/disable Program Trace, Data Trace, Ownership Trace, and Watchpoint Trace Messaging.
- Auxiliary interface for higher data input/output.
  - Configurable (min/max) Message Data Out pins (**nex_mdo[n:0]**)
  - One (1) or two (2) Message Start/End Out pins (**nex_mseo_b[1:0]**)
  - One (1) Read/Write Ready pin (**nex_rdy_b**) pin
  - One (1) Read/Write Error pin (**nex_err_b**) pin
  - One (1) Watchpoint Event output pin (**nex_evto_b**)
  - Four (4) additional Watchpoint Event output pins (**nex_wevto[3:0]**) for SoC use
  - One (1) Event In pin (**nex_evti_b**)
  - One (1) MCKO (Message Clock Out) pin
- Registers for Program Trace, Data Trace, Ownership Trace and Watchpoint Trigger.
- All features controllable and configurable via the JTAG port
- Conditional software control of the module via SoC signaling input (**nex_sfwcntl_en**).

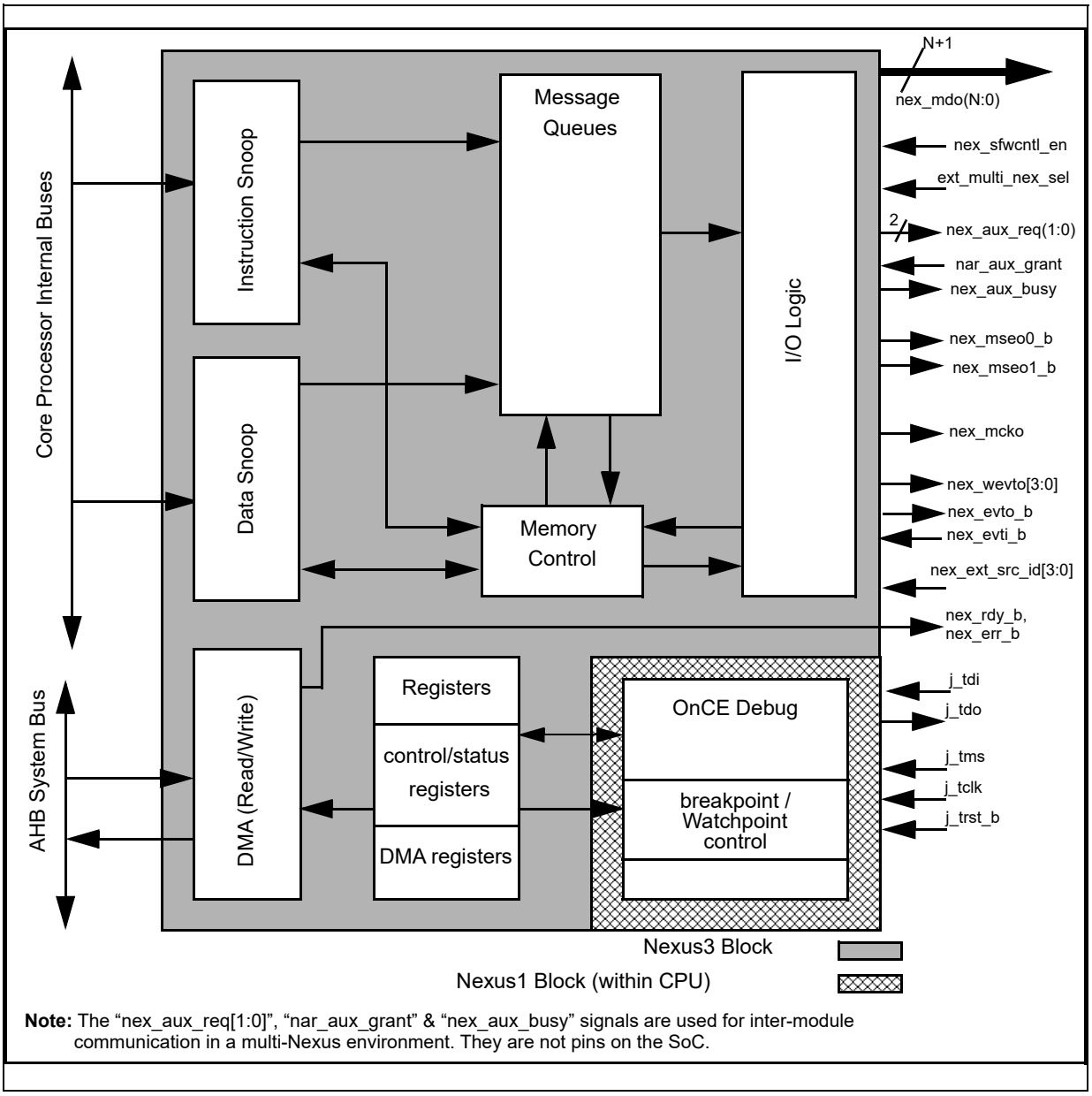
Note: For multi-Nexus implementations, the configuration of the Message Data Out pins is controlled by the Port Control Register (@ the SoC level). For single Nexus implementations (not normally implemented on an SoC), this configuration is controlled by Development Control Register 1 (DC1) within the Nexus 3 module.

In either implementation, Full Port Mode (FPM — maximum number of MDO pins) or Reduced Port Mode (RPM — minimum number of MDO pins) are supported. This setting should not be changed while the system is running.

Note: The configuration of the Message Start/End Out pins (1 or 2) is determined at the SOC integration level. This option will be hard-wired based on SOC bandwidth requirements.

84.1.4 Functional block diagram

Figure 2201. Nexus 3 functional block diagram



Note: The “nex_aux_req[1:0]”, “nar_aux_grant” & “nex_aux_busy” signals are used for inter-module communication in a multi-Nexus environment. They are not pins on the SoC.

84.2 Enabling Nexus 3 operation

The Nexus module is enabled by loading a single instruction (*NEXUS3-ACCESS*) into the JTAG Instruction Register (IR) (OnCE OCMD register). For the Nexus3 module, the OCMD value is 0b0001111100. The Nexus 3 module may alternately be enabled if the nex_evti_b input signal is asserted at the time that j_trst_b is initially negated, or is asserted during the Test-Logic-Reset TAP state. Once enabled, the module will be ready to accept control input via the JTAG/OnCE pins.

Enabling the Nexus 3 module automatically enables the generation of Debug Status Messages.

The Nexus 3 module is disabled when the JTAG state machine initially reaches the Test-Logic-Reset state from any other state. This state can be reached by the assertion of the **j_trst_b** pin or by cycling through the state machine using the **j_tms** pin. The Nexus module will also be disabled if a Power-on-Reset (POR) event occurs. If the Nexus 3 module is disabled, no trace output will be provided, and the module will disable (drive inactive) auxiliary port output pins (**nex_mdo[n:0]**, **nex_mseo[1:0]**, **nex_mcko**). Nexus registers will not be available for reads or writes.

In order to support software control of the Nexus 3 module when no external development tool is present, the Nexus 3 module is not forced to be disabled when the JTAG state remains in the Test-Logic-Reset state. Software is allowed to control the Nexus 3 module when the input signal **nex_sfwcntl_en** is asserted by the SoC. This signal is intended to provide a mechanism for allowing software to use the Nexus 3 module to fill on-chip trace buffers or other visibility mechanisms. It is up to the SoC to determine whether a top-level Nexus 3 controller has been enabled by a hardware debugger, or whether appropriate security mechanisms have granted the capability for software to use these resources, and to drive the appropriate value to the **nex_sfwcntl_en** input. Software can enable the module by a write to any of the module's DCRs when **nex_sfwcntl_en** is asserted.

Reset of the Nexus 3 module is accomplished by a transition on **j_trst_b**, on initial entry into the Test_Logic_Reset state from another state, or if a Power-on-Reset (POR) event occurs. The module is not reset by the CPU's **p_reset_b** signal, even when software has control of the module.

84.2.1 Interaction with Low Power Modes

The Nexus 3 module will continue to operate in the Waiting and Halted states, as long as **nex_clk** remains active. In the Stopped state, **nex_clk** is gated off internally to the Nexus 3 logic, therefore watchpoint or hardware triggering recognition, message generation, and Nexus 3 read-write access to memory is suspended. The Nexus 3 logic will wait to enter the Stopped state until the message FIFOs are empty and any in-progress Nexus R/W transfer has completed. If a block transfer has been requested, the remainder of the block transfer is not completed, and the RWCS AC bit is cleared, and the ERR bit is set. Once the Stopped state has been entered, no further messages are queued and no triggering conditions are monitored. Also, Nexus R/W accesses are no longer available. Upon exiting the Stopped state, normal functions will resume assuming **nex_clk** is active.

84.3 TCODEs supported

The Nexus 3 pins allow for flexible transfer operations via Public Messages. A TCODE defines the transfer format, the number or size of the packets, or both, to be transferred, and the purpose of each packet. The IEEE-ISTO 5001-2003 standard defines a set of public messages and allocates additional TCODEs for vendor-specific features outside the scope of the public messages. The Nexus 3 block supports the TCODEs shown in *Table 2095*.

Table 2095. Supported TCODEs

Message Name	Min Field Size (bits)	Max Field Size (bits)	Field Name	Field Type	Field Description
Debug Status	6	6	TCODE	fixed	TCODE number = 0
	4	4	SRC	fixed	source processor identifier
	8	8	STATUS	fixed	Development Status Register (DS[31:24])
	0	30	TSTAMP	variable	TImestamp value (optionally generated field)
Ownership Trace Message	6	6	TCODE	fixed	TCODE number = 2
	4	4	SRC	fixed	source processor identifier
	1	32	PROCESS	variable	Task/Process ID tag
	0	30	TSTAMP	variable	TImestamp value (optionally generated field)
Program Trace — Direct Branch Message	6	6	TCODE	fixed	TCODE number = 3
	4	4	SRC	fixed	source processor identifier
	1	8	ICNT	variable	# sequential instructions completed since last predicate instruction, transmitted instruction count, or taken change of flow
	0	30	TSTAMP	variable	TImestamp value (optionally generated field)
Program Trace — Indirect Branch Message	6	6	TCODE	fixed	TCODE number = 4
	4	4	SRC	fixed	source processor identifier
	1	1	MAP	fixed	(always set to 0)
	1	8	ICNT	variable	# sequential instructions completed since last predicate instruction, transmitted instruction count, or taken change of flow
	1	32	U-ADDR	variable	unique part of target address for taken branches/exceptions
	0	30	TSTAMP	variable	TImestamp value (optionally generated field)
Data Trace — Data Write Message	6	6	TCODE	fixed	TCODE number = 5
	4	4	SRC	fixed	source processor identifier
	1	1	MAP	fixed	(always set to 0)
	4	4	DSZ	fixed	data size (Refer to *Table 2100*)
	1	32	U-ADDR	variable	unique portion of the data write address
	1	64	DATA	variable	data write value(s) (see Data Trace section for details)
	0	30	TSTAMP	variable	TImestamp value (optionally generated field)

Table 2095. Supported TCODEs (continued)

Message Name	Min Field Size (bits)	Max Field Size (bits)	Field Name	Field Type	Field Description
Data Trace — Data Read Message	6	6	TCODE	fixed	TCODE number = 6
	4	4	SRC	fixed	source processor identifier
	1	1	MAP	fixed	(always set to 0)
	4	4	DSZ	fixed	data size (Refer to *Table 2100*)
	1	32	U-ADDR	variable	unique portion of the data read address
	1	64	DATA	variable	data read value(s) (see Data Trace section for details)
	0	30	TSTAMP	variable	TImestamp value (optionally generated field)
Data Acquisition Message	6	6	TCODE	fixed	TCODE number = 7
	4	4	SRC	fixed	source processor identifier
	8	8	DQTAG	fixed	identification tag taken from $DEVENT_{DQTAG}$ register field
	1	32	DQDATA	variable	exported data taken from DDAM register
	0	30	TSTAMP	variable	TImestamp value (optionally generated field)
Error Message	6	6	TCODE	fixed	TCODE number = 8
	4	4	SRC	fixed	source processor identifier
	4	4	ETYPE	fixed	error type
	8	8	ECODE	fixed	error code
	0	30	TSTAMP	variable	TImestamp value (optionally generated field)
Program Trace — Synchronization Message	6	6	TCODE	fixed	TCODE number = 9
	4	4	SRC	fixed	source processor identifier
	1	1	MAP	fixed	(always set to 0)
	1	8	I-CNT	variable	# sequential instructions completed since last predicate instruction, transmitted instruction count, or taken change of flow. Cleared for most sync conditions.
	1	32	F-ADDR	variable	full target address (leading zero (0) truncated)
	0	30	TSTAMP	variable	TImestamp value (optionally generated field)
Program Trace — Direct Branch Message with Sync	6	6	TCODE	fixed	TCODE number = 11
	4	4	SRC	fixed	source processor identifier
	1	8	ICNT	variable	# sequential instructions completed since last predicate instruction, transmitted instruction count, or taken change of flow
	1	32	F-ADDR	variable	full target address (leading zeros truncated)
	0	30	TSTAMP	variable	TImestamp value (optionally generated field)

Table 2095. Supported TCODEs (continued)

Message Name	Min Field Size (bits)	Max Field Size (bits)	Field Name	Field Type	Field Description
Program Trace — Indirect Branch Message with Sync	6	6	TCODE	fixed	TCODE number = 12
	4	4	SRC	fixed	source processor identifier
	1	1	MAP	fixed	(always set to 0)
	1	8	ICNT	variable	# sequential instructions completed since last predicate instruction, transmitted instruction count, or taken change of flow
	1	32	F-ADDR	variable	full target address (leading zeros truncated)
	0	30	TSTAMP	variable	TImestamp value (optionally generated field)
Data Trace — Data Write Message with Sync	6	6	TCODE	fixed	TCODE number = 13
	4	4	SRC	fixed	source processor identifier
	1	1	MAP	fixed	(always set to 0)
	4	4	DSZ	fixed	data size (Refer to *Table 2100*)
	1	32	F-ADDR	variable	full access address (leading zeros truncated)
	1	64	DATA	variable	data write value(s) (see Data Trace section for details)
	0	30	TSTAMP	variable	TImestamp value (optionally generated field)
Data Trace — Data Read Message with Sync	6	6	TCODE	fixed	TCODE number = 14
	4	4	SRC	fixed	source processor identifier
	1	1	MAP	fixed	(always set to 0)
	4	4	DSZ	fixed	data size (Refer to *Table 2100*)
	1	32	F-ADDR	variable	full access address (leading zeros truncated)
	1	64	DATA	variable	data read value(s) (see Data Trace section for details)
	0	30	TSTAMP	variable	TImestamp value (optionally generated field)
Watchpoint Message	6	6	TCODE	fixed	TCODE number = 15
	4	4	SRC	fixed	source processor identifier
	1	32	WPHIT	variable	Field indicating watchpoint source(s) (leading zeros truncated)
	0	30	TSTAMP	variable	TImestamp value (optionally generated field)
Resource Full Message	6	6	TCODE	fixed	TCODE number = 27
	4	4	SRC	fixed	source processor identifier
	4	4	RCODE	fixed	resource code (Refer to *Table 2098*) — indicates which resource is the cause of this message
	1	32	RDATA	variable	branch / predicate instruction history (Refer to *Section 84.12.4: Resource Full Messages*.)
	0	30	TSTAMP	variable	TImestamp value (optionally generated field)

Table 2095. Supported TCODEs (continued)

Message Name	Min Field Size (bits)	Max Field Size (bits)	Field Name	Field Type	Field Description
Program Trace — Indirect Branch History Message	6	6	TCODE	fixed	TCODE number = 28 (see Note below)
	4	4	SRC	fixed	source processor identifier
	1	1	MAP	fixed	(always set to 0)
	1	8	I-CNT	variable	# sequential instructions completed since last predicate instruction, transmitted instruction count, or taken change of flow
	1	32	U-ADDR	variable	unique part of target address for taken branches/exceptions
	1	32	HIST	variable	branch / predicate instruction history (Refer to *Section 84.12.1: Branch Trace Messaging types*)
	0	30	TSTAMP	variable	TImestamp value (optionally generated field)
Program Trace — Indirect Branch History Message with Sync	6	6	TCODE	fixed	TCODE number = 29 (see Note below)
	4	4	SRC	fixed	source processor identifier
	1	1	MAP	fixed	(always set to 0)
	1	8	I-CNT	variable	# sequential instructions completed since last predicate instruction, transmitted instruction count, or taken change of flow
	1	32	F-ADDR	variable	full target address (leading zero (0) truncated)
	1	32	HIST	variable	branch / predicate instruction history (Refer to *Section 84.12.1: Branch Trace Messaging types*.)
	0	30	TSTAMP	variable	TImestamp value (optionally generated field)
Program Trace — Program Correlation Message	6	6	TCODE	fixed	TCODE number = 33
	4	4	SRC	fixed	source processor identifier
	4	4	EVCODE	fixed	event correlated with program flow (Refer to *Table 2099*)
	2	2	CDF	fixed	# fields of information in CDATA. 00 — reserved, 01 — one field (CDATA1) (reserved), 10 — two fields (CDATA1 + CDATA2), 11 — three fields (reserved)
	1	8	I-CNT	variable	# sequential instructions completed since last predicate instruction, transmitted instruction count, or taken change of flow
	1	32	CDATA1	variable	correlation data field 1 — [branch / predicate instruction history] (Refer to *Section 84.12.5: Program Correlation Messages*.)
	0	32	CDATA2	variable	correlation data field 2— PID info (Refer to *Section 84.12.5: Program Correlation Messages*.)
	0	30	TSTAMP	variable	TImestamp value (optionally generated field)

Note: Program Trace can be implemented using either Branch History/Predicate Instruction Messages, or traditional Direct/Indirect Branch Messages. The user can select between the two types of Program Trace. The advantages for each are discussed in Section 84.12.1: Branch Trace Messaging types. If the Branch History method is selected, the shaded TCODES above will not be messaged out.

Table 2096 shows the error code encodings used when reporting an error via the Nexus 3 Error Message.

Table 2096. Error Code (ECODE) Encoding (TCODE = 8)

Error Code	Description
xxxxxxx1	Watchpoint Trace Message(s) Lost
xxxxxx1x	Data Trace Message(s) Lost
xxxxx1xx	Program Trace Message(s) Lost
xxxx1xxx	Ownership Trace Message(s) Lost
xxx1xxxx	Status Message(s) Lost (such as Debug Status messages)
xx1xxxxx	Data Acquisition Message(s) Lost
x1xxxxxx	Reserved
1xxxxxxx	Reserved

Table 2097 shows the error type encodings used when reporting an error via the Nexus 3 Error Message.

Table 2097. Error Type (ETYPE) Encoding (TCODE = 8)

Error Type	Description
0000	Message Queue Overrun caused one or more messages to be lost
0001	Contention with higher priority messages caused one or more messages to be lost
0010	Reserved
0011	Reserved
0100	Reserved
0101	Invalid access opcode (Nexus Register unimplemented)
0110 – 1111	Reserved

Table 2098 shows the encodings used for resource codes for certain messages.

Table 2098. Resource Code (RCODE) values (TCODE = 27)

Resource Code	Description
0000	Program Trace Instruction counter reached 255 and was reset.
0001	Program Trace, Branch / Predicate Instruction History full. This type of packet is terminated by a stop bit set to 1 after the last history bit.

Table 2099 shows the event code encodings used for certain messages.

Table 2099. Event Code (EVCODE) Encoding (TCODE = 33)

Event Code	Description
0000	Entry into Debug Mode
0001	Entry into Low Power Mode (CPU only)
0010–0011	Reserved for future functionality
0100	Disabling Program Trace
0101	Process ID value is established in PID0/NPIDR via **mtspr PID0/NPIDR**
0110–1000	Reserved for future functionality
1001	Begin masking of program trace messages due to MSR_{PMM} = 0 and $DC4_{PTMARK}$= 1
1010	Branch and link occurrence (direct branch function call)
1011–1111	Reserved for future functionality

Table 2100 shows the data trace size encodings used for certain messages.

Table 2100. Data Trace Size (DSZ) Encodings (TCODE = 5,6,13,14)

DTM Size Encoding	Transfer Size
0000	0 — no data
0001	Byte
0010	Halfword (2 bytes)
0011	Three bytes
0100	Word (4 bytes)
0101	Five bytes
0110	Six bytes
0111	Seven bytes
1000	Doubleword (8 bytes)
1001–1111	Reserved

84.4 Nexus 3 Programmer's Model

This section describes the Nexus 3 programmers model. Nexus 3 registers are accessed using the JTAG/OnCE port in compliance with IEEE 1149.1, or by software via DCRs. Refer to *Section 84.5: Nexus 3 Register Access via JTAG/OnCE* and *Section 84.6: Nexus 3 Register Access via Software* for details on Nexus 3 register access.

Note: Nexus 3 registers and output signals are numbered using bit 0 as the least significant bit. This bit ordering is consistent with the ordering defined by the IEEE-ISTO 5001 standard.

Table 2101 details the register map for the Nexus 3 module.

Table 2101. Nexus 3 Register Map

Nexus Register	Nexus Access Opcode	Read/ Write	Read Address	Write Address	DCR #(1)
Client Select Control (CSC)(2)	0x1	R	0x02	—	
Port Configuration Register (PCR)(3)	PCR_INDEX(2)	R/W	—	—	
Development Control 1 (DC1)	0x2	R/W	0x04	0x05	368
Development Control 2 (DC2)	0x3	R/W	0x06	0x07	369
Development Control 3 (DC3)	0x4	R/W	0x08	0x09	370
Development Control 4 (DC4)	0x5	R/W	0x0A	0x0B	371
Reserved	0x6	R/W	0x18	0x19	—
Read/Write Access Control/Status (RWCS)	0x7	R/W	0x0E	0x0F	—
Reserved	0x8	R/W	0x18	0x19	—
Read/Write Access Address (RWA)	0x9	R/W	0x12	0x13	—
Read/Write Access Data (RWD)	0xA	R/W	0x14	0x15	—
Watchpoint Trigger (WT)	0xB	R/W	0x16	0x17	375
Reserved	0xC	R/W	0x18	0x19	—
Data Trace Control (DTC)	0xD	R/W	0x1A	0x1B	376
Data Trace Start Address 1 (DTSA1)	0xE	R/W	0x1C	0x1D	377
Data Trace Start Address 2 (DTSA2)	0xF	R/W	0x1E	0x1F	378
Data Trace Start Address 3 (DTSA3)	0x10	R/W	0x20	0x21	379
Data Trace Start Address 4 (DTSA4)	0x11	R/W	0x22	0x23	380
Data Trace End Address 1 (DTEA1)	0x12	R/W	0x24	0x25	381
Data Trace End Address 2 (DTEA2)	0x13	R/W	0x26	0x27	382
Data Trace End Address 3 (DTEA3)	0x14	R/W	0x28	0x29	383
Data Trace End Address 4 (DTEA4)	0x15	R/W	0x2A	0x2B	408
Reserved	0x16 → 0x2F	—	0x28→0x5E	0x29→0x5F	—
Development Status (DS)	0x30	R	0x60	—	409
Reserved	0x31	R/W	0x62	0x63	
Overrun Control (OVCR)	0x32	R/W	0x64	0x65	410
Watchpoint Mask (WMSK)	0x33	R/W	0x66	0x67	411
Reserved	0x34	—	0x68	0x69	—
Program Trace Start Trigger Control (PTSTC)	0x35	R/W	0x6A	0x6B	412
Program Trace End Trigger Control (PTETC)	0x36	R/W	0x6C	0x6D	413
Data Trace Start Trigger Control (DTSTC)	0x37	R/W	0x6E	0x6F	414
Data Trace End Trigger Control (DTETC)	0x38	R/W	0x70	0x71	415
Reserved	0x39 → 0x3F	—	0x72→0x7E	0x73→0x7F	—

1. Software access via the **mfdcr** and **mtdcr** instructions use these values for the DCR number. Software writes to these registers via **mtdcr** when **nex_sfwcntl_en** is negated are ignored.

2. The CSC and PCR registers are shown in this table as part of the Nexus programmer's model. They are only present at the top level SoC Nexus controller in a multi-Nexus implementation, not in the Nexus 3 module. The SoC's CSC Register is readable through Nexus, but the PCR is shown for reference only here.
3. The "PCR_INDEX" is a parameter determined by the SoC.

84.4.1 Client Select Control (CSC) register

The CSC Register determines which Nexus client is under development. This register is present at the top-level SOC Nexus 3 controller to select one of multiple on-chip Nexus 3 units.

Reserved				CS			
7	6	5	4	3	2	1	0

Nexus Reg# 0x1; Read-only; Reset: 0x0

Figure 2202. Client Select Control (CSC) register

Table 2102. CSC field descriptions

Bits	Description
CSC[7:4]	Reserved for future Nexus Clients (read as 0)
CSC[3:0]	Client Select Control 0*xx* Nexus client (SoC level)

84.4.2 Port Configuration Register (PCR) — reference only

The Port Configuration Register (PCR) controls the basic port functions for all Nexus modules in a multi-Nexus environment. This includes clock control and auxiliary port width. All bits in this register are writable only once after system reset.

OPC	0	MCK_EN	MCK_DIV	0

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

Nexus Reg# PCR_INDEX; Read/Write; Reset: 0x0

Figure 2203. Port Configuration Register (PCR)

Table 2103. PCR field descriptions

Bit	Name	Description
31	OPC	Output Port Mode Control (SoC Level) 0 Reduced Port Mode configuration (min# **nex_mdo[n:0]** pins defined by SOC) 1 Full Port Mode configuration (max# **nex_mdo[n:0]** pins defined by SOC)
30	—	Reserved for future functionality
29	MCK_EN	MCKO Clock Enable (SoC Level) 0 **nex_mcko** is disabled 1 **nex_mcko** is enabled

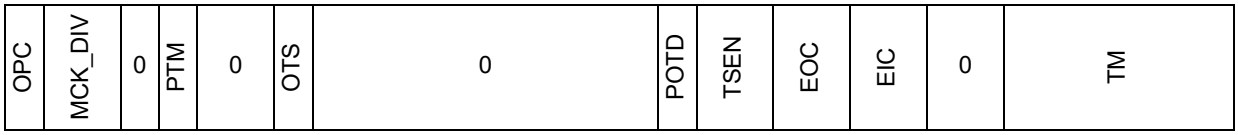
Table 2103. PCR field descriptions (continued)

Bit	Name	Description
28:26	MCK_DIV	MCKO Clock Divide Ratio (see note below) (SoC Level) 000 **nex_mcko** is 1x processor clock freq. 001 **nex_mcko** is 1/2x processor clock freq. 010 Reserved (default to 1/2x processor clock freq.) 011 **nex_mcko** is 1/4x processor clock freq. 100 Reserved (default to 1/2x processor clock freq.) 101 Reserved (default to 1/2x processor clock freq.) 110 Reserved (default to 1/2x processor clock freq.) 111 **nex_mcko** is 1/8x processor clock freq.
25:0	—	Reserved for future functionality

Note: The CSC and PCR Registers exist in a separate module at the SoC level in a multi-Nexus environment. If the core Nexus 3 module is the only Nexus module, these registers are not implemented and the Nexus 3 defined Development Control Register 1 (DC1) is used to control the SoC-level Nexus port functionality.

84.4.3 Nexus Development Control Register 1 (DC1)

Nexus Development Control Register 1 is used to control the basic development features of the Nexus 3 module. Development Control Register 1 is shown in *Figure 2204* and its fields are described in *Table 2104*.



Nexus Reg# 0x2; DCR: 368; Read/Write; Reset: 0x0

Figure 2204. Development Control 1 (DC1) register

Table 2104. DC1 field descriptions

Bits	Name	Description
31	OPC	Output Port Mode Control 0 — Reduced Port Mode configuration (min# **nex_mdo[n:0]** pins defined 1 — Full Port Mode configuration (max# **nex_mdo[n:0]** pins defined
30:29	MCK_DIV	MCKO Clock Divide Ratio (see note below) 00 — **nex_mcko** is 1x processor clock freq. 01 — **nex_mcko** is 1/2x processor clock freq. 10 — **nex_mcko** is 1/4x processor clock freq. 11 — **nex_mcko** is 1/8x processor clock freq.
28	—	Reserved for future functionality
27	PTM	Program Trace Method 0 — Program Trace uses traditional Branch Messages 1 — Program Trace uses Branch History Messages
26:25	—	Reserved for future functionality

Table 2104. DC1 field descriptions (continued)

Bits	Name	Description
24	OTS	Ownership Trace PID Select 0 — PID0 data is transmitted within Ownership Trace Messages 1 — Nexus PID Register (NPIDR) data is transmitted within Ownership Trace Messages
26:15	—	Reserved for future functionality
14	POTD	Periodic Ownership Trace Disable 0 — Periodic Ownership Trace message events are enabled 1 — Periodic Ownership Trace message events are disabled
13:12	TSEN	Timestamp Enable — (not implemented, write to 00) 00 — Timestamp is disabled
11:10	EOC	EVTO Control 00 — **nex_evto_b** upon occurrence of Watchpoints (configured in DC2 and DC3) 01 — **nex_evto_b** upon entry into Debug Mode 1*x* — Reserved
9:8	EIC	EVTI Control 00 — **nex_evti_b** is used for synchronization (Program Trace Sync msg is generated) 01 — **nex_evti_b** is used for Debug request 10 — **nex_evti_b** is disabled 1*x* — Reserved
7:6	—	Reserved for future functionality
5:0	TM	Trace Mode(1) 000000 — All Trace Disabled *xxxxx*1 — Ownership Trace enabled *xxxx*1*x* — Data Trace enabled *xxx*1*xx* — Program Trace enabled *xx*1*xxx* — Watchpoint Trace enabled *x*1*xxxx* — Reserved 1*xxxxx* — Data Acquisition Trace enabled

1. This field may be updated by hardware in response to watchpoint triggering or external hardware trigger inputs if not already enable for tracing by a previous direct write to DC1. Direct writes to this field to enable one or more trace types take precedence over hardware updates. Refer to *Section 84.4.7: Watchpoint Trigger (WT, PTSTC, PTETC, DTSTC, DTETC) registers* for more information on watchpoint triggering. Refer to *Section 84.16: External Hardware Trigger Controls* for more information on external hardware triggering.

Note: The Output Port Mode Control bit (OPC) and MCKO Clock Divide Ratio bits (MCK_DIV) MUST ONLY be modified during system reset or debug mode to insure correct output port and output clock functionality. It is also recommended that all other bits of the DC1 also only be modified in one of these two modes.

84.4.4 Nexus Development Control Registers 2 & 3 (DC2, DC3)

Nexus Development Control Registers 2 and 3 are used to control output signaling on the Nexus 3 module. A table of watchpoints can be found in the Core (e200z425n3) Core Debug Support chapter.

Development Control Register 2 is shown in *Figure 2205* and its fields are described in *Table 2105*.

WEVTO[3]C	WEVTO[2]C	WEVTO[1]C	WEVTO[0]C	EWC

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

Nexus Reg# 0x3; DCR: 369; Read/Write; Reset: 0x0

Figure 2205. Development Control 2 (DC2) register

Table 2105. DC2 field descriptions

Bits	Name	Description
31:28	WEVTO[3]C	Watchpoint Event Out 3 Configuration 0000 — No Watchpoints #0–14 trigger **nex_wevto[3]** 0001 — Watchpoint #0 triggers **nex_wevto[3]** 0010 — Watchpoint #1 triggers **nex_wevto[3]** 0011 — Watchpoint #2 triggers **nex_wevto[3]** 0100 — Watchpoint #3 triggers **nex_wevto[3]** 0101 — Watchpoint #4 triggers **nex_wevto[3]** 0110 — Watchpoint #5 triggers **nex_wevto[3]** 0111 — Watchpoint #6 triggers **nex_wevto[3]** 1000 — Watchpoint #7 triggers **nex_wevto[3]** 1001 — Watchpoint #8 triggers **nex_wevto[3]** 1010 — Watchpoint #9 triggers **nex_wevto[3]** 1011 — Watchpoint #10 triggers **nex_wevto[3]** 1100 — Watchpoint #11 triggers **nex_wevto[3]** 1101 — Watchpoint #12 triggers **nex_wevto[3]** 1110 — Watchpoint #13 triggers **nex_wevto[3]** 1111 — Watchpoint #14 triggers **nex_wevto[3]**
27:24	WEVTO[2]C	Watchpoint Event Out 2 Configuration 0000 — No Watchpoints #0–14 trigger **nex_wevto[2]** 0001 — Watchpoint #0 triggers **nex_wevto[2]** 0010 — Watchpoint #1 triggers **nex_wevto[2]** 0011 — Watchpoint #2 triggers **nex_wevto[2]** 0100 — Watchpoint #3 triggers **nex_wevto[2]** 0101 — Watchpoint #4 triggers **nex_wevto[2]** 0110 — Watchpoint #5 triggers **nex_wevto[2]** 0111 — Watchpoint #6 triggers **nex_wevto[2]** 1000 — Watchpoint #7 triggers **nex_wevto[2]** 1001 — Watchpoint #8 triggers **nex_wevto[2]** 1010 — Watchpoint #9 triggers **nex_wevto[2]** 1011 — Watchpoint #10 triggers **nex_wevto[2]** 1100 — Watchpoint #11 triggers **nex_wevto[2]** 1101 — Watchpoint #12 triggers **nex_wevto[2]** 1110 — Watchpoint #13 triggers **nex_wevto[2]** 1111 — Watchpoint #14 triggers **nex_wevto[2]**

Table 2105. DC2 field descriptions (continued)

Bits	Name	Description
23:20	WEVTO[1]C	Watchpoint Event Out 1 Configuration 0000 — No Watchpoints #0–14 trigger **nex_wevto[1]** 0001 — Watchpoint #0 triggers **nex_wevto[1]** 0010 — Watchpoint #1 triggers **nex_wevto[1]** 0011 — Watchpoint #2 triggers **nex_wevto[1]** 0100 — Watchpoint #3 triggers **nex_wevto[1]** 0101 — Watchpoint #4 triggers **nex_wevto[1]** 0110 — Watchpoint #5 triggers **nex_wevto[1]** 0111 — Watchpoint #6 triggers **nex_wevto[1]** 1000 — Watchpoint #7 triggers **nex_wevto[1]** 1001 — Watchpoint #8 triggers **nex_wevto[1]** 1010 — Watchpoint #9 triggers **nex_wevto[1]** 1011 — Watchpoint #10 triggers **nex_wevto[1]** 1100 — Watchpoint #11 triggers **nex_wevto[1]** 1101 — Watchpoint #12 triggers **nex_wevto[1]** 1110 — Watchpoint #13 triggers **nex_wevto[1]** 1111 — Watchpoint #14 triggers **nex_wevto[1]**

Table 2105. DC2 field descriptions (continued)

Bits	Name	Description
19:16	WEVTO[0]C	Watchpoint Event Out 0 Configuration 0000 — No Watchpoints #0–14 trigger **nex_wevto[0]** 0001 — Watchpoint #0 triggers **nex_wevto[0]** 0010 — Watchpoint #1 triggers **nex_wevto[0]** 0011 — Watchpoint #2 triggers **nex_wevto[0]** 0100 — Watchpoint #3 triggers **nex_wevto[0]** 0101 — Watchpoint #4 triggers **nex_wevto[0]** 0110 — Watchpoint #5 triggers **nex_wevto[0]** 0111 — Watchpoint #6 triggers **nex_wevto[0]** 1000 — Watchpoint #7 triggers **nex_wevto[0]** 1001 — Watchpoint #8 triggers **nex_wevto[0]** 1010 — Watchpoint #9 triggers **nex_wevto[0]** 1011 — Watchpoint #10 triggers **nex_wevto[0]** 1100 — Watchpoint #11 triggers **nex_wevto[0]** 1101 — Watchpoint #12 triggers **nex_wevto[0]** 1110 — Watchpoint #13 triggers **nex_wevto[0]** 1111 — Watchpoint #14 triggers **nex_wevto[0]**
15:0	EWC	EVTO Watchpoint Configuration(1) 0000000000000000 No Watchpoints #0–15 trigger **nex_evto_b** *xxxxxxxxxxxxxxx*1 Watchpoint #0 triggers **nex_evto_b** *xxxxxxxxxxxxxx*1*x* Watchpoint #1 triggers **nex_evto_b** *xxxxxxxxxxxxx*1*xx* Watchpoint #2 triggers **nex_evto_b** *xxxxxxxxxxxx*1*xxx* Watchpoint #3 triggers **nex_evto_b** *xxxxxxxxxxx*1*xxxx* Watchpoint #4 triggers **nex_evto_b** *xxxxxxxxxx*1*xxxxx* Watchpoint #5 triggers **nex_evto_b** *xxxxxxxxx*1*xxxxxx* Watchpoint #6 triggers **nex_evto_b** *xxxxxxxx*1*xxxxxxx* Watchpoint #7 triggers **nex_evto_b** *xxxxxxx*1*xxxxxxxx* Watchpoint #8 triggers **nex_evto_b** *xxxxxx*1*xxxxxxxxx* Watchpoint #9 triggers **nex_evto_b** *xxxxx*1*xxxxxxxxxx* Watchpoint #10 triggers **nex_evto_b** *xxxx*1*xxxxxxxxxxx* Watchpoint #11 triggers **nex_evto_b** *xxx*1*xxxxxxxxxxxx* Watchpoint #12 triggers **nex_evto_b** *xx*1*xxxxxxxxxxxxx* Watchpoint #13 triggers **nex_evto_b** *x*1*xxxxxxxxxxxxxx* Watchpoint #14 triggers **nex_evto_b** 1*xxxxxxxxxxxxxxx* Watchpoint #15 triggers **nex_evto_b**

1. EOC bits in DC1 must be programmed to trigger $\overline{\text{EVTO}}$ on Watchpoint occurrence for EWC bits to have any effect.

Development Control Register 3 is shown in *Figure 2206* and its fields are described in *Table 2106*.

WEVTO[3]C	WEVTO[2]C	WEVTO[1]C	WEVTO[0]C	0	EWC

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

Nexus Reg# 0x4; DCR: 370; Read/Write; Reset: 0x0

Figure 2206. Development Control 3 (DC3) register

Table 2106. DC3 field descriptions

Bits	Name	Description
31:28	WEVTO[3]C	Watchpoint Event Out 3 Configuration 0000 No Watchpoints #15–#26 trigger **nex_wevto[3]** 0001 Watchpoint #15 triggers **nex_wevto[3]** 0010 Watchpoint #16 triggers **nex_wevto[3]** 0011 Watchpoint #17 triggers **nex_wevto[3]** 0100 Watchpoint #18 triggers **nex_wevto[3]** 0101 Watchpoint #19 triggers **nex_wevto[3]** 0110 Watchpoint #20 triggers **nex_wevto[3]** 0111 Watchpoint #21 triggers **nex_wevto[3]** 1000 Watchpoint #22 triggers **nex_wevto[3]** 1001 Watchpoint #23 triggers **nex_wevto[3]** 1010 Watchpoint #24 triggers **nex_wevto[3]** 1011 Watchpoint #25 triggers **nex_wevto[3]** 1100 Watchpoint #26 triggers **nex_wevto[3]** 11*x*1 Reserved
27:24	WEVTO[2]C	Watchpoint Event Out 2 Configuration 0000 No Watchpoints #15–#26 trigger **nex_wevto[2]** 0001 Watchpoint #15 triggers **nex_wevto[2]** 0010 Watchpoint #16 triggers **nex_wevto[2]** 0011 Watchpoint #17 triggers **nex_wevto[2]** 0100 Watchpoint #18 triggers **nex_wevto[2]** 0101 Watchpoint #19 triggers **nex_wevto[2]** 0110 Watchpoint #20 triggers **nex_wevto[2]** 0111 Watchpoint #21 triggers **nex_wevto[2]** 1000 Watchpoint #22 triggers **nex_wevto[2]** 1001 Watchpoint #23 triggers **nex_wevto[2]** 1010 Watchpoint #24 triggers **nex_wevto[2]** 1011 Watchpoint #25 triggers **nex_wevto[2]** 1100 Watchpoint #26 triggers **nex_wevto[2]** 11*x*1 Reserved
23:20	WEVTO[1]C	Watchpoint Event Out 1 Configuration 0000 No Watchpoints #15–#29 trigger **nex_wevto[1]** 0001 Watchpoint #15 triggers **nex_wevto[1]** 0010 Watchpoint #16 triggers **nex_wevto[1]** 0011 Watchpoint #17 triggers **nex_wevto[1]** 0100 Watchpoint #18 triggers **nex_wevto[1]** 0101 Watchpoint #19 triggers **nex_wevto[1]** 0110 Watchpoint #20 triggers **nex_wevto[1]** 0111 Watchpoint #21 triggers **nex_wevto[1]** 1000 Watchpoint #22 triggers **nex_wevto[1]** 1001 Watchpoint #23 triggers **nex_wevto[1]** 1010 Watchpoint #24 triggers **nex_wevto[1]** 1011 Watchpoint #25 triggers **nex_wevto[1]** 1100 Watchpoint #26 triggers **nex_wevto[1]** 1101 Watchpoint #27 triggers **nex_wevto[1]** 1110 Watchpoint #28 triggers **nex_wevto[1]** 1111 Watchpoint #29 triggers **nex_wevto[1]**

Table 2106. DC3 field descriptions (continued)

Bits	Name	Description
19:16	WEVTO[0]C	Watchpoint Event Out 0 Configuration 0000 No Watchpoints #15–#29 trigger **nex_wevto[0]** 0001 Watchpoint #15 triggers **nex_wevto[0]** 0010 Watchpoint #16 triggers **nex_wevto[0]** 0011 Watchpoint #17 triggers **nex_wevto[0]** 0100 Watchpoint #18 triggers **nex_wevto[0]** 0101 Watchpoint #19 triggers **nex_wevto[0]** 0110 Watchpoint #20 triggers **nex_wevto[0]** 0111 Watchpoint #21 triggers **nex_wevto[0]** 1000 Watchpoint #22 triggers **nex_wevto[0]** 1001 Watchpoint #23 triggers **nex_wevto[0]** 1010 Watchpoint #24 triggers **nex_wevto[0]** 1011 Watchpoint #25 triggers **nex_wevto[0]** 1100 Watchpoint #26 triggers **nex_wevto[0]** 1101 Watchpoint #27 triggers **nex_wevto[0]** 1110 Watchpoint #28 triggers **nex_wevto[0]** 1111 Watchpoint #29 triggers **nex_wevto[0]**
15:14	—	Reserved for watchpoint expansion
13:0	EWC	EVTO Watchpoint Configuration(1) 00000000000000 No Watchpoints #16–#29 trigger **nex_evto_b** *xxxxxxxxxxxxx*1 Watchpoint #16 triggers **nex_evto_b** *xxxxxxxxxxxx*1*x* Watchpoint #17 triggers **nex_evto_b** *xxxxxxxxxxx*1*xx* Watchpoint #18 triggers **nex_evto_b** *xxxxxxxxxx*1*xxx* Watchpoint #19 triggers **nex_evto_b** *xxxxxxxxx*1*xxxx* Watchpoint #20 triggers **nex_evto_b** *xxxxxxxx*1*xxxxx* Watchpoint #21 triggers **nex_evto_b** *xxxxxxx*1*xxxxxx* Watchpoint #22 triggers **nex_evto_b** *xxxxxx*1*xxxxxxx* Watchpoint #23 triggers **nex_evto_b** *xxxxx*1*xxxxxxxx* Watchpoint #24 triggers **nex_evto_b** *xxxx*1*xxxxxxxxx* Watchpoint #25 triggers **nex_evto_b** *xxx*1*xxxxxxxxxx* Watchpoint #26 triggers **nex_evto_b** *xx*1*xxxxxxxxxxx* Watchpoint #27 triggers **nex_evto_b** *x*1*xxxxxxxxxxxx* Watchpoint #28 triggers **nex_evto_b** 1*xxxxxxxxxxxxx* Watchpoint #29 triggers **nex_evto_b**

1. EOC bits in DC1 must be programmed to trigger $\overline{\text{EVTO}}$ on Watchpoint occurrence for EWC bits to have any effect.

84.4.5 Nexus Development Control Register 4 (DC4)

Nexus Development Control Register 4 is used to control mark selection for Program and Data Trace Messaging, as well as masking of events that initiate Program Correlation Messages on the Nexus 3 module.

Development Control Register 4 is shown in *Figure 2207* and its fields are described in *Table 2107*.

PTMARK	DTMARK	0	EVCDM

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

Nexus Reg# 0x5; DCR: 371; Read/Write; Reset: 0x0

Figure 2207. Development Control 4 (DC4) register

Table 2107. DC4 field descriptions

Bits	Name	Description
31	PTMARK	Program Trace Mark 0 Ignore MSR_{PMM} for masking program trace messages 1 Mask program trace messages when MSR_{PMM}='0', unmask program trace messages when MSR_{PMM}='1'
30	DTMARK	Data Trace Mark 0 Ignore MSR_{PMM} for masking data trace messages 1 Mask data trace messages when MSR_{PMM}='0', unmask data trace messages when MSR_{PMM}='1'
29:16	—	Reserved
15:0	EVCDM	Event Code (EVCODE) Mask(1) 0000000000000000 No EVCODEs masked for Program Correlation Messages *xxxxxxxxxxxxxxx*1 EVCODE #0 is masked for Program Correlation Messages *xxxxxxxxxxxxxx*1*x* EVCODE #1 is masked for Program Correlation Messages *xxxxxxxxxxxxx*1*xx* EVCODE #2 is masked for Program Correlation Messages *xxxxxxxxxxxx*1*xxx* EVCODE #3 is masked for Program Correlation Messages *xxxxxxxxxxx*1*xxxx* EVCODE #4 is masked for Program Correlation Messages *xxxxxxxxxx*1*xxxxx* EVCODE #5 is masked for Program Correlation Messages *xxxxxxxxx*1*xxxxxx* EVCODE #6 is masked for Program Correlation Messages *xxxxxxxx*1*xxxxxxx* EVCODE #7 is masked for Program Correlation Messages *xxxxxxx*1*xxxxxxxx* EVCODE #8 is masked for Program Correlation Messages *xxxxxx*1*xxxxxxxxx* EVCODE #9 is masked for Program Correlation Messages *xxxxx*1*xxxxxxxxxx* EVCODE #10 is masked for Program Correlation Messages *xxxx*1*xxxxxxxxxxx* EVCODE #11 is masked for Program Correlation Messages *xxx*1*xxxxxxxxxxxx* EVCODE #12 is masked for Program Correlation Messages *xx*1*xxxxxxxxxxxxx* EVCODE #13 is masked for Program Correlation Messages *x*1*xxxxxxxxxxxxxx* EVCODE #14 is masked for Program Correlation Messages 1*xxxxxxxxxxxxxxx* EVCODE #15 is masked for Program Correlation Messages

1. Refer to *Table 2099* for implemented EVCODEs.

84.4.6 Development Status Register (DS)

The Development Status Register is used to report system debug status. When Debug Mode is entered or exited, or an SoC- or core-defined Low Power Mode is entered (see note below), a Debug Status Message is transmitted with DS[31:24]. The external tool can read this register at any time.



Nexus Reg# 0x4; DCR: 409; Read-only; Reset: 0x0

Figure 2208. Development Status (DS) register

Table 2108. DS field descriptions

Bits	Name	Description
31	DBG	CPU Debug Mode Status 0 CPU not in Debug mode 1 CPU in Debug mode (**jd_debug_b** signal asserted)
30:28	LPS	System Low Power Mode Status 000 Normal (Run) mode 001 Waiting State (**p_waiting** signal asserted 010 Halted State (**p_halted** signal asserted 011 Reserved 1*xx* Reserved
27:26	LPC	CPU Low Power Mode Status 00 Normal (Run) mode 01 CPU in Halted state (**p_halted** signal asserted) 10 CPU in Stopped state (**p_stopped** signal asserted) 11 CPU in Waiting state (**p_waiting** signal asserted)
25:0	—	Reserved for future functionality (read as 0)

84.4.7 Watchpoint Trigger (WT, PTSTC, PTETC, DTSTC, DTETC) registers

The Watchpoint Trigger Registers allows the watchpoints defined within the Nexus1 logic to trigger actions. These watchpoints can control Program or Data Trace, or both, enable and disable. The control bits can be used to produce a related “window” for triggering Trace Messages. Watchpoint trigger register WT is used to control triggering by a single selected watchpoint. The Program Trace Start Trigger Control (PTSTC), Program Trace End Trigger Control (PTETC), Data Trace Start Trigger Control (DTSTC), and Data Trace End Trigger Control (DTETC) are used for extended trigger controls for the respective function. If multiple watchpoints are desired for triggering, or a watchpoint beyond watchpoint #13 is required, then one or more of the extended watchpoint trigger registers may be used. A field encoding of 4’b1111 in one of the WT register fields enables the corresponding extended trigger register. For all other WT field encodings, the corresponding extended trigger register is disabled and the contents are ignored. Note that direct writes to enable program trace or data trace in DC1, or both, will override these controls, and trace will remain enabled until another direct write to DC1 to disable program or data trace, or both, occurs.

When a start trigger is detected, the designated trace features become enabled, and the corresponding enable bits of the DC1 register are set. Whenever a stop trigger is detected, the designated trace features become disabled, and the corresponding enable bits of the DC1 register are cleared. If the same trigger condition is used for both start and stop triggering, then the designated trace features will toggle between being enabled and disabled at each occurrence of the trigger condition. Similarly, if start and stop triggers for a trace feature occur simultaneously, then the designated trace feature will toggle between

enabled and disabled depending on the enable state at the time of the trigger events. For example, if tracing is enabled, and a start and stop trigger occur simultaneously, then tracing will be disabled. Direct writes of the DC1 register take precedence over any trace feature enable state that is derived from watchpoint triggering. A table of watchpoints can be found in the Core (e200z425n3) Core Debug Support chapter.

PTS	PTE	DTS	DTE	0

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

Nexus Reg# 0xB; DCR: 375; Read/Write; Reset: 0x0

Figure 2209. Watchpoint Trigger (WT) register

Table 2109 details the Watchpoint Trigger register fields.

Table 2109. WT field descriptions

Bits	Name	Description
31:28	PTS	Program Trace Start Control 0000 Trigger disabled 0001 Use Watchpoint #0 0010 Use Watchpoint #1 . . 1110 Use Watchpoint #13 1111 Use control settings in the PTSTC register
27:24	PTE	Program Trace End Control 0000 Trigger disabled 0001 Use Watchpoint #0 0010 Use Watchpoint #1 . . 1110 Use Watchpoint #13 1111 Use control settings in the PTETC register
23:20	DTS	Data Trace Start Control 0000 Trigger disabled 0001 Use Watchpoint #0 0010 Use Watchpoint #1 . . 1110 Use Watchpoint #13 1111 Use control settings in the DTSTC register
19:16	DTE	Data Trace End Control 0000 Trigger disabled 0001 Use Watchpoint #0 0010 Use Watchpoint #1 . . 1110 Use Watchpoint #13 1111 Use control settings in the DTETC register
15:0	—	Reserved for future functionality (read as 0)

For extended Program Trace start trigger control, the PTSTC register is used.

PTST

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

Nexus Reg# 0x35; DCR: 412; Read/Write; Reset: 0x0

Figure 2210. Program Trace Start Trigger Control (PTSTC) register

Table 2110 details the PTSTC register fields.

Table 2110. PTSTC field descriptions

Bits	Name	Description
31:0	PTST	Program Trace Start Trigger Control 00000000000000000000000000000000 Trigger disabled *xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx*1 Use Watchpoint #0 *xxxxxxxxxxxxxxxxxxxxxxxxxxxxxx*1*x* Use Watchpoint #1 *xxxxxxxxxxxxxxxxxxxxxxxxxxxxx*1*xx* Use Watchpoint #2 *xxxxxxxxxxxxxxxxxxxxxxxxxxxx*1*xxx* Use Watchpoint #3 *xxxxxxxxxxxxxxxxxxxxxxxxxxx*1*xxxx* Use Watchpoint #4 *xxxxxxxxxxxxxxxxxxxxxxxxxx*1*xxxxx* Use Watchpoint #5 *xxxxxxxxxxxxxxxxxxxxxxxxx*1*xxxxxx* Use Watchpoint #6 *xxxxxxxxxxxxxxxxxxxxxxxx*1*xxxxxxx* Use Watchpoint #7 *xxxxxxxxxxxxxxxxxxxxxxx*1*xxxxxxxx* Use Watchpoint #8 *xxxxxxxxxxxxxxxxxxxxxx*1*xxxxxxxxx* Use Watchpoint #9 *xxxxxxxxxxxxxxxxxxxxx*1*xxxxxxxxxx* Use Watchpoint #10 *xxxxxxxxxxxxxxxxxxxx*1*xxxxxxxxxxx* Use Watchpoint #11 *xxxxxxxxxxxxxxxxxxx*1*xxxxxxxxxxxx* Use Watchpoint #12 *xxxxxxxxxxxxxxxxxx*1*xxxxxxxxxxxxx* Use Watchpoint #13 *xxxxxxxxxxxxxxxxx*1*xxxxxxxxxxxxxx* Use Watchpoint #14 *xxxxxxxxxxxxxxxx*1*xxxxxxxxxxxxxxx* Use Watchpoint #15 *xxxxxxxxxxxxxxx*1*xxxxxxxxxxxxxxxx* Use Watchpoint #16 *xxxxxxxxxxxxxx*1*xxxxxxxxxxxxxxxxx* Use Watchpoint #17 *xxxxxxxxxxxxx*1*xxxxxxxxxxxxxxxxxx* Use Watchpoint #18 *xxxxxxxxxxxx*1*xxxxxxxxxxxxxxxxxxx* Use Watchpoint #19 *xxxxxxxxxxx*1*xxxxxxxxxxxxxxxxxxxx* Use Watchpoint #20 *xxxxxxxxxx*1*xxxxxxxxxxxxxxxxxxxxx* Use Watchpoint #21 *xxxxxxxxx*1*xxxxxxxxxxxxxxxxxxxxxx* Use Watchpoint #22 *xxxxxxxx*1*xxxxxxxxxxxxxxxxxxxxxxx* Use Watchpoint #23 *xxxxxxx*1*xxxxxxxxxxxxxxxxxxxxxxxx* Use Watchpoint #24 *xxxxxx*1*xxxxxxxxxxxxxxxxxxxxxxxxx* Use Watchpoint #25 *xxxxx*1*xxxxxxxxxxxxxxxxxxxxxxxxxx* Use Watchpoint #26 *xxxx*1*xxxxxxxxxxxxxxxxxxxxxxxxxxx* Use Watchpoint #27 *xxx*1*xxxxxxxxxxxxxxxxxxxxxxxxxxxx* Use Watchpoint #28 *xx*1*xxxxxxxxxxxxxxxxxxxxxxxxxxxxx* Use Watchpoint #29 *x*1*xxxxxxxxxxxxxxxxxxxxxxxxxxxxxx* Use Watchpoint #30 1*xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx* Use Watchpoint #31

For extended Program Trace end trigger control, the PTETC register is used.

PTET

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

Nexus Reg# 0x36; DCR: 413; Read/Write; Reset: 0x0

Figure 2211. Program Trace End Trigger Control (PTETC) register

Table 2111 details the PTETC register fields.

Table 2111. PTETC field descriptions

Bits	Name	Description
31:0	PTET	Program Trace End Trigger Control 00000000000000000000000000000000 Trigger disabled *xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx*1 Use Watchpoint #0 *xxxxxxxxxxxxxxxxxxxxxxxxxxxxxx*1*x* Use Watchpoint #1 *xxxxxxxxxxxxxxxxxxxxxxxxxxxxx*1*xx* Use Watchpoint #2 *xxxxxxxxxxxxxxxxxxxxxxxxxxxx*1*xxx* Use Watchpoint #3 *xxxxxxxxxxxxxxxxxxxxxxxxxxx*1*xxxx* Use Watchpoint #4 *xxxxxxxxxxxxxxxxxxxxxxxxxx*1*xxxxx* Use Watchpoint #5 *xxxxxxxxxxxxxxxxxxxxxxxxx*1*xxxxxx* Use Watchpoint #6 *xxxxxxxxxxxxxxxxxxxxxxxx*1*xxxxxxx* Use Watchpoint #7 *xxxxxxxxxxxxxxxxxxxxxxx*1*xxxxxxxx* Use Watchpoint #8 *xxxxxxxxxxxxxxxxxxxxxx*1*xxxxxxxxx* Use Watchpoint #9 *xxxxxxxxxxxxxxxxxxxxx*1*xxxxxxxxxx* Use Watchpoint #10 *xxxxxxxxxxxxxxxxxxxx*1*xxxxxxxxxxx* Use Watchpoint #11 *xxxxxxxxxxxxxxxxxxx*1*xxxxxxxxxxxx* Use Watchpoint #12 *xxxxxxxxxxxxxxxxxx*1*xxxxxxxxxxxxx* Use Watchpoint #13 *xxxxxxxxxxxxxxxxx*1*xxxxxxxxxxxxxx* Use Watchpoint #14 *xxxxxxxxxxxxxxxx*1*xxxxxxxxxxxxxxx* Use Watchpoint #15 *xxxxxxxxxxxxxxx*1*xxxxxxxxxxxxxxxx* Use Watchpoint #16 *xxxxxxxxxxxxxx*1*xxxxxxxxxxxxxxxx* Use Watchpoint #17 *xxxxxxxxxxxxx*1*xxxxxxxxxxxxxxxx* Use Watchpoint #18 *xxxxxxxxxxxx*1*xxxxxxxxxxxxxxxx* Use Watchpoint #19 *xxxxxxxxxxx*1*xxxxxxxxxxxxxxxx* Use Watchpoint #20 *xxxxxxxxxx*1*xxxxxxxxxxxxxxxx* Use Watchpoint #21 *xxxxxxxxx*1*xxxxxxxxxxxxxxxx* Use Watchpoint #22 *xxxxxxxx*1*xxxxxxxxxxxxxxxx* Use Watchpoint #23 *xxxxxxx*1*xxxxxxxxxxxxxxxx* Use Watchpoint #24 *xxxxxx*1*xxxxxxxxxxxxxxxx* Use Watchpoint #25 *xxxxx*1*xxxxxxxxxxxxxxxx* Use Watchpoint #26 *xxxx*1*xxxxxxxxxxxxxxxx* Use Watchpoint #27 *xxx*1*xxxxxxxxxxxxxxxx* Use Watchpoint #28 *xx*1*xxxxxxxxxxxxxxxx* Use Watchpoint #29 *x*1*xxxxxxxxxxxxxxxx* Use Watchpoint #30 1*xxxxxxxxxxxxxxxx* Use Watchpoint #31

For extended Data Trace start trigger control, the DTSTC register is used.

0	DTST

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

Nexus Reg# 0x37; DCR: 414; Read/Write; Reset: 0x0

Figure 2212. Data Trace Start Trigger Control (DTSTC) register

The following table details the DTSTC register fields.

Table 2112. DTSTC field descriptions

Bits	Name	Description
31:30	—	Reserved for future functionality (read as 0)
29:0	DTST	Data Trace Start Trigger Control 00000000000000000000000000000000 Trigger disabled *xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx*1 Use Watchpoint #0 *xxxxxxxxxxxxxxxxxxxxxxxxxxxxxx*1*x* Use Watchpoint #1 *xxxxxxxxxxxxxxxxxxxxxxxxxxxxx*1*xx* Use Watchpoint #2 *xxxxxxxxxxxxxxxxxxxxxxxxxxxx*1*xxx* Use Watchpoint #3 *xxxxxxxxxxxxxxxxxxxxxxxxxxx*1*xxxx* Use Watchpoint #4 *xxxxxxxxxxxxxxxxxxxxxxxxxx*1*xxxxx* Use Watchpoint #5 *xxxxxxxxxxxxxxxxxxxxxxxxx*1*xxxxxx* Use Watchpoint #6 *xxxxxxxxxxxxxxxxxxxxxxxx*1*xxxxxxx* Use Watchpoint #7 *xxxxxxxxxxxxxxxxxxxxxxx*1*xxxxxxxx* Use Watchpoint #8 *xxxxxxxxxxxxxxxxxxxxxx*1*xxxxxxxxx* Use Watchpoint #9 *xxxxxxxxxxxxxxxxxxxxx*1*xxxxxxxxxx* Use Watchpoint #10 *xxxxxxxxxxxxxxxxxxxx*1*xxxxxxxxxxx* Use Watchpoint #11 *xxxxxxxxxxxxxxxxxxx*1*xxxxxxxxxxxx* Use Watchpoint #12 *xxxxxxxxxxxxxxxxxx*1*xxxxxxxxxxxxx* Use Watchpoint #13 *xxxxxxxxxxxxxxxxx*1*xxxxxxxxxxxxxx* Use Watchpoint #14 *xxxxxxxxxxxxxxxx*1*xxxxxxxxxxxxxxx* Use Watchpoint #15 *xxxxxxxxxxxxxxx*1*xxxxxxxxxxxxxxxx* Use Watchpoint #16 *xxxxxxxxxxxxxx*1*xxxxxxxxxxxxxxxx* Use Watchpoint #17 *xxxxxxxxxxxxx*1*xxxxxxxxxxxxxxxx* Use Watchpoint #18 *xxxxxxxxxxxx*1*xxxxxxxxxxxxxxxx* Use Watchpoint #19 *xxxxxxxxxxx*1*xxxxxxxxxxxxxxxx* Use Watchpoint #20 *xxxxxxxxxx*1*xxxxxxxxxxxxxxxx* Use Watchpoint #21 *xxxxxxxxx*1*xxxxxxxxxxxxxxxx* Use Watchpoint #22 *xxxxxxxx*1*xxxxxxxxxxxxxxxx* Use Watchpoint #23 *xxxxxxx*1*xxxxxxxxxxxxxxxx* Use Watchpoint #24 *xxxxxx*1*xxxxxxxxxxxxxxxx* Use Watchpoint #25 *xxxxx*1*xxxxxxxxxxxxxxxx* Use Watchpoint #26 *xxxx*1*xxxxxxxxxxxxxxxx* Use Watchpoint #27 *xxx*1*xxxxxxxxxxxxxxxx* Use Watchpoint #28 *xx*1*xxxxxxxxxxxxxxxx* Use Watchpoint #29 *x*1*xxxxxxxxxxxxxxxx* Use Watchpoint #30 1*xxxxxxxxxxxxxxxx* Use Watchpoint #31

For extended Data Trace end trigger control, the DTETC register is used.

0	DTET

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

Nexus Reg# 038; DCR: 415; Read/Write; Reset: 0x0

Figure 2213. Data Trace End Trigger Control (DTETC) register

The following table details the DTETC register fields.

Table 2113. DTETC field descriptions

Bits	Name	Description
31:30	—	Reserved for future functionality (read as 0)
29:0	DTET	Data Trace End Trigger Control 00000000000000000000000000000000 Trigger disabled *xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx*1 Use Watchpoint #0 *xxxxxxxxxxxxxxxxxxxxxxxxxxxxxx*1*x* Use Watchpoint #1 *xxxxxxxxxxxxxxxxxxxxxxxxxxxxx*1*xx* Use Watchpoint #2 *xxxxxxxxxxxxxxxxxxxxxxxxxxxx*1*xxx* Use Watchpoint #3 *xxxxxxxxxxxxxxxxxxxxxxxxxxx*1*xxxx* Use Watchpoint #4 *xxxxxxxxxxxxxxxxxxxxxxxxxx*1*xxxxx* Use Watchpoint #5 *xxxxxxxxxxxxxxxxxxxxxxxxx*1*xxxxxx* Use Watchpoint #6 *xxxxxxxxxxxxxxxxxxxxxxxx*1*xxxxxxx* Use Watchpoint #7 *xxxxxxxxxxxxxxxxxxxxxxx*1*xxxxxxxx* Use Watchpoint #8 *xxxxxxxxxxxxxxxxxxxxxx*1*xxxxxxxxx* Use Watchpoint #9 *xxxxxxxxxxxxxxxxxxxxx*1*xxxxxxxxxx* Use Watchpoint #10 *xxxxxxxxxxxxxxxxxxxx*1*xxxxxxxxxxx* Use Watchpoint #11 *xxxxxxxxxxxxxxxxxxx*1*xxxxxxxxxxxx* Use Watchpoint #12 *xxxxxxxxxxxxxxxxxx*1*xxxxxxxxxxxxx* Use Watchpoint #13 *xxxxxxxxxxxxxxxxx*1*xxxxxxxxxxxxxx* Use Watchpoint #14 *xxxxxxxxxxxxxxxx*1*xxxxxxxxxxxxxxx* Use Watchpoint #15 *xxxxxxxxxxxxxxx*1*xxxxxxxxxxxxxxxx* Use Watchpoint #16 *xxxxxxxxxxxxxx*1*xxxxxxxxxxxxxxxx* Use Watchpoint #17 *xxxxxxxxxxxxx*1*xxxxxxxxxxxxxxxx* Use Watchpoint #18 *xxxxxxxxxxxx*1*xxxxxxxxxxxxxxxx* Use Watchpoint #19 *xxxxxxxxxxx*1*xxxxxxxxxxxxxxxx* Use Watchpoint #20 *xxxxxxxxxx*1*xxxxxxxxxxxxxxxx* Use Watchpoint #21 *xxxxxxxxx*1*xxxxxxxxxxxxxxxx* Use Watchpoint #22 *xxxxxxxx*1*xxxxxxxxxxxxxxxx* Use Watchpoint #23 *xxxxxxx*1*xxxxxxxxxxxxxxxx* Use Watchpoint #24 *xxxxxx*1*xxxxxxxxxxxxxxxx* Use Watchpoint #25 *xxxxx*1*xxxxxxxxxxxxxxxx* Use Watchpoint #26 *xxxx*1*xxxxxxxxxxxxxxxx* Use Watchpoint #27 *xxx*1*xxxxxxxxxxxxxxxx* Use Watchpoint #28 *xx*1*xxxxxxxxxxxxxxxx* Use Watchpoint #29 *x*1*xxxxxxxxxxxxxxxx* Use Watchpoint #30 1*xxxxxxxxxxxxxxxx* Use Watchpoint #31

84.4.8 Nexus Watchpoint Mask (WMSK) register

The Nexus Watchpoint Mask register controls which watchpoint events are enabled to produce Watchpoint Trace Messages ($DC1_{TM}$ must also be programmed to generate Watchpoint Trace Messages).

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

Nexus Reg# 0x33; DCR: 411; Read/Write; Reset: 0x0

Figure 2214. Watchpoint Mask (WMSK) register

Table 2114 details the Watchpoint Trigger register fields.

Table 2114. WMSK field descriptions

Bits	Name	Description
31:0	WEM	Watchpoint Enable for Messaging 00000000000000000000000000000000 No Watchpoints enabled for Watchpoint Trace Messaging *xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx*1 Watchpoint #0 enabled for WTM *xxxxxxxxxxxxxxxxxxxxxxxxxxxxxx*1*x* Watchpoint #1 enabled for WTM *xxxxxxxxxxxxxxxxxxxxxxxxxxxxx*1*xx* Watchpoint #2 enabled for WTM *xxxxxxxxxxxxxxxxxxxxxxxxxxxx*1*xxx* Watchpoint #3 enabled for WTM *xxxxxxxxxxxxxxxxxxxxxxxxxxx*1*xxxx* Watchpoint #4 enabled for WTM *xxxxxxxxxxxxxxxxxxxxxxxxxx*1*xxxxx* Watchpoint #5 enabled for WTM *xxxxxxxxxxxxxxxxxxxxxxxxx*1*xxxxxx* Watchpoint #6 enabled for WTM *xxxxxxxxxxxxxxxxxxxxxxxx*1*xxxxxxx* Watchpoint #7 enabled for WTM *xxxxxxxxxxxxxxxxxxxxxxx*1*xxxxxxxx* Watchpoint #8 enabled for WTM *xxxxxxxxxxxxxxxxxxxxxx*1*xxxxxxxxx* Watchpoint #9 enabled for WTM *xxxxxxxxxxxxxxxxxxxxx*1*xxxxxxxxxx* Watchpoint #10 enabled for WTM *xxxxxxxxxxxxxxxxxxxx*1*xxxxxxxxxxx* Watchpoint #11 enabled for WTM *xxxxxxxxxxxxxxxxxxx*1*xxxxxxxxxxxx* Watchpoint #12 enabled for WTM *xxxxxxxxxxxxxxxxxx*1*xxxxxxxxxxxxx* Watchpoint #13 enabled for WTM *xxxxxxxxxxxxxxxxx*1*xxxxxxxxxxxxxx* Watchpoint #14 enabled for WTM *xxxxxxxxxxxxxxxx*1*xxxxxxxxxxxxxxx* Watchpoint #15 enabled for WTM *xxxxxxxxxxxxxxx*1*xxxxxxxxxxxxxxxx* Watchpoint #16 enabled for WTM *xxxxxxxxxxxxxx*1*xxxxxxxxxxxxxxxxx* Watchpoint #17 enabled for WTM *xxxxxxxxxxxxx*1*xxxxxxxxxxxxxxxxxx* Watchpoint #18 enabled for WTM *xxxxxxxxxxxx*1*xxxxxxxxxxxxxxxxxxx* Watchpoint #19 enabled for WTM *xxxxxxxxxxx*1*xxxxxxxxxxxxxxxxxxxx* Watchpoint #20 enabled for WTM *xxxxxxxxxx*1*xxxxxxxxxxxxxxxxxxxxx* Watchpoint #21 enabled for WTM *xxxxxxxxx*1*xxxxxxxxxxxxxxxxxxxxxx* Watchpoint #22 enabled for WTM *xxxxxxxx*1*xxxxxxxxxxxxxxxxxxxxxxx* Watchpoint #23 enabled for WTM *xxxxxxx*1*xxxxxxxxxxxxxxxxxxxxxxxx* Watchpoint #24 enabled for WTM *xxxxxx*1*xxxxxxxxxxxxxxxxxxxxxxxxx* Watchpoint #25 enabled for WTM *xxxxx*1*xxxxxxxxxxxxxxxxxxxxxxxxxx* Watchpoint #26 enabled for WTM *xxxx*1*xxxxxxxxxxxxxxxxxxxxxxxxxxx* Watchpoint #27 enabled for WTM *xxx*1*xxxxxxxxxxxxxxxxxxxxxxxxxxxx* Watchpoint #28 enabled for WTM *xx*1*xxxxxxxxxxxxxxxxxxxxxxxxxxxxx* Watchpoint #29 enabled for WTM *x*1*xxxxxxxxxxxxxxxxxxxxxxxxxxxxxx* Watchpoint #30 enabled for WTM 1*xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx* Watchpoint #31 enabled for WTM

84.4.9 Nexus Overrun Control (OVCR) register

The Nexus Overrun Control register controls Nexus behavior as the internal message queues fill up. Response options include suppressing selected message types, or stalling processor instruction execution.

0	SPTHOLD	0	SPEN	0	STTHOLD	0	STEN

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

Nexus Reg# 0x32; DCR: 410; Read/Write; Reset: 0x0

Figure 2215. Nexus Overrun Control (OVCR) register

Table 2115. OVCR field descriptions

Bits	Name	Description
31:30	—	Reserved, should be cleared
29:28	SPTHOLD	Suppression Threshold 00 Suppression threshold is when message queues are 1/4 full 01 Suppression threshold is when message queues are 1/2 full 10 Suppression threshold is when message queues are 3/4 full 11 Reserved
27:22	—	Reserved, should be cleared
21:16	SPEN	Suppression Enable 000000 Suppression is disabled *xxxxx*1 Ownership Trace message suppression is enabled *xxxx*1*x* Data Trace message suppression is enabled *xxx*1*xx* Program Trace message suppression is enabled *xx*1*xxx* Watchpoint Trace message suppression is enabled *x*1*xxxx* Reserved 1*xxxxx* Data Acquisition message suppression is enabled
15:14	—	Reserved, should be cleared
13:12	STTHOLD	Stall Threshold 00 Stall threshold is when message queues are 1/4 full 01 Stall threshold is when message queues are 1/2 full 10 Stall threshold is when message queues are 3/4 full 11 Reserved
11:1	—	Reserved, should be cleared
0	STEN	Stall Enable 0 Stalling is disabled 1 Stalling is enabled

84.4.10 Data Trace Control (DTC) register

The Data Trace Control Register controls whether DTM Messages are restricted to reads, writes, or both for a user programmable address range. In addition, control is provided to restrict data trace message generation to only those load or store accesses that are not stack-related, in order to minimize required data trace message bandwidth.

There are four Data Trace channels controlled by the DTC for the Nexus 3 module. Channels can be programmed to trace data accesses or instruction accesses, but not independently.

RWT1	RWT2	RWT3	RWT4	0	STDC1	STDC2	STDC3	STDC4	0	RC1	RC2	RC3	RC4	DI	0
31 30	29 28	27 26	25 24	23 22 21 20 19 18 17 16	15	14	13	12	11 10 9 8	7	6	5	4	3	2 1 0

Nexus Reg# 0xD; DCR: 376; Read/Write; Reset: 0x0

Figure 2216. Data Trace Control (DTC) register

The following table details the Data Trace Control register fields.

Table 2116. DTC field descriptions

Bits	Name	Description
31:30	RWT1	Read/Write Trace 1 00 No trace enabled *x*1 Enable Data Read Trace 1*x* Enable Data Write Trace
29:28	RWT2	Read/Write Trace 2 00 No trace enabled *x*1 Enable Data Read Trace 1*x* Enable Data Write Trace
27:26	RWT3	Read/Write Trace 3 00 No trace enabled *x*1 Enable Data Read Trace 1*x* Enable Data Write Trace
25:24	RWT4	Read/Write Trace 4 00 No trace enabled *x*1 Enable Data Read Trace 1*x* Enable Data Write Trace
23:16	—	Reserved for future functionality (read as 0)
15	STDC1	Stack Trace Disable Control 1 0 Tracing of stack accesses are not disabled for range 1 1 Tracing of stack accesses are disabled for range 1; data accesses using GPR R1 in effective address computations are not traced and do not generate data range watchpoint outputs
14	STDC2	Stack Trace Disable Control 2 0 Tracing of stack accesses are not disabled for range 2 1 Tracing of stack accesses are disabled for range 2; data accesses using GPR R1 in effective address computations are not traced and do not generate data range watchpoint outputs
13	STDC3	Stack Trace Disable Control 3 0 Tracing of stack accesses are not disabled for range 3 1 Tracing of stack accesses are disabled for range 3; data accesses using GPR R1 in effective address computations are not traced and do not generate data range watchpoint outputs
12	STDC4	Stack Trace Disable Control 4 0 Tracing of stack accesses are not disabled for range 4 1 Tracing of stack accesses are disabled for range 4; data accesses using GPR R1 in effective address computations are not traced and do not generate data range watchpoint outputs

Table 2116. DTC field descriptions (continued)

Bits	Name	Description
11:8	—	Reserved for future functionality (read as 0)
7	RC1	Range Control 1 0 Condition trace on address within range 1 Condition trace on address outside of range
6	RC2	Range Control 2 0 Condition trace on address within range 1 Condition trace on address outside of range
5	RC3	Range Control 3 0 Condition trace on address within range 1 Condition trace on address outside of range
4	RC4	Range Control 4 0 Condition trace on address within range 1 Condition trace on address outside of range
3	DI	Data Access / Instruction Access Trace 0 Condition trace on data accesses 1 Condition trace on instruction accesses **Note:** The support for monitoring instruction accesses is not guaranteed to be present in all versions of the Nexus 3 module. **Note:** The setting of the DI bit also affects the information provided on the data trace port outputs (Refer to the “Data Trace Port Signals” section in the Core (e200z425n3) Core Debug Support chapter.)
2:0	—	Reserved for future functionality (read as 0)

84.4.11 Data Trace Start Address Registers (DTSA1–4)

The Data Trace Start Address Registers define the start addresses for each trace channel.

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

Nexus Reg# 0xE; DCR: 377; Read/Write; Reset: 0x0

Figure 2217. Data Trace Start Address 1 (DTSA1) register

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

Nexus Reg# 0xF; DCR: 378; Read/Write; Reset: 0x0

Figure 2218. Data Trace Start Address 2 (DTSA2) register

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

Nexus Reg# 0x10; DCR: 379; Read/Write; Reset: 0x0

Figure 2219. Data Trace Start Address 3 (DTSA3) register

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

Nexus Reg# 0x11; DCR: 380; Read/Write; Reset: 0x0

Figure 2220. Data Trace Start Address 4 (DTSA4) register

84.4.12 Data Trace End Address 1–4 (DTEA1–4) registers

The Data Trace End Address Registers define the end addresses for each trace channel.

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

Nexus Reg# 0x12; DCR: 381; Read/Write; Reset: 0x0

Figure 2221. Data Trace End Address 1 register

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

Nexus Reg# 0x13; DCR: 382; Read/Write; Reset: 0x0

Figure 2222. Data Trace End Address 2 register

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

Nexus Reg# 0x14; DCR: 383; Read/Write; Reset: 0x0

Figure 2223. Data Trace End Address 3 register

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

Nexus Reg# 0x15; DCR: 408; Read/Write; Reset: 0x0

Figure 2224. Data Trace End Address 4 register

Table 2117 illustrates the range that will be selected for Data Trace for various cases of DTSA being less than, greater than, or equal to DTEA.

Table 2117. Data trace — address range options

Programmed values	Range Control Bit Value	Range Selected
DTSA < DTEA	0	DTSA → ← DTEA
DTSA < DTEA	1	← DTSA DTEA →
DTSA > DTEA	N/A	Invalid Range — no trace
DTSA = DTEA	N/A	Invalid Range — no trace

Note: DTSA must be less than DTEA in order to guarantee correct Data Write/Read Traces. Data Trace ranges are inclusive of the DTSA and DTEA addresses for Range Control settings indicating "within range," and are exclusive of the DTSA and DTEA addresses for Range Control settings indicating "outside of range."

Accesses that meet the range and access type qualifiers will cause assertion of a watchpoint output for Ranges 1, 2, and 3. There are three dedicated watchpoint outputs, one for each DTSA/DTEA sets 1, 2, and 3. Range 4 does not provide a watchpoint output. Note that when DTC_{DI} = 1, all instruction fetches and prefetches (including discarded prefetches) are monitored, and thus these range watchpoints differ from the IACx watchpoint outputs, which are not asserted for instructions that are not executed (that is when the instruction prefetch is discarded).

84.4.13 Read/Write Access Control/Status (RWCS) register

The Read Write Access Control/Status Register provides control for Read/Write Access. Read/Write access provides DMA-like access to memory-mapped resources on the AHB System bus either while the processor is halted, or during runtime. Control is provided over access type, size, count, and certain bus attributes. Note that the internal CPU interfaces to the caches and local memory are not accessible or utilized by this mechanism. Since the external interface is used, base address port settings are used to access local memory, and not the settings of local memory control register base address values. The RWCS Register also provides Read/Write Access Status information per *Table 2119*.

AC	RW	SZ	MAP	PR	ATTR	0	CNT	ERR	DV
31	30	29 28 27	26 25 24	23 22	21 20 19 18	17 16	15 14 13 12 11 10 9 8 7 6 5 4 3 2	1	0

Nexus Reg# 0x7; Read/Write(1); Reset: 0x0

Figure 2225. Read/Write Access Control/Status (RWCS) register

1. ERR and DV are read-only.

Table 2118. RWCS field descriptions

Bits	Name	Description
RWCS[31]	AC	Access Control 0 End access/ Access has completed 1 Start access
RWCS[30]	RW	Read/Write Select 0 Read access 1 Write access
RWCS[29:27]	SZ	Word Size 000 8-bit (byte) 001 16-bit (half-word) 010 32-bit (word) 011 64-bit (doubleword, requires two passes through RWD) 1*xx* Reserved (default to word)
RWCS[26:24]	MAP	MAP Select 000 Primary memory map All other values are reserved
RWCS[23:22]	PR(1)	Read/Write Access Priority 00 Reserved (default to highest priority) 01 Reserved (default to highest priority) 10 Reserved (default to highest priority) 11 Highest access priority

Table 2118. RWCS field descriptions (continued)

Bits	Name	Description
RWCS[21:18]	ATTR	Access Attributes 0xxx Reserved 1xxx Reserved x0xx **p_d_hprot[4]** driven to 0 for accesses x1xx **p_d_hprot[4]** driven to 1 for accesses xx0x **p_d_hprot[3]** driven to 0 for accesses xx1x **p_d_hprot[3]** driven to 1 for accesses xxx0 **p_d_hprot[2]** driven to 0 for accesses xxx1 **p_d_hprot[2]** driven to 1 for accesses
RWCS[17:16]	—	Reserved for future functionality
RWCS[15:2]	CNT	Access Control Count hhhh Number of accesses of word size SZ
RWCS[1]	ERR (2)	Read/Write Access Error (Refer to *Table 2119*)
RWCS[0]	DV(2)	Read/Write Access Data Valid (Refer to *Table 2119*)

1. The priority functionality is not currently implemented.
2. ERR and DV are read-only.

Table 2119. Read/Write Access Status Bit Encoding

Read Action	Write Action	ERR	DV
Read Access has not completed	Write Access completed without error	0	0
Read Access error has occurred	Write Access error has occurred	1	0
Read Access completed without error	Write Access has not completed	0	1
Not Allowed	Not allowed	1	1

84.4.14 Read/Write Access Data (RWD) register

The Read/Write Access Data (RWD) register provides the data to/from system bus memory-mapped locations when initiating a read or a write access.

Read/Write Data

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

Nexus Reg# 0xA; Read/Write; Reset: 0x0

Figure 2226. Read/Write Access Data (RWD) register

Read/Write accesses to the AHB require that the debug firmware properly retrieve/place the data in the RWD. *Table 2120* shows the proper placement of data into the RWD. Note that doubleword transfers require two passes through RWD.

Table 2120. RWD data placement for transfers

Transfer Size and byte offset	RWA(2:0)	RWCS[SZ]	RWD			
			31:24	23:16	15:8	7:0
Byte	*x x x*	0 0 0	—	—	—	X
Half	*x x* 0	0 0 1	—	—	X	X
Word	*x* 0 0	0 1 0	X	X	X	X
Doubleword first RWD pass (low order data)	0 0 0	0 1 1	X	X	X	X
second RWD pass (high order data)			X	X	X	X

Table Notes:
"X" indicates byte lanes with valid data
"—" indicates byte lanes that will contain unused data.

Table 2121 shows the mapping of RWD bytes to byte lanes of the AHB read and write data buses.

Table 2121. RWD byte lane mapping

Transfer Size and byte offset	RWA(2:0)	RWD			
		31:24	23:16	15:8	7:0
Byte @000	0 0 0	—	—	—	AHB[7:0]
Byte @001	0 0 1	—	—	—	AHB[15:8]
Byte @010	0 1 0	—	—	—	AHB[23:16]
Byte @011	0 1 1	—	—	—	AHB[31:24]
Byte @100	1 0 0	—	—	—	AHB[39:32]
Byte @101	1 0 1	—	—	—	AHB[47:40]
Byte @110	1 1 0	—	—	—	AHB[55:48]
Byte @111	1 1 1	—	—	—	AHB[63:56]
Half @000	0 0 0	—	—	AHB[15:8]	AHB[7:0]
Half @010	0 1 0	—	—	AHB[31:24]	AHB[23:16]
Half @100	1 0 0	—	—	AHB[47:40]	AHB[39:32]
Half @110	1 1 0	—	—	AHB[63:56]	AHB[55:48]
Word @000	0 0 0	AHB[31:24]	AHB[23:16]	AHB[15:8]	AHB[7:0]
Word @100	1 0 0	AHB[63:56]	AHB[55:48]	AHB[47:40]	AHB[39:32]

Table 2121. RWD byte lane mapping (continued)

Transfer Size and byte offset	RWA(2:0)	RWD			
		31:24	23:16	15:8	7:0
Doubleword @000 first RWD pass second RWD pass	0 0 0	AHB[31:24] AHB[63:56]	AHB[23:16] AHB[55:48]	AHB[15:8] AHB[47:40]	AHB[7:0] AHB[39:32]

Table Notes:
"—" indicates byte lanes that will contain unused data.

84.4.15 Read/Write Access Address (RWA)

The Read/Write Access Address Register provides the system bus address to be accessed when initiating a read or a write access.

Read/Write Address

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

Nexus Reg# 0x9; Read/Write; Reset: 0x0

Figure 2227. Read/Write Access Address (RWA) register

84.5 Nexus 3 Register Access via JTAG/OnCE

Access to Nexus 3 register resources is enabled by loading a single instruction ("*NEXUS3-ACCESS*") into the JTAG Instruction Register (IR) (OnCE OCMD register). For the Nexus 3 block, the OCMD value is 0b0001111100.

Once the NEXUS3-ACCESS instruction has been loaded, the JTAG/OnCE port allows tool/target communications with all Nexus 3 registers according to the register map in *Table 2101*.

Reading/writing of a Nexus 3 register then requires two (2) passes through the Data-Scan (DR) path of the JTAG state machine. (Refer to *Section 84.23: IEEE 1149.1 (JTAG) RD/WR sequences*.)

1. The first pass through the DR selects the Nexus 3 register to be accessed by providing an index (Refer to *Table 2101*), and the direction (read/write). This is achieved by loading an 8-bit value into the JTAG Data Register (DR). This register has the following format:

Figure 2228. JTAG DR format

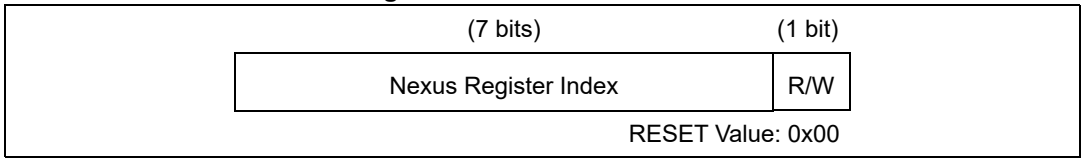


Table 2122. JTAG DR field descriptions

Nexus Register Index	Selected from values in *Table 2101*
Read/Write (R/W)	0 Read 1 Write

2. The second pass through the DR then shifts the data in or out of the JTAG port, LSB first.
   a) During a read access, data is latched from the selected Nexus register when the JTAG state machine passes through the “Capture-DR” state.
   b) During a write access, data is latched into the selected Nexus register when the JTAG state machine passes through the “Update-DR” state.

84.6 Nexus 3 Register Access via Software

Access to most Nexus 3 register resources by software is supported by mapping the Nexus 3 registers to privileged DCRs (device control registers) that are accessible via the **mfdcr** and **mtdcr** instructions. It is intended that these access only occur when no possible conflicts with hardware-based accesses are possible. A conflict between hardware and software accesses will produce undefined results. *Table 2101* shows the mappings of Nexus 3 registers to DCR numbers. Software writes to Nexus 3 register resources are blocked when the **nex_sfwcntl_en** input signal is negated, without signaling an exception. Note that the Nexus 3 Read/Write Access functionality is not available to software, since software can use load and store instructions to access memory.

84.7 Nexus Message Fields

Nexus messages are comprised of fields. Each field contains a distinct piece of information within a message, and each message contains multiple fields. Messages are transferred in packets over the Auxiliary Output protocol. A packet is a collection of fields. A packet may contain any number of fixed length fields, but may contain at most one variable length field. The variable length field must be the last field in a packet. The following sub-sections describe a subset of the message field types.

84.7.1 TCODE Field

The TCODE field is a 6-bit fixed length field that identifies the type of message and its format. The field encodings are assigned by IEEE-ISTO 5001.

84.7.2 Source ID Field (SRC)

Each Nexus module in a device is identified by a unique Client Source Identification Number. The number assigned to each Nexus module is determined by the SoC integrator, and is provided on the **nex3_ext_src_id[0:3]** input signals. Multi-threaded processors may assign additional source ID information to indicate which thread a message is associated with. The Nexus 3 module implements a 4-bit fixed length Source ID field consisting of a Client Source ID.

84.7.3 Relative Address Field (U-ADDR)

The non-sync forms of the Program and Data Trace messages include addresses that are relative to the address that was transmitted in the previous Program or Data Trace message, respectively. The relative address format is compliant with IEEE-ISTO 5001 and is designed to reduce the number of bits transmitted for address fields.

The relative address is generated by XORing the new address with the previous, and then using only the results up to the most significant '1'. To recreate the original address, the relative address is XORed with the previously decoded address.

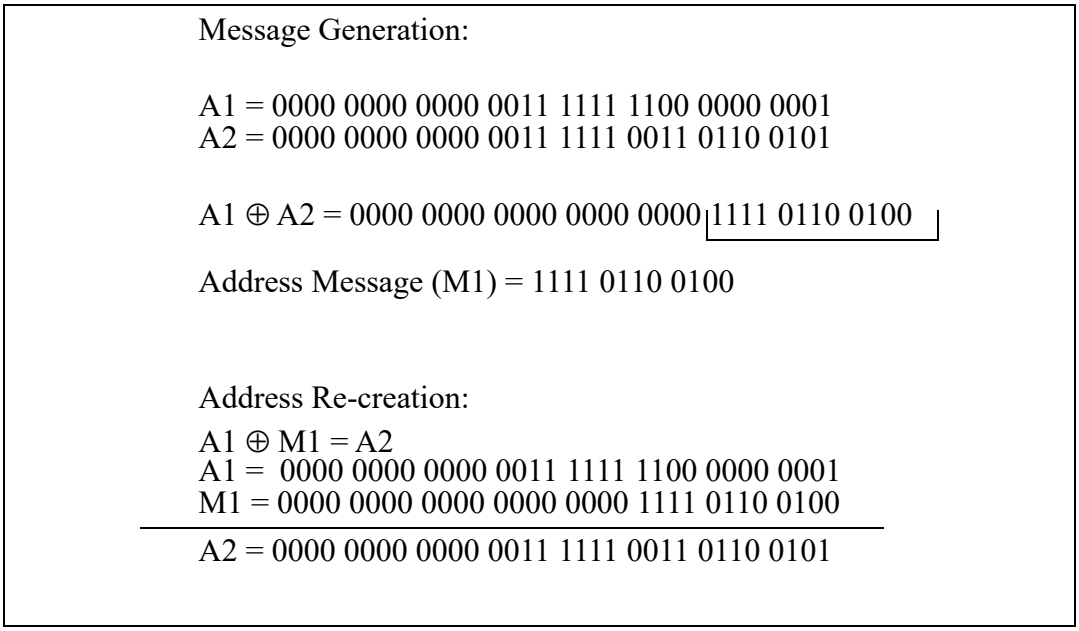
The relative address of a Program Trace message is calculated with respect to the previous Program Trace message, regardless of any address information that may have been sent in any other trace messages in the interim between the two Program Trace messages.

The relative address of a Data Trace message is calculated with respect to the previous Data Trace message, regardless of any address information that may have been sent in any other trace messages in the interim between the two Data Trace messages.

Program trace messages also provide the execution mode status in the least significant bit of the **reconstructed** address field. A value of '0' indicates that preceding instruction count and history information should be interpreted in a non-VLE context. A value of '1' indicates that the preceding instruction count and history information should be interpreted in a VLE context.

Previous Address (A1) =0x0003FC01, New Address (A2) = 0x0003F365

Figure 2229. Relative address generation and recreation



84.7.4 Full Address Field (F-ADDR)

Program Trace synchronization messages provide the full address associated with the trace event (leading zeroes may be truncated) with the intent of providing a reference point for development tools to operate from when reconstructing relative addresses. Synchronization

messages are generated at significant mode switches and are also generated periodically to ensure that development tools are guaranteed to have a reference address given a sufficiently large sample of trace messages. Program trace messages also provide the execution mode status in the least significant bit of the **reconstructed** address field. A value of '0' indicates that preceding instruction count and history information should be interpreted in a non-VLE context. A value of '1' indicates that the preceding instruction count and history information should be interpreted in a VLE context.

84.7.5 Timestamp Field (TSTAMP)

Timestamps may be optionally applied to messages. Enabling of timestamping is controlled by the **nex_tsen** input signal. The timestamping mode (timestamp frequency of occurrence) is controlled by the value of **nex_tsmode[0:1]** and the timestamp value is based on the **nex_timebase[0:29]** inputs. These signals are first synchronized to the **nex_clk** clock domain, and for **nex_timebase[0:29]**, a gray-code to binary value conversion is performed. The timestamp value is applied to messages as they enter the message queues. Leading zeros in the timestamp value are removed. *Table 2123* shows the **nex_tsmode[0:1]** encodings used for the timestamp mode.

Table 2123. nex_tsmode[0:1] Timestamp mode encoding

nex_tsmode[0:1]	Timestamp mode
00	Timestamp applied to every message
01	Timestamp applied to every 4th message
10	Timestamp applied to every 16th message
11	Timestamp applied to every 64th message

84.8 Nexus Message Queues

The Nexus 3 module implements internal message queues capable of storing up to three messages per cycle into a small initial queue, which then fills a larger queue at up to two messages per cycle. Messages that enter the queues are transmitted in the order in which they are received.

If more than three messages attempt to enter the queue in the same cycle, the highest priority messages are stored and the remaining message(s) will be dropped due to a collision. Collision events are expected to be rare.

The Overrun Control register (OVCR) controls the Nexus behavior as the message queue fills. The Nexus block may be programmed to:

- Allow the queue to overflow, drain the contents, queue an overrun error message and resume tracing.
- Stall the processor when the queue utilization reaches the selected threshold.
- Suppress selected message types when the queue utilization reaches the selected threshold.

84.8.1 Message Queue Overrun

In this mode, the message queue will stop accepting messages when an overrun condition is detected. The contents of the queues will be allowed to drain until empty. Incoming

messages are discarded until the queue is emptied. Once empty, an overrun error message is enqueued that contains information about the types of messages that were discarded due to the overrun condition.

84.8.2 CPU Stall

In this mode, processor instruction issue is stalled when the queue utilization reaches the selected threshold. The processor is stalled long enough to drop one threshold level below the level that triggered the stall. For example, if stalling the processor is triggered at 1/4 full, the stall will stay in effect until the queue utilization drops to empty. There may be significant skid from the time that the stall request is made until the processor is able to stop completing instructions. This skid should be taken into consideration when programming the threshold. Refer to *Section 84.4.9: Nexus Overrun Control (OVCR) register*, for complete programming options.

84.8.3 Message Suppression

In this mode, the message queue will disable selected message types when the queue utilization reaches the selected threshold. This allows lower bandwidth tracing to continue and possibly avoid an overrun condition. If an overrun condition occurs despite this message suppression, the queue will respond according to the behavior described in *Section 84.8.1: Message Queue Overrun*. Suppressed message types are reported in the error message if they occur after overrun in addition to other discarded message types. Once triggered, message suppression will remain in effect until queue utilization drops to the threshold below the level selected to trigger suppression.

84.8.4 Nexus Message Priority

Nexus messages may be lost due to contention with other message types under the following circumstance:

- More than three messages are generated in the same cycle

Table 2124 lists the various message types and their relative priority from highest to lowest.

Up to three message requests can be queued into the message buffer in a given cycle. If more than three message requests exist in a given cycle, the three highest priority message classes are queued into the message buffer. The remaining messages that did not successfully queue into the message buffer in that cycle will generate subsequent responses, as detailed in *Table 2124*.

The CPU is capable of completing two instructions per cycle. If multiple trace messages need to be queued at the same time, they will be queued with the following priority: Instruction 0 (oldest instruction) (WPM $\rightarrow$ DQM $\rightarrow$ PCM_{PIDMSG} $\rightarrow$ OTM $\rightarrow$ BTM $\rightarrow$ DTM) $\rightarrow$ Instruction1 (newer instruction) (WPM $\rightarrow$ DQM $\rightarrow$ OTM $\rightarrow$ BTM $\rightarrow$ DTM). As many as three messages may be simultaneously queued. Note that for the cycle following a dropped PTM, non-periodic OTM, or DQM message, only two other messages may be queued in addition to the dropped Error Message.

Watchpoint Messages from instructions that complete at the same time or events that occur during the same cycle will be combined.

Table 2124. Message Type Priority and Message Dropped Responses

Message Type	Message	Priority	Message Dropped Response
Error	Error	0 (highest)	N/A(1)
WP (Watchpoint Trace)	WPM (Watchpoint Message)	1	N/A(1)
DQ (Data Acquisition)	DQM (Data Acquisition Message)	2	DQM Error Message
Program Trace (PID MSG)	PCM — PID/NPIDR update (Program Correlation Message)	2	OTM Error Message
OT (Ownership)	OTM — PID/NPIDR update (Ownership Trace Message)	2	OTM Error Message(2)
Program Trace	BTM (Branch Trace Message)	2	BTM Error Message, Sync upgrade next BTM
	Sync (Program Sync Message)	3	Sync upgrade next BTM
	RFM (Resource Full for Instruction counter or history buffer)	4	BTM Error Message Sync upgrade next BTM
	DS (Debug Status Message)	5	Sync upgrade next BTM
	PCM (Program Correlation Message)	6	BTM Error Message Sync upgrade next BTM
DT (Data Trace)	DTM (Data Trace Message)	7	Sync upgrade next DTM
OT (Ownership)	OTM — Periodic update (Ownership Trace Message)	8 (lowest)	none

1. Error and Watchpoint messages are not dropped due to collisions, due to their priority.
2. Message will always be dropped if program trace is enabled, and program correlation messages for PID messages are not masked (Event Code = 0101). No error message is sent for this case since the PID value is contained in the higher priority message.

84.8.5 Data Acquisition Message Priority Loss Response

If a Data Acquisition Message (DQM) loses arbitration due to contention with higher priority messages, an error message will be generated to indicate that a DQM has been lost due to contention.

84.8.6 Ownership Trace Message Priority Loss Response

If an Ownership Trace message (OTM) due to software updates to the Process ID state loses arbitration due to contention with higher priority messages other than a program correlation message with EVCODE = 0101 (PID update), an error message will be generated to indicate that a OTM has been lost due to contention. If the pending OTM is a periodic update, the event is dropped without generating an error message.

84.8.7 Program Trace Message Priority Loss Response

If a Program Trace message (PTM) loses arbitration due to contention with higher priority messages, and the discarded PTM is a Program Correlation message, a Resource Full message for instruction count or history buffer, or a Branch Trace message, then an Error message is generated to indicate that branch trace information has been lost, and the next Branch Trace message will be upgraded to a sync-type message.

If the discarded PTM is a Program Correlation message with PID information (EVCODE=0101), the Error message will indicate a dropped OTM and a dropped Program Trace (Error code = xxxx11xx).

84.8.8 Program Trace Sync Message Priority Loss Response

If a Program Trace Sync message (SYNC) loses arbitration due to contention with higher priority messages, the Sync event is discarded, and the next Branch Trace message will be upgraded to a sync-type message.

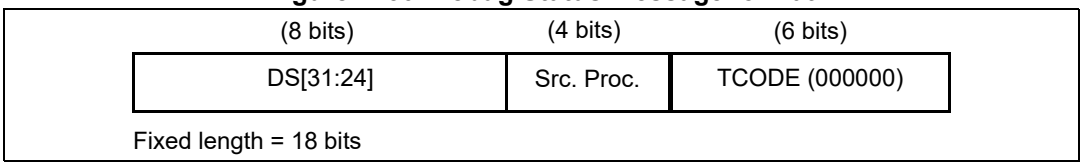
84.8.9 Data Trace Message Priority Loss Response

If a Data Trace message (DTM) loses arbitration due to contention with higher priority messages, the DTM event is discarded, and the next DTM will be upgraded to a sync-type message.

84.9 Debug Status messages

Debug Status Messages report low power mode and debug status. Debug Status messages are enabled when Nexus 3 is enabled. Entering/exiting Debug Mode as well as entering, exiting, or changing Low Power Mode(s) will trigger a Debug Status message, indicating the value of the most significant byte in the Development Status register. Debug status information is sent out in the following format:

Figure 2230. Debug Status message format



84.10 Error messages

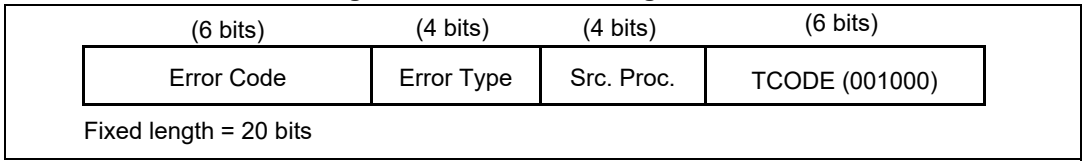
Error messages are enabled whenever the debug logic is enabled. There are two conditions that will produce an error message, each receiving a separate error type designation:

- A message is discarded due to contention with other (higher priority) message types. These errors will have an Error Type value of 1.
- The message queue overruns. After the queue is drained, an error message is enqueued with an error code that indicates what types of messages were discarded during the interim. These errors will have an Error Type value of 0.

Note: The OVCR Register can be used in order to alleviate potential overrun situations.

Error information is messaged out in the following format (also, refer to *Table 2096* and *Table 2097*):

Figure 2231. Error message format



84.11 Ownership trace

This section details the ownership trace features of the Nexus 3 module.

84.11.1 Overview

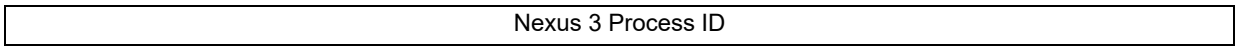
Ownership trace provides a macroscopic view, such as task flow reconstruction, when debugging software written in a high level (or object-oriented) language. It offers the highest level of abstraction for tracking operating system software execution. This is especially useful when the developer is not interested in debugging at lower levels.

84.11.2 Ownership Trace Messaging (OTM)

Ownership trace information is messaged via the auxiliary port using an Ownership Trace Message (OTM). Core (e200z425n3) processors contain a <asisitalic>PowerISA 2.06 defined “Process ID” register within the CPU. It is updated by the operating system software to provide task/process ID information. The contents of this register are replicated on the pins of the processor and connected to Nexus. The Process ID (PID) register value can be accessed using the **mfspr**/**mtspr** instructions.

Note: The CPU includes a Process ID register (PID0), thus the Nexus UBA functionality is not implemented.

In addition, to decouple trace information from the PID0 register and to provide a full independent 32-bit process ID for debug use, the Nexus PID Register (NPIDR) may be used instead of PID0 to provide OTM process ID information. It is updated by the operating system software to provide task/process ID information. The Nexus Process ID (NPIDR) register value can be accessed using the **mfspr**/**mtspr** instructions. This register is accessible in user or supervisor mode.



SPR — 517; Read/Write; Reset: 0

Figure 2232. Nexus 3 Process ID Register (NPIDR)

Note: OS updates to NPIDR should be performed in addition to normal PID0 updates when a process switch occurs, in order to properly generate OTM messages with new process ID information when NPIDR is selected for OTM use.

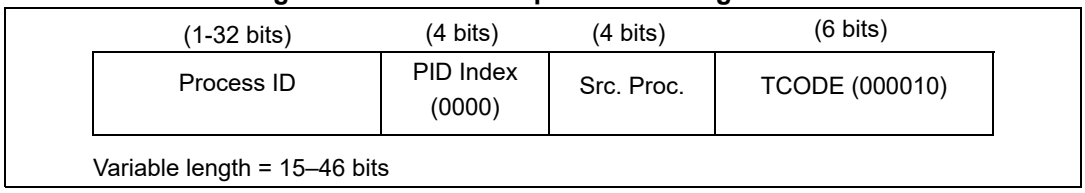
The process ID source (PID0 or NPIDR) is selected by the setting of the $DC1_{OTS}$ control bit.

There are two conditions that will cause an Ownership Trace Message when Ownership Trace is enabled:

- When the PID0 register is written to by the processor and $DC1_{OTS}$ indicates PID0 should be used, or when the NPIDR register is written to by the processor and $DC1_{OTS}$ indicates NPIDR should be used, the data is latched within Nexus, and is messaged out via the auxiliary port, allowing development tools to trace ownership flow. However, if Program Trace is enabled, and program correlation messages for PID updates are not masked (Event Code = 0101), then an OTM will not be generated for an update to the selected process ID register, since the program correlation message will provide this process ID update information.
- Periodically, at least once every 256 messages, the most recent state of the selected process ID register is messaged out. The resulting Ownership Trace message will indicate in the PID Index sub-field that PID0/NPIDR status is being reported and the most recent value of the PID0/NPIDR register will be conveyed in the Process ID value sub-field. These periodic Ownership Trace message events can be disabled by setting $DC1_{POTD}$.

Ownership trace information is messaged out in the following format:

Figure 2233. Ownership Trace Message format



84.12 Program Trace

This section details the program trace mechanism supported by Nexus3 for the e200z4256 processor. Program trace is implemented via Branch Trace Messaging (BTM) as per the **IEEE-ISTO 5001** standard definition. Branch Trace Messaging for Core (e200z425n3) processors is accomplished by snooping the internal address bus (between the CPU and Cache), attribute signals, and CPU Status (**p_mode[0:3]**, **p_pstat_pipe{0,1}[0:5]**).

84.12.1 Branch Trace Messaging types

Traditional Branch Trace Messaging facilitates program trace by providing the following types of information:

- Messaging for taken direct branches includes how many sequential instructions were executed since the last taken branch or exception, including the taken direct branch. Branch instructions are included in the count of sequential instructions.
- Messaging for taken indirect branches and exceptions includes how many sequential instructions were executed since the last taken branch or exception and the unique portion of the branch target address or exception vector address. Branch instructions are included in the count of sequential instructions. For taken indirect branches that trigger generation of a message, the branch is also included in the count.

Branch History Messaging facilitates program trace by providing the following information.

- Messaging for taken indirect branches and exceptions includes a) how many sequential instructions (I-CNT) were executed since the last predicate instruction,

taken/not taken direct branch, taken/not-taken indirect branch, or exception, b) the unique portion of the branch target address or exception vector address, and c) a branch/predicate instruction history field. Each bit in the history field represents a direct branch or predicated instruction where a value of one (1) indicates taken, and a value of zero (0) indicates not taken. (**evsel**)Not-taken indirect branches will generate a history bit with a value of zero (0). Instructions that generate history bits are not included in instruction counts. For taken indirect branches that trigger generation of this message type, the branch is included in the count, but not in the history field.

84.12.1.1 Indirect Branch Message instructions

Table 2125 shows the types of instructions and events that cause Indirect Branch Messages or Branch History Messages to be encoded.

Table 2125. Indirect Branch Message sources

Source of Indirect Branch Message	Instructions / Detail
Taken branch relative to a register value	**bcctr**, **bcctrl**, **bclr**, **bclrl, se_bctr, se_bctrl, se_blr, se_blrl**
System Call / Trap exceptions taken	se_sc, tw
Return from interrupts / exceptions	se_rfi, se_rfci, se_rfdi
Exit from reset with Program Trace Enabled	Indirect branch with Sync, target address is initial instruction, count = 1

84.12.1.2 Direct Branch Message Instructions

Table 2126 shows the types of instructions that will cause Direct Branch Messages or will toggle a bit in the instruction history buffer to be messaged out in a Resource Full Message or Branch History Message.

Table 2126. Direct Branch Message sources

Source of Direct Branch Message	Instructions
Taken direct branch instructions Instruction Synchronize	**b**, **ba**, **bl**, **bla**, **bc**, **bca**, **bcl**, **bcla, se_b. se_bc, se_bl, e_b, e_bc, e_bl, e_bcl, se_isync**

84.12.1.3 BTM using Branch History Messages

Traditional BTM Messaging can accurately track the number of sequential instructions between branches, but cannot accurately indicate which instructions were conditionally executed, and which were not.

Branch History Messaging solves this problem by providing a predicated instruction history field in each Indirect Branch Message. Each bit in the history represents a predicated instruction or direct branch, or a not-taken indirect branch. A value of one (1) indicates the conditional instruction was executed or the direct branch was taken. A value of zero (0) indicates the conditional instruction was not executed or the branch was not taken. (**evsel**)

Branch History Messages solve predicated instruction tracking and save bandwidth since only indirect branches cause messages to be queued.

84.12.1.4 BTM using Traditional Program Trace Messages

Based on the PTM bit in the DC1 Register, Program Tracing can utilize either Branch History Messages (PTM = 1) or traditional Direct/Indirect Branch Messages (PTM = 0).

Branch History will save bandwidth and keep consistency between methods of Program Trace, yet may lose temporal order between BTM messages and other types of messages. Since direct branches are not messaged, but are instead included in the history field of the Indirect Branch History Message, other types of messages may enter the FIFO between Branch History Messages. The development tool cannot determine the ordering of “events” that occurred with respect to direct branches simply by the order in which messages are sent out.

Traditional BTM messages maintain their temporal ordering because each event that can cause a message to be queued will enter the FIFO in the order it occurred and will be messaged out maintaining that order.

84.12.2 BTM Message For mats

The Nexus 3 block supports three types of traditional BTM Messages — Direct, Indirect, and Synchronization Messages. It supports two types of branch history BTM Messages — Indirect Branch History, and Indirect Branch History with Synchronization Messages.

84.12.2.1 Indirect branch messages (history)

Indirect branches include all taken branches whose destination is determined at run time, interrupts, and exceptions. If $DC1_{PTM}$ is set to ‘1’, indirect branch information is messaged out in the following format:

Figure 2234. Indirect branch message (history) format

(1-32 bits)	(1–32 bits)	(1–8 bits)	(1 bit)	(4 bits)	(6 bits)
Branch History	Relative Address	Sequence Count	(0)	Source Proc.	TCODE (011100)

Max length = 83 bits; Min length = 14 bits

84.12.2.2 Indirect branch messages (traditional)

If $DC1_{PTM}$ is cleared to ‘0’, indirect branch information is messaged out in the following format:

Figure 2235. Indirect branch message format

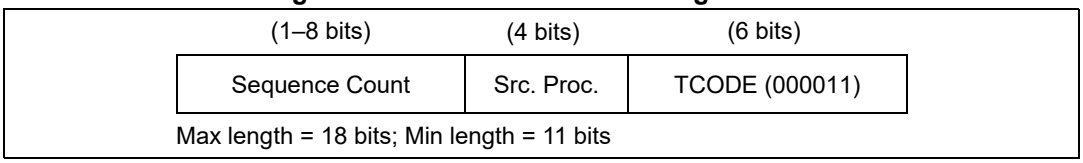
(1–32 bits)	(1–8 bits)	(1 bit)	(4 bits)	(6 bits)
Relative Address	Sequence Count	(0)	Source Proc.	TCODE (000100)

Max length = 51 bits; Min length = 13 bits

84.12.2.3 Direct branch messages (traditional)

Direct branches (conditional or unconditional) are all taken branches whose destination is fixed in the instruction opcode. Direct branch information is messaged out in the following format:

Figure 2236. Direct branch message format



Note: When $DC1_{PTM}$ is set, Direct Branch Messages will not be transmitted. Instead, each direct branch, not-taken indirect branch, or predicated instruction will be recorded in the history buffer.

84.12.3 Program Trace Message Fields

The following subsections describe specific fields used for Program Trace messages.

84.12.3.1 Sequential Instruction Count Field (ICNT)

Most of the program trace messages include an instruction count field. For traditional Branch messages, ICNT represents the number of sequential instructions including non-taken branches since the last Direct/Indirect Branch messages. Branch instructions that trigger message generation are included in the ICNT.

For Branch History messages, ICNT represents the number of instructions executed since the last taken/non-taken direct branch, predicate instruction, last taken/not-taken indirect branch, or exception. Branch instructions that trigger message generation are included in the ICNT. Instructions that generate history bits are not included in the ICNT.

The sequential instruction counter overflows after its value reaches 255 and is reset to 0. In addition, the next BTM Message (corresponding to the 256th or later instruction) will be converted to a with sync type message.

The instruction counter is reset every time the instruction count is transmitted in a message or whenever there is a branch/predicate history event, as well as on exiting from debug mode.

84.12.3.2 Branch/Predicate Instruction History (HIST)

If $DC1_{PTM}$ is set, BTM messaging will use the Branch History format. The branch history (HIST) field in these messages provides a history of branch execution used for reconstructing the program flow. The branch/predicate history buffer stores information about branch and predicate instruction execution. The buffer is implemented as a left-shifting register. The buffer is preloaded with a one (1), which acts as a stop bit (the most significant 1 in the history field is a termination bit for the field). The pre-loaded bit itself is not part of the history, but is transmitted with the packet.

A value of one (1) is shifted into the history buffer for each taken direct branch (program counter relative branch) or predicate instruction whose condition evaluates to true. A value of zero (0) is shifted into the history buffer for each not-taken branch (including indirect branch instructions) or predicate instruction whose condition evaluates to false. For the **evsel** instruction, two bits are shifted in, corresponding to the low element (shifted in first) and the high element (shifted in second) conditions.

This history buffer information is transmitted as part of an Indirect Branch with History message, as part of a Program Correlation message, or as part of a Resource Full message if the history buffer becomes full. The history buffer is reset every time the history information is transmitted in a message, as well as on exiting from debug mode.

Table 2127. Branch/Predicate history events

Branch/Predicate History Event	History Bit(s)	Relevant Instructions
Not taken register indirect branches	0	**bcctr, bcctrl, bclr, bclrl**
Not taken direct branches	0	**b, ba, bc, bca, bla, bcla, bl, bcl**
Taken direct branches	1	**b, ba, bc, bca, bla, bcla, bl, bcl**(1)
evsel instruction	00,01,10, or 11	evsel

1. If the EVCODE for direct branch function calls is not masked in DC4, taken bl and bcl instructions will generate Program Correlation Messages and will not be logged in the history buffer.

84.12.3.3 Execution Mode Indication

In order for a development tool to properly interpret instruction count and history information, it must be aware of the execution mode context of that information. VLE instructions will be interpreted differently from non-VLE instructions.

Program trace messages provide the execution mode status in the least significant bit of the **reconstructed** address field. A value of ‘0’ indicates that preceding instruction count and history information should be interpreted in a non-VLE context. A value of ‘1’ indicates that the preceding instruction count and history information should be interpreted in a VLE context.

84.12.4 Resource Full Messages

The Resource Full Message is used in conjunction with Branch Trace and Branch History Messages. The Resource Full Message is generated when either the internal branch/predicate history buffer is full, or if the BTM Instruction sequence counter (I-CNT) overflows. If synchronization is needed at the time this message is generated, the synchronization is delayed until the next Branch Trace Message that is not a Resource Full Message.

For history buffer overflow, the Resource Full Message transmits a Resource Code (RCODE) of 0b0001 and the current contents of the history buffer, including the stop bit, are transmitted in the Resource Data (RDATA) field. This history information can be concatenated by the development tool with the branch/predicate history information from subsequent messages to obtain the complete branch/predicate history between indirect changes of flow.

For instruction counter overflow, the Resource Full Message transmits an RCODE of 0b0000 and a value of 0xFF is transmitted in the RDATA field, indicating that 255 sequential instructions have been executed since the last change of flow; or, if program trace is in history mode, since the last instruction that recorded history information.

Figure 2237. Resource Full Message format

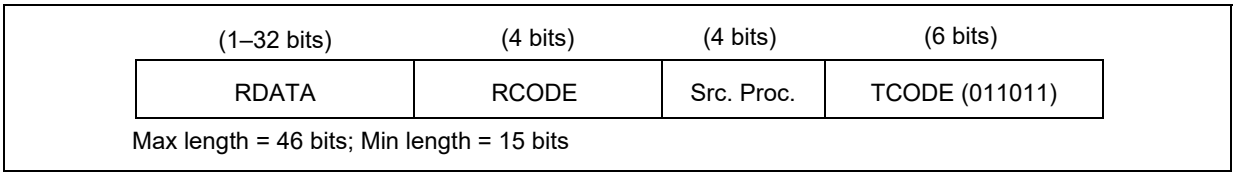


Table 2128 shows the RCODE encodings and RDATA information used for Resource Full Messages.

Table 2128. RCODE encoding

RCODE	Description	RDATA field
0000	Program Trace Instruction counter reached 255 and was reset.	0xFF
0001	Program Trace, Branch / Predicate Instruction History full.	Branch HIstory. This type of packet is terminated by a stop bit set to 1 after the last history bit.

84.12.5 Program Correlation Messages

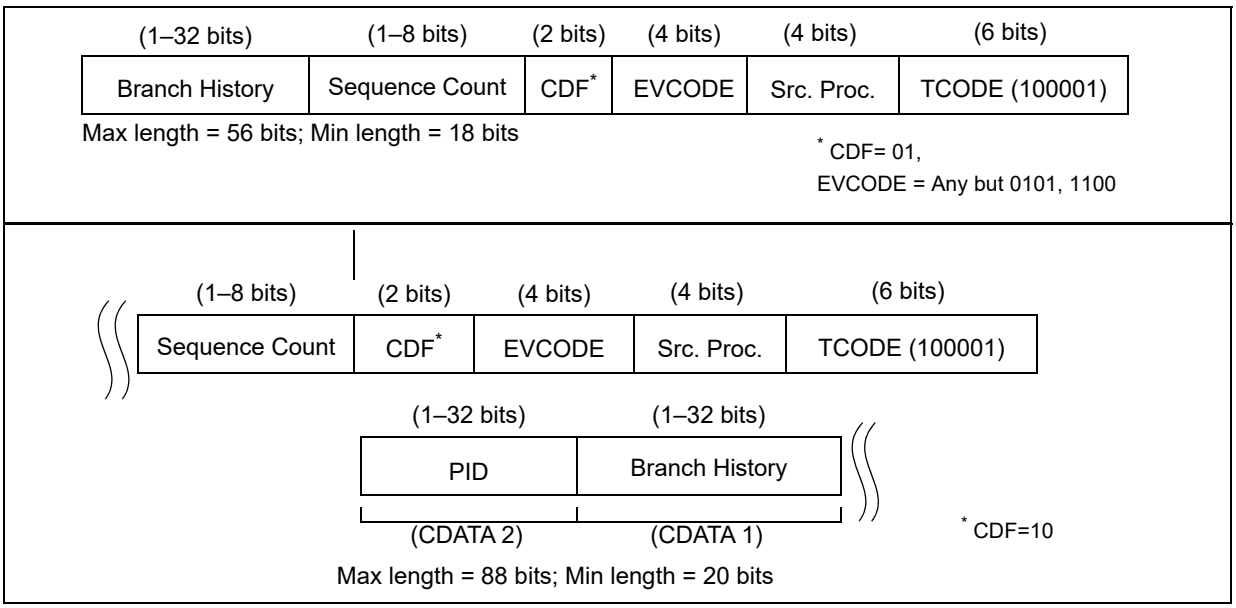
Program Correlation Messages (PCMs) are used to correlate events to the program flow that may or may not be associated with the instruction stream. The following events will result in a PCM when program trace is enabled:

- When the CPU enters debug mode, a PCM is generated. The instruction count and history information provided by the PCM can be used to determine the last sequence of instructions executed prior to debug mode entry.
- When the CPU first enters a low power mode in which instructions are no longer executed, a PCM is generated. The instruction count and history information provided by the PCM can be used to determine the last sequence of instructions executed prior to low power mode entry.
- Whenever program trace is disabled by any means, a PCM is generated. The instruction count and history information provided by the PCM can be used to determine the last sequence of instructions executed prior to disabling program trace.
- When a “Branch and Link” instruction executes (direct branch function call — **bl**/**bcl**/**bla**/**bcla**-type instructions).
- When program trace becomes masked due to MSR_{PMM}=‘0’ and $DC4_{PTMARK}$=‘1’.
- When a write to the process ID register selected by $DC1_{OTS}$ (PID0 or NPIDR) is made via a **mtspr** PID0 or **mtspr** NPIDR.

Refer to *Table 2099* for the event codes that are supported in this implementation. Event code masking is available via the EVCDM field of the DC4 register to allow for control over generation of Program Correlation messages for each event type.

Program Correlation is messaged out in the following formats:

Figure 2238. Program Correlation message formats



If timestamping is enabled, the TSTAMP field is appended to the end of the message and adds up to 30 additional bits.

84.12.5.1 Program Correlation message generation for PID updates

When a (potentially) new value is established in the selected process ID register via a **mtspr** PID0/NPIDR, a Program Correlation message is generated containing the information regarding the new process ID value. This PCM also contains the current history and instruction count. The message is provided so that address translation information can be processed by the development tool in the proper program flow. The **mtspr** PID0/NPIDR is included in the instruction count information. Note that Ownership Trace Messages (other than the periodic OTM) are redundant with the information provided, and may be disabled to avoid unnecessary message bandwidth or collisions.

84.12.6 Program Trace Overflow Error messages

An Error Message occurs when a new message cannot be queued due to the message queue being full. The FIFO will discard incoming messages until it has completely emptied the queue. Once emptied, an Error Message will be queued. The error encoding will indicate which type(s) of messages attempted to be queued while the FIFO was being emptied.

84.12.7 Program Trace Synchronization messages

By default, program trace messages will perform XOR compression on the branch target address to produce the address field for the message. This compression is consistent with the specification in IEEE-ISTO 5001.

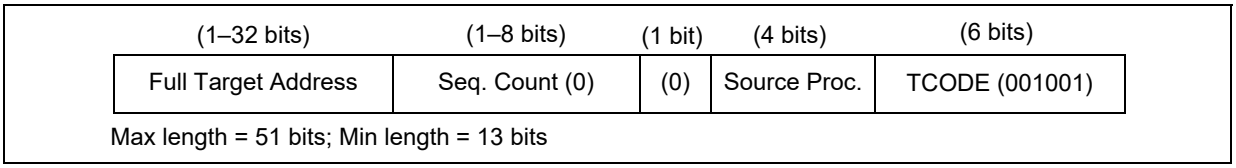
Under some conditions an uncompressed address is sent to provide development tools with a baseline reference address. A Program Trace Synchronization message is messaged via the auxiliary port (provided Program Trace is enabled) for the following conditions (refer to *Table 2129*):

- Initial Program Trace message after exit from system reset or whenever program trace is enabled.
- Upon exiting from a CPU Low Power state.
- Upon exiting from Debug Mode.
- Upon occurrence of queue overrun (can be caused by any trace message), provided Program Trace is enabled.
- Upon assertion of the Event In (**nex_evti_b**) pin if the EIC bits within the DC1 Register have enabled this feature.
- When program trace becomes unmasked due to $MSR_{PMM} \rightarrow$ '1' with $DC4_{PTMARK}$='1'.

Note that the ICNT information for these messages is driven to zero, thus will not always be meaningful for some of these cases.

The format for Program Trace Synchronization messages is as follows:

Figure 2239. .Program Trace Synchronization message format



If timestamping is enabled, the TSTAMP field is appended to the end of the message and adds up to 30 additional bits.

Exception conditions that result in Program Trace Synchronization are summarized in *Table 2129*.

Table 2129. Program Trace exception summary

Exception Condition	Exception Handling
System Reset Negation	At the negation of JTAG reset (**j_trst_b**), queue pointers, counters, state machines, and registers within the Nexus 3 module are reset. Upon exiting system reset (**p_reset_b**), if Program Trace is already enabled, a Program Trace Sync Message is sent.
Program Trace Enabled	The first Program Trace Message (after Program Trace has been enabled or re-enabled) is a synchronization message.
Exit from Low Power/Debug	Upon exit from a Low Power mode or Debug mode, a Program Trace Sync Message is sent.
Queue Overrun	An Error Message occurs when a new message cannot be queued due to the message queue being full. The FIFO will discard messages until it has completely emptied the queue. Once emptied, an Error Message will be queued. The error encoding will indicate which type(s) of messages attempted to be queued while the FIFO was being emptied. The next BTM message in the queue will be a Program Trace Sync Message.
Event In	If the Nexus module is enabled, a **nex_evti_b** assertion initiates a Program Trace Sync Message if Program Trace is enabled and the EIC bits of the DC1 Register have enabled this feature.

Note that for cases where program trace sync messages are generated due to program trace being enabled, the address contained in the sync message may either be the address

of the instruction that caused program trace to be enabled, or may be the address of the first instruction of an exception handler that is executed in response to unsuccessful completion of that instruction.

84.12.8 Program Trace Direct/Indirect Branch with Sync messages

Under some conditions an uncompressed address is sent to provide development tools with a baseline reference address. A Program Trace Synchronization or a Direct/Indirect Branch with Sync message is messaged via the auxiliary port (provided Program Trace is enabled) for the following conditions (refer to *Table 2129*):

- Upon direct/indirect branch after a Program Sync Message was lost due to a collision while attempting to enter the message queue.
- Upon direct/indirect branch after the sequential instruction counter has expired indicating 255 instructions have occurred since the last change of flow.
- When the periodic program trace counter has expired indicating 255 *without-sync* Program Trace messages have occurred since the last *with-sync* message occurred.

Note that the ICNT and History information for the first message will not always be meaningful, since the temporary masking of program trace may result in ambiguous values. Subsequent with Sync messages will not have this issue.

The format for Program Trace Direct/Indirect Branch with Sync messages is as follows:

Figure 2240. Direct/Indirect Branch with Sync message format

(1–32 bits)	(1–8 bits)	(1 bit)	(4 bits)	(6 bits)
Full Target Address	Sequence Count	(0)	Source Proc.	TCODE (001011 or 001100)

Max length = 51 bits; Min length = 13 bits

The format for Program Trace Indirect Branch History with Sync Messages is as follows:

Figure 2241. Indirect Branch History with Sync message format

(1–32 bits)	(1–32 bits)	(1–8 bits)	(1 bit)	(4 bits)	(6 bits)
Branch History	Full Target Address	Sequence Count	(0)	Source Proc.	TCODE (011101)

Max length = 83 bits; Min length = 14 bits

Exception conditions that result in Program Trace Synchronization using a with Sync message type are summarized in *Table 2130*.

Table 2130. Program Trace Exception Summary for with Sync BTM messages

Exception Condition	Exception Handling
Periodic Program Trace Sync.	A forced synchronization occurs periodically after 255 non-sync Program Trace Messages have been queued. A Direct/Indirect Branch with Sync Message is queued. The periodic program trace message counter then resets.
Sequential Instruction Count Overflow	After the sequential instruction counter reaches its maximum count (up to 255 sequential instructions may be executed), a forced synchronization occurs. The sequential counter then resets. A Program Trace Direct/Indirect Branch with Sync Message is queued upon execution of the next branch. A Resource Full Message is Queued on the overflow event. If a branch instruction is the 255th instruction to occur, and causes a Program Trace message to be queued, then no Resource Full Message is queued, and the with Sync message will be queued for the *next* Program Trace Direct/Indirect Branch Message.
Collision Priority	All Messages have the following priority: Instruction 0 (WPM → DQM → PCM_{PIDMSG}→OTM → BTM → DTM)→ Instruction1 (WPM → DQM → OTM → BTM → DTM), where instruction0 is the oldest instruction. A BTM Message from Instruction1 that attempts to enter the queue at the same time as three higher priority messages from either instruction will be lost. An Error Message will be sent indicating the BTM was lost. The following direct/indirect branch will queue a Direct/Indirect Branch with Sync Message. The count value within this message will reflect the number of sequential instructions executed after the last successful BTM Message was generated. This count will include the branch that did not generate a message due to the collision.

84.12.9 Enabling Program Trace

Program Trace Messaging can be enabled in one of below ways:

- Setting the TM field of the DC1 Register to enable Program Trace
- Using the PTS field of the WT Register to enable Program Trace on Watchpoint hits (watchpoints are configured within the CPU)
- Using the external hardware trace enable input signal (nex_ptm_starttr)
- Filtering of Program Trace messages may be performed using the MSR_{PMM} bit and the setting of $DC4_{PTMARK}$

84.12.10 Program Trace Timing Diagrams (2 MDO / 1 MSEO configuration)

Figure 2242. Program Trace — Indirect Branch message (Traditional)

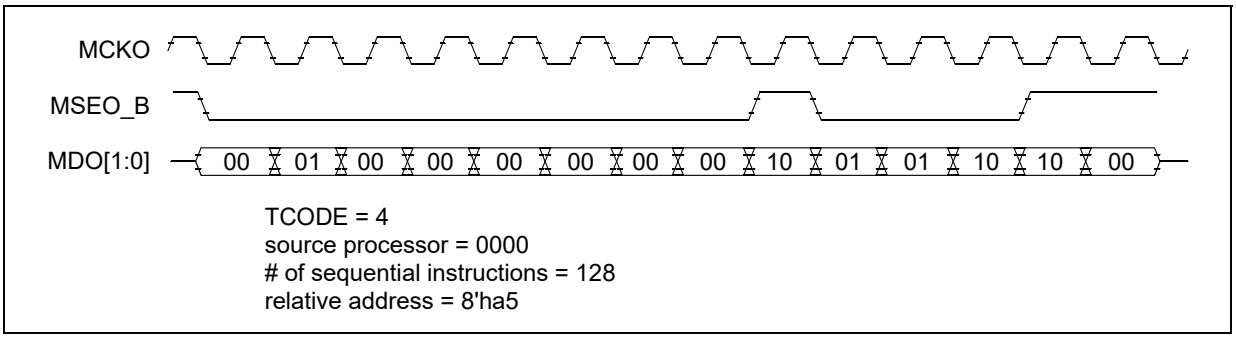


Figure 2243. Program Trace — Indirect Branch message (history)

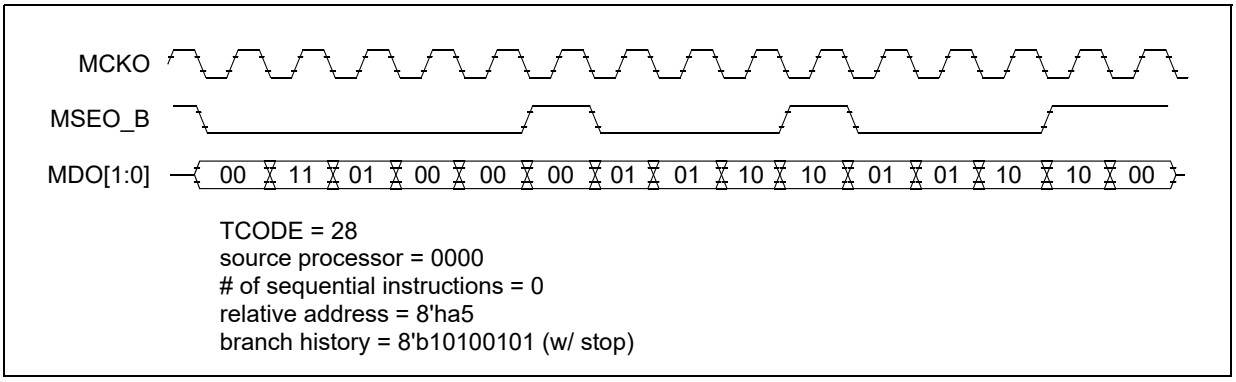


Figure 2244. Program Trace — Direct Branch (traditional) & Error messages

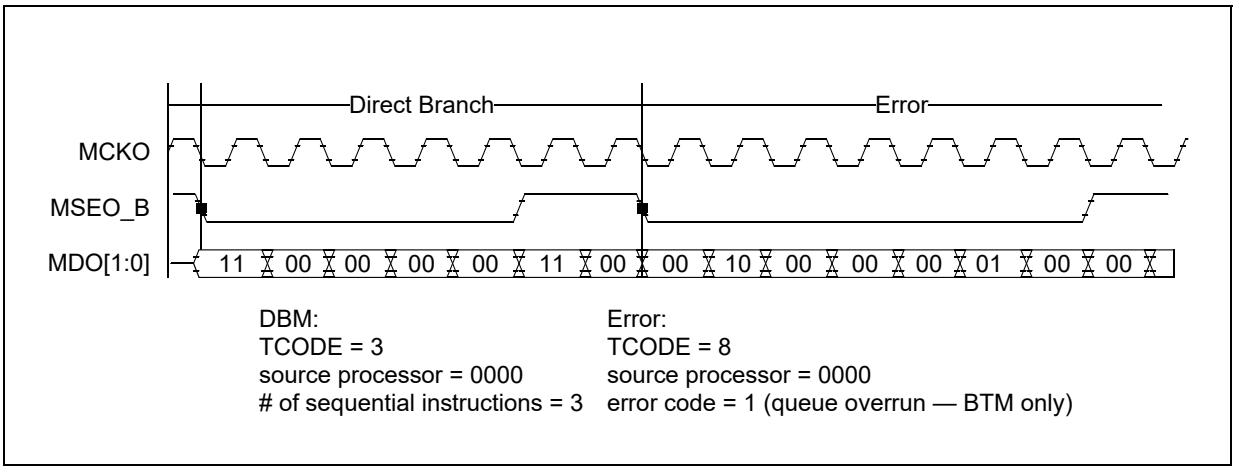
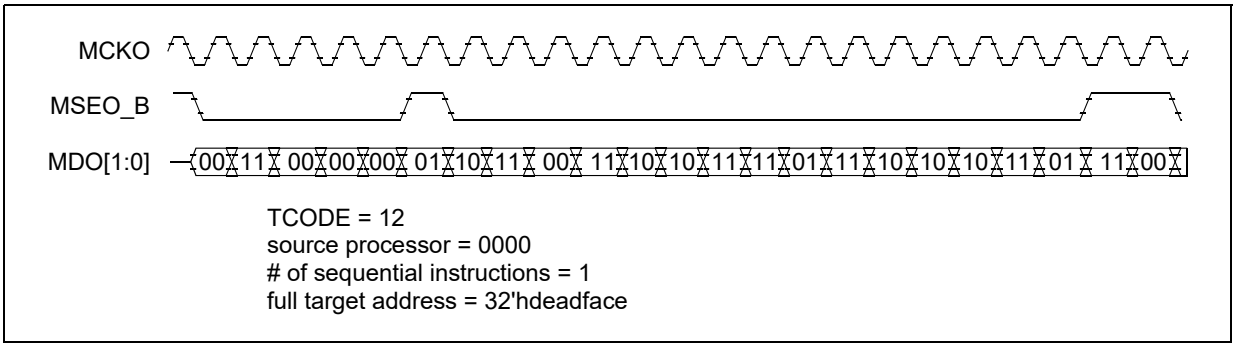


Figure 2245. Program Trace — Indirect Branch with Sync message



84.13 Data Trace

This section deals with the Data Trace mechanism supported by the Nexus 3 module. Data Trace is implemented via Data Write Messaging (DWM) and Data Read Messaging (DRM), as per the **IEEE-ISTO 5001** standard.

84.13.1 Data Trace Messaging (DTM)

Data Trace Messaging is accomplished by snooping the internal address and data buses, and storing the information for qualifying accesses (based on enabled features and matching target addresses). The Nexus 3 module traces all data access that meet the selected range and attributes.

Note: Data Trace is only performed on the internal data buses. This allows for data visibility for Core (e200z425n3) processors that incorporate a data cache. Only CPU initiated accesses will be traced. No DMA accesses to the AHB system bus will be traced.

Data Trace Messaging can be enabled in one of three ways.

- Setting the TM field of the DC1 Register to enable Data Trace.
- Using the DTS field of the WT Register to enable Data Trace on Watchpoint hits (watchpoints are configured within the Nexus1 module).
- Using the external hardware trace enable signal (**nex_dtm_starttr**).

84.13.2 DTM Message formats

The Nexus 3 block supports five types of DTM Messages — Data Write, Data Read, Data Write Synchronization, Data Read Synchronization, and Error Messages.

84.13.2.1 Data Write messages

The Data Write message contains the data write value and the address of the write access, relative to the previous Data Trace message. Data Write message information is messaged out in the following format:

Figure 2246. Data Write message format

(1–64 bits)	(1–32 bits)	(4 bits)	(1 bit)	(4 bits)	(6 bits)
Data Value(s)*	Relative Address	Data Size	(0)	Src. Proc	TCODE (000101)

Max length = 111 bits; Min length = 17 bits

If timestamping is enabled, the TSTAMP field is appended to the end of the message and adds up to 30 additional bits.

84.13.2.2 Data Read messages

The Data Read message contains the data read value and the address of the read access, relative to the previous Data Trace message. Data Read message information is messaged out in the following format:

Figure 2247. Data Read message format

(1–64 bits)	(1–32 bits)	(4 bits)	(1 bit)	(4 bits)	(6 bits)
Data Value(s)*	Relative Address	Data Size	(0)	Src. Proc	TCODE (000110)

Max length = 111 bits; Min length = 17 bits

*Note: Core (e200z425n3)-based CPUs are capable of generating two (2) reads or writes per clock cycle in cases where multiple registers are accessed with a single instruction (lmw/stmw). These will have a double word pair size encoding (**p_tsiz** = 0b000). In these cases, the*

Nexus 3 module will send one (1) Data Trace Message with the two 32-bit data values as one combined 64-bit value for each message.

*For Core (e200z425n3)-based CPUs, the doubleword encoding (**p_tsiz** = 0b000) may also indicate a doubleword access and will be sent out as a single Data Trace Message with a single 64-bit data value.*

The debug/development tool will need to distinguish the two cases based on the family of processor.

84.13.2.3 Data Trace Synchronization messages

A Data Trace Write/Read with Sync message is messaged via the auxiliary port (provided Data Trace is enabled) for the following conditions (refer to *Table 2131*):

- Initial Data Trace Message after exit from system reset or whenever Data Trace is enabled.
- Upon returning from a CPU Low Power state.
- Upon returning from Debug Mode.
- After occurrence of queue overrun (can be caused by any trace message), provided Data Trace is enabled.
- After the periodic data trace counter has expired indicating 255 *without-sync* Data Trace messages have occurred since the last *with-sync* message occurred.
- Upon assertion of the Event In (**nex_evti_b**) pin, the first Data Trace Message will be a synchronization message if the EIC bits of the DC1 Register have enabled this feature.
- Upon Data Trace Write/Read after the previous DTM message was lost due to an attempted access to a secure memory location (for SOC's with security).
- Upon Data Trace Write/Read after the previous DTM message was lost due to a collision entering the FIFO between the DTM message and any two of the following: Watchpoint message, Ownership Trace message, or Program Trace message.

Data Trace Synchronization Messages provide the full address (without leading zeros) and insure that development tools fully synchronize with Data Trace regularly. Synchronization messages provide a reference address for subsequent DTMs, in which only the unique portion of the Data Trace address is transmitted. The format for Data Trace Write/Read with Sync Messages is as follows:

Figure 2248. Data Write/Read with Sync message format

(1–64 bits)	(1–32 bits)	(4 bits)	(1 bit)	(4 bits)	(6 bits)
Data Value	Full Address	Data Size	(0)	Source Proc.	TCODE (001101 or 001110)

Max length = 111 bits; Min length = 17 bits

If timestamping is enabled, the TSTAMP field is appended to the end of the message and adds up to 30 additional bits.

Exception conditions that result in Data Trace Synchronization are summarized in *Table 2131*.

Table 2131. Data Trace Exception summary

Exception Condition	Exception Handling
System Reset Negation	At the negation of JTAG reset (**j_trst_b**), queue pointers, counters, state machines, and registers within the Nexus 3 module are reset. If Data Trace is enabled, the first Data Trace Message is a Data Write/Read with Sync Message.
Data Trace Enabled	The first Data Trace Message (after Data Trace has been enabled) is a synchronization message.
Exit from Low Power/Debug	Upon exit from a Low Power mode or Debug mode the next Data Trace Message will be converted to a Data Write/Read with Sync Message.
Queue Overrun	An Error Message occurs when a new message cannot be queued due to the message queue being full. The FIFO will discard messages until it has completely emptied the queue. Once emptied, an Error Message will be queued. The error encoding will indicate which type(s) of messages attempted to be queued while the FIFO was being emptied. The next DTM message in the queue will be a Data Write/Read with Sync Message.
Periodic Data Trace Sync.	A forced synchronization occurs periodically after 255 Data Trace Messages have been queued. A Data Write/Read with Sync Message is queued. The periodic data trace message counter then resets.
Event In	If the Nexus module is enabled, a **nex_evti_b** assertion initiates a Data Trace Write/Read with Sync Message upon the next data write/read (if Data Trace is enabled and the EIC bits of the DC1 Register have enabled this feature).
Collision Priority	All Messages have the following priority: Instruction 0 (WPM → DQM → PCM_{PIDMSG} → OTM → BTM → DTM)→ Instruction1 (WPM → DQM → OTM → BTM → DTM), where instruction0 is the oldest instruction. A DTM Message that attempts to enter the queue at the same time as three other higher priority messages will be lost. A subsequent read/write will queue a Data Trace Read/Write with Sync Message.

84.13.3 DTM operation

84.13.3.1 Data Trace windowing

Data Write/Read Messages are enabled via the RWT field in the Data Trace Control Register (DTC) for each DTM channel. Data Trace windowing is achieved via the address range defined by the DTEA and DTSA Registers and by the RC field in the DTC register. All CPU initiated read/write accesses that fall inside or outside these address ranges, as programmed, are candidates to be traced.

84.13.3.2 Data Access / Instruction Access Data Tracing

The Nexus3 module is capable of tracing either instruction access data or data access data and can be configured for either type of data trace by setting the DI1 field within the Data Trace Control Register. This setting applies to all DTM channels.

84.13.3.3 Data Trace filtering

Data Trace filtering is available base on the settings of MSR_{PMM} and $DC4_{DTMARK}$.

84.13.3.4 Bus cycle special cases

Table 2132. Bus cycle special cases

Special case	Action
Bus cycle aborted	Cycle ignored
Bus cycle with data error ($\overline{TEA}$)(1)	Data Trace Message discarded
Bus cycle completed without error(1)	Cycle captured & transmitted
AHB bus cycle initiated by Nexus 3	Cycle ignored
Bus cycle is an instruction fetch	Cycle selectively ignored based on DTC_{DI} setting
Bus cycle accesses misaligned data (across 64-bit boundary) — both 1st & 2nd transactions within data trace range	1st & 2nd cycle captured & a single or a pair of DTM(s) is (are) transmitted (see Note)
Bus cycle accesses misaligned data (across 64-bit boundary) — 1st transaction within data trace range — 2nd transaction out of data trace range	1st & 2nd cycle captured & a single or a pair of DTM(s) is (are) transmitted (see Note)
Bus cycle accesses misaligned data (across 64-bit boundary) — 1st transaction within data trace range — 2nd transaction (regardless of within range or not) receives a bus error	Data Trace Message discarded
Bus cycle accesses misaligned data (across 64-bit boundary) — 1st transaction out of data trace range — 2nd transaction within data trace range	1st & 2nd cycle captured & a single or a pair of DTM(s) is (are) transmitted (see Note)
Bus cycle accesses misaligned data (across 64-bit boundary) — 1st transaction out of data trace range — 2nd transaction within range, receives a bus error	Data Trace Message discarded

1. Buffering of stores in the CPU store buffer may generate a DTM prior to the actual memory access, regardless of an error termination condition from memory.

Note: For misaligned accesses (crossing 64-bit boundary), the access is broken into two accesses by the CPU. If either access is within the data trace range, a single DTM will be sent with a size encoding indicating the size of the original access (that is word), and the address indicating the original misaligned accesses, unless the misaligned access wraps into the doubleword at address 0. In this case, since the two portions of the misaligned access are not contiguous, two DTMs will be sent, one for each portion. The size encodings and the addresses of the DTMs will indicate the accessed bytes of data. The data trace port handles these cases in the same manner. (Refer to the Data Trace Port section in the Core (e200z425n3) Core Debug Support chapter.)

A store to the cache's store buffer within the data trace range may initiate a DTM message prior to completion of the actual memory access.

84.13.4 Data Trace Timing Diagrams (8 MDO / 2 MSEO configuration)

Figure 2249. Data Trace — Data Write message

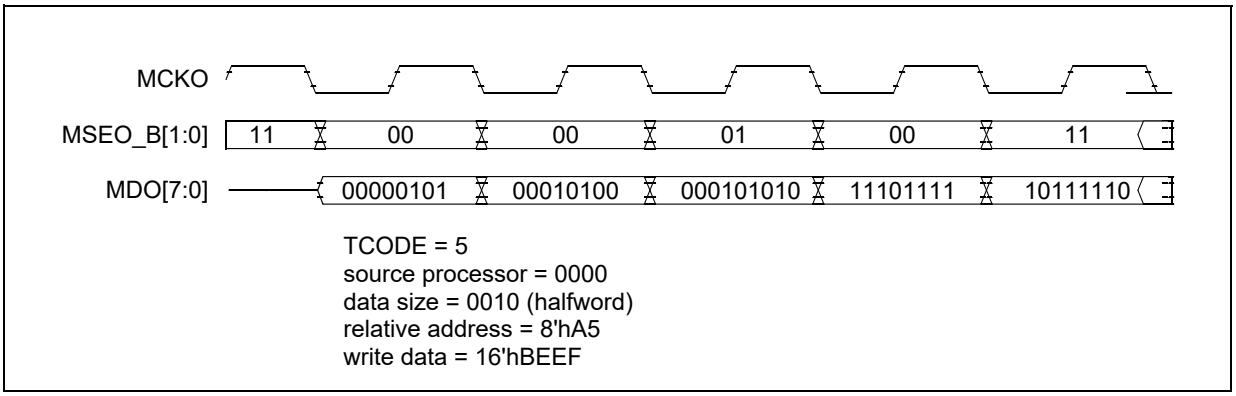
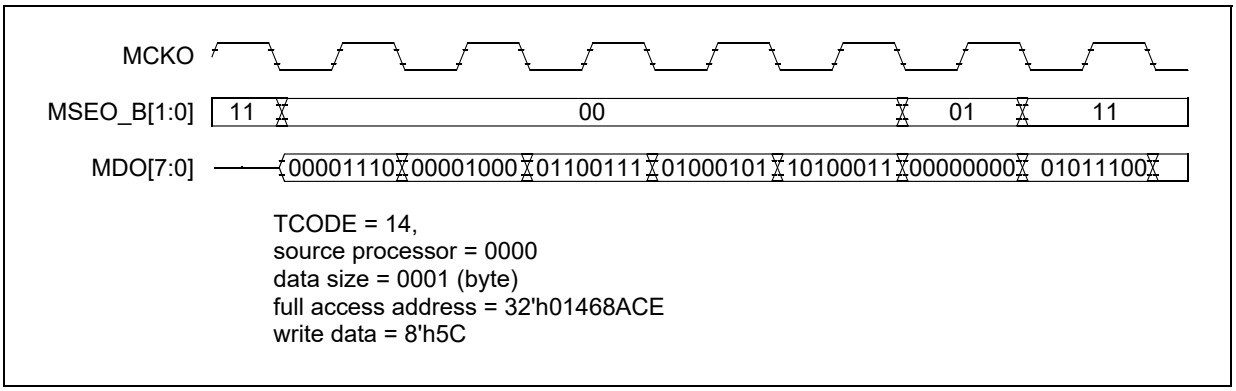


Figure 2250. Data Trace — Data Read with Sync message



84.14 Data Acquisition messaging

This section details the Data Acquisition mechanisms supported by the Nexus 3 module. Data Acquisition Trace is implemented using Data Acquisition Trace Messages in accordance with IEEE-ISTO 5001 definitions. The control mechanism to export the data is different from the recommendations of the standard, however.

Data Acquisition Trace provides a convenient and flexible mechanism for the debugger to observe the architectural state of the machine through software instrumentation.

84.14.1 Data Acquisition ID Tag field

The DQTAG Tag field (DQTAG) is an 8-bit value specifying control or attribute information for the data included in the Data Acquisition message. DQTAG is sampled from $DEVENT_{DQTAG}$ when a write to DDAM is performed via **mtspr** operations. The usage of the DQTAG is left to the discretion of the development tool to be used in whatever manner is deemed appropriate for the application.

84.14.2 Data Acquisition Data field

The Data Acquisition Data field (DQDATA) is the data captured from the DDAM write operation via **mtspr** operations. Leading zeros are omitted from the message.

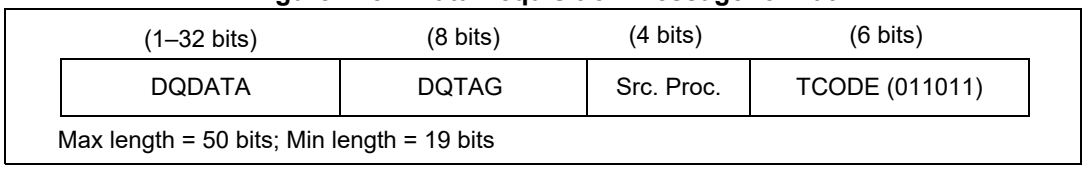
84.14.3 Data Acquisition Trace event

For DQM, a dedicated SPR has been allocated (DDAM). It is expected that the general use case is to instrument the software and use **mtspr** operations to generate Data Acquisition messages.

There is no explicit error response for failed accesses as a result of contention between an internal and external debugger. Software may be blocked or given ownership of DDAM and the DQTAG field of the DEVENT register via control in EDBRAC0 while in External Debug Mode. Hardware always has access to these registers. Refer to the “External Debug Resource Allocation Control Register (EDBRAC0)” section in the Core (e200z425n3) Core Debug Support chapter for more detail on EDBRAC0.

Reads from the Data Acquisition channel do not generate a Data Acquisition event and will return zeroes for the read data.

Figure 2251. Data Acquisition message format



If timestamping is enabled, the TSTAMP field is appended to the end of the message and adds up to 30 additional bits.

84.15 Watchpoint Trace messaging

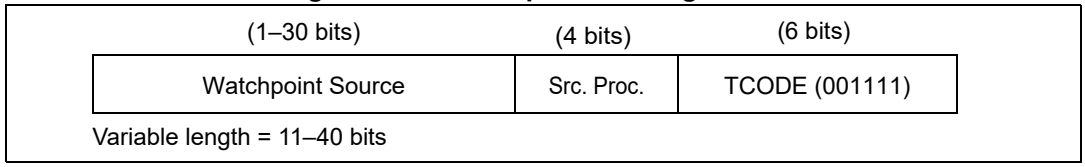
Enabling Watchpoint messaging is done by setting the Watchpoint Trace Enable bit in the DC1 Register. Setting the individual Watchpoint sources is supported through the Nexus1 module and the Performance Monitor unit. The Nexus1 module is capable of setting multiple types of watchpoints. Refer to the Core (e200z425n3) Core Debug Support chapter for details on watchpoint initialization.

When watchpoints occur due to one or more asserted watchpoint event signals and Watchpoint Trace Messaging is enabled, a Watchpoint Trace message will be sent to the message queue to be messaged out. This message includes the watchpoint number indicating which watchpoint(s) caused the message. If more than one enabled watchpoint occurs in a single cycle, only one Watchpoint Trace message is generated and multiple bits of the watchpoint hit field will be set. The settings of the $WMSK_{WEM}$ field control which watchpoints are enabled to generate watchpoint trace messages.

The occurrence of any of the defined watchpoints can also be programmed to assert the Event Out (**nex_evto_b**) pin for one (1) period of the output clock (**nex_mcko**) based on settings in the DC2 and DC3 registers. Refer to *Table 2136* for details on **nex_evto_b**.

Watchpoint information is messaged out in the following format:

Figure 2252. Watchpoint message format



If timestamping is enabled, the TSTAMP field is appended to the end of the message and adds up to 30 additional bits.

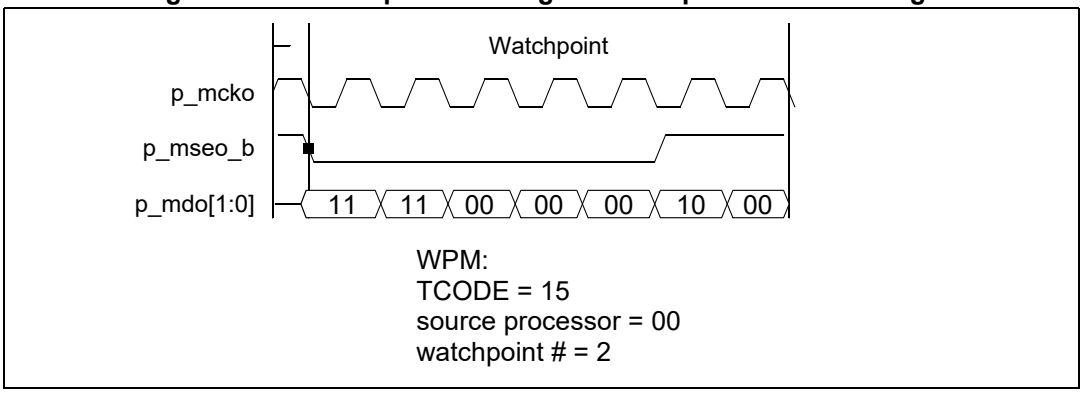
The Watchpoint Source message field will contain a ‘1’ for each asserted watchpoint. Leading zeros are truncated.

Table 2133. Watchpoint source encoding

Watchpoint source (1–30 bits)	Watchpoint description
00000000000000000000000000000000	No Watchpoints enabled for Watchpoint Trace Messaging
*xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx*1	Watchpoint #0 enabled for WTM
*xxxxxxxxxxxxxxxxxxxxxxxxxxxxxx*1*x*	Watchpoint #1 enabled for WTM
*xxxxxxxxxxxxxxxxxxxxxxxxxxxxx*1*xx*	Watchpoint #2 enabled for WTM
*xxxxxxxxxxxxxxxxxxxxxxxxxxxx*1*xxx*	Watchpoint #3 enabled for WTM
*xxxxxxxxxxxxxxxxxxxxxxxxxxx*1*xxxx*	Watchpoint #4 enabled for WTM
*xxxxxxxxxxxxxxxxxxxxxxxxxx*1*xxxxx*	Watchpoint #5 enabled for WTM
*xxxxxxxxxxxxxxxxxxxxxxxxx*1*xxxxxx*	Watchpoint #6 enabled for WTM
*xxxxxxxxxxxxxxxxxxxxxxxx*1*xxxxxxx*	Watchpoint #7 enabled for WTM
*xxxxxxxxxxxxxxxxxxxxxxx*1*xxxxxxxx*	Watchpoint #8 enabled for WTM
*xxxxxxxxxxxxxxxxxxxxxx*1*xxxxxxxxx*	Watchpoint #9 enabled for WTM
*xxxxxxxxxxxxxxxxxxxxx*1*xxxxxxxxxx*	Watchpoint #10 enabled for WTM
*xxxxxxxxxxxxxxxxxxxx*1*xxxxxxxxxxx*	Watchpoint #11 enabled for WTM
*xxxxxxxxxxxxxxxxxxx*1*xxxxxxxxxxxx*	Watchpoint #12 enabled for WTM
*xxxxxxxxxxxxxxxxxx*1*xxxxxxxxxxxxx*	Watchpoint #13 enabled for WTM
*xxxxxxxxxxxxxxxxx*1*xxxxxxxxxxxxxx*	Watchpoint #14 enabled for WTM
*xxxxxxxxxxxxxxxx*1*xxxxxxxxxxxxxxx*	Watchpoint #15 enabled for WTM
*xxxxxxxxxxxxxxx*1*xxxxxxxxxxxxxxxx*	Watchpoint #16 enabled for WTM
*xxxxxxxxxxxxxx*1*xxxxxxxxxxxxxxxx*	Watchpoint #17 enabled for WTM
*xxxxxxxxxxxxx*1*xxxxxxxxxxxxxxxx*	Watchpoint #18 enabled for WTM
*xxxxxxxxxxxx*1*xxxxxxxxxxxxxxxx*	Watchpoint #19 enabled for WTM
*xxxxxxxxxxx*1*xxxxxxxxxxxxxxxx*	Watchpoint #20 enabled for WTM
*xxxxxxxxxx*1*xxxxxxxxxxxxxxxx*	Watchpoint #21 enabled for WTM
*xxxxxxxxx*1*xxxxxxxxxxxxxxxx*	Watchpoint #22 enabled for WTM
*xxxxxxxx*1*xxxxxxxxxxxxxxxx*	Watchpoint #23 enabled for WTM
*xxxxxxx*1*xxxxxxxxxxxxxxxx*	Watchpoint #24 enabled for WTM
*xxxxxx*1*xxxxxxxxxxxxxxxx*	Watchpoint #25 enabled for WTM
*xxxxx*1*xxxxxxxxxxxxxxxx*	Watchpoint #26 enabled for WTM
*xxxx*1*xxxxxxxxxxxxxxxx*	Watchpoint #27 enabled for WTM
*xxx*1*xxxxxxxxxxxxxxxx*	Watchpoint #28 enabled for WTM
*xx*1*xxxxxxxxxxxxxxxx*	Watchpoint #29 enabled for WTM
*x*1*xxxxxxxxxxxxxxxx*	Watchpoint #30 enabled for WTM
1*xxxxxxxxxxxxxxxx*	Watchpoint #31 enabled for WTM

84.15.1 Watchpoint Timing Diagram (2 MDO / 1 MSEO configuration)

Figure 2253. Watchpoint Message & Watchpoint Error message



84.16 External Hardware Trigger Controls

The Nexus 3 module provides a set of hardware trigger inputs to allow for external hardware to also independently control enabling/disabling of Program Trace, Data Trace, Ownership Trace, and Watchpoint Trace messaging. These signals can be used to produce a “window” for trace messaging for each type.

When a start trigger condition is detected for one of the trace types, and the corresponding trace type has not already been enabled by a software write to the DC1 register, the designated trace features become enabled, and the corresponding enable bit(s) of the DC1 register is set. When a stop trigger condition is detected and the corresponding trace type has not already been enabled by a software write to the DC1 register, the designated trace feature becomes disabled, and the corresponding enable bit of the DC1 register is cleared. If a simultaneous trigger condition is received for both start and stop triggering, then the designated trace feature will toggle between enabled and disabled depending on the enable state at the time of the trigger events. For example, if tracing is enabled, and a start and stop trigger condition occur simultaneously, then tracing will be disabled. Direct writes of the DC1 register take precedence over any trace feature control action that is derived from external hardware triggering, and if the corresponding trace feature has been enabled by a direct write to DC1, then the corresponding hardware trigger signals will have no effect. *Table 2134* details the external hardware trigger signals.

Table 2134. External Hardware Trigger Signals

Name	Description
nex_dtm_starttr	Data Trace Messaging start trigger input. A transition from 0→1 on this input signals a start trace condition.
nex_dtm_stoptr	Data Trace Messaging stop trigger input. A transition from 0→1 on this input signals a stop trace condition.
nex_otm_starttr	Ownership Trace Messaging start trigger input. A transition from 0→1 on this input signals a start trace condition.
nex_otm_stoptr	Ownership Trace Messaging stop trigger input. A transition from 0→1 on this input signals a stop trace condition.

Table 2134. External Hardware Trigger Signals (continued)

Name	Description
nex_ptm_starttr	Program Trace Messaging start trigger input. A transition from 0→1 on this input signals a start trace condition.
nex_ptm_stoptr	Program Trace Messaging stop trigger input. A transition from 0→1 on this input signals a stop trace condition.
nex_wtm_starttr	Watchpoint Trace Messaging start trigger input. A transition from 0→1 on this input signals a start trace condition.
nex_wtm_stoptr	Watchpoint Trace Messaging stop trigger input. A transition from 0→1 on this input signals a stop trace condition.

These signals are active-high inputs and are transition sensitive. A 0→1 transition on the signal indicates the corresponding condition is requested. These signals are not synchronized, and must meet setup and hold requirements relative to **nex_clk**.

84.17 Nexus 3 Read/Write access to memory-mapped resources

The Read/Write access feature allows access to memory-mapped resources via the JTAG/OnCE port. The Read/Write mechanism supports single as well as block reads and writes to AHB resources.

The Nexus 3 module is capable of accessing resources on the system bus (AHB). Memory-mapped registers and other non-cached memory can be accessed via the standard memory map settings.

All accesses are setup and initiated by the Read/Write Access Control/Status Register (RWCS), as well as the Read/Write Access Address (RWA) and Read/Write Access Data Registers (RWD). Nexus 3 read/write accesses are run as privileged data non-cacheable accesses by default, and drive the **p_d_hprot[5:0]** bus access attributes to 6‘b000011 accordingly. The $RWCS_{ATTR}$ field is provided to allow a portion of these default values to be modified when performing read or write accesses using the Nexus 3 Read/Write access mechanism.

Using the Read/Write Access Registers (RWCS/RWA/RWD), memory mapped AHB resources can be accessed through Nexus 3. The following subsections describe the steps required to access memory-mapped resources.

Note: Read/Write Access can only access memory mapped resources when system reset is de-asserted and clocks are running.

Misaligned accesses are NOT supported in the Nexus 3 module.

Uncorrectable ECC errors on Nexus 3 read access will result in the RWD register being updated with the raw data received.

84.17.1 Single write access

1. Initialize the Read/Write Access Address Register (RWA) through the access method outlined in *Section 84.5: Nexus 3 Register Access via JTAG/OnCE*. Configure as follows:
   a) Write Address → 32h'xxxxxxxx (write address)
2. Initialize the Read/Write Access Control/Status Register (RWCS) through the access method outlined in *Section 84.5: Nexus 3 Register Access via JTAG/OnCE*. Configure the bits as follows:
   a) Access Control (AC) → 1b'1 (to indicate start access)
   b) Map Select (MAP) → 3b'000 (primary memory map)
   c) Access Priority (PR) → 2b'11 (highest priority)
   d) Read/Write (RW) → 1b'1 (write access)
   e) Word Size (SZ) → 3b'0xx (32-bit, 16-bit, 8-bit)
   f) Access Count (CNT) → 14h'0000 or 14h'0001(single access)

Note: Access Count (CNT) of 14'h0000 or 14'h0001 will perform a single access.

3. Initialize the Read/Write Access Data Register (RWD) through the access method outlined in *Section 84.5: Nexus 3 Register Access via JTAG/OnCE*. Configure as follows:
   a) Write Data → 32h'xxxxxxxx (write data)
4. The Nexus block will then arbitrate for the AHB system bus and transfer the data value from the data buffer RWD Register to the memory mapped address in the Read/Write Access Address Register (RWA). The **nex_ahb_start** output will be asserted during the first clock of the address phase of the transfer. When the access has completed, Nexus clears the AC and DV bits in the RWCS Register. If the access has completed without error, (ERR=1'b0), Nexus asserts the **nex_rdy_b** pin (refer to *Table 2136* for details) and clears the ERR bit in the RWCS Register. Otherwise, if the access completes with an error, the **nex_err_b** pin will be asserted and the ERR bit will be set to indicate an error has occurred, and the **nex_rdy_b** pin will not be asserted. Once DV has been cleared, this indicates that the device is ready for the next access, or that an error has occurred.

*Note: Only the **nex_ahb_start**, **nex_rdy_b**, and **nex_err_b** pins as well as the AC, DV, and ERR bits within the RWCS provide Read/Write Access status to the external development tool.*

84.17.2 Block write access

1. For a block write access, follow Steps 1, 2, and 3 outlined in *Section 84.17.1: Single write access* to initialize the registers, but using a value greater than one (14'h0001) for the CNT field in the RWCS Register.
2. The Nexus block will then arbitrate for the AHB system bus and transfer the first data value from the RWD Register to the memory mapped address in the Read/Write Access Address Register (RWA). The **nex_ahb_start** output will be asserted during the first clock of the address phase of the transfer. When the transfer has completed without error, the address from the RWA Register is incremented to the next word size (specified in the SZ field), the number from the CNT field is decremented, and the **nex_rdy_b** pin is asserted. If the access has completed without error, Nexus clears the ERR and DV bits in the RWCS Register, otherwise the ERR bit will be set and the DV bit will be cleared to indicate an error has occurred, the **nex_err_b** pin is asserted, the block transfer is aborted, and the AC bit in the RWCS register is cleared. Once DV has

been cleared, this indicates that the device is ready for the next access in the block transfer, or an error has occurred.

3. If AC has not been cleared due to an error, repeat Step 3 in *Section 84.17.1: Single write access* until the internal CNT value is zero (0). When this occurs, the AC bit within the RWCS will be cleared to indicate the end of the block write access.

Note: The actual RWA value as well as the CNT field within the RWCS are not changed when executing a block write access. The original values can be read by the external development tool at any time.

84.17.3 Single read access

1. Initialize the Read/Write Access Address Register (RWA) through the access method outlined in *Section 84.5: Nexus 3 Register Access via JTAG/OnCE*. Configure as follows:
   a) Read Address → 32h'xxxxxxxx (read address)
2. Initialize the Read/Write Access Control/Status Register (RWCS) through the access method outlined in *Section 84.5: Nexus 3 Register Access via JTAG/OnCE*. Configure the bits as follows:
   a) Access Control (AC) → 1b'1 (to indicate start access)
   b) Map Select (MAP) → 3b'000 (primary memory map)
   c) Access Priority (PR) → 2b'11 (highest priority)
   d) Read/Write (RW) → 1b'0 (read access)
   e) Word Size (SZ) → 3b'0xx (32-bit, 16-bit, 8-bit)
   f) Access Count (CNT) → 14h'0000 or 14h'0001(single access)

Note: Access Count (CNT) of 14'h0000 or 14'h0001 will perform a single access.

3. The Nexus block will then arbitrate for the AHB system bus and the read data will be transferred from the AHB to the RWD Register. The **nex_ahb_start** output will be asserted during the first clock of the address phase of the transfer. When the access has completed without error, Nexus clears the ERR bit and sets the DV bit in the RWCS Register and asserts the **nex_rdy_b** pin (refer to *Table 2136* for detail on **nex_rdy_b**). Otherwise, if the access has completed with an error, the **nex_err_b** pin will be asserted, the ERR bit will be set and the DV bit will be cleared to indicate an error has occurred, and the **nex_rdy_b** pin will not be asserted. Once ERR or DV has been set, this indicates that the device is ready for the next access, or that an error has occurred. The AC bit will be cleared in either case.
4. The data can then be read from the Read/Write Access Data Register (RWD) through the access method outlined in *Section 84.5: Nexus 3 Register Access via JTAG/OnCE*.

*Note: Only the **nex_ahb_start**, **nex_rdy_b**, and **nex_err_b** pins as well as the AC, DV and ERR bits within the RWCS provide Read/Write Access status to the external development tool.*

84.17.4 Block read access

1. For a block read access, follow Steps 1 and 2 outlined in *Section 84.17.3: Single read access* to initialize the registers, but using a value greater than one (14'h0001) for the CNT field in the RWCS Register.
2. The Nexus block will then arbitrate for the AHB system bus and the read data will be transferred from the AHB to the RWD Register. When the access has completed without error, Nexus clears the ERR bit and sets the DV bit in the RWCS Register and

asserts the **nex_rdy_b** pin (refer to *Table 2136* for detail on **nex_rdy_b**). Otherwise, if the access has completed with an error, the **nex_err_b** pin will be asserted, the ERR bit will be set and the DV bit will be cleared to indicate an error has occurred, and the **nex_rdy_b** pin will not be asserted. Once ERR or DV has been set, this indicates that the device is ready for the next access, or that an error has occurred. The AC bit will be cleared in either case.

3. When the transfer has completed without error, the address from the RWA Register is incremented to the next word size (specified in the SZ field), the number from the CNT field is decremented, Nexus clears the ERR bit and sets the DV bit in the RWCS Register, and asserts the **nex_rdy_b** pin (refer to *Table 2136* for detail on **nex_rdy_b**). Otherwise, if the access has completed with an error, the **nex_err_b** pin will be asserted, the ERR bit will be set and the DV bit will be cleared to indicate an error has occurred, the AC bit is cleared and the block transfer is aborted, and the **nex_rdy_b** pin will not be asserted. Once ERR or DV has been set, this indicates that the device is ready for the next access, or that an error has occurred. Once DV has been set, this indicates that the device is ready for the next access in the block transfer, or if ERR is set (AC will be cleared), the block transfer has been aborted.
4. The data can then be read from the Read/Write Access Data Register (RWD) through the access method outlined in *Section 84.5: Nexus 3 Register Access via JTAG/OnCE*.
5. If AC has not been cleared due to an error, repeat Steps 3 and 4 in *Section 84.17.3: Single read access* until the CNT value is zero (0). When this occurs, the AC bit within the RWCS is cleared to indicate the end of the block read access.

Note: The data values must be shifted out 32 bits at a time LSB first (that is doubleword read = two word reads from the RWD).

The actual RWA value as well as the CNT field within the RWCS are not changed when executing a block read access. The original values can be read by the external development tool at any time.

84.17.5 Error handling

The Nexus 3 module handles various error conditions as follows:

84.17.5.1 AHB Read/Write error

All address and data errors that occur on read/write accesses to the AHB system bus will return a transfer error encoding on the **p_hresp[1:0]** signals. If this occurs:

1. The access is terminated without retrying (AC bit is cleared).
2. The ERR bit in the RWCS Register is set.
3. The Error Message is sent (TCODE = 8) indicating Read/Write error.

84.17.5.2 Access termination

The following cases are defined for sequences of the Read/Write protocol that differ from those described in the above sections.

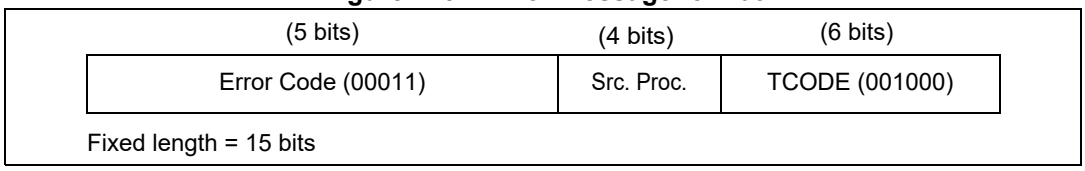
1. If the AC bit in the RWCS Register is set to start Read/Write accesses and invalid values are loaded into the RWD or RWA, or both, then an AHB access error may occur. This is handled as described above.
2. If a block access is in progress (all cycles not completed), and the RWCS Register is written, then the original block access is terminated at the boundary of the nearest completed access.
   a) If the RWCS is written with the AC bit set, the next Read/Write access will begin and the RWD can be written to/ read from.
   b) If the RWCS is written with the AC bit cleared, the Read/Write access is terminated at the nearest completed access. This method can be used to break (early terminate) block accesses.

84.17.6 Read/Write access error message

The Read/Write Access Error Message is sent out when an AHB system bus access error (read or write) has occurred.

Error information is messaged out in the following format:

Figure 2254. Error message format



84.18 Nexus 3 pin interface

This section details information regarding the Nexus 3 pins and pin protocol.

The Nexus 3 pin interface provides the function of transmitting messages from the messages queues to the external tools. It is also responsible for handshaking with the message queues.

84.18.1 Pins implemented

The Nexus 3 module implements an auxiliary port consisting of one (1) **nex_evti_b** and one (1) **nex_mseo_b** or two (2) **nex_mseo_b[1:0]**. It also implements a configurable number of **nex_mdo[n:0]** pins, (1) **nex_rdy_b** pin, (1) **nex_err_b** pin, (1) **nex_evto_b** pin, (4) **nex_wevto[3:0]** pins, and one (1) clock output pin (**nex_mcko**), as well as additional configuration pins described in *Table 2136*. The output pins are synchronized to the Nexus 3 output clock (**nex_mcko**).

All Nexus 3 input functionality may be controlled through the JTAG/OnCE port in compliance with IEEE 1149.1 (refer to *Section 84.5: Nexus 3 Register Access via JTAG/OnCE* for details). The JTAG pins are incorporated as I/O to the processor, and are further described in the “JTAG/OnCE Pins” section of the Core (e200z425n3) Core Debug Support chapter.

Table 2135. JTAG pins for Nexus 3

JTAG pins	Input/ Output	Description of JTAG pins (included in Nexus 1)
j_tdo	O	The Test Data Output (**j_tdo**) pin is the serial output for test instructions and data. **j_tdo** is three-stateable and is actively driven in the “Shift-IR” and “Shift-DR” controller states. **j_tdo** changes on the falling edge of **j_tclk**.
j_tdi	I	The Test Data Input (**j_tdi**) pin receives serial test instruction and data. TDI is sampled on the rising edge of **j_tclk**.
j_tms	I	The Test Mode Select (**j_tms**) input pin is used to sequence the OnCE controller state machine. **j_tms** is sampled on the rising edge of **j_tclk**.
j_tclk	I	The Test Clock (**j_tclk**) input pin is used to synchronize the test logic, and control register access through the JTAG/OnCE port.
j_trst_b	I	The Test Reset (**j_trst_b**) input pin is used to asynchronously initialize the JTAG/OnCE controller.

The auxiliary pins are used to send and receive messages and are described in *Table 2136*.

Table 2136. Nexus 3 auxiliary pins

Auxiliary pins	Input/ Output	Description of auxiliary pins
nex_mcko	O	Message Clock Out (**nex_mcko**) is a free running output clock to development tools for timing of **nex_mdo[n:0]** & **nex_mseo_b[1:0]** pin functions. **nex_mcko** is programmable through the DC1 Register.
nex_mdo[n:0]	O	Message Data Out (**nex_mdo[n:0])** are output pins used for OTM, BTM, and DTM. External latching of **nex_mdo[n:0]** shall occur on the rising edge of the Nexus3 clock (**nex_mcko**).
nex_mseo_b[1:0]	O	Message Start/End Out (**nex_mseo_b[1:0]**) are output pins that indicate when a message on the **nex_mdo[n:0]** pins has started, when a variable length packet has ended, and when the message has ended. External latching of **nex_mseo_b[1:0]** shall occur on the rising edge of the Nexus3 clock (**nex_mcko**). One or two pin MSEO functionality is determined at integration time per SOC implementation
nex_rdy_b	O	Ready (**nex_rdy_b**) is an output pin used to indicate to the external tool that the Nexus block is ready for the next Read/Write Access. If Nexus is enabled, this signal is asserted upon successful (without error) completion of an AHB system bus transfer (Nexus read or write) & is held asserted until the JTAG/OnCE state machine reaches the “Capture_DR” state. Upon exit from system reset or if Nexus is disabled, **nex_rdy_b** remains deasserted
nex_err_b	O	Error (**nex_err_b**) is an output pin used to indicate to the external tool that the Nexus block is ready for the next Read/Write Access and an error has occurred on the previous access. If Nexus is enabled, this signal is asserted upon unsuccessful (with error) completion of an AHB system bus transfer (Nexus read or write) & is held asserted until the JTAG/OnCE state machine reaches the “Capture_DR” state. Upon exit from system reset or if Nexus is disabled, **nex_err_b** remains deasserted
nex_evto_b	O	Event Out (**nex_evto_b**) is an output that, when asserted, indicates one of two events has occurred based on the EOC bits in the DC1 Register. **nex_evto_b** is held asserted for one (1) cycle of **nex_mcko**: 1) One (or more) watchpoints has occurred (from Nexus1) & EOC = 2’b00 2) Debug mode was entered (jd_debug_b asserted from Nexus1) & EOC = 2’b01

Table 2136. Nexus 3 auxiliary pins (continued)

Auxiliary pins	Input/ Output	Description of auxiliary pins
nex_evti_b	I	Event In (**nex_evti_b**) is an input that, when asserted, will initiate one of two events based on the EIC bits in the DC1 Register (if the Nexus module is enabled at reset): 1) Program Trace & Data Trace synchronization messages (provided Program Trace & Data Trace are enabled & EIC = 2'b00). 2) Debug request to Nexus1 module (provided EIC = 2'b01 and this feature is implemented).
nex_wevto[3:0]	O	Watchpoint Event Out 3:0 (**nex_wevto[3:0]**) are outputs that, when asserted, indicates one or more watchpoint events has occurred based on the settings in the DC2 and DC3 registers. **nex_wevto[3:0]** is held asserted for one (1) cycle of **nex_mcko**.
nex_ext_src_id[0:3]	I	nex_ext_src_id[0:3] is used to provide the SRC field value used in each message. These pins are tied to a predetermined value at SoC integration time
nex_sfwcntl_en]	I	nex_sfwcntl_en is used to allow software to control the module resources via the DCR register mappings. SoC logic should drive this signal appropriately in a semi-static manner based on the presence of an external hardware debugger and appropriate security precautions.

The Nexus auxiliary port arbitration pins are used when the Nexus 3 module is implemented in a multi-Nexus SoC that shares a single auxiliary output port. The arbitration is controlled by an SoC level Nexus Aurora Router (NAR). Refer to *Section 84.20: Auxiliary port arbitration*, for detail on Nexus port arbitration.

Table 2137. Nexus port arbitration signals

Nexus port arbitration pins	Input/ Output	Description of arbitration pins
nex_aux_req[1:0]	O	Nexus Auxiliary Request (**nex_aux_req[1:0]**) output signals indicate to an SoC level Nexus arbiter a request for access to the shared Nexus auxiliary port in a multi-Nexus implementation. The priority encodings are determined by how many messages are currently in the message queues (See *Table 2139*.)
nex_aux_busy	O	Nexus Auxiliary Busy (**nex_aux_busy**) is an output signal to an SoC level Nexus arbiter indicating that the Nexus 3 module is currently transmitting its message after being granted the Nexus auxiliary port.
nar_aux_grant	I	Nexus Auxiliary Grant (**nar_aux_grant**) is an input from the SoC level NAR that the auxiliary port has been granted to the Nexus 3 module to transmit its message.
ext_multi_nex_sel	I	Multi-Nexus Select (**ext_multi_nex_sel**) is a static signal indicating that the Nexus 3 module is implemented within a multi-Nexus environment. If set, port control and arbitration is controlled by the SoC level arbitration module (NAR).

84.18.2 Pin protocol

The protocol for the processor transmitting messages via the auxiliary pins is accomplished with the MSEO pin function outlined in *Table 2138*. Both single and dual pin cases are shown.

nex_mseo_b[1:0] is used to signal the end of variable-length packets, and not fixed length packets. **nex_mseo_b[1:0]** is sampled on the rising edge of the Nexus 3 clock (**nex_mcko**).

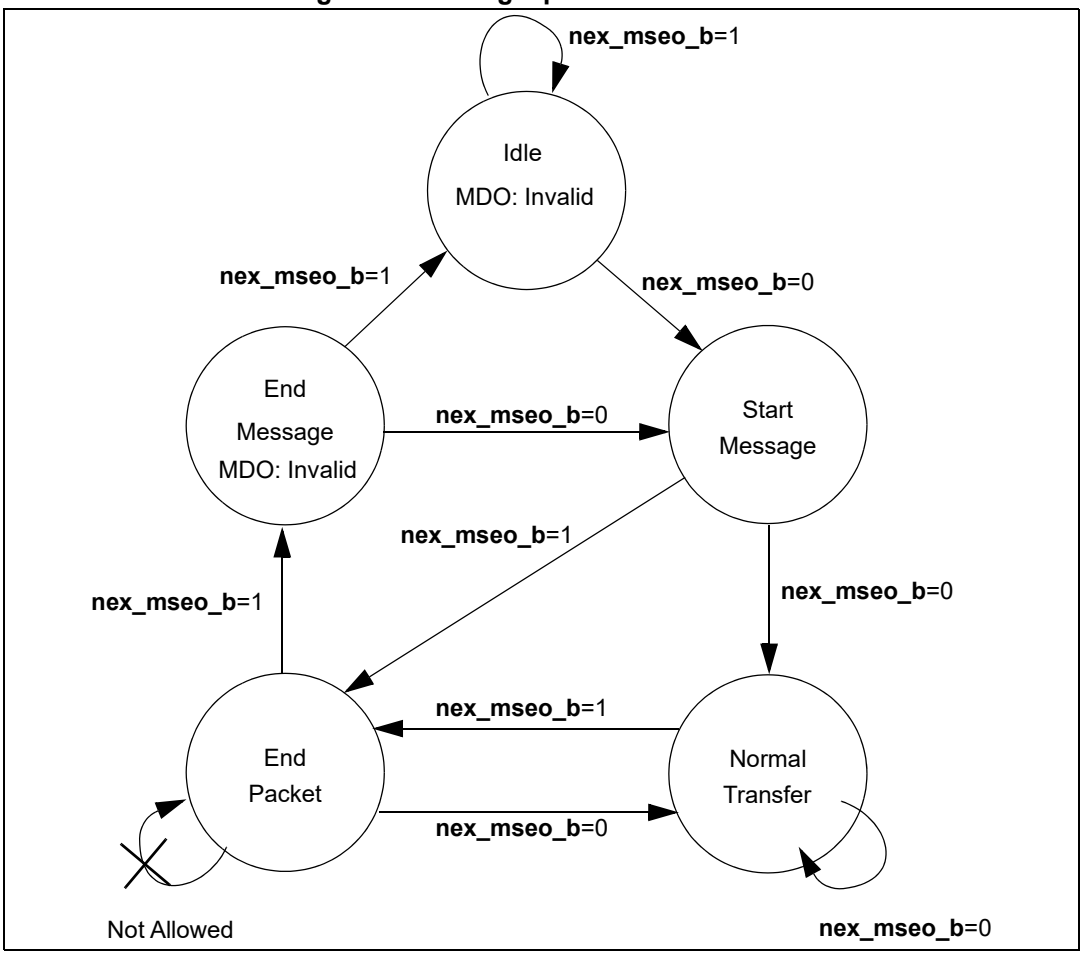
Single pin MSEO is not supported on the SPC58xEx/SPC58xGx.

Table 2138. MSEO pin(s) protocol

nex_mseo_b function	Single nex_mseo_b data (serial)	Dual nex_mseo_b[1:0] data
Start of message	1-1-0	11-00
End of message	0-1-1-(more 1's)	00 (or 01)-11-(more 1's)
End of variable length packet	0-1-0	00-01
Message transmission	0's	00's
Idle (no message)	1's	11's

Figure 2255 illustrates the state diagram for single pin MSEO transfers.

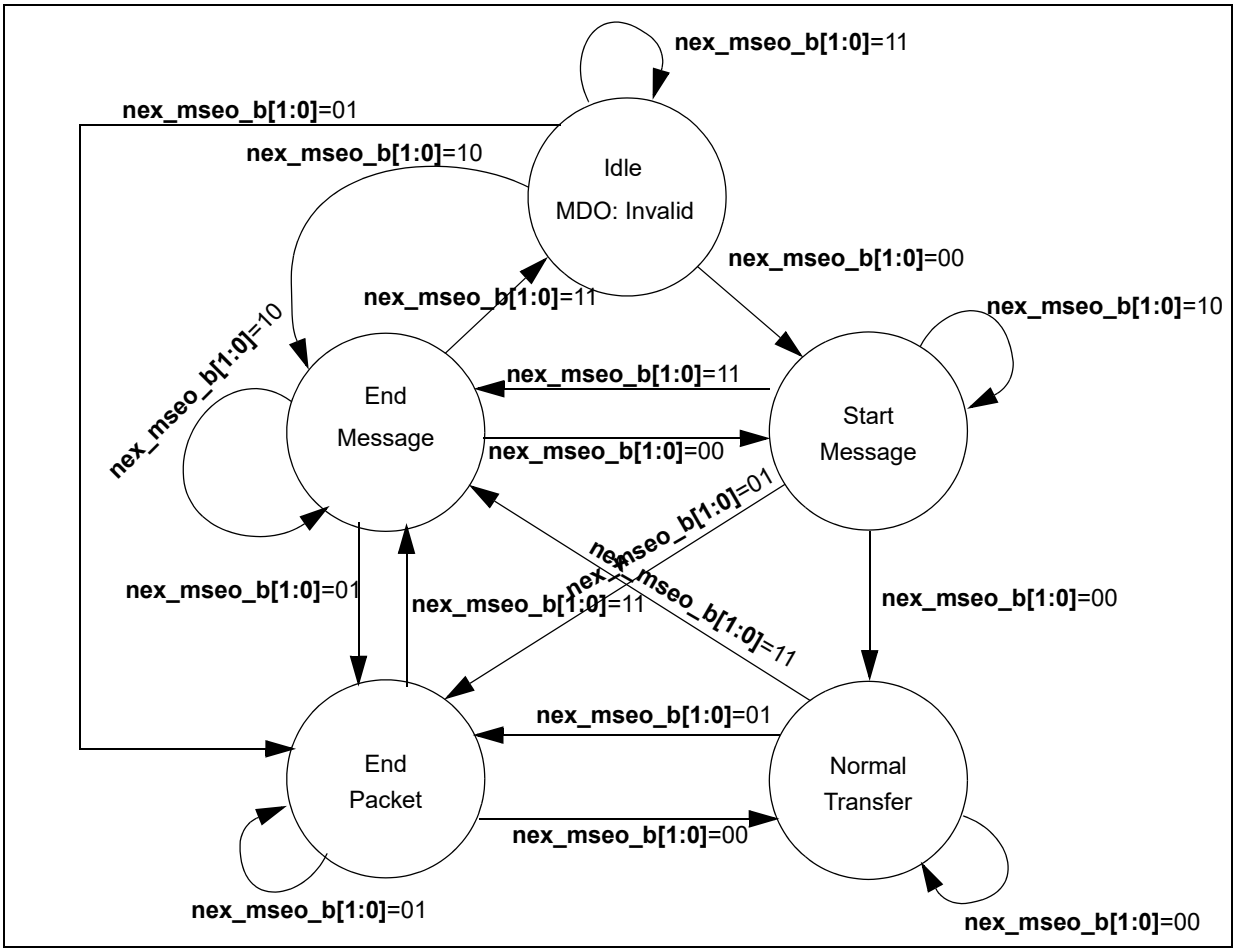
Figure 2255. Single pin MSEO transfers



Note that the “End Message” state does not contain valid data on the **nex_mdo[n:0]** pins. Also, It is not possible to have two consecutive “End Packet” messages. This implies the minimum packet size for a variable length packet is 2x the number of **nex_mdo[n:0]** pins. This ensures that a false end of message state is not entered by emitting two consecutive ‘1’s on the **nex_mseo_b** pin before the actual end of message.

Figure 2256 illustrates the state diagram for dual pin MSEO transfers.

Figure 2256. Dual pin MSEO transfers



The dual pin MSEO option is more robust that the single pin option. Termination of the current message may immediately be followed by the start of the next message on the consecutive clocks. An extra clock to end the message is not necessary as with the one MSEO pin option. The dual pin option also allows for consecutive “End Packet” states. This can be an advantage when small, variable sized packets are transferred.

Note: The “End Message” state may also indicate the end of a variable-length packet as well as the end of the message when using the dual pin option.

84.19 Rules for output messages

Class 3 compliant embedded processors must provide messages via the auxiliary port in a consistent manner as described below:

- A variable-sized packet within a message must end on a port boundary.
- A variable-sized packet may start within a port boundary only when following a fixed length packet. (If two variable-sized packets end and start on the same clock, it is

impossible to know which bit is from the last packet and which bit is from the next packet.)

- Whenever a variable-length packet is sized such that it does not end on a port boundary, it is necessary to extend and zero fill the remaining bits after the highest-order bit so that it can end on a port boundary.

  For example, if the **nex_mdo[n:0]** port is 2 bits wide, and the unique portion of an indirect address TCODE is 5 bits, then the remaining 1 bit of **nex_mdo[n:0]** must be packed with a 0.

84.20 Auxiliary port arbitration

In a multi-Nexus environment, the Nexus 3 module must arbitrate for the shared Nexus port at the SoC level.The request scheme is implemented as a 2-bit request with various levels of priority. The priority levels are defined in *Table 2139* below. The Nexus 3 module will receive a 1-bit grant signal (**nar_aux_grant**) from the SoC level arbiter. When a grant is received, the Nexus 3 module will begin transmitting its message following the protocol outlined in *Section 84.18.2: Pin protocol*. The Nexus 3 module will maintain control of the port, by asserting the **nex_aux_busy** signal, until the MSEO state machine reaches the “End Message” state.

Table 2139. MDO request encodings

Request level	MDO request encoding (nex_aux_req[1:0])	Condition of queue
No Request	00	No message to send
Low Priority	01	Message queue less than 1/2 full
—	10	Reserved
High Priority	11	Message queue 1/2 full or more

84.21 Examples

The following are examples of Program Trace and Data Trace Messages.

Table 2140 illustrates an example Indirect Branch Message with 2 MDO / 1MSEO configuration. *Table 2141* illustrates the same example with an 8 MDO / 2 MSEO configuration.

Note that T0 and S0 are the least significant bits where:

- Tx = TCODE number (fixed)
- Sx = Source processor (fixed)
- MAP = (always set to 0)
- Ix = Number of instructions (variable)
- Ax = Unique portion of the address (variable)

Note that during clock 13, the **nex_mdo[n:0]** pins are ignored in the single MSEO case.

Table 2140. Indirect branch message example (2 MDO / 1 MSEO)

Clock	nex_mdo[1:0]		nex_mseo_b	State
0	X	X	1	Idle (or end of last message)
1	T1	T0	0	Start Message
2	T3	T2	0	Normal Transfer
3	T5	T4	0	Normal Transfer
4	S1	S0	0	Normal Transfer
5	S3	S2	0	Normal Transfer
6	I0	MAP	0	Normal Transfer
7	I2	I1	0	Normal Transfer
8	I4	I3	1	End Packet
9	A1	A0	0	Normal Transfer
10	A3	A2	0	Normal Transfer
11	A5	A4	0	Normal Transfer
12	A7	A6	1	End Packet
13	0	0	1	End Message
14	T1	T0	0	Start Message

Table 2141. Indirect branch message example (8 MDO / 2 MSEO)

Clock	nex_mdo[7:0]								nex_mseo_b[1:0]		State
0	X	X	X	X	X	X	X	X	1	1	Idle (or end of last message)
1	S1	S0	T5	T4	T3	T2	T1	T0	0	0	Start Message
2	I4	I3	I2	I1	I0	MAP	S3	S2	0	1	End Packet
3	A7	A6	A5	A4	A3	A2	A1	A0	1	1	End Packet/End Message
4	S1	S0	T5	T4	T3	T2	T1	T0	0	0	Start Message

Table 2142 and *Table 2143* illustrate examples of Direct Branch Messages: one with 2 MDO / 1 MSEO, and one with 8 MDO / 2 MSEO.

Note that T0 and I0 are the least significant bits where:

- Tx = TCODE number (fixed)
- Sx = Source processor (fixed)
- Ix = Number of Instructions (variable)

Table 2142. Direct branch message example (2 MDO / 1 MSEO)

Clock	nex_mdo[1:0]		nex_mseo_b	State
0	X	X	1	Idle (or end of last message)
1	T1	T0	0	Start Message
2	T3	T2	0	Normal Transfer
3	T5	T4	0	Normal Transfer
4	S1	S0	0	Normal Transfer
5	S3	S2	0	Normal Transfer
6	I1	I0	1	End Packet
7	0	0	1	End Message

Table 2143. Direct branch message example (8 MDO / 2 MSEO)

Clock	nex_mdo[7:0]								nex_mseo_b[1:0]		State
0	X	X	X	X	X	X	X	X	1	1	Idle (or end of last message)
1	S1	S0	T5	T4	T3	T2	T1	T0	0	0	Start Message
2	0	0	0	0	I1	I0	S3	S2	1	1	End Packet/End Message
3	S1	S0	T5	T4	T3	T2	T1	T0	0	0	Start Message

Table 2144 illustrates an example Data Write Message with 8 MDO / 1 MSEO configuration, and *Table 2145* illustrates the same DWM with 8 MDO / 2 MSEO configuration

Note that T0, A0, D0 are the least significant bits where:

- Tx = TCODE number (fixed)
- Sx = Source processor (fixed)
- MAP = (always set to 0)
- Zx = Data size (fixed)
- Ax = Unique portion of the address (variable)
- Dx = Write data (variable 8-, 16- or 32-bit)

Table 2144. Data write message example (8 MDO / 1 MSEO)

Clock	nex_mdo[7:0]								nex_mseo_b	State
0	X	X	X	X	X	X	X	X	1	Idle (or end of last message)
1	S1	S0	T5	T4	T3	T2	T1	T0	0	Start Message
2	A0	Z3	Z2	Z1	Z0	0	S3	S2	1	End Packet
3	D7	D6	D5	D4	D3	D2	D1	D0	0	Normal Transfer
4	0	0	0	0	0	0	0	0	1	End Packet
5	0	0	0	0	0	0	0	0	1	End Message

Table 2145. Data write message example (8 MDO / 2 MSEO)

Clock	nex_mdo[7:0]								nex_mseo_b[1:0]		State
0	X	X	X	X	X	X	X	X	1	1	Idle (or end of last message)
1	S1	S0	T5	T4	T3	T2	T1	T0	0	0	Start Message
2	A0	Z3	Z2	Z1	Z0	0	S3	S2	0	1	End Packet
3	D7	D6	D5	D4	D3	D2	D1	D0	1	1	End Packet/ End Message

84.22 Electrical characteristics

For all electrical characteristics related to core processor and Nexus 3 operation, refer to the SPC58xEx/SPC58xGx Data Sheet.

84.23 IEEE 1149.1 (JTAG) RD/WR sequences

This section contains example JTAG/OnCE sequences used to access resources.

84.23.1 JTAG sequence for accessing internal Nexus registers

Table 2146. Accessing internal Nexus 3 registers via JTAG/OnCE

Step #	TMS Pin	Description
1	1	IDLE → SELECT-DR_SCAN
2	0	SELECT-DR_SCAN → CAPTURE-DR (Nexus Command Register value loaded in shifter)
3	0	CAPTURE-DR → SHIFT-DR
4	0	(7) TCK clocks issued to shift in direction (rd/wr) bit and first 6 bits of Nexus reg. addr.
5	1	SHIFT-DR → EXIT1-DR (7th bit of Nexus reg. shifted in)
6	1	EXIT1-DR → UPDATE-DR (Nexus shifter is transferred to Nexus Command Register)
7	1	UPDATE-DR → SELECT-DR_SCAN
8	0	SELECT-DR_SCAN → CAPTURE-DR (Register value is transferred to Nexus shifter)
9	0	CAPTURE-DR → SHIFT-DR
10	0	(31) TCK clocks issued to transfer register value to TDO pin while shifting in TDI value
11	1	SHIFT-DR → EXIT1-DR (MSB of value is shifted in/out of shifter)
12	1	EXIT1-DR → UPDATE -DR (if access is write, shifter is transferred to register)
13	0	UPDATE-DR → RUN-TEST/IDLE (transfer complete — Nexus controller to Reg. Select state)

84.23.2 JTAG sequence for read access of memory-mapped resources

Table 2147. Accessing memory-mapped resources (reads)

Step #	TCLK clocks	Description
1	13	Nexus Command = write to Read/Write Access Address Register (RWA)
2	37	Write RWA (initialize starting read address — data input on TDI)
3	13	Nexus Command = write to Read/Write Control/Status Register (RWCS)
4	37	Write RWCS (initialize read access mode and CNT value — data input on TDI)
5	—	Wait for falling edge of **nex_rdy_b** or **nex_err_b** pin
6	13	Nexus Command = read Read/Write Access Data Register (RWD)
7	37	Read RWD (data output on TDO)
8	—	If CNT > 0, go back to Step #5

84.23.3 JTAG sequence for write access of memory-mapped resources

Table 2148. Accessing memory-mapped resources (writes)

Step #	TCLK clocks	Description
1	13	Nexus Command = write to Read/Write Access Control/Status Register (RWCS)
2	37	Write RWCS (initialize write access mode and CNT value — data input on TDI)
3	13	Nexus Command = write to Read/Write Address Register (RWA)
4	37	Write RWA (initialize starting write address — data input on TDI)
5	13	Nexus Command = read Read/Write Access Data Register (RWD)
6	37	Write RWD (data output on TDO)
7	—	Wait for falling edge of **nex_rdy_b** or **nex_err_b** pin
8	—	If CNT > 0, go back to Step #5

85 Nexus Crossbar Multi-Master Client (NXMC)

85.1 Introduction

The Nexus Crossbar Multi-master Client (NXMC) module provides real-time trace capabilities in compliance with the IEEE-ISTO Nexus 5001-2012 standard. This module provides development support capabilities for devices without requiring address and data pins for internal visibility. A portion of the pin interface is also compliant with the IEEE 1149.1 JTAG standard. The IEEE-ISTO 5001 standard defines an extensible auxiliary port which is used in conjunction with the JTAG port.

Many bus masters such as DMA AHB, and CPU Data Bus AHB start accesses to internal address locations via the system bus. The NXMC module monitors the system bus and provides real-time trace information to debug or development tools.This also provides the capability of sending user defined messages for the peripherals attached to it.

85.1.1 Features

The NXMC module is compliant with the IEEE-ISTO 5001-2012 standard. The following features are implemented:

- Data trace via Data Write Messaging (DWM) and Data Read Messaging (DRM). This provides the capability for the development tool to trace reads or writes, or both, to (selected) internal memory resources
- Multi-master tracing capability for multiple master accesses
- HSM master(s) filtering from the tracing function based on external control
- Watchpoint messaging based on internal/external triggers, via the auxiliary pins
- Watchpoint trigger events out based on internal/external triggers
- Watchpoint trigger enable of data trace messaging
- Peripheral interface for data message transfers based on In Circuit Trace Message (ICTM) format
- Start or stop of Tracing (Data Trace, Watchpoint Trace) based on external triggers
- Optional timestamping of the messages based on timer value and controls received at the interface
- Nexus Auxiliary port for higher data throughput
  - One $\overline{\text{EVTO}}$ (Watchpoint Event) pin
  - One $\overline{\text{EVTI}}$ (Event In) pin
- Registers for data trace, watchpoint generation, and watchpoint trigger
- All features controllable and configurable via the JTAG port

85.2 External signal description

This section details information regarding the standard ports and protocol. The NXMC pin interface provides the function of transmitting messages from the messages queues to NAR. It is also responsible for handshaking with the message queues.

The NXMC module implements the following signals:

- One $\overline{\text{EVTI}}$
- One $\overline{\text{EVTO}}$

These pins are described in *Table 2150*.

All NXMC input functionality is controlled through the JTAG port in compliance with IEEE 1149.1 (refer to *Section 85.5.1: IEEE 1149.1 (JTAG) Test Access Port* for details). A separate JTAG TAP controller is instantiated within the NXMC. *Table 2149* describes the JTAG pins.

Table 2149. JTAG pins for NXMC

JTAG port	Input/ output	Description
TDO	O	The Test Data Output (TDO) pin is the serial output for test instructions and data. TDO is three-stateable and is actively driven in the shift-IR and shift-DR controller states. TDO changes on the falling edge of TCK.
TDI	I	The Test Data Input (TDI) pin receives serial test instruction and data. TDI is sampled on the rising edge of TCK and has an internal pull-up resistor.
TMS	I	The Test Mode Select (TMS) input pin is used to sequence the JTAG test controllers' state machine. TMS is sampled on the rising edge of TCK and has an internal pull-up resistor.
TCK	I	The Test Clock (TCK) input pin is used to synchronize the test logic, and control register access through the JTAG port.
TRST	I	The Test Reset ($\overline{\text{TRST}}$) input pin is used to asynchronously initialize the JTAG controller. The $\overline{\text{TRST}}$ pin has an internal pull-down resistor.

Table 2150. NXMC **auxiliary pins**

Auxiliary port	Input/ output	Description
EVTO	O	Event Out ($\overline{\text{EVTO}}$) is an output which, when asserted, indicates one of two events has occurred based on the EOC bits in the DC Register: – one of the internal or external watchpoints has occurred and EOC = 2'b00 – system-level debug mode was entered (ipg_debug) and EOC = 2'b01 $\overline{\text{EVTO}}$ is held asserted for one cycle.
EVTI	I	Event In ($\overline{\text{EVTI}}$) is an input which initiates synchronization messages. If the Nexus module is enabled at reset, (refer to *Section 85.5.2: Enabling NXMC operation*), assertion initiates data trace synchronization messages (provided data trace is enabled).

85.3 Supported messages and TCODEs

The NXMC pins allow for flexible transfer operations via public messages. A TCODE defines the transfer format, the number or size of the packets, or both, to be transferred, and the purpose of each packet. The IEEE-ISTO 5001-2012 standard defines a set of public messages. The NXMC block supports the public TCODEs listed in *Table 2151*. The SRC

and MASTER field values are described in the device-specific chapter that describes how the modules are configured.

Table 2151. Supported public TCODEs

Message name	Min packet size (bits)	Max packet size (bits)	Packet name	Packet type	Packet description
Data Trace - Data Write Message	6	6	TCODE	fixed	TCODE number = 58
	4	4	SRC	fixed	Source processor identifier (multiple Nexus configuration)
	4	4	MASTER	fixed	Indicates which master (ID) initiated the data access
	3	3	DSZ	fixed	Data size (refer to *Table 2152*)
	1	32	U-ADDR	variable	Unique portion of the data write address
	8	64	DATA	fixed	Data write value (size fixed based on DSZ field)
	1	26	TSTAMP	variable	Timestamp (Optional)
Data Trace - Data Read Message	6	6	TCODE	fixed	TCODE number = 59
	4	4	SRC	fixed	Source processor identifier (multiple Nexus configuration)
	4	4	MASTER	fixed	Indicates which master (ID) initiated the data access
	3	3	DSZ	fixed	Data size (refer to *Table 2152*)
	1	32	U-ADDR	variable	Unique portion of the data read address
	8	64	DATA	fixed	Data read value (size fixed based on DSZ field)
	1	26	TSTAMP	variable	Timestamp (Optional)
Error Message	6	6	TCODE	fixed	TCODE number = 8
	4	4	SRC	fixed	Source processor identifier (multiple Nexus configuration)
	4	4	ETYPE	fixed	Error type (refer to *Table 2153*)
	14	14	ECODE	fixed	Error code (refer to *Table 2154*)
	1	26	TSTAMP	variable	Timestamp (Optional)
Data Trace - Data Write Message with Sync	6	6	TCODE	fixed	TCODE number = 60
	4	4	SRC	fixed	Source processor identifier (multiple Nexus configuration)
	4	4	MASTER	fixed	Indicates which master (ID) initiated the data access
	3	3	DSZ	fixed	Data size (refer to *Table 2152*)
	1	32	F-ADDR	variable	Full access address (leading zero (0) truncated)
	8	64	DATA	fixed	Data write value (size fixed based on DSZ field)
	1	26	TSTAMP	variable	Timestamp (Optional)

Table 2151. Supported public TCODEs (continued)

Message name	Min packet size (bits)	Max packet size (bits)	Packet name	Packet type	Packet description
Data Trace - Data Read Message with Sync	6	6	TCODE	fixed	TCODE number = 61
	4	4	SRC	fixed	Source processor identifier (multiple Nexus configuration)
	4	4	MASTER	fixed	Indicates which master (ID) initiated the data access
	3	3	DSZ	fixed	Data size (refer to *Table 2152*)
	1	32	F-ADDR	variable	Full access address (leading zero (0) truncated)
	8	64	DATA	fixed	Data read value (size fixed based on DSZ field)
	1	26	TSTAMP	variable	Timestamp (Optional)
Watchpoint Message	6	6	TCODE	fixed	TCODE number = 15
	4	4	SRC	fixed	Source processor identifier (multiple Nexus configuration)
	1	8	WPHIT	variable	Watchpoint source(s) number (refer to *Table 2166*)
	1	26	TSTAMP	variable	Timestamp (Optional)
Peripheral Message (In circuit Trace Message: ICTM)	6	6	TCODE	fixed	TCODE number = 34
	4	4	SRC	fixed	Source processor identifier (multiple Nexus configuration)
	6	6	CKSRC	fixed	Peripheral source identifier
	1	44	CKDATA	variable	Peripheral data
	1	26	TSTAMP	variable	Timestamp (Optional)

Table 2152. Data trace size (DSZ) encodings (TCODE = 5,6,13,14)

DSZ	Transfer size
001	8-bit
010	16-bit
011	32-bit
100	64-bit
000, 101-111	Reserved

Table 2153. Error Type (ETYPE) encodings (applicable to all error messages)

ETYPE	Description
0000	Queue overrun caused message (one or more) to be lost
0001	Contention with higher priority message caused message(s) to be lost
0010	Reserved

Table 2153. Error Type (ETYPE) encodings (applicable to all error messages)

ETYPE	Description
0011	Reserved
0100	Reserved
0101	Invalid access opcode (Nexus register unimplemented)
0110–1111	Reserved

Table 2154. Error Code (ECODE) encodings (applicable based on error source)

ECODE		Description[1]
13 to 8	7 to 0	
Source (SRC)		
000000	XXXXXXX1	Watchpoint Message(s) lost
000000	XXXXXX1X	Data Trace Message(s) lost
000000	XXXXX1XX	Reserved
000000	XXXX1XXX	Reserved
000000	XXX1XXXX	Reserved
000000	XX1XXXXX	Reserved
000000	X1XXXXXX	Reserved
000000	1XXXXXXX	Reserved
Source (SRC) n = peripheral index		
CKSRCn	XXXXXXX1	Reserved
CKSRCn	XXXXXX1X	Reserved
CKSRCn	XXXXX1XX	Reserved
CKSRCn	XXXX1XXX	Reserved
CKSRCn	XXX1XXXX	Reserved
CKSRCn	XX1XXXXX	Reserved
CKSRCn	X1XXXXXX	In-Circuit Trace Message(s) Lost
CKSRCn	1XXXXXXX	Reserved

1. The SRC field values are described in the device-specific chapter that describes how the modules are configured.

85.4 Register description

This section describes the NXMC register definition. Nexus registers are accessed using the JTAG port in compliance with IEEE 1149.1. Refer to *Section 85.5.1: IEEE 1149.1 (JTAG) Test Access Port* for details. The complete list of registers is shown in *Table 2155*.

Table 2155. NXMC registers

Nexus register	Nexus access opcode	Read/ write	Read address	Write address
Development Control 1 (DC1)	0x2	R/W	0x04	0x05
Development Control 2 (DC2)	0x3	R/W	0x06	0x07
Watchpoint Trigger (WT)	0xB	R/W	0x16	0x17
Data Trace Control (DTC)	0xD	R/W	0x1A	0x1B
Data Trace Start Address1 (DTSA1)	0xE	R/W	0x1C	0x1D
Data Trace Start Address2 (DTSA2)	0xF	R/W	0x1E	0x1F
Data Trace End Address1 (DTEA1)	0x12	R/W	0x24	0x25
Data Trace End Address2 (DTEA2)	0x13	R/W	0x26	0x27
Breakpoint/Watchpoint Control Register1 (BWC1)	0x16	R/W	0x2C	0x2D
Breakpoint/Watchpoint Control Register2 (BWC2)	0x17	R/W	0x2E	0x2F
Breakpoint/Watchpoint Address Register1 (BWA1)	0x1E	R/W	0x3C	0x3D
Breakpoint/Watchpoint Address Register2 (BWA2)	0x1F	R/W	0x3E	0x3F

85.4.1 Register definitions

Detailed register definitions for the current NXMC implementation are described in the following sections.

85.4.1.1 Development control register 1 (DC1)

The Development Control Registers control the basic development features of the NXMC module. *Figure 2257* shows the format of the DC1.

Address: refer to *Table 2155* Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0(1)	0(1)	0(1)	EOC		0	0	WEN	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	EIC		TM		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. This bit is not implemented and its value has no impact on the system.

Figure 2257. Development control register 1 (DC1)

The DC1 is described in *Table 2156*.

Table 2156. DC1 register field description

Field	Description
3:4 EOC	$\overline{\text{EVTO}}$ Control. 00 $\overline{\text{EVTO}}$ upon occurrence of watchpoint (internal or external) 01 $\overline{\text{EVTO}}$ upon entry into system-level Debug Mode 1x Reserved
7 WEN	Watchpoint Trace Enable. 0 Watchpoint Messaging disabled 1 Watchpoint Messaging enabled
27:28 EIC	$\overline{\text{EVTI}}$ Control 00 $\overline{\text{EVTI}}$ for synchronization (Data Trace) 01 Reserved 10 $\overline{\text{EVTI}}$ disabled 11 Reserved
29:31 TM	Trace Mode 000 No Trace 1xx Reserved x1x Data Trace enabled xx1 Reserved

85.4.1.2 Development control register 2 (DC2)

The Development Control Registers control the basic development features of the NXMC module. *Figure 2258* shows the format of the DC2.

Address: refer to *Table 2155* Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	EWC								0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	PME0(1)
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. This bit is not implemented and its value has no impact on the system.

Figure 2258. Development control register 2 (DC2)

The DC2 is described in *Table 2157*.

Table 2157. DC2 register field description

Field	Description
0:7 EWC	$\overline{\text{EVTO}}$ Watchpoint Configuration(1). 00000000 No Watchpoints trigger $\overline{\text{EVTO}}$ 1XXXXXXX External Watchpoint #1 triggers $\overline{\text{EVTO}}$ X1XXXXXX External Watchpoint #2 triggers $\overline{\text{EVTO}}$ XX1XXXXX Reserved(2) XXX1XXXX Reserved(2) XXXX1XXX Internal Watchpoint #1 triggers $\overline{\text{EVTO}}$ XXXXX1XX Internal Watchpoint #2 triggers $\overline{\text{EVTO}}$ XXXXXX1X Reserved(2) XXXXXXX1 Reserved(2)
31 PME0	Peripheral Messaging Enable for the In-Circuit Trace (ICT) feature.(3) 0 Peripheral Interface disabled 1 Peripheral Interface enabled

1. The EOC bits in DC1 must be programmed to trigger EVTO on watchpoint occurrence for the EWC bits to have any effect.
2. Writing bits 2,3,6,7 has no impact and read value will be 0 for these bits.
3. The peripheral details are described in the device-specific chapter that describes how the modules are configured.

85.4.1.3 Watchpoint trigger register (WT)

The Watchpoint Trigger Register allows the watchpoints defined either internally to the NXMC module or by an external module to trigger actions. These watchpoints can control data trace enable and disable. The WT bits can be used to produce an address related window for triggering trace messages. *Figure 2259* shows the format of the WT register.

Address: refer to *Table 2155* Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	DTS		DTE				0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

Figure 2259. Watchpoint trigger register (WT)

The WT register is described in *Table 2158*.

Table 2158. WT register field description

Field	Description(1)
6;8 DTS	Data Trace Start Control. 000 Trigger disabled 001 Use External Watchpoint #1 010 Use External Watchpoint #2 011 Reserved 100 Reserved 101 Use Internal Watchpoint #1 (BWA1 Register) 110 Use Internal Watchpoint #2 (BWA2 Register) 111 Reserved
9;11 DTE	Data Trace End Control. 000Trigger disabled 001 Use External Watchpoint #1 010 Use External Watchpoint #2 011 Reserved 100 Reserved 101 Use Internal Watchpoint #1 (BWA1 Register) 110 Use Internal Watchpoint #2 (BWA2 Register) 111 Reserved

1. The WT bits only enable data trace if the TM bits in the Development Control Register (DC) have not already been set to enable data trace.

85.4.1.4 Data trace control register (DTC)

The Data Trace Control Register controls whether DTM messages are restricted to reads, writes or both for a user programmable address range. There are two data trace channels controlled by the DTC for the NXMC module. *Figure 2260* shows the format of the DTC register.

Address: refer to *Table 2155* Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	RWT1		RWT2		0	0	0	MME	0	0			MID			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	RC1	RC2	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 2260. Data trace control (DTC) register

The DTC register is described in *Table 2159*.

Table 2159. DTC field description

Field	Description
0:1 RWT1	Read/Write Trace 1. 00 No trace messages generated x1 Enable data read trace 1x nable data write trace
2:3 RWT2	Read/Write Trace 2. 00 No trace messages generated x1 Enable data read trace 1x Enable data write trace
7 MME	Multi Master Tracing enable. 0 Tracing enabled only for Master with ID = MID (DTC[19:16]) 1 Tracing enabled for all the Masters
12:15 MID	ID of the system bus Master to be traced(1)
24 RC1	Range Control 1. 0 Condition trace on address within range (endpoints inclusive) 1 Condition trace on address outside of range (endpoints exclusive)
25 RC2	Range Control 2. 0 Condition trace on address within range (endpoints inclusive) 1 Condition trace on address outside of range (endpoints exclusive)

1. The MASTER field values are described in the device-specific chapter that describes how the modules are configured.

85.4.1.5 Data trace start address registers 1 and 2 (DTSA1 and DTSA2)

The Data Trace Start Address Registers define the start addresses for each trace channel. *Figure 2261* shows the format of the DTSA1 and DTSA2 registers.

Address: refer to *Table 2155* Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	DATA TRACE START ADDRESS															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	DATA TRACE START ADDRESS															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 2261. Data trace start address registers 1 and 2 (DTSA1 and DTSA2)

85.4.1.6 Data trace end address registers 1 and 2 (DTEA1 and DTEA2)

The Data Trace End Address Registers define the end addresses for each Trace channel. *Figure 2262* shows the format of the DTEA1 and DTEA2 registers.

Address: refer to *Table 2155* Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	DATA TRACE END ADDRESS															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	DATA TRACE END ADDRESS															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 2262. Data trace end address registers 1 and 2 (DTEA1 and DTEA2)

Table 2160 illustrates the ranges that are selected for data trace for various cases of DTSA being less than, greater than, or equal to DTEA.

Table 2160. Data trace address range options

Programmed values	Range control bit value	Range selected
DTSA ≤ DTEA	0	DTSA -> <- DTEA
DTSA ≤ DTEA	1	<- DTSA DTEA ->
DTSA > DTEA	N/A	Invalid range – no trace

Note: DTSA must be less than (or equal to) DTEA in order to guarantee correct data write/read traces.

When the range control bit is 0 (internal range), accesses to DTSA and DTEA addresses are traced. When the range control bit is 1 (external range), accesses to DTSA and DTEA are not traced.

85.4.1.7 Breakpoint/watchpoint control register 1 (BWC1)

Breakpoint/watchpoint control register 1 controls the attributes for generation of NXMC watchpoint#1. *Figure 2263* shows the format of the BWC1 register.

Address: refer to *Table 2155* Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	BWE1		BRW1		0	0	0	0	0	0	0	0	0	0	BWR1	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	BWT1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 2263. Breakpoint/watchpoint control register 1(BWC1)

The BWC1 register is described in *Table 2161*.

Table 2161. BWC1 field description

Field	Description
0:1 BWE1	Breakpoint/Watchpoint #1 Enable 00 Internal Nexus Watchpoint #1 disabled 01 Reserved 10 Reserved 11 Internal Nexus Watchpoint #1 enabled
2:3 BRW1	Breakpoint/Watchpoint #1 Read/Write Select 00 Watchpoint #1 hit on read accesses 01 Watchpoint #1 hit on write accesses 10 Watchpoint #1 on read or write accesses 11 Reserved
14:15 BWR1	Breakpoint/Watchpoint #1 Register Compare 00 No Register Compare (same as BWC1[31:30] = 2'b00) 01 Reserved 10 Compare with BWA1 value 11 Reserved
16 BWT1	Breakpoint/Watchpoint #1 Type 0 Reserved 1 Watchpoint #1 on data accesses

85.4.1.8 Breakpoint/watchpoint control register 2 (BWC2)

Breakpoint/Watchpoint Control Register2 controls attributes for generation of NXMC watchpoint#2. *Figure 2264* shows the format of the BWC2 register.

Address: refer to *Table 2155* Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	BWE2		BRW2		0	0	0	0	0	0	0	0	0	0	BWR2	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	BWT2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 2264. Breakpoint/watchpoint control register 2(BWC2)

The BWC2 register is described in *Table 2162*.

Table 2162. BWC2 field description

Field	Description
0:1 BWE2	Breakpoint/Watchpoint #2 Enable 00 Internal Nexus Watchpoint #2 disabled 01 Reserved 10 Reserved 11 Internal Nexus Watchpoint #2 enabled
2:3 BRW2	Breakpoint/Watchpoint #2 Read/Write Select 00 Watchpoint #2 hit on read accesses 01 Watchpoint #2 hit on write accesses 10 Watchpoint #2 on read or write accesses 11 Reserved
14:15 BWR2	Breakpoint/Watchpoint #2 Register Compare. 00 No Register Compare (same as BWC2[31:30] = 2b00) 01 Reserved 10 Compare with BWA2 value 11 Reserved
16 BWT2	Breakpoint/Watchpoint #2 Type 0 Reserved 1 Watchpoint #2 on data accesses

85.4.1.9 Breakpoint/watchpoint address registers 1 and 2 (BWA1 and BWA2)

The Breakpoint/Watchpoint Address Registers are compared with system bus addresses in order to generate internal watchpoints. *Figure 2265* shows the format of the BWA1 and BWA2 registers.

Address: refer to *Table 2155* Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	BREAKPOINT / WATCHPOINT ADDRESS															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	BREAKPOINT / WATCHPOINT ADDRESS															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 2265. Breakpoint /watchpoint address registers 1 and 2 (BWA1 and BWA2)

85.4.1.10 Unimplemented registers

Unimplemented registers are those with client select and index value combinations other than those listed in *Table 2155*. For unimplemented registers, the NXMC module drives TDO to zero during the SHIFT-DR state. It also transmits an error message with the invalid access opcode encoding (ETYPE = 0101).

85.5 Programming considerations (RESET)

If NXMC register configuration is to occur during system reset (as opposed to debug mode), all NXMC configuration should be completed between the negation of TRST and system reset de-assertion, after the JTAG ID register has been read by the tool after negation of TRST and should be programmed before deassertion of system reset.

85.5.1 IEEE 1149.1 (JTAG) Test Access Port

The NXMC module uses the IEEE 1149.1 TAP controller for accessing Nexus resources. The JTAG signals themselves are shared by all TAP controllers on the device. The NXMC module implements a 4-bit Instruction Register (IR). The valid instructions and method for register access are outlined in *Section 85.5.4: NXMC register access via JTAG*.

85.5.2 Enabling NXMC operation

The Nexus module is enabled by loading a single instruction into the JTAG Instruction Register (IR). Once enabled, the module is ready to accept control input via the JTAG pins.

The Nexus module is disabled when the JTAG state machine reaches the Test-Logic-Reset state. This state can be reached by the assertion of the TRST pin or by cycling through the state machine using the TMS pin. The Nexus module is also disabled if a Power-on-Reset (POR) event occurs.

If the NXMC module is disabled, no trace output is provided. Nexus registers are not available for reads or writes.

85.5.3 Enabling NXMC TAP controller

Assertion of a Power-on-Reset signal or assertion of the TRST pin resets all TAP controllers. Upon exit from the Test-Logic-Reset state, the IR value is loaded with the JTAG ID. When the NXMC TAP is accessed, this information helps the development tool obtain information about the Nexus module it is accessing, such as version, sequence, feature set.

85.5.4 NXMC register access via JTAG

Access to Nexus register resources is enabled by loading a single instruction into the JTAG Instruction Register (IR). Once the JTAG NEXUS-ACCESS instruction has been loaded, the JTAG port allows tool/target communications with all Nexus registers according to the map in *Table 2155*.

Reading/writing of a Nexus register then requires two passes through the Data-Scan (DR) path of the JTAG state machine.

1. The first pass through the DR selects the Nexus register to be accessed by providing an index (refer to *Table 2155*), and the direction (read/write). This is achieved by loading an 8-bit value into the JTAG Data Register (DR). This register has the format shown in *Figure 2266* and *Table 2163*.



Figure 2266. JTAG DR for Nexus register access

Table 2163. JTAG DR field description for Nexus register access

Field	Description
Nexus Register Index	Selected from values in *Table 2155*
Read/Write (R/W)	0 Read 1 Write

2. The second pass through the DR then shifts the data in or out of the JTAG port, LSB first.
   a) During a read access, data is latched from the selected Nexus register when the JTAG state machine passes through the Capture-DR state.
   b) During a write access, data is latched into the selected Nexus register when the JTAG state machine passes through the Update-DR state.

85.6 Functional description

85.6.1 Data trace

This section describes the data trace mechanism supported by the NXMC module. Data trace is implemented via Data Write Messaging (DWM) and Data Read Messaging (DRM), as per the IEEE-ISTO 5001-2012 standard.

85.6.1.1 Data trace messaging (DTM)

NXMC data trace messaging is accomplished by snooping the system bus and storing the information for qualifying accesses (based on enabled features and matching target addresses). The NXMC module traces all data access that meet the selected range and attributes.

85.6.1.2 DTM message formats

The NXMC block supports five types of DTM messages:

- Data write
- Data read
- Data write synchronization
- Data read synchronization
- Error messages

85.6.1.2.1 Data write and data read messages

The data write and data read messages contain the data write/read value and the address of the write/read access, relative to the previous data trace message. The DATA field is considered fixed based on the DSZ encodings to either 8, 16, 32, 64. The TSTAMP field is 26 bits. Data write message and data read message information is messaged out in the format shown in *Figure 2267*. The SRC field values are described in the device-specific chapter that describes how the modules are configured.

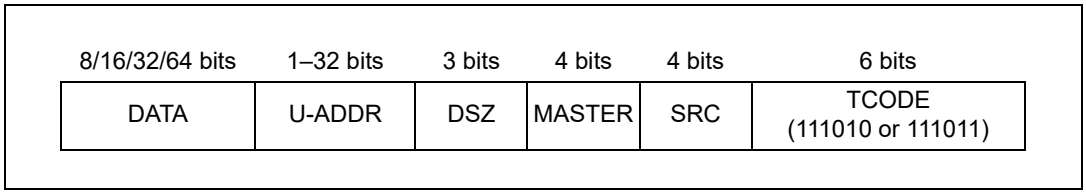


Figure 2267. DTM message format for DATA width = 64 bits

TSTAMP is another optional field which is inserted based on the controls received from the NAR, refer to *Section 85.6.4: Timestamping feature*.

85.6.1.2.2 Data trace with non-contiguous byte strobes asserted

The v6 extensions to AMBA 2.0 allow system bus transfers with bytes non-asserted byte strobes in between asserted byte strobes. The NXMC does not support transfers with non-contiguous byte strobes asserted. In such a condition, the invalid bytes in the DATA packet are driven to the value of 0xFF.

85.6.1.2.3 DTM overflow error messages

An error message occurs when a new message cannot be queued due to the message queue being full. The FIFO discards incoming messages until it has completely emptied the queue. Once emptied, an error message is queued. The error encoding indicates which type(s) of messages attempted to be queued while the FIFO was being emptied.

If only a data trace message attempts to enter the queue while it is being emptied, the error message incorporates the data trace only error encoding (ECODE = 00000000000010). If a watchpoint also attempts to be queued while the FIFO is being emptied, then the error message incorporates error encoding (ECODE = 00000000000011).

Error information is messaged out in the format shown in *Figure 2268*. The SRC field values are described in the device-specific chapter that describes how the modules are configured.

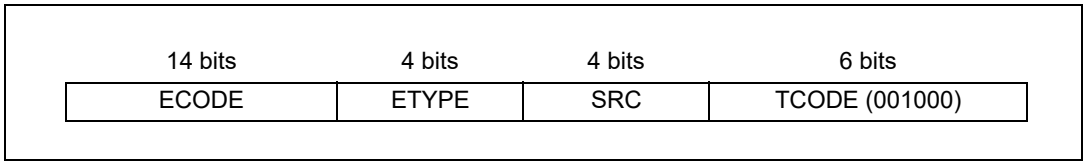


Figure 2268. DTM error message format (fixed length = 48 bits)

85.6.1.2.3.1 Data trace synchronization messages

A data trace write/read with synchronization message is messaged via the auxiliary port (provided data trace is enabled) for the following conditions (refer to *Table 2164*):

- Initial data trace message upon exit from system reset or whenever data trace is enabled is a synchronization message
- Upon returning from a low power state, the first data trace message is a synchronization message
- Upon returning from debug mode, the first data trace message is a synchronization message
- After occurrence of queue overrun (can be caused by any trace message), the first data trace message is a synchronization message
- After the periodic data trace counter has expired indicating 255 without-sync data trace messages have occurred since the last with-sync message occurred
- Upon assertion of the event in (EVTI) pin, the first data trace message is a synchronization message if the EIC bits of the DC Register have enabled this feature
- Upon data trace write/read after the previous DTM Message was lost due to a collision while entering the FIFO between the DTM message and any of the following: error message, or watchpoint message or peripheral message.This is a very rare occurrence and possible only in case of continuous burst on the system bus as well as watchpoint events which is unrealistic

Data trace synchronization messages provide the full address (without leading zeros) and ensure that development tools fully synchronize with data trace regularly. Synchronization messages provide a reference address for subsequent DTMs, in which only the unique portion of the data trace address is transmitted. The format for data trace write/read with synchronization messages is shown in *Figure 2269*. The SRC and MASTER field values are described in the device-specific chapter that describes how the modules are configured.

8-64 bits	1–32 bits	3 bits	4 bits	4 bits	6 bits
DATA	F-ADDR	DSZ	MASTER	SRC	TCODE (111100 or 111101)

Figure 2269. Data write/read with synchronization message format

Exception conditions that result in data trace synchronization are summarized in *Table 2164*.

Table 2164. Data trace exception summary

Exception condition	Exception handling
System Reset Negation	At the negation of JTAG reset ($\overline{\text{TRST}}$), queue pointers, counters, state machines, and registers within the NXMC module are reset. If data trace is enabled, the first data trace message is a data write/read with synchronization message.
Data Trace Enabled	The first data trace message (after data trace has been enabled) is a synchronization message.
Exit from Low Power/Debug	Upon exit from a low power mode or debug mode the next data trace message is converted to a data write/read with synchronization message.

Table 2164. Data trace exception summary (continued)

Exception condition	Exception handling
Queue Overrun	An error message occurs when a new message cannot be queued due to the message queue being full. The FIFO discards messages until it has completely emptied the queue. Once emptied, an error message is queued. The error encoding indicates which type(s) of messages attempted to be queued while the FIFO was being emptied. The next DTM message in the queue is a data write/read with synchronization message.
Periodic Data Trace Synchronization	A forced synchronization occurs periodically after 255 data trace messages have been queued. A data write/read with synchronization message is queued. The periodic data trace message counter then resets.
Event In	If the Nexus module is enabled, an $\overline{\text{EVTI}}$ assertion initiates a data write/read with synchronization message upon the next data write/read (if data trace is enabled and the EIC bits of the DC Register have enabled this feature).
Collision Priority	All messages have the following priority: Error > WPM > ICTM > DTM. Although a very rare occurrence, during simultaneous burst lower priority messages can be lost due to burst of high priority messages. Proper error messages are generated and in case of DTM lost a subsequent read/write queues a data write/read with synchronization message.

85.6.1.3 DTM operation

85.6.1.3.1 Enabling data trace messaging

Data trace messaging can be enabled in one of three ways:

- Setting the TM field of the DC Register via JTAG to enable data trace (DC[1]).
- Using the DTS field of the WT Register to enable data trace on watchpoint hits (either watchpoints configured within the NXMC module or external watchpoint inputs controlled by the device).
- By toggling the external start control, which in turn sets the TM field of the DC register to enable data trace. The TM bit clears and hence the data tracing stops on the toggle of external stop control. In case of conflict between JTAG write and external start/stop control, JTAG write takes precedence.

Note: When enabling the DTM, either via JTAG or external start/stop control, it is important to make sure that all the other controls (that is DTC, DTSA/E1/2 and WT register fields) are correctly configured to ensure the correct DTM functionality.

85.6.1.3.2 DTM queueing

The NXMC implements a programmable depth queue for queuing all messages. Messages that enter the queue are transmitted via the auxiliary pins in the order in which they are queued.

Note: If multiple trace messages need to be queued at the same time, watchpoint messages have the highest priority (WPM > ICTM > DTM).

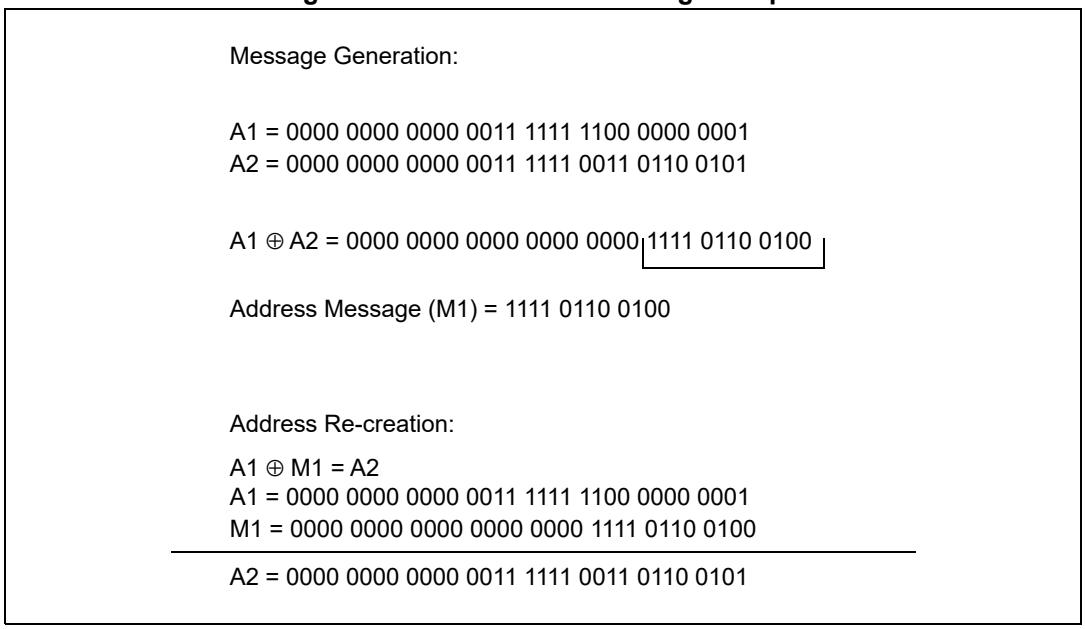
85.6.1.3.3 Relative addressing

The relative address feature is compliant with IEEE-ISTO 5001-2012 and is designed to reduce the number of bits transmitted for addresses of data trace messages. The address

transmitted is relative to the address of the previous data trace message. It is generated by XORing the new address with the previous address, and then using only the results up to the most significant 1 in the result. To recreate this address, an XOR of the (most-significant 0-padded) message address with the previously decoded address gives the current address.

For example, if the previous address (A1) = 0x0003FC01, the new address (A2) = 0x0003F365 as shown in *Figure 2270*.

Figure 2270. Relative addressing example



85.6.1.3.4 Data trace windowing

Data write/read messages are enabled via the RWT1(2) field in the Data Trace Control Register (DTC) for each DTM channel. Data trace windowing is achieved via the address range defined by the DTEA and DTSA Registers and by the RC1(2) field in the DTC. All system bus DMA initiated read/write accesses which fall inside or outside these address ranges, as programmed, are candidates to be traced.

85.6.1.3.5 System bus cycle special cases

The system bus cycle special cases are shown in *Table 2165*.

Table 2165. System Bus cycle special cases

Special case	Action
System bus cycle with data error	Data trace message discarded
System bus cycle completed without error	Cycle captured and transmitted
System bus cycle is an instruction fetch	Cycle ignored

85.6.1.3.6 System bus multi-master tracing

DTM tracing can be done either for all the masters or for a single master (indicated by the MID field of the DTC register) based on the DTC register MME field. In case of single master, the correct value needs to be configured in the MID register before enabling the data trace operation.

The MASTER field of the DTM message (refer to figure in *Section 85.6.1.2.1: Data write and data read messages* for an example) indicates the system bus master for which the trace has been generated. For each masters (0 to 15) there is a parameter associated (MSID0-15) for indicating its actual ID, which finally gets inserted in the MASTER field. The MASTER field values are described in the device-specific chapter that describes how the modules are configured.

85.6.1.3.7 HSM master filtering

For security reasons certain HSM masters need to be disabled or filtered from data tracing operations.

85.6.2 Watchpoint support

The NXMC module provides watchpoint messaging via the auxiliary pins, as defined by IEEE-ISTO 5001-2012. Watchpoint messages can be generated by either using the NXMC defined internal watchpoints or by externally defined watchpoint signals.

85.6.2.1 Watchpoint messaging

Enabling watchpoint messaging is accomplished by setting the WEN bit in the DC Register. This bit is set either via JTAG write 1 access or external start indication, and cleared via JTAG write 0 access or external start indication. In case of conflict between JTAG write and external start/stop control, JTAG write takes precedence.

Note: In case of enabling the watchpoint messaging, either via JTAG or external start/stop control, it is important to make sure that all the other Watch Point Message (WPM) controls (that are BWC1/2, BWA1/2 register fields) are configured with the correct values in order to ensure correct WPM operation.

Watchpoint setting is supported through two methods:

- Internal watchpoints: Using the BWC1(2) registers, two independently controlled internal watchpoints can be initialized. When a system bus access address matches on BWA1(2), a watchpoint message is transmitted. The system bus multi-master or single master controls applies here too as explained for DTM (in *Section 85.6.1.3.6: System bus multi-master tracing*).
- External watchpoints: The NXMC module supports two external watchpoint inputs. The optional logic to generate these watchpoint signals is implemented at the device level.The watchpoints are described in the device-specific chapter that describes how the modules are configured. If either of these signals asserts (that is on 0-to-1 transition), a watchpoint message is transmitted.

The Nexus module provides watchpoint messaging using the IEEE-ISTO 5001 defined TCODE. When any of the four possible watchpoint sources asserts, a message is sent to the queue to be messaged out. This message indicates the watchpoint number as shown in *Figure 2271* and *Table 2166*.

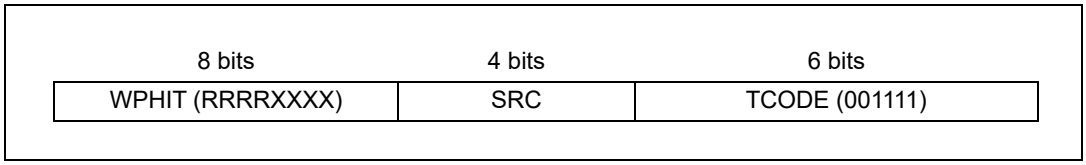


Figure 2271. Watchpoint message format (fixed length = 18 bits)

Table 2166. Watchpoint source description

Watchpoint source (8-bits)	Watchpoint description
RRRRXXX1(1)	External Watchpoint #1(device defined)
RRRRXX1X	External Watchpoint #2 (device defined)
RRRRX1XX	Internal Watchpoint #1 (BWA1 match)
RRRR1XXX	Internal Watchpoint #2 (BWA2 match)

1. R = reserved for future use (insert 0s).

85.6.2.2 Watchpoint error message

An error message occurs when a new message cannot be queued due to the message queue being full. The FIFO discards messages until it has completely emptied the queue. Once emptied, an error message is queued. The error encoding indicates which type(s) of messages attempted to be queued while the FIFO was being emptied.

If only a watchpoint message attempts to enter the queue while it is being emptied, the error message incorporates the watchpoint only error encoding (ECODE = 00000000000001). If a data trace message also attempts to enter the queue while it is being emptied, the error message incorporates error encoding (00000000000011). Error information is messaged out in the format shown in *Figure 2272*.

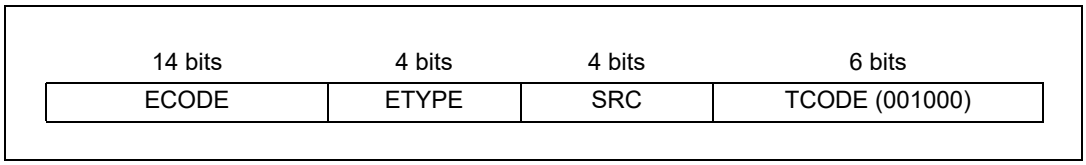


Figure 2272. Error message format (fixed length = 28 bits)

85.6.3 Peripheral interface

The NXMC provides an asynchronous interface to peripherals for sending data messages. The NXMC can back pressure or delay the data transfer using a data accept signal. A round-robin scheme is followed between the peripherals for forwarding received data to the queue; therefore, each peripheral has an equal priority and hence bandwidth for data transfers.

To enable messaging from peripherals there are controls present in the DC2 register for individual peripherals: DC2[PME0]. There are no message generated and hence data accepted from a peripheral unless the corresponding enable bit is set to 1.

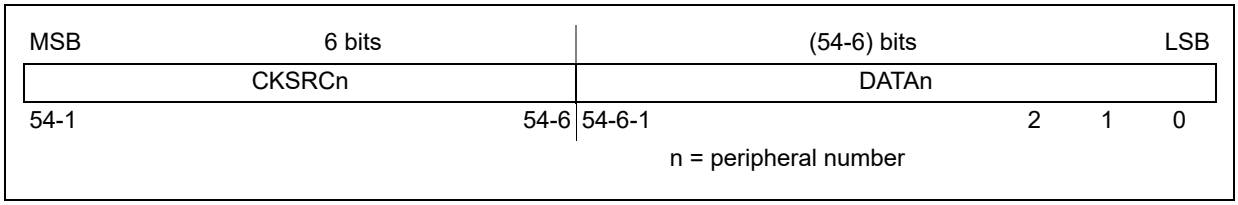


Figure 2273. Peripheral DATA bus fields

Another input port is present which indicates data loss inside the peripheral itself due to over-run, the module in turn generates error message to communicate such events to the debugger.

85.6.3.1 Peripheral data message format (ICTM)

The message formats for the peripheral data messages (formatted as ICTM) is shown in *Figure 2274*. The SRC field values are described in the device-specific chapter that describes how the modules are configured. The CKSRC field is received from the MSB 6 bits of the individual peripheral data bus, and CKDATA is a variable field and contains 44 LSBits received on the databus of the peripheral interface.

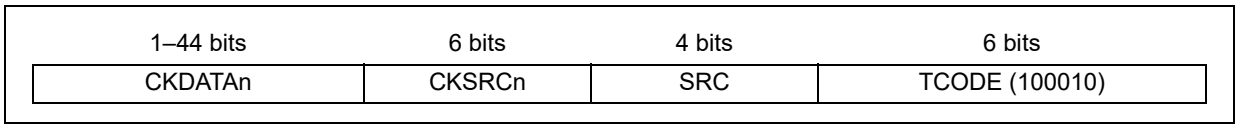


Figure 2274. Peripheral message format (max length = 60 bits, min length = 17 bits)

85.6.3.2 Peripheral error message format

The error messages are generated based data over-run indication from a peripheral. Error information is messaged out in the format shown in *Figure 2275*.

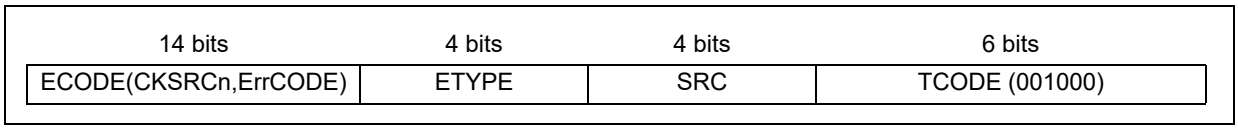


Figure 2275. Error message format (fixed length = 28 bits)

The SRC field values are described in the device-specific chapter that describes how the modules are configured. ETYPE and ECODE encoding are listed in *Table 2153* and *Table 2154*.

85.6.4 Timestamping feature

This feature allows a timestamp packet to be appended to all messages, based on the timer value and controls received externally from the NAR. If enabled, this packet is added as the last field to all the outgoing messages. The TSTAMP packet width is 26 bits.

The TSTAMP packet is appended at the end of a message during message formatting, before transfer. *Figure 2276* shows an example of ICTM message with a TSTAMP field.

26 bits	1–44 bits	6 bits	4 bits	6 bits
TSTAMP	CKDATAn	CKSRCn	SRC	TCODE (100010)

Figure 2276. Message with TSTAMP field example

The enable or disable indication should remain static for the complete debug session. Dynamically enabling or disabling of the timestamping is not allowed during running debug tracing. The following steps allow seamless enabling/disabling of the timestamping:

1. All tracing (DTM/WPM/ICTM) must be disabled
2. Wait for the internal queues to be emptied (no further request is asserted at the auxiliary interface)
3. Change the timestamp enable indication to assert/deassert
4. Enable the required tracing controls (via such as TM, PME0 control bits)

86 Cyclic Redundancy Check (CRC) Unit

86.1 Introduction

The Cyclic Redundancy Check (CRC) computing unit is dedicated to the computation of CRC, off-loading this work from the CPU. Each context has a separate CRC computation engine in order to allow the concurrent computation of the CRC of multiple data streams. The CRC computation is performed at speed without wait-state insertion. Bit-swap and bit-inversion operations can be applied on the final CRC signature. Each context can be configured with one of four hard-wired polynomials, normally used for most of the standard communication protocols. The data stream supports multiple data width (byte/halfword/word) formats.

86.2 Main features

- 4 contexts for the concurrent CRC computation
- Separate CRC engine for each context
- Zero-wait states during the CRC computation (pipeline scheme)
- 4 hard-wired polynomials (two CRC-8, CRC-32 Ethernet, and CRC-16-CCITT)
  - Support for byte, halfword, or word width of the input data stream

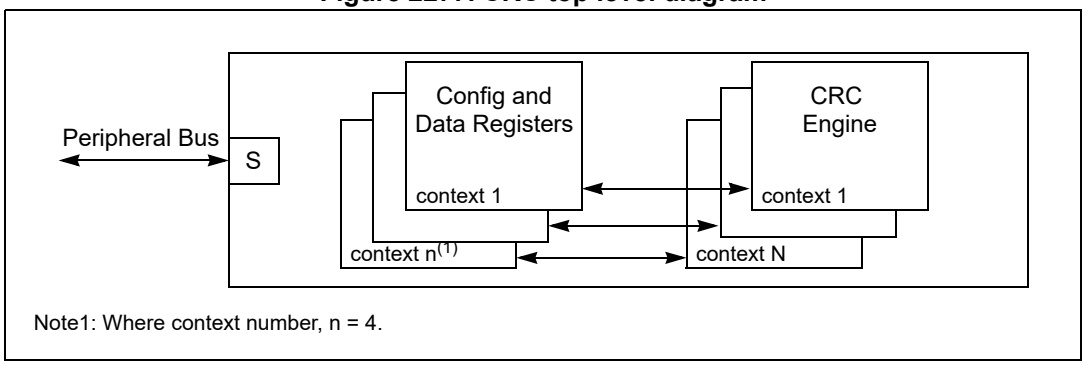
86.2.1 Standard features

- Peripheral bus interface
- CRC-8 VDA CAN
- CRC-16-CCITT
- CRC-32 Ethernet
- CRC-8-H2F Autosar polynomial

86.3 Block diagram

The top level diagram of the CRC module is given in *Figure 2277*.

Figure 2277. CRC top level diagram



Note1: Where context number, n = 4.

86.4 Signal description

The CRC module does not generate any external signals.

86.4.1 Peripheral bus interface

The peripheral bus interface is a slave bus used for configuration and data streaming (CRC computation) purposes via CPU or DMA. The following bus operations (contiguous byte enables) are supported:

- Word (32 bits) data write/read operations to any registers
- Low and high halfword (16 bits, data[31:16] or data[15:0]) data write/read operations to any registers
- Byte (8 bits, data[31:24], data[23:16], data[15:8], or data[7:0]) data write/read operations to any registers

Any other operation (free byte enables or other operations) must be avoided.

The CRC module generates a transfer error in the following cases:

- Any write/read access to the register addresses not mapped on the peripheral but included in the address space of the peripheral
- Any write/read operation different from byte/halfword/word (free byte enables or other operations) on each register

The registers of the CRC module are accessible (read/write) in each access mode: user, supervisor, and test.

The following summarizes bus operation performance:

- Zero wait states (single bus cycle) for each write/read operation to the CRC_CFG and CRC_INP registers
- Zero wait states (single bus cycle) for each write operation to the CRC_CSTAT register
- Double wait states (3 bus cycles) for each read operation to the CRC_CSTAT or CRC_OUTP registers immediately following (next clock cycle) a write operation to the CRC_CSTAT, CRC_INP, or CRC_CFG registers belonging to the same context; in all the other cases, no wait states are inserted

86.5 Memory map and registers

This section describes, in address-order, all the CRC registers.

86.5.1 Register description

Table 2167 shows the memory map for the CRC registers.

Table 2167. CRC unit memory map

Offset	Register	Register classification(1)	Location
CRC registers per context (n=0 to CRC_CNTX_NUM - 1 with CRC_CNTX_NUM = 4)			
0x000 + *n**0x10	CRC Configuration Register *n* (CRC_CFG*n*)	Saf-Lat	*Section 86.5.1.1*
0x004 + *n**0x10	CRC Input Register *n* (CRC_INP*n*)	Saf-Relv	*Section 86.5.1.2*
0x008 + *n**0x10	CRC Current Status Register *n* (CRC_CSTAT*n*)	Saf-Relv	*Section 86.5.1.3*
0x00C + *n**0x10	CRC Output Register *n* (CRC_OUTP*n*)	Saf-Relv	*Section 86.5.1.4*

1. No-Saf = Not safety relevant, Saf-Lat = Safety Latent, Saf-Relv = Safety Relevant, Saf-Crit = Safety Critical.

86.5.1.1 CRC Configuration Register *n* (CRC_CFG*n*)

Offset: 0x000 + *n* *0x10 (*n* = 0 to CRC_CNTX_NUM - 1) Access: User read/write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	SWAP_BYTEWISE	SWAP_BITWISE	POLYG		SWAP	INV
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	f(n)(1)	0	0

Figure 2278. CRC Configuration Register *n* (CRC_CFG*n*)

1. Reset value is a function of n. When n%2 = 0, reset value = 0. When n%2 = 1, reset value = 1.

Table 2168. CRC_CFG*n* field descriptions

Field	Description
26 SWAP_BYTEWISE	SWAP_BYTEWISE 0 No swap 1 Byte wise (MSB to LSB, LSB to MSB) bytes swap performed on CRC_INP input data internally for CRC-32 polynomial calculations **Note:** INV and SWAP bits are set to '1' for CRC-32 polynomial calculations.
27 SWAP_BITWISE	SWAP_BITWISE 0 No swap 1 Bit wise (MSB to LSB, LSB to MSB) bits swap performed on CRC_INP input data internally for CRC-16 polynomial calculations

Table 2168. CRC_CFG*n* field descriptions (continued)

Field	Description
28:29 POLYG	Polynomial selection 00 CRC-CCITT polynomial 01 CRC-32 polynomial 10 CRC-8 polynomial 11 CRC-8H2F Autosar polynomial This bit can be read and written by software. **Note:** This bit can be written only during the configuration phase.
30 SWAP	SWAP selection 0 No swap selection applied on the CRC_OUTP content 1 Swap selection (MSB to LSB, LSB to MSB) applied on the CRC_OUTP content. The swap operation is a bit-by-bit rotation of the content. In case of CRC-CCITT polynomial, the swap operation is applied on the 16 LSB bits. This bit can be read and written by software. **Note:** This bit can be written only during the configuration phase.
31 INV	INV selection 0 No inversion selection applied on the CRC_OUTP content 1 Inversion selection (bit x bit) applied on the CRC_OUTP content. The inversion operation is a complement or negation of the content. In case of CRC-CCITT polynomial the inversion operation is applied on the 16 LSB bits. This bit can be read and written by software. **Note:** This bit can be written only during the configuration phase.

86.5.1.2 CRC Input Register *n* (CRC_INP*n*)

Offset: 0x004 + *n* * 0x10 (*n* = 0 to CRC_CNTX_NUM - 1) Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	INP															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	INP															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 2279. CRC Input Register *n* (CRC_INP*n*)

Table 2169. CRC_INP*n* field descriptions

Field	Description
0:31 INP	Input data for the CRC computation The INP register can be written at byte, halfword (high and low), or word in any sequence. In case of halfword write operation, the bytes must be contiguous. Only the bits written are fed to the CRC engine. This register can be read and written by software.

86.5.1.3 CRC Current Status Register *n* (CRC_CSTAT*n*)

Offset: 0x008 + *n* * 0x10 (*n* = 0 to CRC_CNTX_NUM - 1) Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	CSTAT															
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	CSTAT															
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Figure 2280. CRC Current Status Register *n* (CRC_CSTAT*n*)

Table 2170. CRC_CSTATn field descriptions

Field	Description
0:31 CSTAT	Status of the CRC signature The CSTAT register includes the current status of the CRC signature. No bit swap and inversion are applied to this register. In the case of the CRC-CCITT polynomial, only the16 LSB bits are significative. The 16 MSB bits are tied at 0b during the computation. In the case of the CRC-8 polynomial, only the 8 LSB bits are significative. The 24 MSB bits are tied at 0b during the computation. The CSTAT register can be written at byte, halfword, or word. This register can be read and written by software. **Note:** This register can be written only during the configuration phase.

86.5.1.4 CRC Output Register *n* (CRC_OUTP*n*)

Offset: 0x00C + *n**0x10 (*n* = 0 to CRC_CNTX_NUM - 1) Access: User read-only

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	OUTP															
W																
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	OUTP															
W																
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Figure 2281. CRC Output Register *n* (CRC_OUTP*n*)

Table 2171. CRC_OUTP*n* field descriptions

Field	Description
0:31 OUTP	Final CRC signature The OUTP register includes the final signature corresponding to the CRC_CSTAT register value eventually swapped and inverted. In the case of the CRC-CCITT polynomial, only the16 LSB bits are significative. The 16 MSB bits are tied at 0b during the computation. In the case of the CRC-8 polynomial, only the 8 LSB bits are significative. The 24 MSB bits are tied at 0b during the computation. This register can be read by software.

86.6 Functional description

The CRC module supports the CRC computation for each context. Each context has its own complete set of registers, including the CRC engine. The data flow of each context can be interleaved. The data stream can be structured as a sequence of bytes, halfwords, or words. The input data sequence is provided, eventually mixing the data formats (byte, halfword, and word), writing to the input data register (CRC_INP).

The data stream is generally executed by *n* concurrent DMA data transfers (mem2mem) where *n* is less than or equal to the number of contexts.

- The standard generator polynomials are given in *Equation 92*, *Equation 93* and *Equation 94* for the CRC computation of each context. Two separate registers are available: CRC_INP for writing data and CRC_CSTAT for retrieving the (accumulated up to now) CRC.

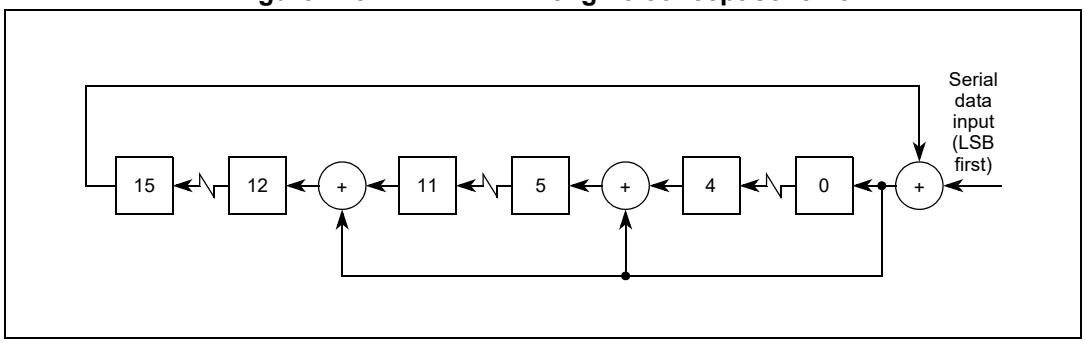
Equation 92 CRC8 VDA CAN:
$$x^8 + x^4 + x^3 + x^2 + 1$$

Equation 93 CRC-CCITT (x.25 protocol):
$$x^{16} + x^{12} + x^5 + 1$$

Equation 94 CRC-32 (Ethernet protocol):
$$x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^8 + x^7 + x^5 + x^4 + x^2 + x + 1$$

Equation 95 CRC-8-H2F (Autosar 4.0 protocol):
$$x^8 + x^5 + x^3 + x^2 + x + 1$$

Figure 2282. CRC-CCITT engine concept scheme

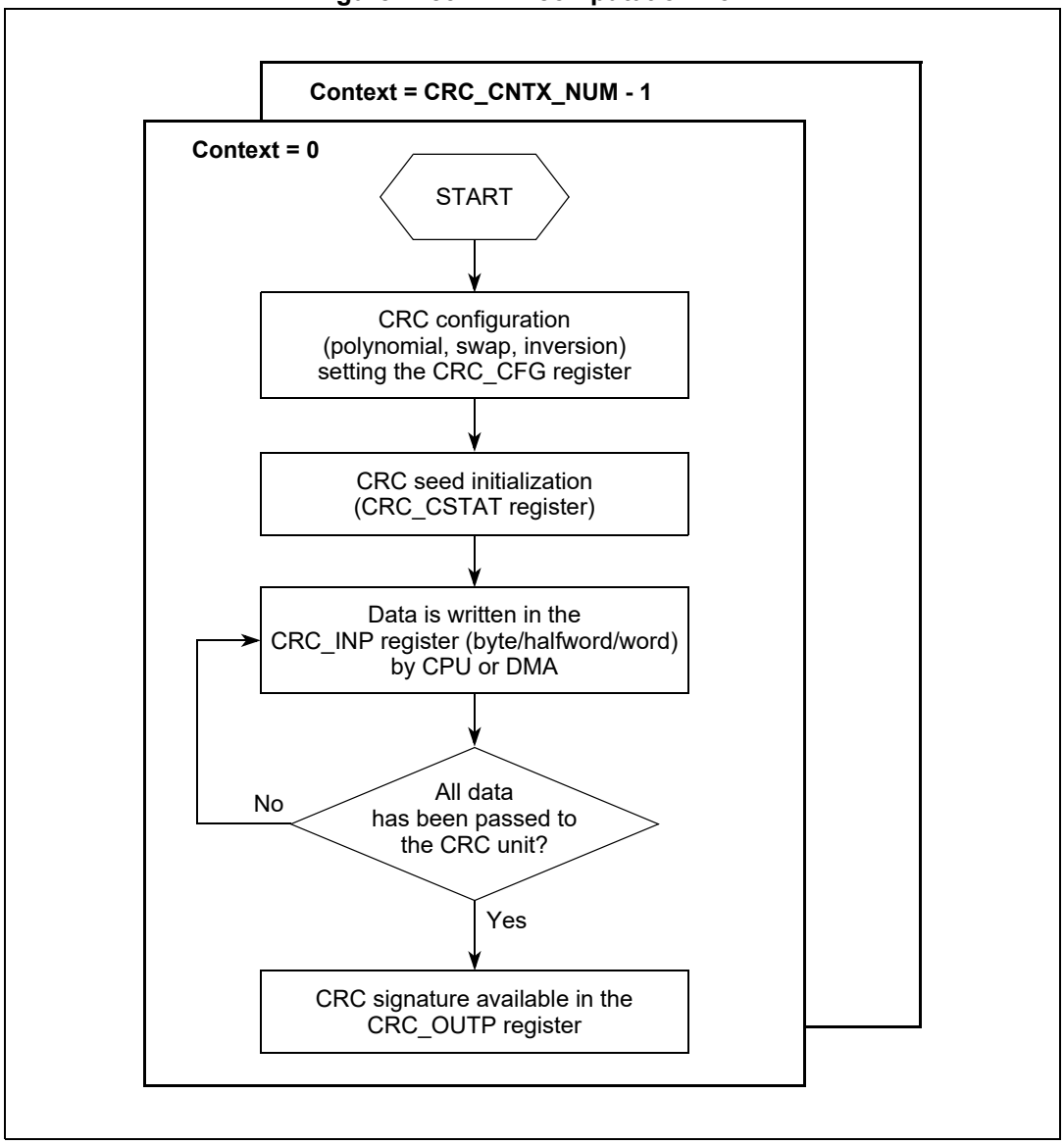


The initial seed value of the CRC can be programmed by initializing the CRC_CSTAT register. The concept scheme (serial data loading) of the CRC engine is given in *Figure 2282* for the CRC-CCITT. The design implementation executes the CRC computation in a single clock cycle (parallel data loading). A pipeline scheme has been adopted to decouple the peripheral bus interface from the CRC engine in order to allow the computation of the CRC at speed (zero wait states).

If the CRC signature is used for encapsulation in the data frame of a communication protocol (such as SPI), a bit swap (MSB → LSB, LSB → MSB) or bit inversion of the final CRC signature, or both can be applied (CRC_OUTP register) before transmitting the CRC.

Use of the CRC module is summarized in the flow chart given in *Figure 2283*.

Figure 2283. CRC computation flow



86.7 Use cases

86.7.1 Programming example

The number of contexts depends on the application. The overall number of contexts for the CRC peripheral depends on the number of peripherals that concurrently require intervention by the CRC module. Two main use cases are considered:

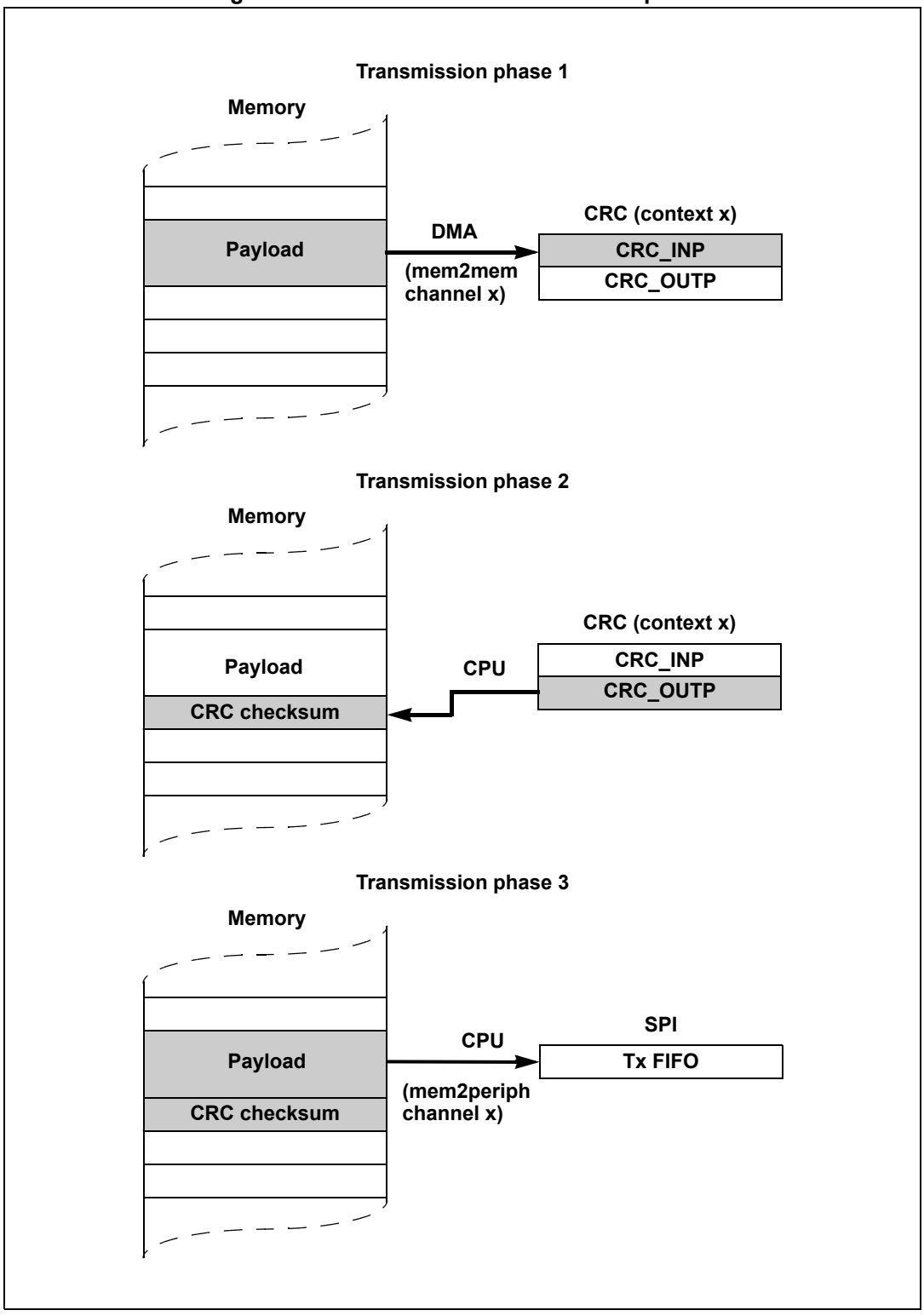
- Calculation of the CRC of the configuration registers during the process safety time
- Calculation of the CRC on the incoming/outcoming frames for the communication protocols (not protected with CRC by definition of the protocol itself) used as a safety-relevant peripheral

The signature of the configuration registers is computed correctly only if these registers do not contain any status bit.

Assuming that the DMA engine has *n* channels (greater than or equal to the number of contexts) configurable for the following types of data transfer: mem2mem, periph2mem, mem2periph; then this sequence, as shown in *Figure 2284*, is applied to manage the transmission data flow:

1. DMA/CRC module configuration (context x, channel x) by CPU
2. Payload transfer from the MEM to the CRC module (CRC_INP register) after MSB → LSB change (input mirroring) to calculate the CRC signature (phase1) by DMA (mem2mem data transfer, channel x)
3. CRC signature copy from the CRC module (CRC_OUTP register) to the MEM (phase 2) by CPU
4. Data block (payload + CRC) transfer from the MEM to the PERIPH module (for example, SPI Tx FIFO) (phase 3) by DMA (mem2periph data transfer, channel x)

Figure 2284. DMA-CRC transmission sequence



86.7.2 Register programming

- INV or SWAP, or both (affecting the output only) can be applied to CRC-8, CRC-16 and CRC-32. Both can be applied together.
- SWAP_BITWISE (affecting the input only) can be applied for all polynomials if required by an application.
- SWAP_BYTEWISE (affecting the input only) can be applied for CRC-16 and CRC-32 polynomials only.
- SWAP_BITWISE has priority over SWAP_BYTEWISE when both are applied together.
- When generating CRC-32 for the Ethernet standard the user needs to set SWAP_BYTEWISE together with INV and SWAP.
- When generating CRC-16 the user needs to set SWAP_BITWISE bit.

87 Memory Error Management Unit (MEMU)

87.1 Introduction

The MEMU is responsible for collection and reporting of error events associated with ECC (Error Correction Code) logic used on:

- SRAM
- Peripheral RAM
- Flash memory.

When any of the following events occur the MEMU receives an error signal which causes an event to be recorded and corresponding error flags to be set and reported to FCCU (Fault Collection and Control Unit).

- Correctable Error: It comprises the following:
  - Single/Double bit error in the data part that is detected via ECC for flash memory. Reporting of single/double bit error as correctable error depends on CTRL.FLASH_ERR_SELECT configuration.
  - Single Bit error in the data part that is detected via MBIST on any RAM.
- Uncorrectable Error: It comprises the following:
  - Double/Multiple(more than two) bit error that is detected via ECC for Flash Memory. Reporting of double bit error as uncorrectable error depends on CTRL.FLASH_ERR_SELECTION configuration.
  - Multiple bit error that is detected via MBIST on any SRAM. Based on MBIST implementation, the MBIST might report several correctable errors in the same word instead of an uncorrectable one. It is the task of the software after MBIST to aggregate these reports and detect the uncorrectable error.
  - Addressing errors and unused data bit errors detected by ECC logic.

The MEMU system connections are chip-specific; see *Chapter 7: Device configuration* that describes how modules are configured and connected.

When multiple errors are indicated from various sources at the same instant, an Overflow can be indicated by the MEMU to the FCCU. Overflow can also be indicated if the reporting table entries are full and a new unique error is reported by the system.

MEMU processes the errors based on whether they are correctable or uncorrectable (from either ECC or MBIST logic), independent of the source generating these errors.

The MEMU has multiple instances of the basic reporting block; one each for:

- System RAM ECC and MBIST
- Peripheral SRAM ECC
- Flash memory ECC.

Each instance has two reporting tables–Correctable errors are reported in one table while the uncorrectable are reported in the other.

See *Section 87.4: Design overview* for the reporting block details.

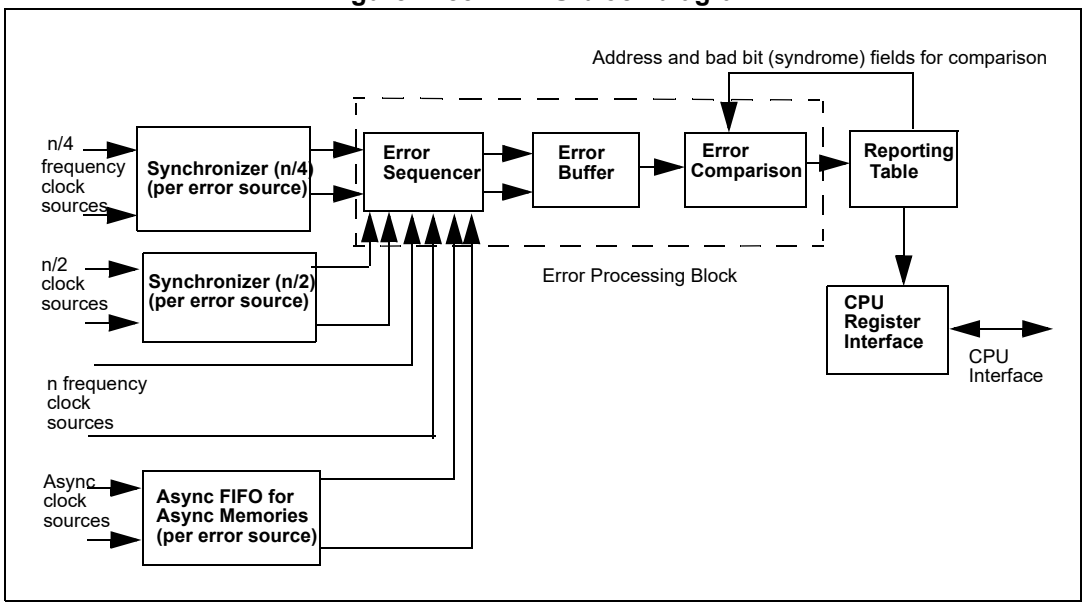
87.2 Features

The MEMU has the following features:

- Supports up to 128 error sources per Reporting Block (System RAM ECC and MBIST, Peripheral SRAM ECC or flash memory ECC): The number of error sources are chip-specific; see *Chapter 7: Device configuration* that describes how modules are configured and connected.
- Support for error reporting from the following category of error sources (both for ECC and MBIST):
  - System RAM
  - Peripheral SRAM
  - Flash memory
- Unique reported errors are logged into the module's reporting table which is accessible to CPU via memory mapped CPU register programming interface.
- MEMU handles overflow during error assertion and reports status accordingly.
- Unique errors are stored in correctable and non correctable section of the reporting table and corresponding indication is generated to the FCCU.
- CPU can program the known errors into the reporting table to avoid their re-reporting by MEMU.
- CPU can clear the reporting table errors.
- The reporting table for uncorrectable errors supports only 1 entry.

87.3 Block diagram

Figure 2285. MEMU block diagram



87.4 Design overview

The MEMU block has three instantiations of the basic reporting block along with a common CPU interface for the user software to access the storage block.

The MEMU should be enabled on power-on reset. The user software can write known error addresses into the reporting table to prevent reporting of those errors addresses to FCCU should they later be accessed in operation. All writes into the reporting table by the CPU that assert the Valid Bit (of the respective reporting table entry) will be indicated as a new error detected and will be reported to the FCCU. This feature can also be used as self-checking option of the MEMU and the MEMU-FCCU connection.

The MEMU design does not limit any particular sequence or errors. See *Section 87.7.3: Handling overflows (Multiple error reporting)* for details on overflow behaviors.

The basic algorithm flow/architecture for error reporting in the MEMU is summarized below:

1. Synchronizer block

   For asynchronous inputs, a FIFO-based structure (per asynchronous channel) is implemented. All the error inputs (correctable error reported, uncorrectable error reported, error address and bad bits/syndrome) are concatenated and stored in the FIFO. While storing the entry in the FIFO, a duplicity check is performed with the previously stored entry (the entry which was stored in the last cycle). If both the entries are duplicate the new entry is deleted and the pointers are rolled back to the old value, otherwise the new entry is stored. Whenever there is an overflow (FULL condition) in the Asynchronous FIFO, it is denoted by setting the Error Buffer Overflow or the EBO flag to the FCCU and the corresponding bit in the Concurrent Overflow Register is set.

   For synchronous inputs, inputs are sampled whenever the corresponding clock synchronizing signal is asserted for a particular channel.

2. Error Processing block

   This block is used to process the incoming errors coming from the device.

   – Error Sequencer is used for sequencing the errors so that these are processed one per clock. In case of more than one errors detected by the sequencer logic, one is sent for the processing (error comparison block), the other is stored in the error buffer and the rest are marked as overflows in the Concurrent overflow register.

   – All the sequences of incoming errors are supported. However, in case of multiple errors being asserted in back to back clock cycles, MEMU will process the errors being forwarded by the sequencer logic and register the rest of the errors as Concurrent Overflows in the Concurrent Overflow Register.

   – While storing the error in the error buffer, uniqueness is determined by comparing the error address, correctable or uncorrectable error type and the bad bit/syndrome (depending on ECC or MBIST logic) of the incoming error with the one stored in buffer. If the error buffer is full, and the incoming error to be stored in the error buffer is not yet in the error table, Error Buffer Overflow (EBO) is registered and the corresponding bit in the Concurrent Overflow Register is set, else the incoming error is dropped.

   – While doing the error buffer uniqueness check, the type of error (Correctable Error: CE or Uncorrectable Error: UCE) is also compared, hence even if the Error Address and the Bad bit/syndrome fields (depending on whether the incoming error is generated from ECC or MBIST logic) of the incoming error matches with

the error buffer, if the type of errors (CE and UCE) differ they are treated as unique.

– Determine the type and uniqueness of errors in the error comparison block and store them in the correct reporting table (correctable or uncorrectable).
– Generate signals to assert the necessary flags. Be aware that it takes several MEMU clock cycles (<=5 unless the error gets temporarily stored in the error buffer due to a collision) for an error report to cause a signal to be sent to the FCCU.

3. Reporting table

This is a pre-defined table in the MEMU to store the details for the device. For the reporting table if the entry is unique it is stored in the reporting table. If the entry is not unique it is not stored and discarded.

The MEMU asserts proper flags when error is stored in the reporting table or an overflow is asserted. Flags are also asserted when the software writes the VLD bit to the reporting table.

The CPU can clear the flags signaling errors or overflows to the FCCU directly by writing 1 at the corresponding valid bit of the status register (*Section 87.6.2.4: System RAM correctable error reporting table status register (SYS_RAM_CERR_STSn)* or *Section 87.6.2.6: System RAM uncorrectable error reporting table status register (SYS_RAM_UNCERR_STS)*). The clearing of flags is totally independent from the status of VLD bits in the reporting table.

The reporting table sizes are chip-specific; see *Chapter 7: Device configuration* that describes how modules are configured and connected.

4. Register block

The CPU register programming interface provides the memory-mapped registers necessary for the CPU to access the reporting table and other control and status registers for the modules. Flags to FCCU will be asserted on write by CPU to the valid bit or by MEMU itself if a unique error is stored. However, in the situation when both MEMU and the CPU are accessing the same Register location, a wait cycle will be signaled to the CPU and the CPU command will be completed in the next clock cycle.

The software, in principle, is not supposed to write VLD to 1 if it is already 1.

The CPU register interface provides the decoding of the peripheral signals to allow access to the reporting table and other registers in module.

Figure 2286. MEMU error buffer uniqueness check flow chart

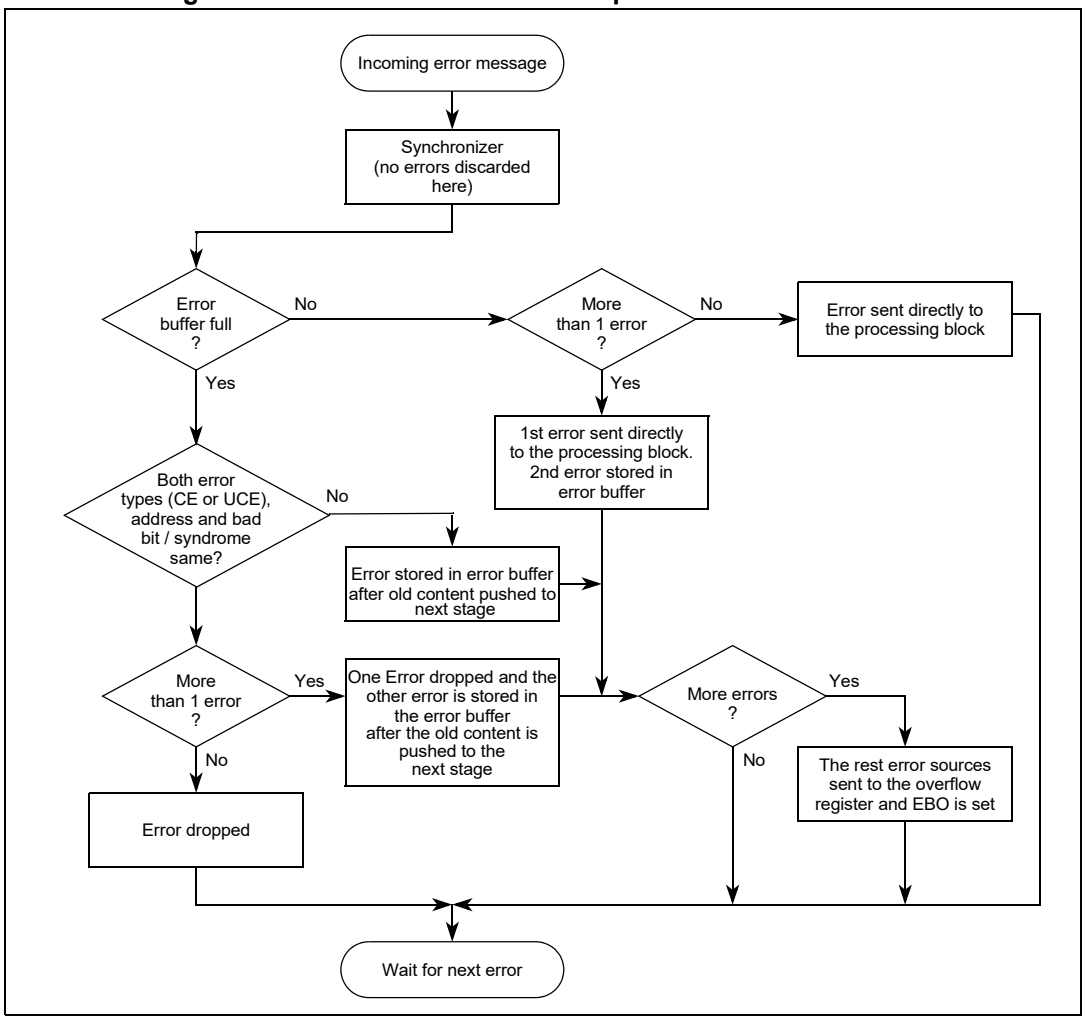
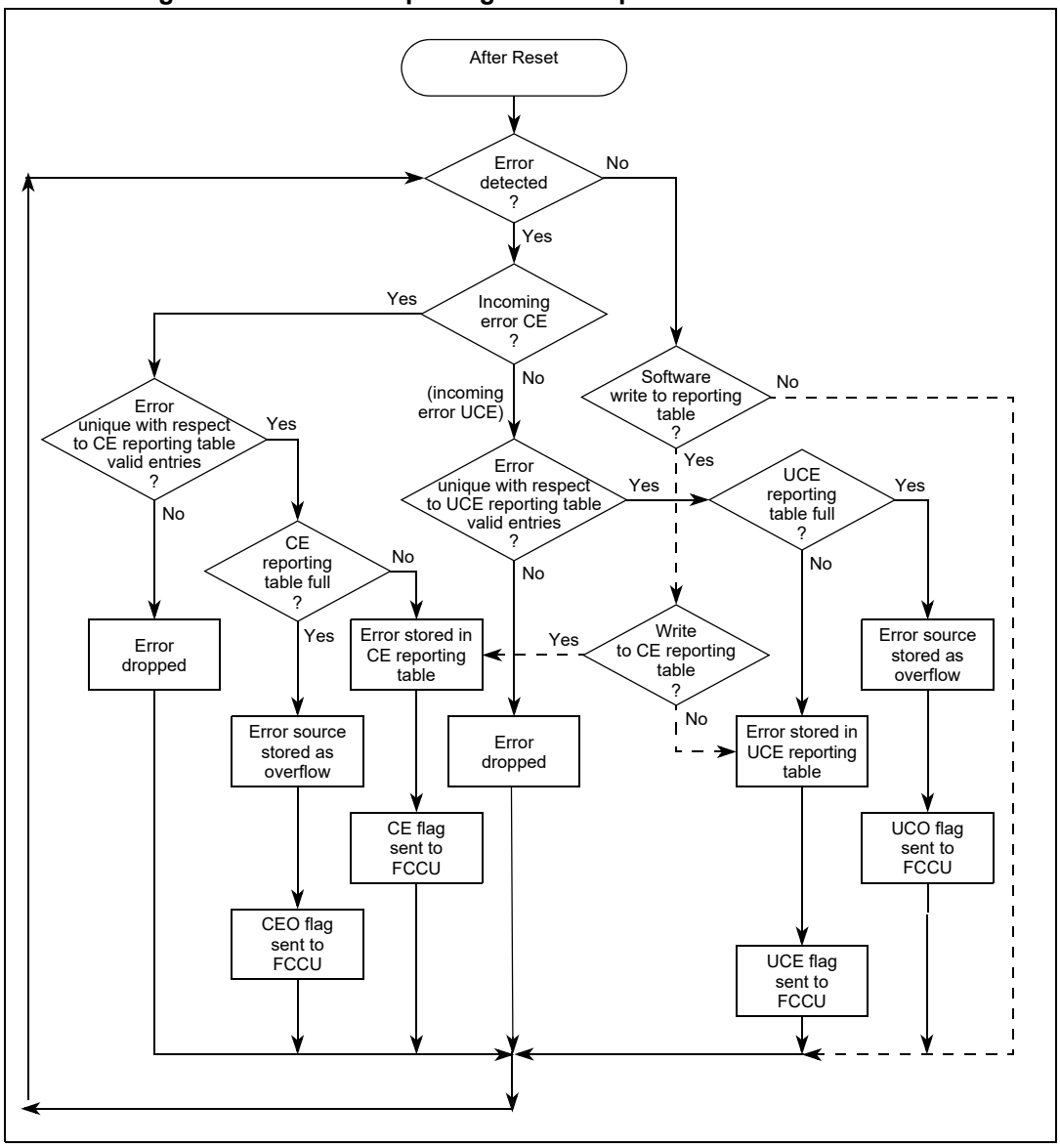


Figure 2287. MEMU reporting table uniqueness check flow chart



87.5 External signal description

MEMU has no external signals.

87.6 Memory map and register definition

87.6.1 Memory map

The memory map comprises 32-bit aligned registers that can be accessed via 8-bit,16-bit, or 32-bit accesses. Write access to reserved address locations will generate a transfer error. Read data from reserved locations will also generate a transfer error and the read data bus will return all 0's.

Note: The module does not check for correctness of the programmed values in registers. Software must ensure that the correct values are being written.

The Control, Error Flag and Debug registers are common for all MEMU error reporting blocks. The correctable error status, correctable error address, uncorrectable error status, uncorrectable error address and concurrent error overflow registers are replicated for each MEMU instance.

Note: The actual availability, number of registers and number of reported errors in the reporting table are chip-specific; see Chapter 7: Device configuration that describes how modules are configured and connected.

Table 2172. MEMU memory map

Offset or address	Register	Access	Reset value	Page
0x0000	Control register (CTRL)	R/W	0x0000_0000	*Section 87.6.2.1*
0x0004	Error Flag register (ERR_FLAG)	R/W	0x0000_0000	*Section 87.6.2.2*
0x000C	Debug register (DEBUG)	R/W	0x0000_0000	*Section 87.6.2.3*
0x0020 + *n**0x8 (*n* = 0 to 10 - 1)	System RAM correctable error reporting table status registers (SYS_RAM_CERR_STS*n*)	R/W	0x0000_0000	*Section 87.6.2.4*
0x0024 + *n**0x8 (*n* = 0 to 10 - 1)	System RAM correctable error reporting table address registers (SYS_RAM_CERR_ADDR*n*)	R/W	0x0000_0000	*Section 87.6.2.5*
0x0028 + (10 - 1)*0x8	System RAM uncorrectable error reporting table status register (SYS_RAM_UNCERR_STS)	R/W	0x0000_0000	*Section 87.6.2.6*
0x002C + (10 - 1)*0x8	System RAM uncorrectable error reporting table address register (SYS_RAM_UNCERR_ADDR)	R/W	0x0000_0000	*Section 87.6.2.7*
0x0030 + *n**0x4 + (10 - 1)*0x8 (*n* = 0 to 128÷32 - 1)(1)	System RAM concurrent overflow registers(2) (SYS_RAM_OFLW*n*)	w1c	0x0000_0000	*Section 87.6.2.8*
0x0620 + *n**0x8 (*n* = 0 to 2 - 1)	Peripheral RAM correctable error reporting table status registers (PERIPH_RAM_CERR_STS*n*)	R/W	0x0000_0000	*Section 87.6.2.9*
0x0624 + *n**0x8 (*n* = 0 to 2 - 1)	Peripheral RAM correctable error reporting table address registers (PERIPH_RAM_CERR_ADDR*n*)	R/W	0x0000_0000	*Section 87.6.2.10*
0x0628 + (2 - 1)*0x8	Peripheral RAM uncorrectable error reporting table status register (PERIPH_RAM_UNCERR_STS)	R/W	0x0000_0000	*Section 87.6.2.11*
0x062C + (2 - 1)*0x8	Peripheral RAM uncorrectable error reporting table address register (PERIPH_RAM_UNCERR_ADDR)	R/W	0x0000_0000	*Section 87.6.2.12*
0x0630 + *n**0x4 + (2 - 1)*0x8 (*n* = 0 to 32÷32 - 1)(1)	Peripheral RAM concurrent overflow registers(1) (PERIPH_RAM_OFLW*n*)	w1c	0x0000_0000	*Section 87.6.2.13*

Table 2172. MEMU memory map (continued)

Offset or address	Register	Access	Reset value	Page
0x0C20 + *n**0x8 (*n* = 0 to 20 - 1)	Flash memory correctable error reporting table status registers (FLASH_CERR_STS*n*)	R/W	0x0000_0000	*Section 87.6.2.14*
0x0C24 + *n**0x8 (*n* = 0 to 20 - 1)	Flash memory correctable error reporting table address registers (FLASH_CERR_ADDR*n*)	R/W	0x0000_0000	*Section 87.6.2.15*
0x0C28 + (20 - 1)*8	Flash memory uncorrectable error reporting table status register (FLASH_UNCERR_STS)	R/W	0x0000_0000	*Section 87.6.2.16*
0x0C2C + (20 - 1)*8	Flash memory uncorrectable error reporting table address register (FLASH_UNCERR_ADDR)	R/W	0x0000_0000	*Section 87.6.2.17*
0x0C30 + *n**0x4 + (20 - 1)*0x8 (*n* = 0 to 5÷32 - 1)(1)	Flash memory concurrent overflow registers(1)(FLASH_OFLW*n*)	w1c	0x0000_0000	*Section 87.6.2.18*

1. In this calculation, round any fractional result of the division to the next highest integer. For example, if the division is 94 ÷ 32 = 2.9375 or 66 ÷ 32 = 2.0625, round the result to 3.
2. The corresponding error sources for the Concurrent Overflow Register (OFLW) are chip-specific; see the Device Configuration chapter that describes how modules are configured and connected.

87.6.2 Register descriptions

87.6.2.1 Control register (CTRL)

Offset: 0x0000 Access: R/W

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R		0	0	0	0	0	0	0	0	0	0	0	0	0	FLASH_ERR_SELECT	
W	SWR															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 2288. Control register (CTRL)

Table 2173. CTRL field descriptions

Field	Description
16 SWR	Software Reset bit. Setting this will clear status flags and overflow register. However, the reporting table registers are not cleared. This bit will auto clear on next clock cycle. 0 No reset. 1 Reset asserted.
30:31 FLASH_ERR_SELECT	FLASH_ERR_SELECT[1:0] 00 Store 20xDEC (Default). Flash Double-bit error corrections only are stored as correctable errors. 01 Store 20xDEC or SEC. Flash Single-bit or Double-bit error corrections, both are stored as correctable errors. 10 Store 20xSEC. Flash Single-bit error corrections only are stored as correctable errors 11 Reserved

87.6.2.2 Error Flag register (ERR_FLAG)

Bits in the ERR_FLAG register indicate:

- A new entry in an error reporting table has been made (indicated by PR_CE, PR_UCE, F_CE, F_UCE, SR_CE, and SR_UCE)
- An overflow condition has been encountered when attempting to make a new entry in an error reporting table (indicated by SR_UCO, SR_CEO, F_UCO,F_CEO, PR_UCO, and PR_CEO)
- An overflow condition has been encountered in the internal error processing buffer of a MEMU reporting block.

Individual flags can be cleared by writing a ‘1’. This also resets the corresponding error indication to FCCU. Flags will not automatically reset if entries are removed from the reporting table. They have to be explicitly cleared by SW.

Offset: 0x0004 Access: R/W

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	PR_CE	PR_UCE	PR_CEO	PR_UCO	PR_EBO
W												w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	F_CE	F_UCE	F_CEO	F_UCO	F_EBO	0	0	0	SR_CE	SR_UCE	SR_CEO	SR_UCO	SR_EBO
W				w1c	w1c	w1c	w1c	w1c				w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 2289. Error flag register (ERR_FLAG)

Bits SR_UCO, SR_CEO, F_UCO, F_CEO, PR_UCO, and PR_CEO are asserted when an overflow in the uncorrectable error reporting table is detected.

Table 2174. ERR_FLAG field descriptions

Field	Description
11 PR_CE	Peripheral RAM ECC Correctable Error detect flag. Indicates that a new and unique Peripheral RAM ECC correctable error is detected. Asserted when a new entry is inserted into the correctable error reporting table (either by CPU or module). 0 No new and unique error detected. 1 New entry in correctable error reporting table is made.
12 PR_UCE	Peripheral RAM ECC Uncorrectable Error Detect flag. Asserted when a new Peripheral RAM ECC uncorrectable error was detected or a new entry is inserted into the uncorrectable error reporting table is done (either by CPU or module). 0 No new and unique error detected. 1 New entry in uncorrectable error reporting table is made.
13 PR_CEO	Peripheral RAM ECC Correctable error Overflow flag. Asserted when the Peripheral RAM ECC correctable error reporting table is full and a new entry is attempted to be made to this table. 0 No Overflow. 1 Overflow in the correctable error reporting table detected.
14 PR_UCO	Peripheral RAM ECC Uncorrectable error Overflow flag. Asserted when the Peripheral RAM ECC uncorrectable error reporting table is full and a new entry is made to this table. 0 No Overflow. 1 Overflow in the uncorrectable error reporting table detected.
15 PR_EBO	Peripheral RAM ECC Error buffer Overflow flag. Asserted when the internal error processing buffer of Peripheral RAM ECC MEMU reporting block has overflowed. This can happen when too many errors are reported in a short interval of time for processing by MEMU. The channel(s) from which error reports were dropped due to the overflow are indicated in the XXXXX_OFLW registers. This field will also be set to ‘1’ if the input ASYNC FIFOs overflow. 0 No Overflow. 1 Overflow in the internal error processing buffer detected.
19 F_CE	Flash ECC Correctable Error Detect flag. Indicates that a new and unique Flash ECC correctable error was detected. Asserted when a new entry to the correctable error reporting table is done (either by CPU or module). 0 No new and unique error detected. 1 New entry in correctable error reporting table is made.
20 F_UCE	Flash ECC Uncorrectable Error Detect flag. Asserted when: – a new Flash ECC uncorrectable error was detected. – a new entry to the uncorrectable error reporting table is done (either by CPU or module) 0 No new and unique error detected. 1 New entry in uncorrectable error reporting table is made.

Table 2174. ERR_FLAG field descriptions (continued)

Field	Description
21 F_CEO	Flash ECC Correctable Error Overflow flag. Asserted when the Flash ECC correctable error reporting table is full and a new entry is made to this table. 0 No Overflow. 1 Overflow in the correctable error reporting table detected.
22 F_UCO	Flash ECC Uncorrectable Error Overflow flag. Asserted when the Flash ECC uncorrectable error reporting table is full and a new entry is made to this table. 0 No Overflow. 1 Overflow in the uncorrectable error reporting table detected.
23 F_EBO	Flash ECC Error buffer Overflow. Flag Asserted when the internal error processing buffer of Flash ECC MEMU reporting block has overflowed. This can happen when too many errors are reported in a short interval of time for MEMU to process them all. The channel(s) from which error reports were dropped due to the overflow are indicated in the XXXXX_OFLW registers. This field will also be set to ‘1’ if the input ASYNC FIFOs overflow. 0 No Overflow. 1 Overflow in the internal error processing buffer detected.
27 SR_CE	System RAM ECC and MBIST Correctable Error detect flag. Indicates that a new and unique System RAM ECC and MBIST correctable error is detected. Asserted when a new entry to the correctable error reporting table is done (either by CPU or module). 0 No new and unique error detected. 1 New entry in correctable error reporting table is made.
28 SR_UCE	System RAM ECC and MBIST Uncorrectable Error Detect flag. Asserted when: – A new System RAM ECC and MBIST uncorrectable error is detected – A new entry to the uncorrectable error reporting table is done (either by CPU or module) 0 No new and unique error detected. 1 New entry in uncorrectable error reporting table is made.
29 SR_CEO	System RAM ECC and MBIST Correctable error Overflow flag. Asserted when the System RAM ECC and MBIST correctable error reporting table is full and a new entry is made to this table. 0 No Overflow. 1 Overflow in the correctable error reporting table detected.

Table 2174. ERR_FLAG field descriptions (continued)

Field	Description
30 SR_UCO	System RAM ECC and MBIST Uncorrectable error Overflow flag. Asserted when the System RAM ECC and MBIST uncorrectable error reporting table is full and a new entry is made to this table. 0 No Overflow. 1 Overflow in the uncorrectable error reporting table detected.
31 SR_EBO	System RAM ECC and MBIST Error buffer Overflow. Flag asserted when the internal error processing buffer of System RAM ECC and MBIST MEMU reporting block has overflowed. This can happen when too many errors are reported in a short interval of time for processing by MEMU. The channel(s) from which error reports were dropped due to the overflow are indicated in the XXXXX_OFLW registers. This field will also be set to '1' if the input ASYNC FIFOs overflow. 0 No Overflow. 1 Overflow in the internal error processing buffer detected.

87.6.2.3 Debug register (DEBUG)

This register is used to check the connections between MEMU & FCCU.

The error output of the MEMU towards FCCU shall stay set if forced until reset by SW via the MEMU in the normal way.

Offset 0x000C Access: R/W

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	FR_PR_CE	FR_PR_UCE	FR_PR_CEO	FR_PR_UCO	FR_PR_EBO
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	FR_F_CE	FR_F_UCE	FR_F_CEO	FR_F_UCO	FR_F_EBO	0	0	0	FR_SR_CE	FR_SR_UCE	FR_SR_CEO	FR_SR_UCO	FR_SR_EBO
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 2290. Debug register (DEBUG)

Table 2175. DEBUG field descriptions

Field	Description
11 FR_PR_CE	Force Peripheral RAM Correctable Error detect flag. 0 Forcing is disabled. 1 Forces PR_CE flag going towards FCCU to 1.
12 FR_PR_UCE	Force Peripheral RAM Uncorrectable Error detect flag. 0 Forcing is disabled. 1 Forces PR_UCE flag going towards FCCU to 1.
13 FR_PR_CEO	Force Peripheral RAM Correctable Error overflow flag 0 Forcing is disabled. 1 Forces PR_CEO flag going towards FCCU to 1.
14 FR_PR_UCO	Forces Peripheral RAM Uncorrectable Error overflow flag. 0 Forcing is disabled. 1 Forces PR_UCO flag going towards FCCU to 1.
15 FR_PR_EBO	Forces Peripheral RAM Error Buffer Overflow Flag. 0 Forcing is disabled. 1 Forces PR_EBO flag going towards FCCU to 1.
19 FR_F_CE	Forces Flash Correctable Error detect flag 0 Forcing is disabled. 1 Forces F_CE flag going towards FCCU to 1.
20 FR_F_UCE	Forces Flash Uncorrectable Error detect flag 0 Forcing is disabled. 1 Forces F_UCE flag going towards FCCU to 1.
21 FR_F_CEO	Forces Flash Correctable Error overflow flag 0 Forcing is disabled. 1 Forces F_CEO flag going towards FCCU to 1.
22 FR_F_UCO	Forces Flash Uncorrectable Error overflow flag 0 Forcing is disabled. 1 Forces F_UCO flag going towards FCCU to 1.
23 FR_F_EBO	Forces Flash Error buffer Overflow flag 0 Forcing is disabled. 1 Forces F_EBO flag going towards FCCU to 1.
27 FR_SR_CE	Forces System RAM Correctable Error detect flag 0 Forcing is disabled. 1 Forces SR_CE flag going towards FCCU to 1.
28 FR_SR_UCE	Forces System RAM Uncorrectable Error detect flag 0 Forcing is disabled. 1 Forces SR_UCE flag going towards FCCU to 1.
29 FR_SR_CEO	Forces System RAM Correctable Error overflow flag 0 Forcing is disabled. 1 Forces SR_CEO flag going towards FCCU to 1.

Table 2175. DEBUG field descriptions (continued)

Field	Description
30 FR_SR_UCO	Forces System RAM Uncorrectable Error overflow flag 0 Forcing is disabled. 1 Forces SR_UCO flag going towards FCCU to 1.
31 FR_SR_EBO	Forces System RAM Error buffer Overflow Flag 0 Forcing is disabled. 1 Forces SR_EBO flag going towards FCCU to 1.

87.6.2.4 System RAM correctable error reporting table status register (SYS_RAM_CERR_STS*n*)

Note: The reporting table is not cleared on software reset.

Offset: 0x0020 + *n**0x8 (*n* = 0 to (10 - 1)) Access: R/W

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	VLD	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0(1)	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	BAD_BIT							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. Write 1 to this bit may lead to non-zero invalid data in the fields 7:15

Figure 2291. System RAM correctable error reporting table status register (SYS_RAM_CERR_STS*n*)

Table 2176. SYS_RAM_CERR_STS*n* field descriptions

Field	Description
0 VLD	Valid bit This indicates that the entry in reporting table is valid. Assertion of this bit causes the correctable error detect flag to be asserted. 0 Entry in table is invalid. 1 Entry in table is valid.
24:31 BAD_BIT	Bad bit field Indicates the bit position in RAM where the error is detected. This is also known as the syndrome. For non-BIST type of errors, this field is not used and reads as 0xFF. Any value other than 0xFF - Position of the bit in RAM where error is found. 0xFF - Invalid Bad Bit. This field should not be used when processing errors.

87.6.2.5 System RAM correctable error reporting table address register (SYS_RAM_CERR_ADDR*n*)

Offset: 0x0024 + *n**0x8 (*n* = 0 to (10 - 1)) Access: R/W

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	ERR_ADD															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	ERR_ADD															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 2292. System RAM correctable error reporting table address register (SYS_RAM_CERR_ADDR*n*)

Table 2177. SYS_RAM_CERR_ADDR*n* field descriptions

Field	Description
0:31 ERR_ADD	Error address field Indicates the address on which the error was detected.

87.6.2.6 System RAM uncorrectable error reporting table status register (SYS_RAM_UNCERR_STS)

Note: The reporting table is not cleared on software reset.

Offset: 0x0028 + (10 - 1)*0x8 Access: R/W

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	VLD	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

Figure 2293. System RAM uncorrectable error reporting table status register (SYS_RAM_UNCERR_STS)

Table 2178. SYS_RAM_UNCERR_STS field descriptions

Field	Description
0 VLD	Valid bit This indicates that the entry in reporting table is valid. Assertion of this bit causes the uncorrectable error detect flag to be asserted. 0 Entry in table is invalid. 1 Entry in table is valid.

87.6.2.7 System RAM uncorrectable error reporting table address register (SYS_RAM_UNCERR_ADDR)

Note: The reporting table is not cleared on software reset.

Offset: 0x002C + (10 - 1)*0x8 Access: R/W

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	ERR_ADD															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	ERR_ADD															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 2294. System RAM uncorrectable error reporting table address register (SYS_RAM_UNCERR_ADDR)

Table 2179. SYS_RAM_UNCERR_ADDR field descriptions

Field	Description
0:31 ERR_ADD	Error address field Indicates the address on which the error was detected.

87.6.2.8 System RAM concurrent overflow registers (SYS_RAM_OFLW*n*)

Note: The reporting table is not cleared on software reset.

Offset: 0x0030 + *n**0x4 + (10 - 1)*0x8 (*n* = 0 to (128÷32 - 1))(1) Access: R/W

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	OFLW															
W	w1c															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	OFLW															
W	w1c															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. In this calculation, round any fractional result of the division to the next highest integer. For example, if the division is 94 ÷ 32 = 2.9375 or 66 ÷ 32 = 2.0625, round the result to 3.

Figure 2295. System RAM concurrent overflow registers (SYS_RAM_OFLW*n*)

Table 2180. SYS_RAM_OFLW*n* field descriptions

Field	Description
0:31 OFLW	Overflow Bit Each bit corresponds to an error source and is asserted when any of the conditions mentioned in *Section 87.7.3: Handling overflows (Multiple error reporting)* occurs.

87.6.2.9 Peripheral RAM correctable error reporting table status register (PERIPH_RAM_CERR_STSn)

Note: The reporting table is not cleared on software reset.

Offset: 0x0620 + *n**0x8 (*n* = 0 to(2 - 1)) Access: R/W

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	VLD	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0(1)	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	BAD_BIT							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. Write 1 to this bit may lead to non-zero invalid data in the fields 7:15

Figure 2296. Peripheral RAM correctable error reporting table status register (PERIPH_RAM_CERR_STS*n*)

Table 2181. PERIPH_RAM_CERR_STS*n* field descriptions

Field	Description
0 VLD	Valid bit This indicates that the entry in reporting table is valid. Assertion of this bit causes the correctable error detect flag to be asserted. 0 Entry in table is invalid. 1 Entry in table is valid.
24:31 BAD_BIT	Bad bit field Indicates the bit position in RAM where the error is detected. This is also known as the syndrome. For non-BIST type of errors, this field is not used and reads as 0xFF. Any value other than 0xFF - Position of the bit in RAM where error is found. 0xFF - Invalid Bad Bit. This field should not be used when processing errors.

87.6.2.10 Peripheral RAM correctable error reporting table address register (PERIPH_RAM_CERR_ADDR*n*)

Offset: 0x0624 + *n**0x8 (*n* = 0 to (2 - 1)) Access: R/W

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	ERR_ADD															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	ERR_ADD															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 2297. Peripheral RAM correctable error reporting table address register (PERIPH_RAM_CERR_ADDR*n*)

Table 2182. PERIPH_RAM_CERR_ADDR*n* field descriptions

Field	Description
0:31 ERR_ADD	Error address field Indicates the address on which the error was detected.

87.6.2.11 Peripheral RAM uncorrectable error reporting table status register (PERIPH_RAM_UNCERR_STS)

Note: The reporting table is not cleared on software reset.

Offset: 0x0628 + (2 - 1)*0x8 Access: R/W

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	VLD	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

Figure 2298. Peripheral RAM uncorrectable error reporting table status register (PERIPH_RAM_UNCERR_STS)

Table 2183. PERIPH_RAM_UNCERR_STS field descriptions

Field	Description
0 VLD	Valid bit This indicates that the entry in reporting table is valid. Assertion of this bit causes the uncorrectable error detect flag to be asserted. 0 Entry in table is invalid. 1 Entry in table is valid.

87.6.2.12 Peripheral RAM uncorrectable error reporting table address register (PERIPH_RAM_UNCERR_ADDR)

Note: The reporting table is not cleared on software reset.

Offset: 0x062C + (2 - 1)*0x8 Access: R/W

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	ERR_ADD															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	ERR_ADD															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 2299. Peripheral RAM uncorrectable error reporting table address register (PERIPH_RAM_UNCERR_ADDR)

Table 2184. PERIPH_RAM_UNCERR_ADDR field descriptions

Field	Description
0:31 ERR_ADD	Error address field Indicates the address on which the error was detected.

87.6.2.13 Peripheral RAM concurrent overflow registers (PERIPH_RAM_OFLW*n*)

Note: The reporting table is not cleared on software reset.

Offset: 0x0630 + *n**0x4 + (2 - 1)*0x8 (*n* = 0 to (32÷32 - 1))(1) Access: R/W

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	OFLW															
W	w1c															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	OFLW															
W	w1c															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. In this calculation, round any fractional result of the division to the next highest integer. For example, if the division is 94 ÷ 32 = 2.9375 or 66 ÷ 32 = 2.0625, round the result to 3.

Figure 2300. Peripheral RAM concurrent overflow registers (PERIPH_RAM_OFLW*n*)

Table 2185. PERIPH_RAM_OFLW*n* field descriptions

Field	Description
0:31 OFLW	Overflow Bit Each bit corresponds to an error source and is asserted when any of the conditions mentioned in *Section 87.7.3: Handling overflows (Multiple error reporting)* occurs.

87.6.2.14 Flash memory correctable error reporting table status register (FLASH_CERR_STS*n*)

Note: The reporting table is not cleared on software reset.

MEMU captures flash ecc error address on 128bit boundary.

Offset: 0x0C20 + *n**0x8 (*n* = 0 to (20 - 1)) Access: R/W

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	VLD	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0(1)	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	BAD_BIT															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. Write 1 to this bit may lead to non-zero invalid data in the fields 7:15

Figure 2301. Flash memory correctable error reporting table status register (FLASH_CERR_STS*n*)

Table 2186. FLASH_CERR_STS*n* register field descriptions

Field	Description
0 VLD	Valid bit This indicates that the entry in reporting table is valid. Assertion of this bit causes the correctable error detect flag to be asserted. 0 Entry in table is invalid. 1 Entry in table is valid.
16:31 BAD_BIT	Bad bit field Indicates the bit position in FLASH where the error is detected. This is also known as the syndrome. For non-BIST type of errors, this field is not used and reads as 0xFFFF. Any value other than 0xFFFF - Position of the bit in FLASH where error is found. 0xFFFFInvalid Bad Bit. This field should not be used when processing errors.

87.6.2.15 Flash memory correctable error reporting table address register (FLASH_CERR_ADDR*n*)

Offset: 0x0C24 + *n**0x8 (*n* = 0 to (20 - 1)) Access: R/W

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	ERR_ADD															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	ERR_ADD															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 2302. Flash memory correctable error reporting table address register (FLASH_CERR_ADDR*n*)

Table 2187. FLASH_CERR_ADDR*n* register field descriptions

Field	Description
0:31 ERR_ADD	Error address field Indicates the address on which the error was detected.

87.6.2.16 Flash memory uncorrectable error reporting table status register (FLASH_UNCERR_STS)

Note: The reporting table is not cleared on software reset.

Offset: 0x0C28 + (20 - 1)*0x8 Access: R/W

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	VLD	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

Figure 2303. Flash memory uncorrectable error reporting table status register (FLASH_UNCERR_STS)

Table 2188. UNCERR_STS field descriptions

Field	Description
0 VLD	Valid bit This indicates that the entry in reporting table is valid. Assertion of this bit causes the uncorrectable error detect flag to be asserted. 0 Entry in table is invalid. 1 Entry in table is valid.

87.6.2.17 Flash memory uncorrectable error reporting table address register (FLASH_UNCERR_ADDR)

Note: The reporting table is not cleared on software reset.

Offset: 0x0C2C + (20 - 1)*8 Access: R/W

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	ERR_ADD															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	ERR_ADD															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 2304. Flash memory uncorrectable error reporting table address register (FLASH_UNCERR_ADDR)

Table 2189. FLASH_UNCERR_ADDR field descriptions

Field	Description
0:31 ERR_ADD	Error address field Indicates the address on which the error was detected.

87.6.2.18 Flash memory concurrent overflow registers (FLASH_OFLWn)

Note: The reporting table is not cleared on software reset.

Offset: 0x0C30 + *n**0x4 + (20 - 1)*0x8 (*n* = 0 to (5÷32 - 1))[1] Access: R/W

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	OFLW															
W	w1c															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	OFLW															
W	w1c															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. In this calculation, round any fractional result of the division to the next highest integer. For example, if the division is 94 ÷ 32 = 2.9375 or 66 ÷ 32 = 2.0625, round the result to 3.

Figure 2305. Flash memory concurrent overflow registers (FLASH_OFLWn)

Table 2190. FLASH_OFLWn field descriptions

Field	Description
0:31 OFLW	Overflow Bit Each bit corresponds to an error source and is asserted when any of the conditions mentioned in *Section 87.7.3: Handling overflows (Multiple error reporting)* occurs.

87.7 Functional description

In this section, the term “bad bit” and “syndrome” are equivalent, and are used to indicate the position of the bit in error.

87.7.1 Initializing MEMU

MEMU is always enabled and can be configured by user software for known errors which the system/application has collected over a period of time. The MEMU logic will not update the reporting table corresponding to the programmed values which have the VLD bit SET.

The user software can write into the reporting table and must make VLD bit = ‘1’ for the programmed entry to be valid. Any write by the CPU that asserts the VLD bit will cause a corresponding flag to FCCU.

The software can program the table with known error addresses by setting the valid bit and storing the corresponding error address.The sequence of programming to prevent a collision of reporting table entries written by the HW and the SW is as follows:

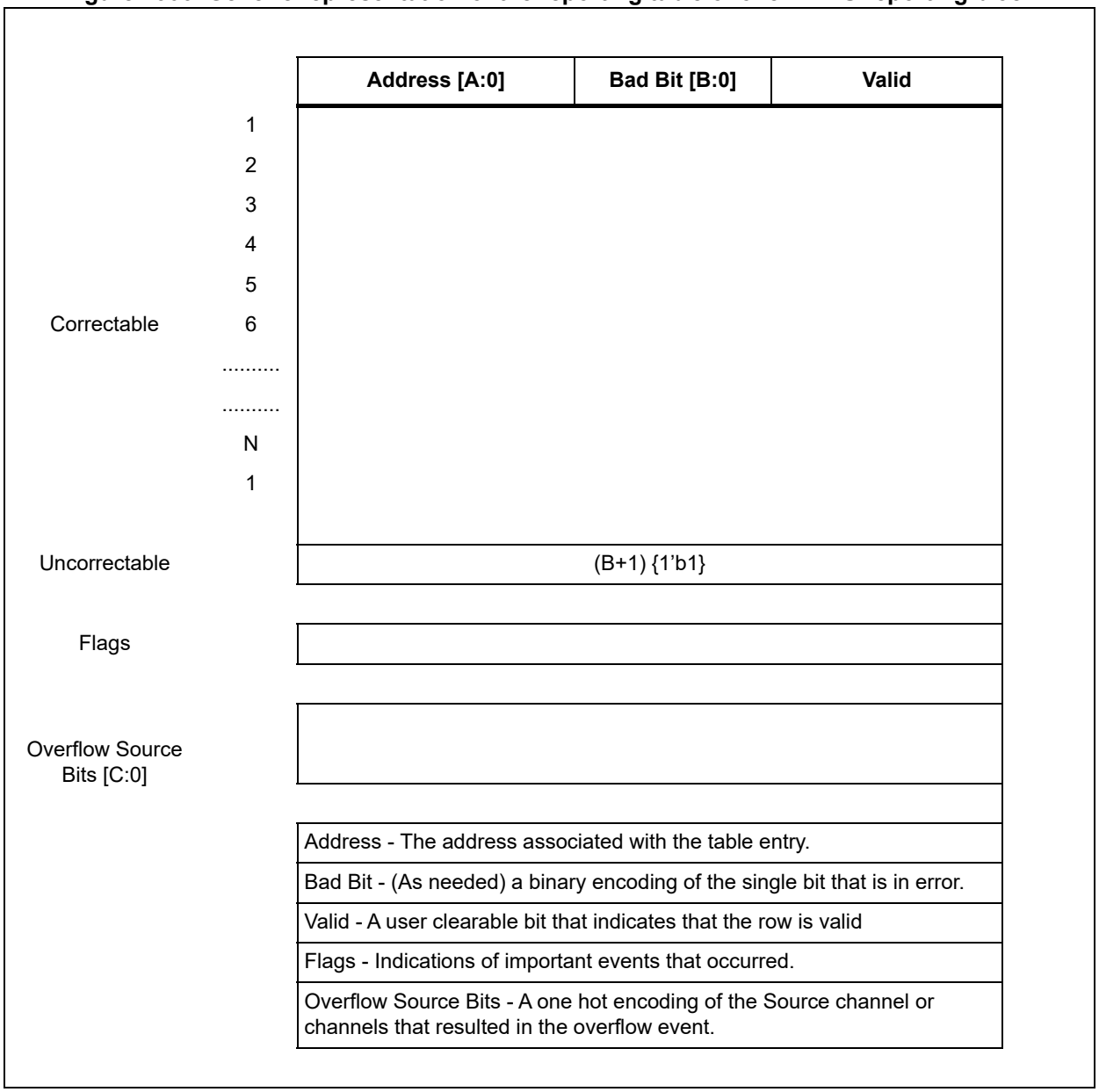
1. Check that the VLD bit is not set,
2. Write an invalid address/bad bit information,
3. Set VLD bit,
4. Check that the address still has an invalid value,
5. Write address/bad bit information to desired address.

If the FCCU is programmed to cause an IRQ on new MEMU table entries, steps 3 to 5 should be executed with interrupts disabled to prevent any error handling SW to read the invalid address written in step 2). Note that in the case the software is in between the 3rd and the 5th step, and the MEMU writes an error with the same address that Software wants to program, both the errors will be stored in the reporting table (duplicate entries).

87.7.2 Reading the reporting table

The software at any time can read the reporting table via the software programming interface. It should read the entries with the valid bit set and take necessary action. To invalidate any entry, the valid bit must be cleared. Reading an entry which does not have the VLD bit set will return invalid data.

Figure 2306. Generic representation of the reporting table of one MEMU reporting block



87.7.3 Handling overflows (Multiple error reporting)

When a bit in the overflow registers (either error buffer or reporting table though the overflow registers are same) is asserted it can indicate the occurrence of one of the following conditions:

1. Error buffer overflow–More than two memories reporting an error at the same instant or the input ASYNC FIFOs reach the FULL level (overflow)
   – More than 2 errors detected at the same time. This would lead to overflow in the error buffer. MEMU can process only one error at a time while storing the other in the error buffer. So this would be indicated as an overflow. The uniqueness check in the input buffer does not “guarantee” that an ECC-supervised memory with only one error will not be marked as overflow source. If that one error occurs together with two (or more) additional errors at the same cycle it can get flagged as an overflow.
2. Correctable/Uncorrectable Error Overflow–Overflow in correctable and uncorrectable reporting table occurs only when a unique entry is to be stored but the tables are already full (all entries have valid bit set).
   – An ECC error which matches the address of an MBIST error is dropped (same as if it matched the address of an ECC error). In case, where a MBIST error comes with ECC error already in the table, both the address and bad bits are compared to determine uniqueness.
3. The Overflows will be stored in the bit-wise order which means that if the error bit 31 reports an error, the bit 31 of the overflow register will be set to ‘1’.

88 Indirect Memory Access (IMA)

88.1 Introduction

Indirect Memory Access (IMA) refers to the activity of accessing any chip memory for the purpose of reading or modifying data, or both, including ECC check bits. This capability is useful for test activities, for example, verifying ECC functionality.

The SPC58xEx/SPC58xGx MCU provides two mechanisms for IMA:

- The built-in CPU End2End ECC test capability provides a path for accessing data and ECC check bits from some SRAM arrays.
- An IMA controller module provides registers for selecting, reading, and writing memory data.

Note: Not all SRAM data and ECC bits are accessible. Refer to the Device Configuration chapter for details.

88.2 Accessing memory via CPU End2End ECC test

The processor cores can access all bits (including ECC check bits) in SRAM arrays that support E2E ECC with the exceptions of addresses that are blocked by MPU configuration.

Note: All memory accesses by the processor cores are filtered through the MPU. There are no backdoors to blocked memory addresses.

SRAM is selected by absolute memory address with check bit data in a special register. The register used depends on which memory is being accessed: I-cache/D-cache, or other memory.

Since all CPU memory accesses are arbitrated by the crossbar (XBAR), there is no need to halt normal memory access in software.

Memory is accessed via processor core registers. The register(s) used depends on the memory being accessed. This is described in the following sections.

88.2.1 Accessing I-cache and D-cache memory via processor core

For I-cache and D-cache memory, the SRAM location is selected by cache way, set and word with data and check bits in registers provided specifically for reading and writing cache tag bits and ECC check bits. There are two Device Control Registers (DCRs) provided for cache access:

- Cache Debug Access Control register (CDACNTL)
- Cache Debug Access Data register (CDADATA)

These registers are summarized in the following sections.

88.2.1.1 Cache Debug Access Control register (CDACNTL)

The Cache Debug Access Control register (CDACNTL) is a 32-bit privileged register (available only in supervisor and debug modes) that provides access to I-cache and D-cache memory.

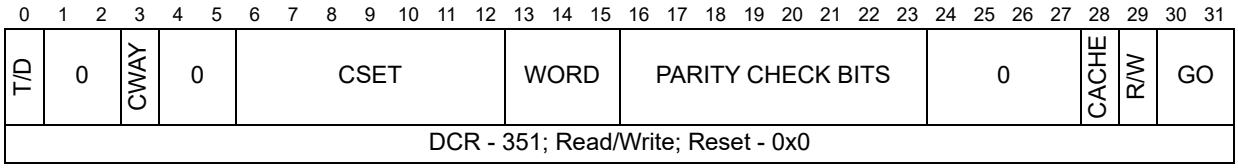


Figure 2307. Cache Debug Access Control register (CDACNTL)

The CDACNTL bits are described in *Table 2191*.

Table 2191. CDACNTL field descriptions

Bits	Name	Description
0	T/D	Tag / Data: 0 Data array selected 1 Tag array selected
1:2	—	Reserved - read as zeros.
3	CWAY	Cache Way: Specifies the cache way to be selected.
4:5	—	Reserved - read as zeros.
6:12	CSET	Cache Set: Specifies the cache set to be selected. Only valid set numbers for the selected cache are supported, otherwise results are undefined.
13:15	WORD	Word (Used for data array access only, I or D-cache) Specifies one of eight words of selected set.
16:23	PARITY CHECK BITS	Parity check bits (I or D-cache): D-cache Data array, I-cache Data Array: parity check bits for data. Tag Array: parity check bits for tag.
24:27	—	Reserved - read as zeros.
28	CACHE	Cache Select: Specifies the cache to be selected. 0 Selects the data cache for the operation. 1 Selects the instruction cache for the operation.
29	R/W	Read / Write: 0 Selects write operation. Write the data in the CDADATA register to the location specified by this CDACNTL register. 1 Selects read operation. Read the cache memory location specified by this CDACNTL register and store the resulting data in the CDADATA register and store the parity bits in this CDACNTL register.
30:31	GO	GO command bits: 00 Inactive or complete (no action taken) hardware sets GO=00 when an operation is complete 01 Read or write cache memory location specified by this CDACNTL register. 1x Reserved

88.2.1.2 Cache Debug Access Data register (CDADATA)

The Cache Debug Access Data register (CDADATA) is a 32-bit privileged register (available only in supervisor and debug modes) that provides access to I-cache and D-cache memory.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
TAG or DATA																															
DCR - 350; Read/Write; Reset - Undefined/Unaffected																															

Figure 2308. Cache Debug Access Data register (CDADATA)

The CDADATA bits are described in *Table 2192*.

Table 2192. CDADATA field descriptions

Bits	Name	Description
0:31	TAG	TAG Array Access Data - when accessing the tag array of the I-cache: 0–19 Tag compare bits 20–22 Reserved 23 Valid bit 24 R bit 25 SO bit 26–28 Reserved 29–31 Lockout bits. These three bits should have the same value, 1- Locked out, 0- not locked out. TAG Array Access Data - when accessing the tag array of the D-cache: 0–20 Tag compare bits 21–22 Reserved 23 Valid bit 24 R bit 25 SO bit 26 SW bit 27 UW bit 28 Reserved 29–31 Lockout bits. These three bits should have the same value, 1-Locked out, 0- not locked out.
	DATA	DATA Array Access Data (Bytes 0:3 of the selected word) - when accessing the data array of either cache: 0–7 byte 0 8–15 byte 1 16–23 byte 2 24–31 byte 3

88.2.2 Accessing non I-cache or D-cache memory via processor core

The e2eECC Error Control/Status register 0 (E2EECSR0) is provided specifically for reading and writing E2E ECC check bits. This register is described in the following section.

88.2.2.1 e2eECC Error Control/Status register 0 (E2EECSR0)

The e2eECC Error Control/Status register 0 (E2EECSR0) is a 32-bit privileged register (available only in supervisor and debug modes) that provides access to E2E ECC data. It is shown in the following figure.

0 1 2 3	4 5 6 7 8 9 10 11	12 13 14 15	16 17	18 19	20 21 22 23	24 25 26 27 28 29 30 31
0	RCHKBIT	0	WRC	INVC	0	WCHKBIT/CHKINVT
DCR - 511; Read/Write; Reset - 0x0; Privileged						

Figure 2309. e2eECC Error Control/Status register 0 (E2EECSR0)

The E2EECSR0 bits are described in *Table 2193*.

Table 2193. E2EECSR0 field descriptions

Bits	Name	Description
0:3	—	Reserved - read as zeros.
4:11	RCHKBIT	Read Checkbits: This field provides the raw checkbits received on the last CPU data access to the DMEM or to external memory via the data BIU. Software may use this information to determine the stored checkbits of external memories or the DMEM by performing CPU load operations and then accessing this field prior to the next load operation, assuming interrupts are masked.
12:15	—	Reserved - read as zeros.
16:17	WRC	Write Control: 00 No write checkbit substitution is performed by this control bit 01 Checkbit substitution is performed for the next CPU external write access (only) with the values in the WCHKBIT control field. Software must clear this field before re-writing to 01 in order to generate a subsequent checkbit substitution operation. 10 Checkbit substitution is performed for each CPU external write access while this field remains set to 10, by driving the values in the WCHKBIT control field. This field must be cleared by software to discontinue further checkbit substitution. 11 Reserved **Note:** Checkbit substitution has priority over Error injection
18:19	INVC	Invert Control: 00 No error injection is performed by this control bit 01 Error injection is performed for the next CPU external write access (only) by modifying p_hwchkbit[7:0] based on CHKINVT. Software must clear this field before re-writing to 01 in order to generate a subsequent error injection operation. 10 Error injection is performed for each CPU external write access while this field remains set to 10 by modifying p_hwchkbit[7:0] based on CHKINVT. This field must be cleared by software to discontinue further error generation. 11 Reserved **Note:** Checkbit substitution has priority over Error injection

Table 2193. E2EECSR0 field descriptions (continued)

Bits	Name	Description
20:23	—	Reserved - read as zeros.
24:31	WCHKBIT/CHKINVT	Write Checkbits / Checkbit Invert Mask: This field provides the checkbit substitution values when performing checkbit substitution via the WRC control field, and controls which checkbits are inverted when performing error injection via the INVC control field. For Checkbit inversion operations via error injection: 0 Checkbit is driven normally 1 Checkbit is inverted before being driven out when error injection occurs.

88.3 Accessing memory via the IMA module

The Indirect Memory Access (IMA) module provides an alternative way to access on-chip memory without going through standard peripheral or system RAM interfaces to check the data or change the data.

This alternative path can be used for diagnostic purposes. For example, for SRAM arrays that do not support E2E ECC, the IMA block provides a way to write corrupted ECC values so that the ECC error decoding logic can be checked.

Note the following restrictions on memory access via IMA:

- When a bus master performs an IMA access, no other bus master shall perform an access to the same memory.
- Before any IMA access, the user shall disable the FCCU fault related to the IMA. If not, the FCCU triggers a fault.
- The IMA must be unlocked for reads.

The order of precedence for access to a memory is shown below. Once the IMA has been granted access, no other accesses are allowed until the IMA select line to the selected memory is negated by the IMA.

Note: Once the IMA has selected a RAM, it expects no other accesses from any other master until deselected by the IMA.

The IMA module signals the FCCU when it is reading or writing to a memory address. This enables the FCCU to assess whether the access is intentional or spurious. For an intentional access the respective FCCU error input needs to be disabled, otherwise it should be enabled.

88.3.1 Features

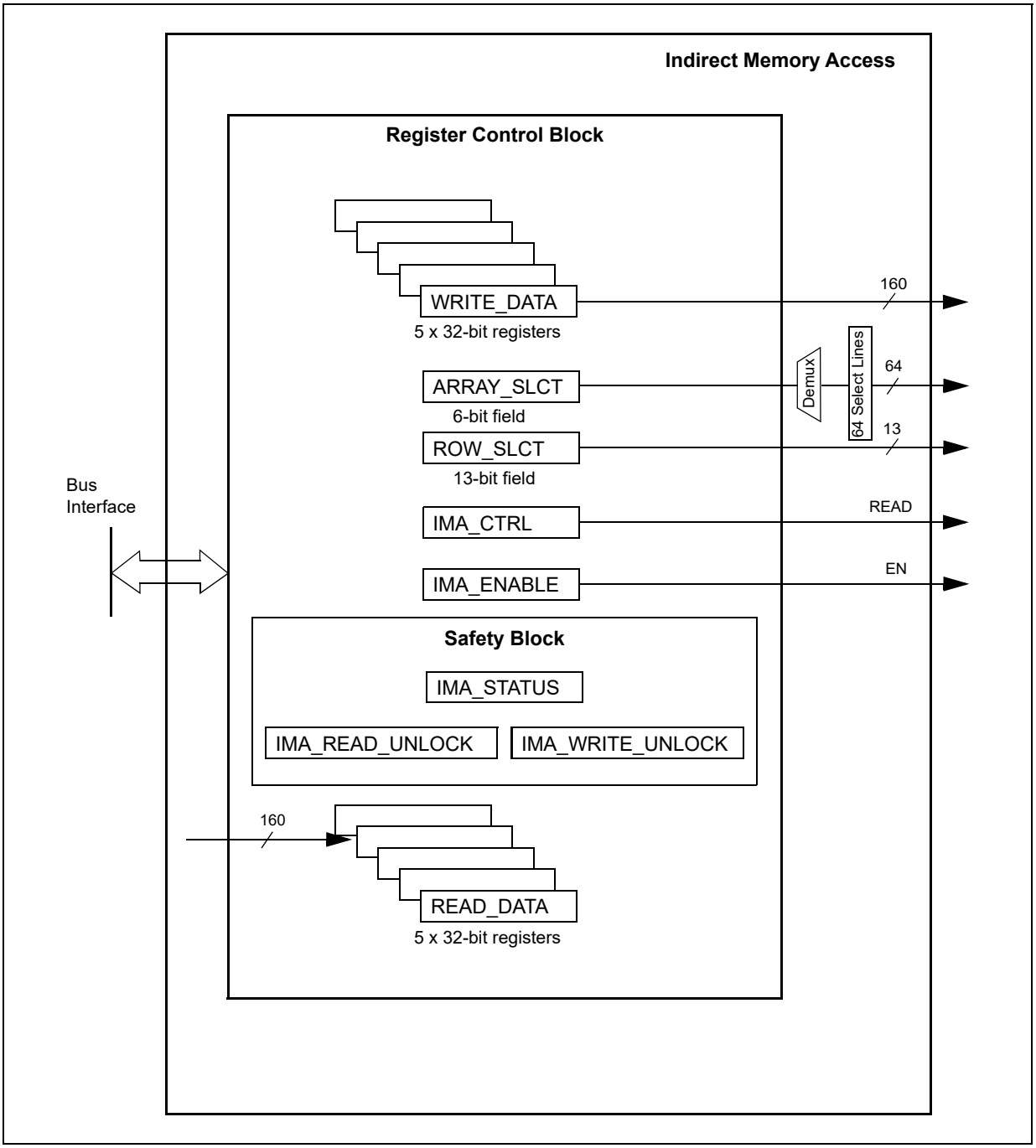
The IMA module includes the following features:

- Able to read all bits in all RAM arrays connected to the IMA, once the IMA is unlocked for reads.
- Enables reading and manipulation of ECC check bits.
- During an IMA SRAM access, other accesses are affected as follows:
  - Reads return random data
  - Writes have no effect
- Any IMA access to memory are indicated in the Fault Collection and Control Unit (FCCU).
- IMA accesses to memory do not bypass any SRAM repair which may have occurred during production test of the device.
- IMA accesses to memory use a physical address, not a logical address, and thus memory addresses used by the IMA are not memory mapped.

88.3.2 Block diagram

The following figure shows the structure and function of the IMA module. The Register Control Block contains the memory-mapped registers accessible by software. The Safety Block prevents spurious activation of the IMA module. It ensures the IMA does not access RAM unless the system needs it too.

Figure 2310. IMA block diagram



88.3.3 IMA Module Memory Map and Registers

This section provides details of the IMA memory map. It also provides detailed descriptions for each of the registers in that map.

The Indirect Memory Access unit's memory map allows for the CPU to access RAM's on the chip to write test data so that single-bit ECCs and multi-bit ECC logic can be tested.

Table 2194. IMA memory map

Offset Address	Register	Section
0x0000	IMA Control register (IMA_CTRL)	*Section 88.3.3.1*
0x0004	IMA Enable Access register (IMA_ENABLE)	*Section 88.3.3.2*
0x0008	IMA Status register (IMA_STATUS)	*Section 88.3.3.3*
0x000C	IMA RAM SELECT register (IMA_SLCT)	*Section 88.3.3.4*
0x0010	IMA Write Unlock register (IMA_WRITE_UNLOCK)	*Section 88.3.3.5*
0x0014	IMA Read Unlock register (IMA_READ_UNLOCK)	*Section 88.3.3.6*
0x0018–0x002B	Reserved	
0x002C	IMA RAM Write Data register 4 (IMA_WRITE_DATA_4)	*Section 88.3.3.7*
0x0030	IMA RAM Write Data register 3 (IMA_WRITE_DATA_3)	*Section 88.3.3.7*
0x0034	IMA RAM Write Data register 2 (IMA_WRITE_DATA_2)	*Section 88.3.3.7*
0x0038	IMA RAM Write Data register 1 (IMA_WRITE_DATA_1)	*Section 88.3.3.7*
0x003C	IMA RAM Write Data register 0 (IMA_WRITE_DATA_0)	*Section 88.3.3.7*
0x0040–0x004B	Reserved	
0x004C	IMA RAM Read Data register 4 (IMA_READ_DATA_4)	*Section 88.3.3.8*
0x0050	IMA RAM Read Data register 3 (IMA_READ_DATA_3)	*Section 88.3.3.8*
0x0054	IMA RAM Read Data register 2 (IMA_READ_DATA_2)	*Section 88.3.3.8*
0x0058	IMA RAM Read Data register 1 (IMA_READ_DATA_1)	*Section 88.3.3.8*
0x005C	IMA RAM Read Data register 0 (IMA_READ_DATA_0)	*Section 88.3.3.8*

88.3.3.1 IMA Control register (IMA_CTRL)

The IMA CTRL register controls read/write accesses. Since the IMA may interfere with proper read/write access of RAMs, a locking scheme is implemented to prevent accidental activation of the IMA. Using this scheme, the IMA read/write accesses can be locked when desired.

Offset: 0x00 Access: User read/write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	READ
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 2311. IMA Control register (IMA_CTRL)

Table 2195. IMA_CTRL field descriptions

Field	Description
31 READ	IMA Read This bit controls whether the IMA will perform a read or write 0 The IMA access will be a write. 1 The IMA access will be a read.

88.3.3.2 IMA Enable Access register (IMA_ENABLE)

The Enable Access Register is a single-bit register that starts an IMA access. The bit automatically clears itself after two clocks thus providing an enable pulse.

Note: Back to back writes to the enable bit is not valid. It will still generate a two-clock enable pulse and will be considered as a single access initiation.

Offset: 0x04 Access: User read/write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																EN
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 2312. IMA Enable Access register (IMA_ENABLE)

Table 2196. IMA_ENABLE field descriptions

Field	Description
31 EN	IMA Enable This bit controls RAM access initiation by the IMA. Once set, this bit is cleared after two clocks to provide an enable pulse. 0 No IMA access will occur or the IMA has finished the previous access. 1 Start an IMA access or an IMA access is occurring.

88.3.3.3 IMA Status register (IMA_STATUS)

The IMA Status register shows the current status of the IMA as to which operations are unlocked.

Offset: 0x08 Access: User read

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	WRITE_LOCK	0	0	0	0	0	0	0	READ_LOCK
W																
Reset	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1

Figure 2313. IMA Status register (IMA_STATUS)

Table 2197. IMA_STATUS field descriptions

Field	Description
23 WRITE_LOCK	The ability of the IMA to do write accesses. The WRITE_LOCK bit shows the current status of the lock on IMA write accesses. If set, the IMA does not write to the memories. If cleared, the IMA is allowed to do writes to the memories. 0 Write accesses from the IMA to memories are allowed. 1 Write accesses from the IMA to memories are not allowed. Any memory writes from the IMA are ignored.
31 READ_LOCK	The ability of the IMA to do read accesses. The READ_LOCK bit shows the current status of the lock on IMA read accesses. If set, the IMA does not do reads from the memories. If cleared, the IMA is allowed to do reads from the memories. 0 Read accesses from the IMA to memories are allowed. 1 Read accesses from the IMA to memories are not allowed. Any memory reads from the IMA are ignored.

88.3.3.4 IMA RAM Select register (IMA_SLCT)

The IMA RAM Select register provides a method of selecting which RAM memory block will be accessed. This method is used because RAM is to be accessed in a non-functional way, such as accessing all bits including the ECC, and accessing on RAM word boundaries instead of byte or 32-bit word boundary.

Note: Only one RAM can be selected at a time. Each SoC has a different encoding for the RAMs. Refer to the Device Configuration chapter for details.

Offset: 0x0C Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	ROW_SLCT												
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	ARRAY_SLCT					
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 2314. IMA RAM SELECT register (IMA_SLCT)

Table 2198. IMA_SLCT field descriptions

Field	Description
3:15 ROW_SLCT	The ROW_SLCT value specifies the row of the current RAM array under test to be selected. Hence the largest RAM which the IMA can access is 32 KB RAM.
26:31 ARRAY_SLCT	The ARRAY_SLCT specifies the RAM array to be tested. Up to 64 RAM instances can be supported.

88.3.3.5 IMA Write Unlock register (IMA_WRITE_UNLOCK)

The Unlock register is a safety feature to ensure that the IMA does not access RAMs unless the system needs for it to access the memories. The unlock register has to be written with two unique values for the IMA to unlock. The WRITE_LOCK bit in the IMA_STATUS register indicates whether the IMA is unlocked for writes.

Offset: 0x10 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	WRITE_KEY															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	WRITE_KEY															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 2315. IMA Write Unlock register (IMA_WRITE_UNLOCK

Table 2199. IMA_WRITE_UNLOCK field descriptions

Field	Description
0:31 WRITE_KEY	Key to unlock the IMA. – 0x04A43F95 is the first key that has to be written to start the unlock process. – 0xE4A9EBF7 is the second key that has to be written to unlock the IMA.

88.3.3.6 IMA Read Unlock register (IMA_READ_UNLOCK)

The Unlock register is a safety feature to ensure that the IMA does not access RAMs unless the system needs for it to access the memories. The unlock register has to be written with two unique values for the IMA to unlock. The READ_LOCK bit in the IMA_STATUS register indicates whether the IMA is unlocked for reads.

Offset: 0x14 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	READ_KEY															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	READ_KEY															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 2316. IMA Read Unlock register (IMA_READ_UNLOCK)

Table 2200. IMA_READ_UNLOCK field descriptions

Field	Description
0:31 READ_KEY	Key to unlock the IMA Read Access to RAMs. – 0xF06AB5BC is the first key that has to be written to start the unlock process. – 0x14081B56 is the second key that has to be written to unlock the IMA.

88.3.3.7 IMA RAM Write Data register n (IMA_WRITE_DATA_*n*)

The IMA_WRITE_DATA_*n* registers hold the data values to be written to the RAM. The number of active data bits depends on the number of bits for the RAM selected. Upper unused write data bits are ignored during the RAM access. With IMA_CTRL[READ] set to 0, by writing a 1 to the IMA_ENABLE[EN] bit, the IMA will begin a write access.

0x2C (WRITE_DATA_4)
0x30 (WRITE_DATA_3)
Offset: 0x34 (WRITE_DATA_2) Access: User read/write
0x38 (WRITE_DATA_1)
0x3C (WRITE_DATA_0)

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	WRITE															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	WRITE															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 2317. IMA RAM Write Data register *n* (IMA_WRITE_DATA_*n*)

Table 2201. IMA_WRITE_DATA_*n* field descriptions

Field	Description
0:31 WRITE	Contains data to be written to RAM. – WRITE_DATA_4[WRITE] contains write data bits[159:128] – WRITE_DATA_3[WRITE] contains write data bits[127:96] – WRITE_DATA_2[WRITE] contains write data bits[95:64] – WRITE_DATA_1[WRITE] contains write data bits[63:32] – WRITE_DATA_0[WRITE] contains write data bits[31:0]

88.3.3.8 IMA RAM Read Data register n (IMA_READ_DATA_*n*)

The IMA_READ_DATA_*n* registers hold the data values read from the RAM. The number of active data bits depends on the number of bits for the RAM selected. Upper unused READ DATA bits are updated to zeroes during the RAM access. With IMA_CTRL[READ] set to 1, by writing a 1 to the IMA_ENABLE[EN] bit, the IMA will begin a read access. Performing a read operation on IMA_READ_DATA_*n* registers causes data from RAM to be latched into these registers and presented to the CPU on the next clock.

Offset: 0x4C (READ_DATA_4), 0x50 (READ_DATA_3), 0x54 (READ_DATA_2), 0x58 (READ_DATA_1), 0x5C (READ_DATA_0) — Access: User read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	READ															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	READ															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 2318. IMA RAM Read Data register n (IMA_READ_DATA_*n*)

Table 2202. IMA_READ_DATA_*n* field descriptions

Field	Description
0:31 READ	Contains data to be written to RAM. – READ_DATA_4[READ] contains read data bits[159:128] – READ_DATA_3[READ] contains read data bits[127:96] – READ_DATA_2[READ] contains read data bits[95:64] – READ_DATA_1[READ] contains read data bits[63:32] – READ_DATA_0[READ] contains read data bits[31:0]

88.3.4 IMA module functional description

This section describes the operation of the IMA, beginning with the unlocking of reads and writes, the software initialization, then the software interface for generating single-bit ECC errors, generating multiple-bit ECC errors, and a software supported memory BIST.

88.3.4.1 Initialization sequence

This section describes which registers are reset due to hardware reset and what locations the user must initialize prior to doing an access by the IMA.

88.3.4.2 Hardware controlled initialization

In the IMA, registers and control logic are reset by hardware (system reset). A system reset deasserts output signals and resets general configuration bits.

88.3.4.3 User initialization (prior to asserting IMA_ENABLE[EN])

The user needs to initialize portions of the SoC prior to setting the IMA_ENABLE[EN] bit. The exact values depend on the particular application. Refer to the specific chapter's memory map to see how to turn off that peripheral so that accesses are not happening at the same time as the IMA. Without the peripheral shut down, there is a small chance that the IMA and the peripheral can interfere with each other in accessing the RAM.

88.3.4.3.1 Shut down the peripheral

The IMA can cause problems with the system or the peripheral if both the IMA and the other system components are accessing the RAM during normal operation.

To shut down the peripheral, read the peripheral's documentation. The documentation will describe how to put the peripheral into a quiet state so that it will not access the RAM during the IMA access. If the other functional path is running at the same time, the data for the other peripheral will be corrupted and unknown.

88.3.4.3.2 Unlocking reads

The IMA will not allow a read to any memory unless it has been unlocked. This is a safety feature so that it will take two unique writes to the READ_KEY register to unlock writes. To unlock the read feature, the system will have to do the following:

1. Write 0xF06AB5BC to READ_KEY.
2. Write 0x14081B56 to READ_KEY.

The IMA has two features to make the READ_KEY lock in the following instance:

- Writing an incorrect key to READ_KEY.

88.3.4.3.3 Unlocking writes

The IMA will not allow a write to any memory unless it has been unlocked. This is a safety feature so that it will take two unique writes to the WRITE_KEY register to unlock writes. To unlock the write feature, the system will have to do the following:

1. Write 0x04A43F95 to WRITE_KEY.
2. Write 0xE4A9EBF7 to WRITE_KEY.

The IMA has a feature to make the WRITE_KEY lock in the following two instances:

1. Writing an incorrect key to WRITE_KEY.
2. Not setting the IMA_ENABLE[bit] for 16,384 system clocks.

88.3.4.4 IMA lock

The IMA module also allows for read/write access locking via an input pin. If this input is asserted, both read/write accesses via IMA are locked. This locking scheme overrides the above KEY based locking scheme. Hence it provides additional safety over the KEY based locking method. However, usage of this feature is entirely SOC specific.

88.3.4.5 IMA access

There are two differences between a read access and a write access. The first difference is that the IMA_STATUS[WRITE_LOCK] bit has to be cleared for writes and the IMA_STATUS[READ_LOCK] has to be cleared for reads. The second difference is that the IMA_CTRL[READ] has to be set for reads and IMA_CTRL[READ] has to be cleared for writes.

The following sections show the different accesses that are available depending on the options.

88.3.4.5.1 Read access

Read Accesses are expected to follow the sequence below:

1. Unlock Read Access by performing appropriate consecutive writes to READ_UNLOCK register.
2. Based on the address to be accessed, write the ARRAY_SLCT and ROW_SLCT fields in the IMA_SLCT register.
3. Set the READ bit in the IMA_CTRL register.
4. Set the EN bit in the IMA_ENABLE register to initiate read access. The EN bit will self-clear after two clocks.
5. Software must wait an appropriate number of clocks depending on the speed of the SRAM for the access to complete.
6. Read data from READ_DATA registers. This causes data from RAM to be latched into the IMA and returned to the CPU on the next clock.

88.3.4.5.2 Write access

Write Accesses are expected to follow the sequence below:

1. Unlock Write Access by performing appropriate consecutive writes to WRITE_UNLOCK register.
2. Based on the address to be accessed, write the ARRAY_SLCT and ROW_SLCT fields in the IMA_SLCT register.
3. Write the Write Data in WRITE_DATA registers.
4. Clear the READ bit in the IMA_CTRL register
5. Set the EN bit in the IMA_ENABLE register to initiate write access. The EN bit will self-clear after two clocks.
6. Software must wait an appropriate number of clocks for the IMA write to occur, depending on the speed of the SRAM being written.

88.3.5 Application information

Caution: The IMA will inhibit correct operation of the underlying peripherals. The application software is responsible for ensuring there is not a resource conflict.

The primary function of the IMA module is to provide a means for the system to program invalid ECC bits and see that the appropriate peripheral is able to detect the error. For a single-bit error, the peripheral should be able to correct the error. For a multiple-bit error, the peripheral should report an error.

89 Fault Collection and Control Unit (FCCU)

89.1 Introduction

The Fault Collection and Control Unit (FCCU) offers a systematic approach to fault detection and control. The FCCU provides a hardware channel to collect errors and to place the device into a safety state when a failure in the device is detected. No CPU intervention is requested for collection and control operation.

89.2 Features

The FCCU offers a systematic approach to fault detection and control. The distinctive features of the module are:

- 1 to 128 recoverable faults management
- HW or SW fault recovery management
- Fault detection collection
- Fault injection (fake faults; refer to the implementation-specific details)
- Bi-stable, Dual Rail and Time Switching output protocols on EOUT
- Watchdog timer for the reconfiguration phase
- Redundant collection of hardware checker (for example RCCU) results
- Redundant collection of error information from safety relevant modules on the device
- Collection of test results
- Configurable and graded fault control
- Internal (SoC) reactions
  - ALARM state: interrupt request
  - FAULT state: long functional reset request pulse, short functional reset request pulse, NMI
- Internal reactions (independently configurable for each RF)
  - No reset reaction
  - IRQ
  - Short functional reset
  - Long functional reset
  - NMI
- External reaction (failure is reported to the outside world via one or more output pins)
- External reaction (fault state): EOUT signaling. Failure indication via the pin(s) is controlled by the FCCU
- Configuration lock: The FCCU’s configuration is lockable in two ways
  - Permanently locked until next reset or transiently locked until a specific key is written. FCCU gets transiently locked again if an invalid key is written into

FCCU_TRANS_LOCK register (a value different from 0xBC). Key to lock the FCCU permanently is 0xFF, written in the FCCU_PERMNT_LOCK register

- One of the failure indication pins is high to indicate operational (OK) state (in bi-stable operation mode). The above is not true in case the failure indication protocol is configured to be a toggling protocol that is dual rail and time switching
- After power-on the error out pins have high impedance(31). FCCU goes to operational state only on software request
- In case of a failure event or on software request for Error pin indication, the pin(s) are set to faulty state for a minimum time T_min, even if SW tries to release it before (for the case of error pin configured in bi-stable mode only). T_min = 250 µs + delta_T, with delta_T parameter being configurable by SW up to 10 ms
- The overall error detection, processing and indication time is less than 10 ms
- The actual maximum time is five safe (IRCOSC) clock cycles

Two classes of fault are identified based on the criticality and the related reactions. Internal (short or long functional reset request pulse, interrupt request) and external (EOUT signaling) reactions are statically defined or programmable based on the fault criticality. The default configuration can be modified only in a specific FCCU state for application/test/debugging purposes. FCCU is designed for assuming that the system clock is faster than the IRC clock.

89.2.1 Standard features

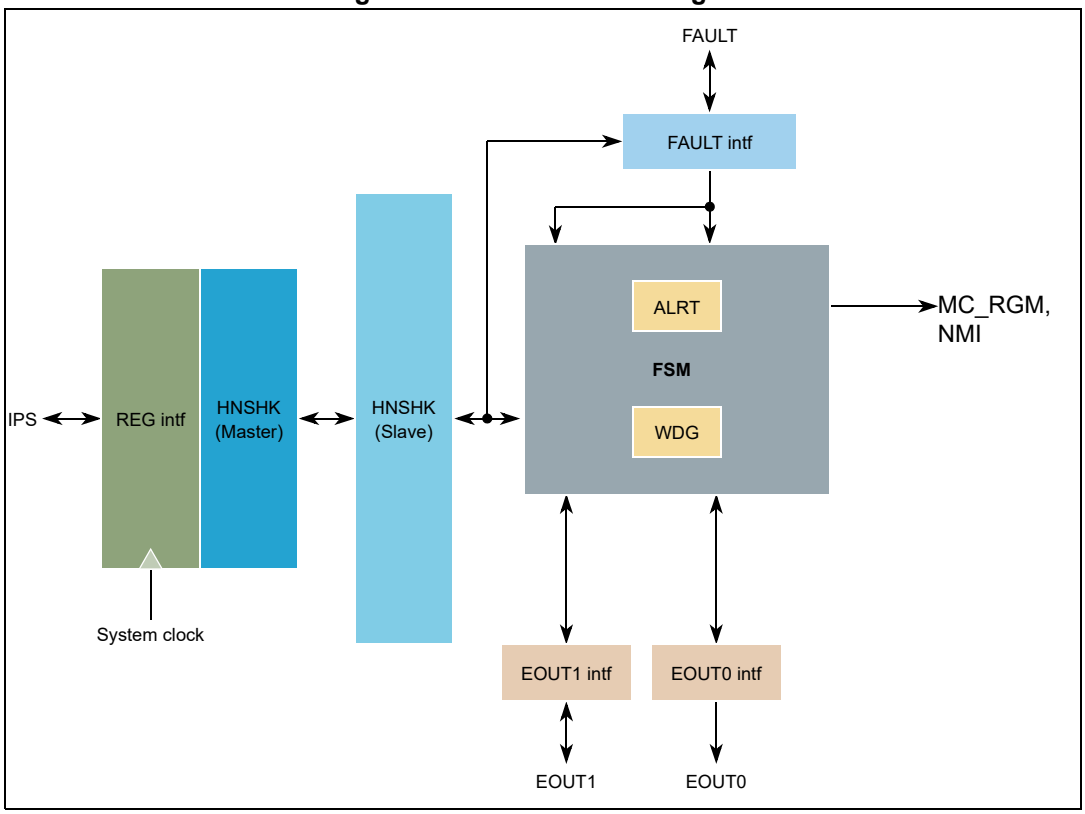
- IPS (slave bus signals) bus interface

89.3 Block diagram

The top-level diagram of the FCCU module is given in *Figure 2319*.

31. Actual value depends on the SoC settings at pad level.

Figure 2319. FCCU block diagram



The FCCU module includes the submodules listed in *Table 2203*.

Table 2203. FCCU submodules

Submodule	Description
REG intf	Includes the register file, the IPS bus interface, the IRQ interface
HNSHK blocks (master and slave blocks)	Includes the FSMs to support the handshake between the REG if and the FSM unit due to the usage of two asynchronous clocks (IPS system clock and RC oscillator clock)
FSM unit	Implements the main functions of the FCCU. The FSM also includes the: Watchdog timer (WDG) Alarm timer (ALRT)
FAULT intf	Implements the interface for the fault conditioning and management
EOUTx units	Implement the output stage to manage the EOUT interfaces

89.4 Signal description

89.4.1 Pinout

The FCCU generates two external signals, EOUT[0] and EOUT[1]. These are described in *Section 89.5.9: EOUT interface*.

For the availability of these signals on a chip, refer to the device configuration chapter.

89.4.2 IPS bus interface

The IPS bus interface is a slave bus used for configuration purposes via CPU. The following bus operations (contiguous byte enables) are supported:

- word (32 bits) data write/read operations to any registers.
- low and high half-words (16 bits, data[31:16] or data[15:0]) data write/read operations to any registers.
- byte (8 bits, data[31:24] or data[23:16] or data[15:8] or data[7:0] data write/read operations to any registers.
- any other operation (free byte enables or other operations) must be avoided.

The FCCU module generates a transfer error in the following cases:

- any write/read access executed outside the register address space of the peripheral. Note that if some registers are not implemented (for example FCCU_CFS_CFGx registers), they generate no bus transfer error when accessed in writing.
- any write/read operation different from byte/hword/word (free byte enables or other operations) on each register.
- any write access executed on configuration registers, when the fccu is not in the CONFIG state. Configuration registers are those registers which can be written only in CONFIG state. Refer to *Table 2212: FCCU memory map*.
- any write executed on FCCU_CFG_TO register, when the fccu is in the CONFIG state.
- any write access to the Transient and permanent Lock Register in any mode other than supervisor access mode.

The registers of the FCCU module are accessible (read/write) in each access mode: user, supervisor or test.

89.5 Functional description

89.5.1 Definitions

In general, the following definitions are applicable for the fault management:

- HW recoverable fault: the fault indication is an edge-triggered and level-sensitive signal that remains asserted until the fault cause is asserted. That is, if 0 on the fault signal indicates fault, then the status flags are valid until the fault line is ‘0’. The status is automatically cleared when the fault signal goes to 1. Typically the fault signal is latched in an external module at the FCCU. The FCCU state transitions are consequently executed on the state changes of the input fault signal. No SW intervention in the FCCU is required to recover the fault condition.
- SW recoverable fault: the fault indication is a signal asserted without a defined time duration. The fault signal is latched in the FCCU. The fault recovery is executed following a SW recovery procedure (status/flag register clearing).

HW recoverable is an option to exclude the handling of error source/s by FCCU management SW, in case it is known that the error is recoverable by itself when the error condition gets corrected.

The following types of reset are applicable:

- Destructive reset: any type of reset related to a power failure condition that implies a complete system re-initialization.
- Long functional reset: implies the Flash and digital circuitry (most of it except FCCU, STCU) initialization.
- Short functional reset: implies the digital circuitry (most of it except FCCU, STCU) initialization.

89.5.2 FSM description

The FCCU module functionality is depicted by the FSM diagram given in *Figure 2320*.

Basically four states are identified with the following meaning:

- CONFIG: The configuration state is used only to modify the default configuration of the FCCU. A subset of the FCCU registers, dedicated to define the FCCU configuration (global configuration, reactions to fault, timeout, recoverable fault masking) can be accessed in write mode only in the CONFIG state.
  The CONFIG state is accessible only in NORMAL state and if the configuration is not locked. The permanent configuration lock can be disabled by a reset of the FCCU, the transient lock register is unlocked by writing 0xBC into it. FCCU gets transiently locked again if an invalid key is written into FCCU_TRANS_LOCK register (a value different from 0xBC). Key to lock the FCCU permanently is 0xFF, written in the FCCU_PERMNT_LOCK register. After reset lifts, the state shall be transiently locked (permanent lock → is unlocked and transient lock → is locked).
  The CONFIG to NORMAL state transition can be executed by SW or automatically following a timeout condition of the watchdog. In case the timeout information and the SW request to mode change to NORMAL appears at the same time, watchdog timeout had the priority and hence the Configuration registers (those that are writable only in CONFIG mode) are reset to their default values. Also, FCCU_CFG_TO and FCCU_EINOUT are reset to their default values, although they do not meet the condition of being a configuration register. The movement to NORMAL is made.
  The incoming faults, occurring during the configuration phase (CONFIG state) are anyway latched in order to process them when the FCCU is moved into the NORMAL state, according to the new configuration.
- NORMAL: This is the FCCU's operating state when no faults are occurring. It is also the default state on the reset exit. Succeeding one of the following events:
  - unmasked recoverable faults with the timeout disabled → the FCCU moves to the FAULT state.
  - unmasked recoverable faults with the timeout enabled → the FCCU moves to the ALARM state.
  - masked recoverable faults → the FCCU stays in NORMAL state.
- ALARM: the FCCU moves into the ALARM state when an unmasked recoverable fault occurs and the timeout is enabled. The transition to the ALARM state goes along with an interrupt request (ALARM state), if enabled. By definition, this fault may be recovered within a programmable timeout period, before it generates a transition to the

FAULT state. The timeout is re-initialized if the FCCU state moves to the NORMAL state. The timeout restarts following the recovery from the FAULT state.

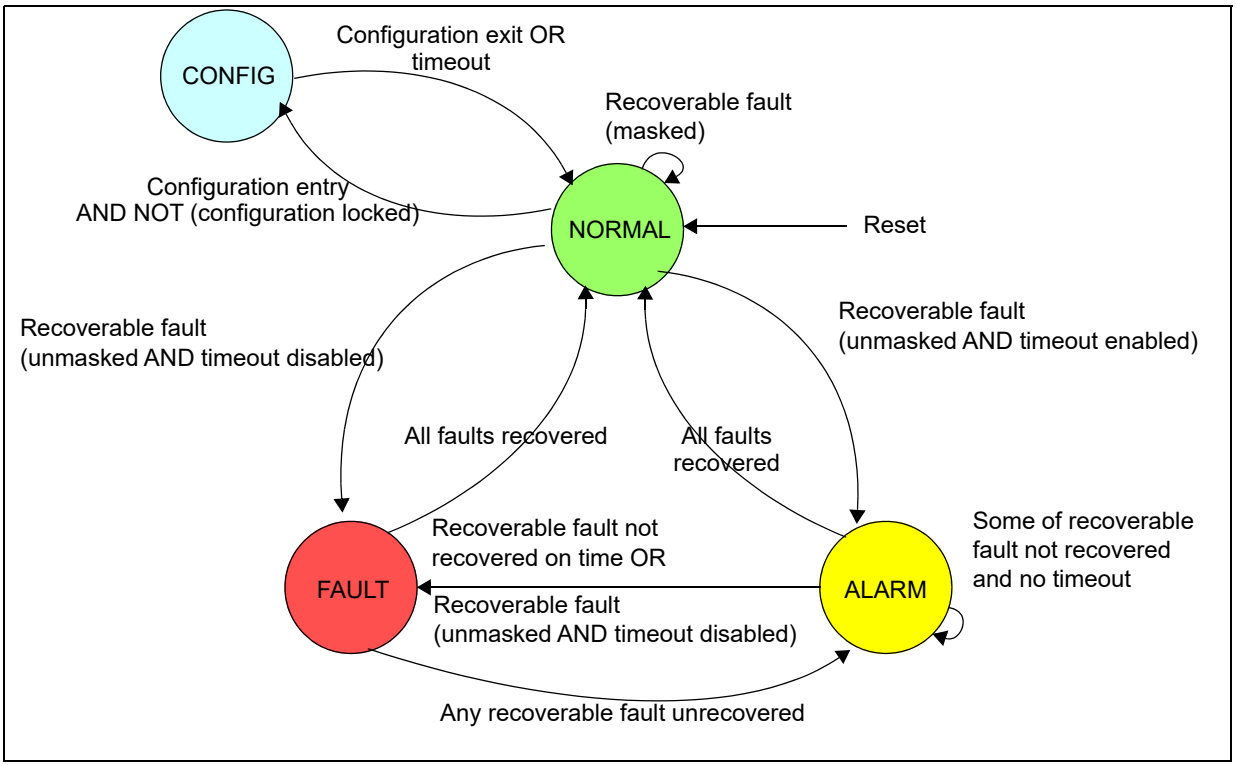
- FAULT: the FCCU moves into the FAULT state when one of the following conditions occurs:
  - timeout related to a recoverable fault when the FCCU is in the ALARM state.
  - unmasked recoverable faults with the timeout disabled.

The transition from NORMAL/ALARM state goes along with the generation of:

- NMI interrupt (optional).
- EOUT signaling (optional).
- SW option: Soft reaction (Short functional reset request pulse if configured).
- SW option: Hard reaction (Long functional reset request pulse if configured).

Non Maskable Interrupt (NMI) is routed only to the Safety Core.

Figure 2320. FCCU state diagram



89.5.3 Reset interface

The FCCU has two input resets, as described in *Table 2204* and two output resets related to the FAULT state. In FAULT state, each fault can be programmed to generate a soft or hard reaction request pulse.

In *Table 2205* for each type of fault reaction (hard, soft or self-checking), the respective output reset (Long functional reset, Short functional reset) is activated. In *Table 2206* for each type of reset source (external reset, POR, STCU, FCCU failure, short functional, long functional), the respective FCCU input reset state is described.

The MC unit generates the corresponding input reset as described in *Table 2204*.

Table 2204. Input reset

Input reset	Description
System reset	Global reset of the FCCU synchronous to the system clock.
System reset (RC oscillator)	Global reset of the FCCU synchronous to the IRCOSC.

Table 2205 and *Table 2206* describe a typical reset strategy at system level.

Table 2205. Reset reactions

FAULT reaction	Active reset	System reset signal
Hard	Long functional reset	disabled
Soft	Short functional reset	disabled

Table 2206 shows the Reset sources.

Table 2206. Reset sources

Reset source	System reset signal
External reset	enabled
POR	enabled
STCU	enabled
FCCU failure	application dependent
Functional reset	disabled

Table 2207 summarizes the MC actions corresponding to each reset generated by FCCU.

Table 2207. MC actions

Active reset	MC action
Long functional reset	Reset Flash and digital modules excluded FCCU, STCU
Short functional reset	Reset digital modules excluded FCCU, STCU

In reset state the FCCU is not able to collect any RF. The FCCU reset phase (POR, external reset, STCU, destructive reset, FCCU failure) must guarantee the assertion of the FCCU input resets at least for two IRCOSC clock periods.

89.5.4 Fault priority scheme and nesting

The FAULT state has a higher priority than the ALARM state in case of concurrent fault events (possible for RFs only) that occur in the NORMAL state.

The ALARM to FAULT state transition occurs if a RF (unmasked and with timeout disabled) is asserted in the ALARM state.

The ALARM to NORMAL state transition occurs only if all the RF (including the faults that have been collected after the entry in the ALARM state) have been cleared (SW or HW recovery) otherwise the FCCU will remain in the ALARM state.

The FAULT to NORMAL state transition occurs only if all the RF (including the faults that have been collected after the entry in the FAULT/ALARM state) have been cleared (SW or HW recovery) otherwise the FCCU will return in the ALARM state (if any RF is still pending and the timeout is not elapsed).

In general, no fault nesting is supported except for the RF that causes an ALARM to FAULT state transition. In this case the alarm timer is stopped until the FAULT state is recovered. If FCCU is in ALARM state and another fault comes, which has its alarm timeout enabled, then the alarm timer shall not reload and shall not start again.

89.5.5 Fault recovery

The following timing diagrams describe the main use cases of the FCCU in terms of fault events and related recovery.

A typical sequence related to a RF management (ALARM state), refer to *Figure 2321* and *Figure 2322*, is as follows:

- RF assertion
- FCCU state transition (automatic): NORMAL → ALARM
  - Alarm interrupt request (if configured)
  - Timeout running
- System state: RUN
- Alarm interrupt management: FAULT recovery (by SW): FCCU state transition ALARM → NORMAL

Figure 2321. RF (ALARM state) recovery (a)

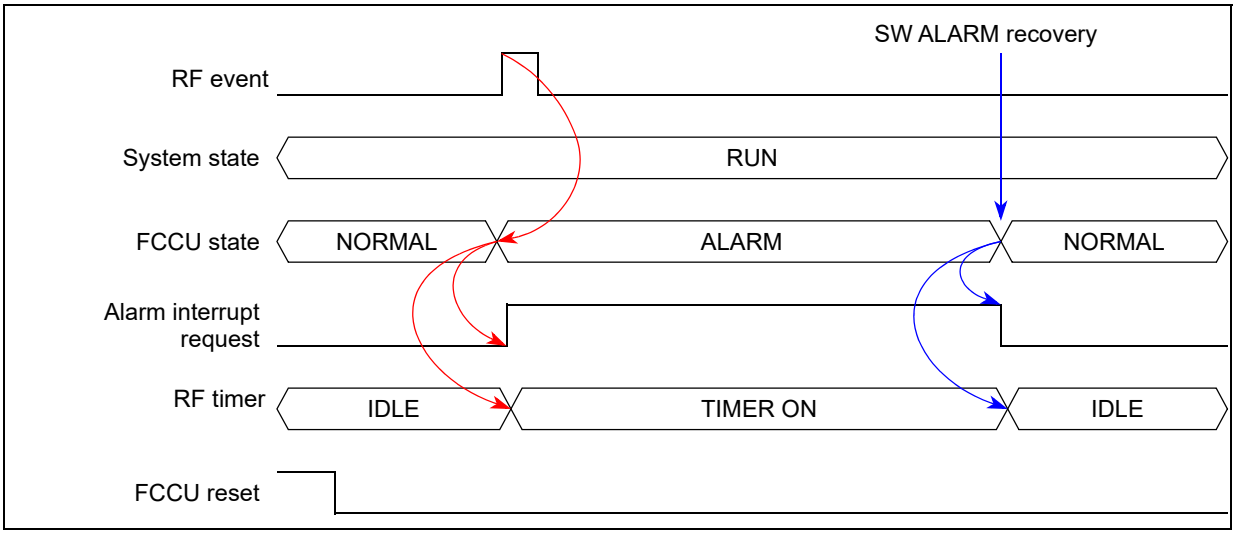
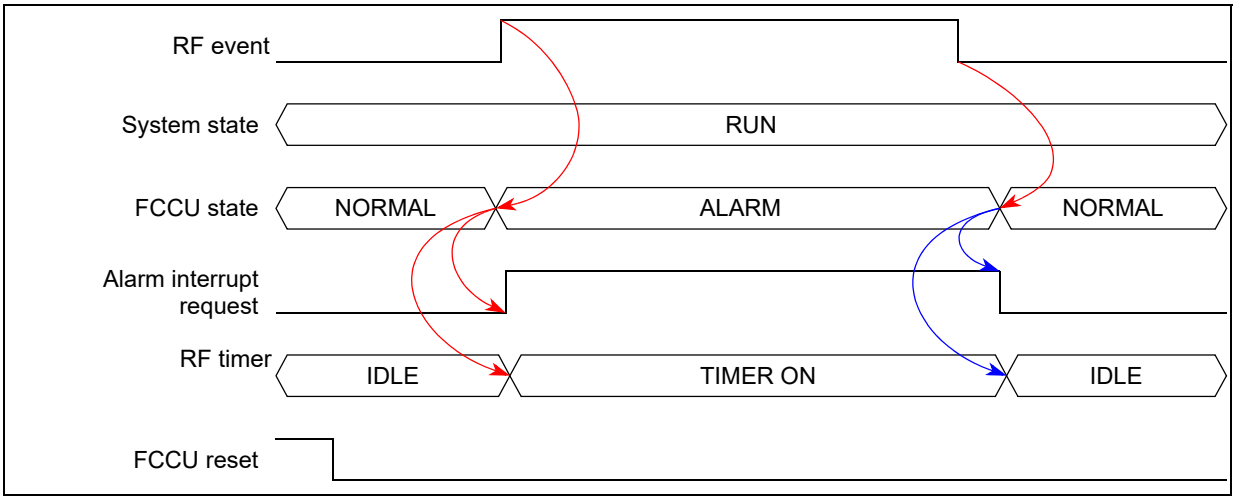


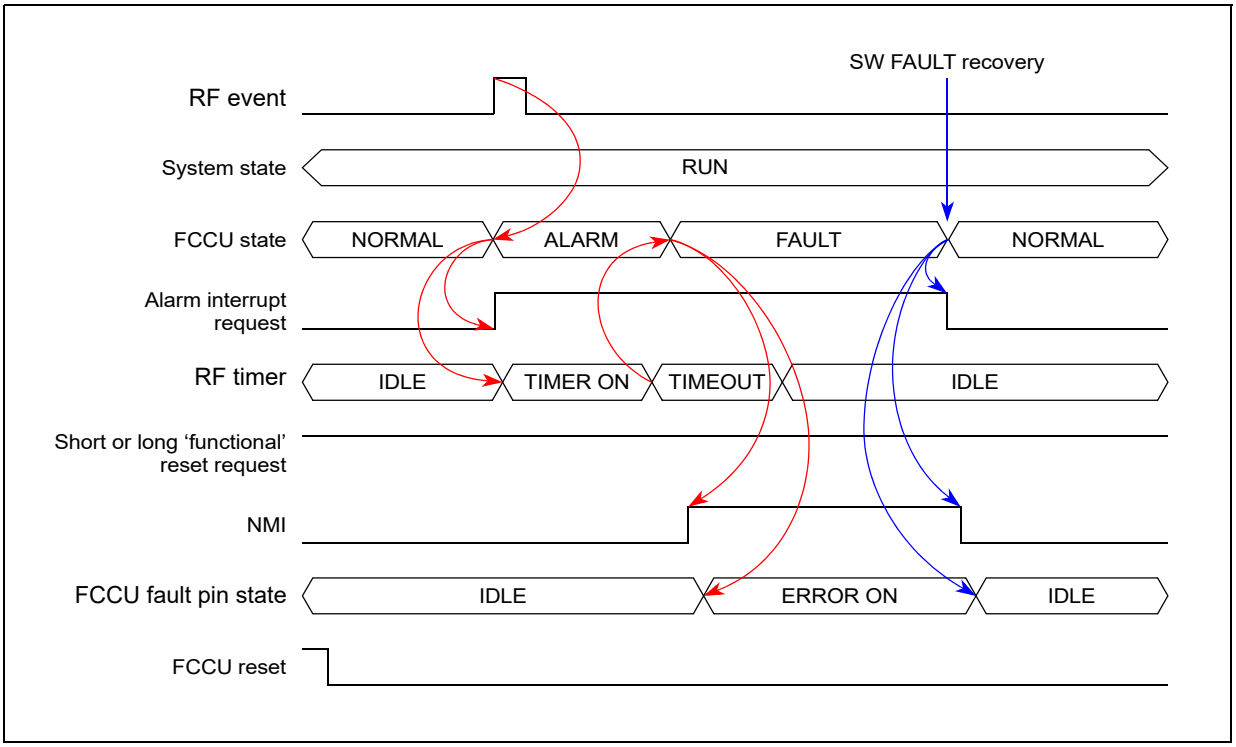
Figure 2322. RF (ALARM state) recovery (b)



A typical sequence related to a RF management (ALARM → FAULT state), refer to *Figure 2323*, is as follows:

- RF assertion
- FCCU state transition (automatic): NORMAL → ALARM
  - Alarm interrupt request (if enabled)
  - Timeout running
- FCCU state transition (following the timeout trigger): ALARM → FAULT: NMI assertion (if enabled)
- NMI interrupt management (if enabled)
  - FAULT recovery (by SW): FCCU state transition FAULT → NORMAL

Figure 2323. RF(ALARM -> FAULT state) recovery



89.5.6 NMI/WKPU interface

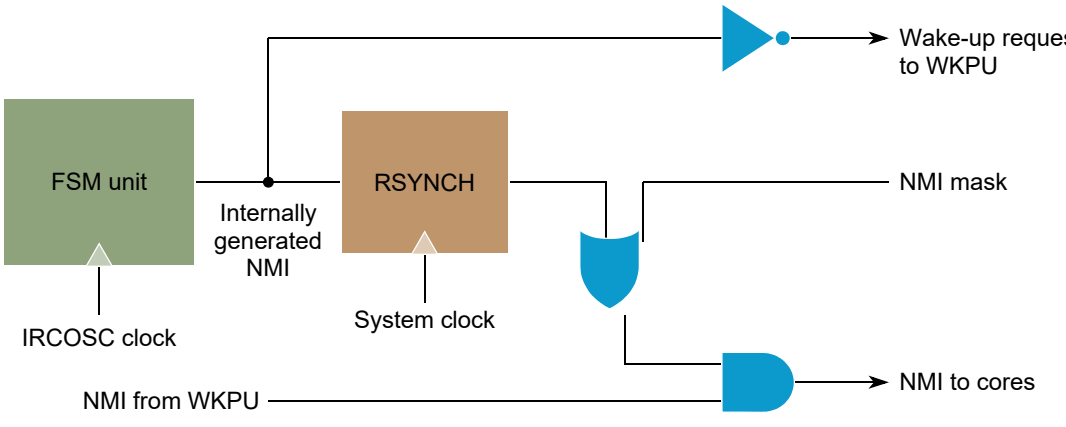
The wake-up interface includes Non-maskable interrupt: NMI interrupt source generated by the WKPU. It is synchronous with the system clock and active low. A logical AND interconnects the NMI sources (internally generated by FCCU and by WKPU) in order to provide a single NMI output.

The NMI signal internally generated by FCCU is masked when:

- FCCU asynchronous reset
- MC's current mode (MC state[3:0]) = RESET

and unmasked when a SW change request of the MC state is triggered (MC mode SW change request = 1).

Figure 2324. NMI/WKUP scheme



Note: All NMI signals are shown active-low.

89.5.7 FAULT interface

The basic functionality of the fault interface is to provide a simple, programmable edge-sensitive interface (active high or low) to signal a fault condition to the FCCU module.

To support a fault injection mechanism an additional and optional signal is available. It is an active high signal and is asserted for a single clock cycle synchronously at the system clock. The behavior is given in *Figure 2325*. The fault injection is executed by a write operation in the FCCU_RFF register.The reaction following a fake RF cannot be masked. The fault injection mechanism is optional; refer to the implementation-specific details. Refer to “FCCU failure inputs” table in Functional Safety chapter, for more details.

To support a fault clearing mechanism, in order to clear a fault latched directly in the FAULT root, an additional and optional signal (FCCU recoverable faults clear) is available. This active-high signal is asserted for a single clock cycle synchronously at the system clock. The behavior is given in *Figure 2326*. The fault clearing is executed when the SW “clears” the status (flag) bits of the FCCU_RF_S*n*. The fault clearing mechanism is optional and its support must be specified at system level.

Figure 2325. FAULT injection

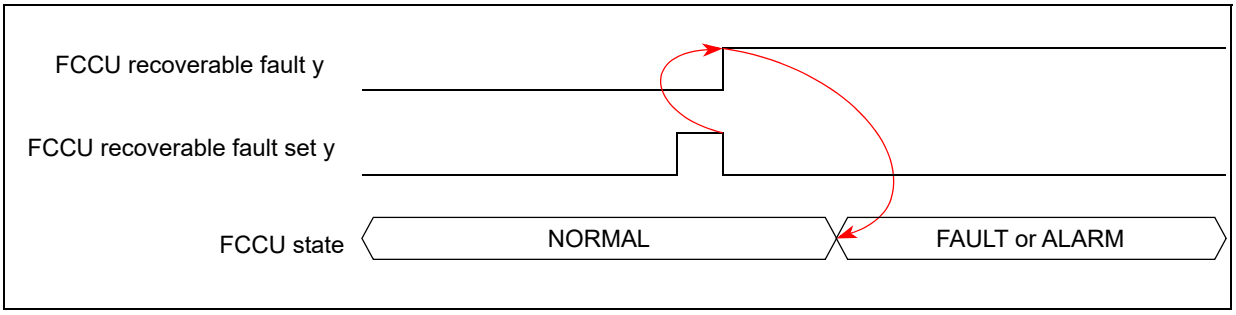
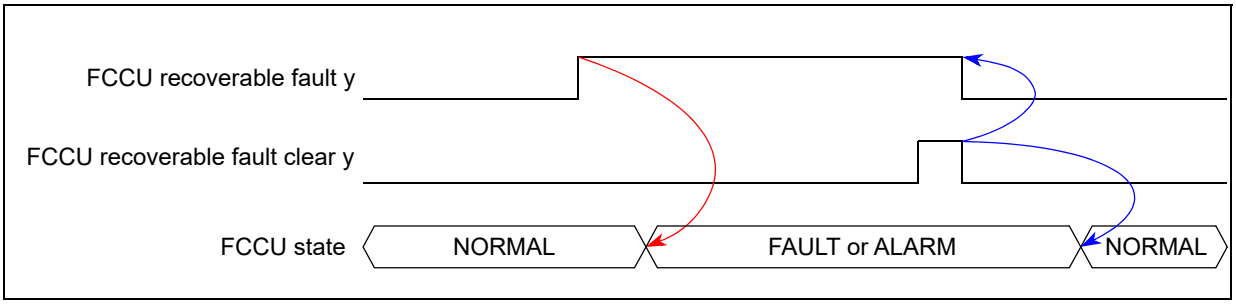


Figure 2326. FAULT clearing



89.5.8 STCU interface

The STCU interface includes:

- A set of signals resulting from the self-checking procedure connected externally at the following FCCU fault:
  - Recoverable
- The STCU fault signals are processed by the FCCU when the SOC is rebooted following the self-testing procedure. The STCU includes also a status register that stores the self-testing results (flags).
- A mask (EOUT mask[1:0]) that inhibits the EOUT dummy signaling until the STCU self-checking procedure has been completed.

During the self testing procedure, depending on the STCU results, three cases are applicable:

a) STCU completes the self testing procedure successfully. The SOC reboots and the FCCU is responsible for providing a reaction.

b) STCU completes the self testing procedure with low severity failures. The FCCU is responsible for providing the proper reactions according to the fault occurred.

c) STCU completes the self testing procedure with serious failures. The FCCU or other critical parts of the SoC could not be able to provide the proper reaction. STCU should keep permanently the SoC in RESET state. This feature is optionally programmable inside the STCU.

Figure 2327. STCU-FCCU (case a)

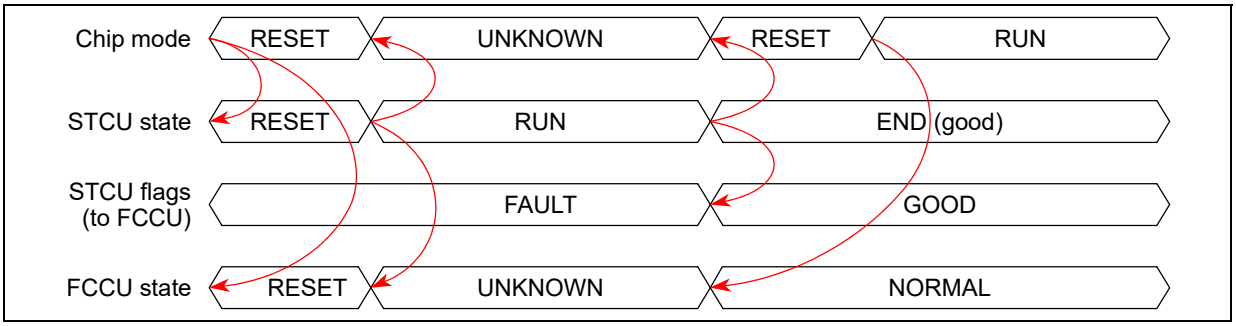


Figure 2328. STCU-FCCU (case b)

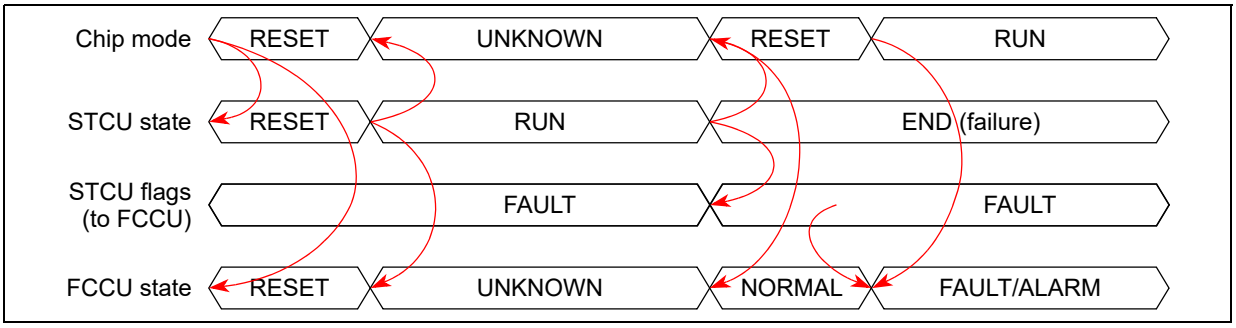
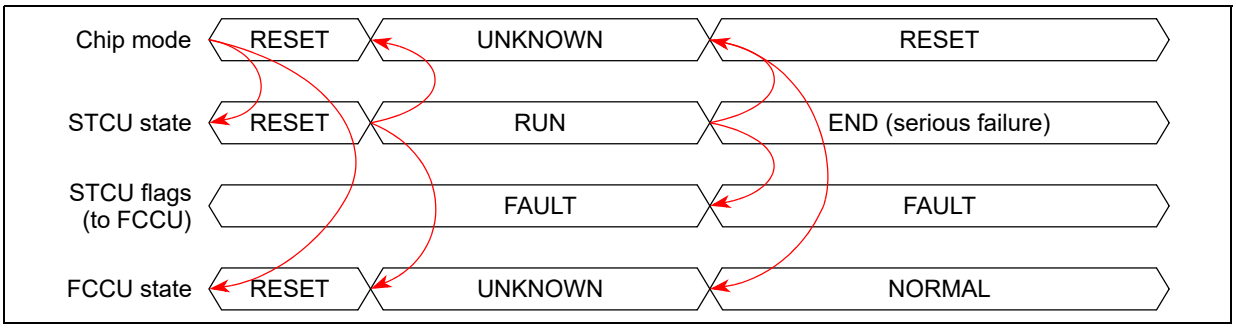


Figure 2329. STCU-FCCU (case c)



89.5.9 EOUT interface

The FCCU provides up to two bidirectional signals (EOUT interface) as a failure indication to the external world (refer to table “FCCU failure inputs” in the “Functional safety” chapter for details). These signals need to be mapped to appropriate pins for external visibility. Since SIUL2 does not control the pad buffer enable during Alternate function, FCCU provides a mechanism to drive the pad in OD (Open Drain) mode where the pad is driven Hi-Z when FCCU drives logic ‘1’ else it drives logic ‘0’ (when FCCU drives logic '0'). The selection to OD is done by OD bit in FCCU_CFG register. Different protocols for the EOUT interface are supported, selecting the FCCU_CFG.FOM register field:

- Dual rail protocol
- Time switching protocol
- Bi-stable protocol
- Test mode

The signal polarity can be programmed by setting the PS field in the FCCU_CFG register. For example, for PS = 0, FCCU Error output[0] = 0 and FCCU Error output[1] = 1 in FAULT state for bi-stable and for PS = 1, it is inverted. All the diagrams and tables are related to the default configuration selection: switching mode (FCCU_CFG.SM = 0b). In case of inverted polarity (FCCU_CFG.PS = 1b) all the values on the EOUT output pins are inverted.

Two modes can be programmed to define the EOUT protocol transitions in dual rail or time switching mode:

- slow switching mode: no EOUT frequency violation during the FCCU state transition (NORMAL to FAULT or vice-versa and CONFIG to NORMAL). The EOUT protocol

transition occurs after a max delay equal to the duration of the semi-period of the EOUT frequency.

- fast switching mode: The EOUT protocol transition (NORMAL to FAULT or vice-versa and CONFIG to NORMAL) occurs immediately. A pulse with the minimum duration corresponding to 16 MHz/1024 (IRCOSC clock) period can occur in fast switching mode. It implies a frequency violation of the EOUT protocol.

The EOUT frequency is generated by dividing IRCOSC clock by a fixed factor of 2^{18}. With an IRCOSC clock of 16 MHz, this drives a signal of 61 Hz on EOUT.

The external monitor of the EOUT protocol should oversample the EOUT signals in order to synchronize periodically the external clock (used by the monitor) and the IRCOSC clock detecting the edge transition of the EOUT protocol in dual rail or time switching mode.

In case of a failure event or on software request for Error pin indication, the pin(s) are set to faulty state for a minimum time T_min, even if SW tries to release it before. If SW configures the error pins to OK(1) and if a fault comes trying to drive the pin to NOK(0), then priority is given to the fault indication and error pins indicate NOK, that is an incoming fault is not masked when SW has set the error pin to high. During the T_min by a non-SW fault, the FCCU FSM moves independently of this pin state (low) and as soon as the timer expires, the pin behavior is dictated by the state in which the FSM finds itself in and it is not possible to set the pins to OK by SW moving FCCU to CONFIG state, as long as this timer is running. No SW intervention is needed to bring the pin from the low state. The SW can bring the pin back to OK state by clearing the faults and waiting for the T_min to expire, after which the FCCU automatically enters NORMAL state and the error pin indicates OK.

In case another failure event happens within T_min after a first one, the T_min counter is restarted.

Resets (except the power on reset, and software bit FCCU_RST in RGM_PRST5) should not influence the state of the failure indication, pins.

Note: The transition from FAULT to CONFIG state is not possible but is shown to display the behavior in all four states: RESET, NORMAL, FAULT and CONFIG.

Table 2208. FCCU error pin behavior on STATE transitions(1)

Fault signaling state	Transition from NORMAL to FAULT state			NORMAL+CONFIG states			Transition from NORMAL to ALARM state		
	with Fault disabled | enabled						with Fault disabled | enabled		
	Internal Error pin	Error pin enable	Error pad	Internal error pin	Error pin enable	Error pad	Internal error pin	Error pin enable	Error pad
Disabled	NA | OK	NA | 1	NA | OK	OK	1	OK	NA | OK	NA | 1	NA | OK
Enabled	NA | NOK	NA | 1	NA | NOK	OK	1	OK	NA | OK	NA | 1	NA | OK

1. Table valid only after SW has written FCCU_SET_AFTER_RESET bit to 1.

Note: FCCU remains in NORMAL state when fault is disabled hence, “NA” (Not Applicable) appears in the above table.

89.5.9.1 Dual rail protocol

Dual rail encoding is an alternate method for encoding bits. In contrast with classical encoding, where each signal carries a single-bit value, dual rail encoded circuits use two wires to carry each bit. The encoding scheme is given in *Table 2209* and the related timing diagram is given in *Figure 2330* and *Figure 2331*.

Table 2209. Dual rail encoding

Logical state	Dual rail encoding (FCCU Error output pins [1:0])	Note
nonfaulty	10	toggling
nonfaulty	01	
faulty	00	toggling
faulty	11	
reset	high-Z(1)	no toggling
configuration	same as NORMAL(2)	toggling

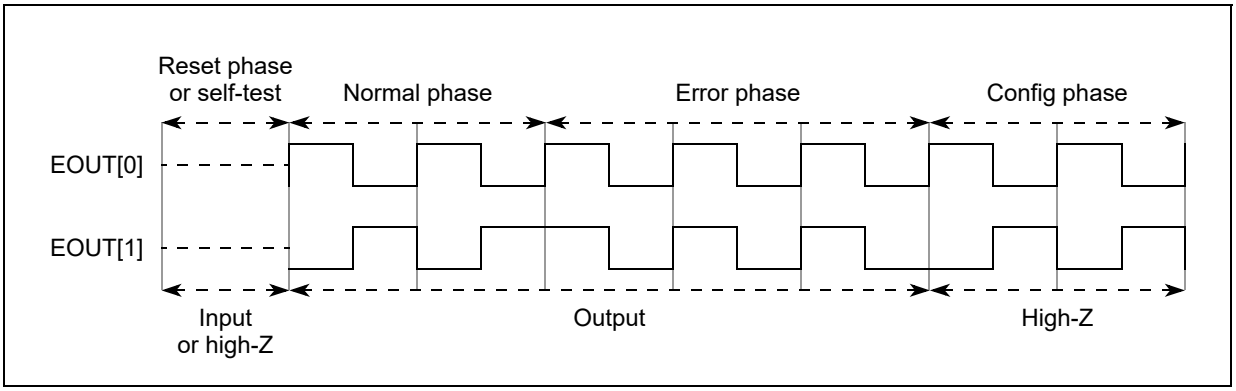
1. Final value depends on the SoC settings at pad level.
2. Till FCCU_SET_AFTER_RESET is written, the Error indicating pads are driven to High impedance from FCCU's side. That bit is write-able only after the FCCU enters CONFIG state. Hence on entry to CONFIG state, user may still see high impedance.

As long as FCCU is in NORMAL or ALARM state, output will show “nonfaulty” signal. Output pins #0, #1 will toggle between 01 and 10.

In the RESET phase the output pins are set as “high impedance”(32).

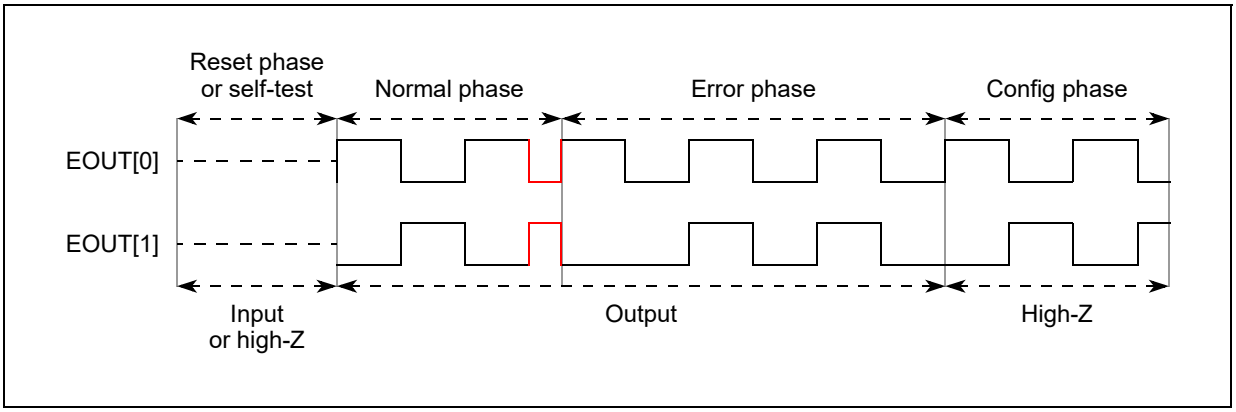
Note: Figure 2330 and Figure 2331 are formatted to display the behavior in all four phases (reset, normal, error, and config), not to imply transitions between one phase to another. In particular, transition from error phase to config phase is not possible.

Figure 2330. Dual rail protocol (slow switching mode)



32. Actual value depends on the SOC settings at pad level.

Figure 2331. Dual rail protocol (fast switching mode)



89.5.9.2 Time switching protocol

The encoding scheme is given in *Table 2210* and the related timing diagram is given in *Figure 2332*.

Table 2210. Time switching encoding

Logical state	Time switching encoding (FCCU Error output pins [1:0])	Note
nonfaulty	10	toggling
nonfaulty	01	
faulty	10	no toggling
reset	high-Z(1)	no toggling
configuration	same as NORMAL(2)	toggling

1. Final value depends on the SoC settings at pad level.
2. Till FCCU_SET_AFTER_RESET is written, the Error indicating pads are driven to High impedance from FCCU's side. That bit is write-able only after the FCCU enters CONFIG state. Hence on entry to CONFIG state, user may still see high impedance.

As long as FCCU is in NORMAL or ALARM state, outputs will show “nonfaulty” signal. Output pins #0, #1 will toggle between 01 and 10.

In the FAULT state, the output pin FCCU Error output pin[0] is set as low.

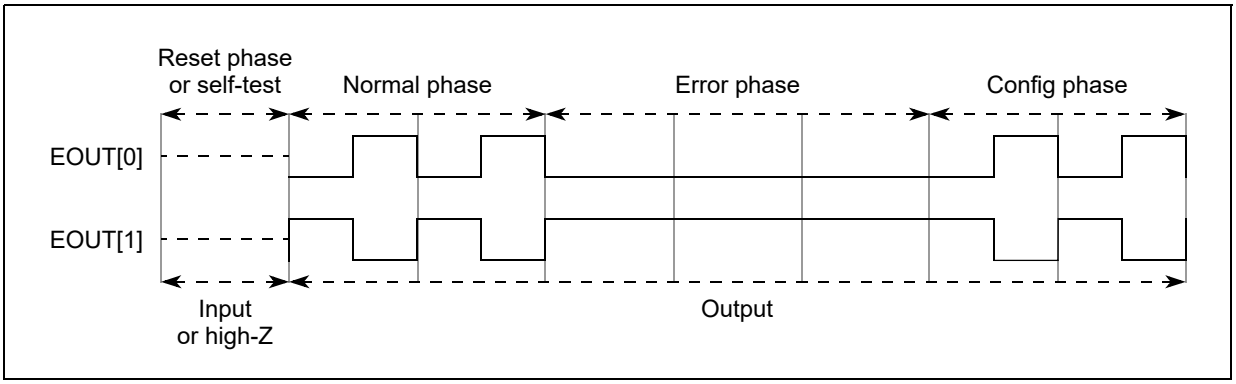
In Time Switching mode the second output (FCCU Error output pin[1]) is the inverted signal of first output (FCCU Error output pin[0]).

In the RESET phase the output pins are set as “high impedance”(33).

Note: Figure 2332 is formatted to display the behavior in all four phases (reset, normal, error, and config), not to imply transitions between one phase to another. In particular, transition from error phase to config phase is not possible.

33. Actual value depends on the SOC settings at pad level.

Figure 2332. Time switching protocol



89.5.9.3 Bi-stable protocol

The encoding scheme is given in *Table 2211* and the related timing diagram is given in *Figure 2333*.

Table 2211. Bi-stable encoding

Logical state	Bi-stable encoding (FCCU Error output pins [1:0])	Note
nonfaulty	01	no toggling
faulty	10	no toggling
reset	high-Z(1)	no toggling
configuration	same as NORMAL(2)	no toggling

1. Final value depends on the SoC settings at pad level.
2. Till FCCU_SET_AFTER_RESET is written, the Error indicating pads are driven to High impedance from FCCU's side. That bit is write-able only after the FCCU enters CONFIG state. Hence on entry to CONFIG state, user may still see high impedance.

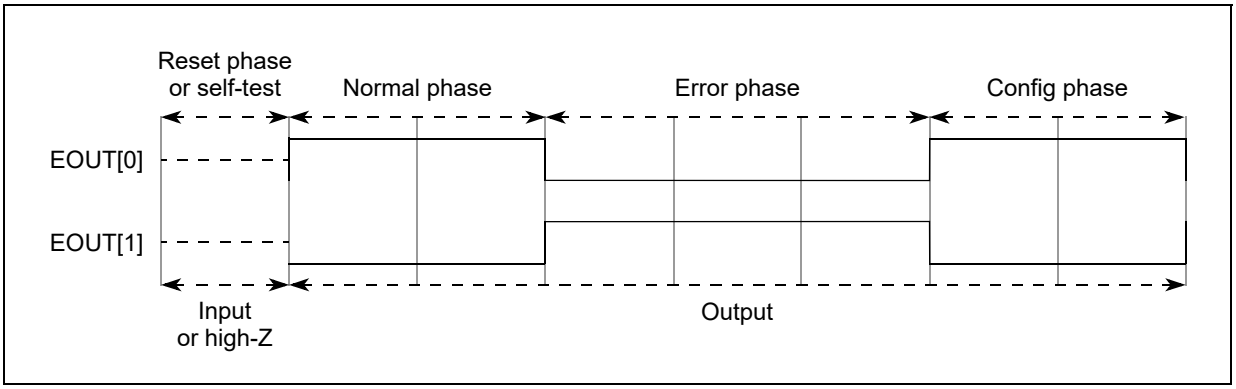
In the FAULT state, the faulty logical state is indicated. In NORMAL or ALARM state, “no-faulty” state is indicated. In Bi-stable mode the second output (FCCU Error output[1]) is the inverted signal of first output (FCCU Error output[0]).

In the RESET phase the output pins are set as “high impedance”(34).

Note: Figure 2333 is formatted to display the behavior in all four phases (reset, normal, error, and config), not to imply transitions between one phase to another. In particular, transition from error phase to config phase is not possible.

34. Actual value depends on the SOC settings at pad level.

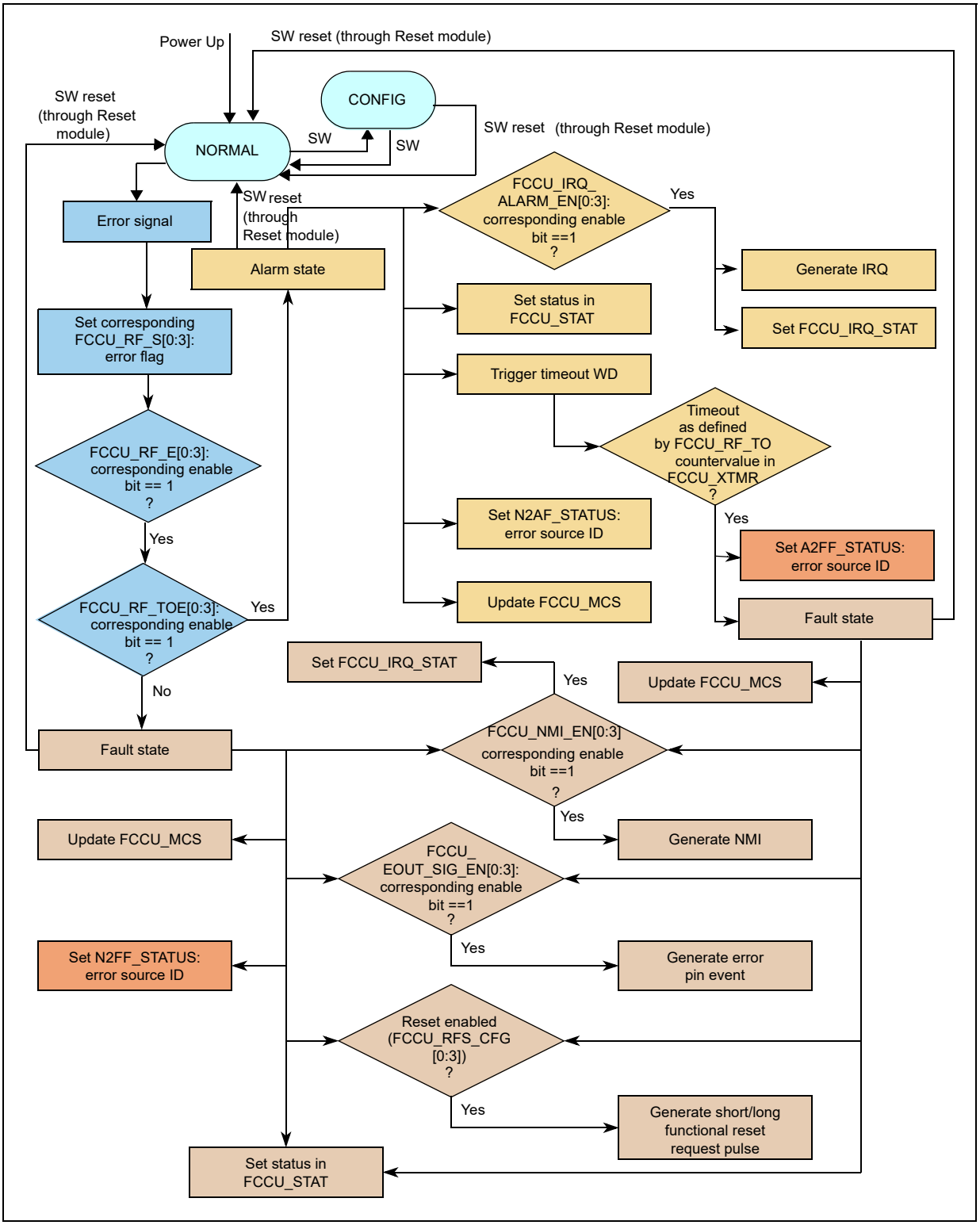
Figure 2333. Bi-stable protocol



89.5.10 Error signal flow for RF case

On Power up, FCCU moves to NORMAL state. SW can configure it by moving it to CONFIG and back again. SW reset leads to movement back to NORMAL. On reaching ALARM state, depending on ALARM being enabled, FCCU generates IRQ (alarm) and the FCCU_IRQ_STAT's alarm status is also set. FSM status is set in FCCU_STAT along with N2AF_STATUS. If Error is enabled (FCCU_RF_E) and timeout disabled (FCCU_RF_TOE), FCCU moves straight to FAULT state and depending on FCCU_NMI_EN or FCCU_SIG_EN, or both, being set, it generates a NMI or signals error, or both, out to the external world. FCCU FSM status is also captured in FCCU_STAT.

Figure 2334. Error signal flow



89.6 Register description

The FCCU registers are listed in *Table 2212*. Any address offset not explicitly mentioned in *Table 2212* is reserved.

All the registers accessible in write mode only in CONFIG state (last column of the *Table 2212*) are referred as configuration registers. These configuration registers return to the default value after configuration watchdog timer expires. These are the registers protected by FCCU_TRANS_LOCK and FCCU_PERMNT_LOCK registers. The configuration register setting has effect only when the FCCU state exits from the CONFIG state.

For each possible RF failure source a different reaction shall be configurable through the use of NMI, IRQ, long/short reset selection registers as well as no reaction by disabling the former registers. It is not possible for a single event upset to switch off all reactions on failures as implementation is per fault source (but it will be possible to switch them all off by SW if intended). Failures themselves are not able to disable all reactions and indications.

The FCCU is not reset by short or long functional resets.

Table 2212. FCCU memory map

Address offset	Register	Access[1]	Location
0x000	FCCU Control Register (FCCU_CTRL)	R/W always	*Section 89.6.1*
0x004	FCCU CTRL Key Register (FCCU_CTRLK)	W always	*Section 89.6.2*
0x008	FCCU Configuration Register (FCCU_CFG)	R always; W in CONFIG state only	*Section 89.6.3*
0x01C	FCCU RF Configuration Register 0 (FCCU_RF_CFG0)	R always; W in CONFIG state only	*Section 89.6.4*
0x020	FCCU RF Configuration Register 1 (FCCU_RF_CFG1)		
0x024	FCCU RF Configuration Register 2 (FCCU_RF_CFG2)		
0x028	FCCU RF Configuration Register 3 (FCCU_RF_CFG3)		
0x04C	FCCU RFS Configuration Register 0 (FCCU_RFS_CFG0)	R always; W in CONFIG state only	*Section 89.6.5*
0x050	FCCU RFS Configuration Register 1 (FCCU_RFS_CFG1)		
0x054	FCCU RFS Configuration Register 2 (FCCU_RFS_CFG2)		
0x058	FCCU RFS Configuration Register 3 (FCCU_RFS_CFG3)		
0x05C	FCCU RFS Configuration Register 4 (FCCU_RFS_CFG4)		
0x060	FCCU RFS Configuration Register 5 (FCCU_RFS_CFG5)		
0x064	FCCU RFS Configuration Register 6 (FCCU_RFS_CFG6)		
0x068	FCCU RFS Configuration Register 7 (FCCU_RFS_CFG7)		
0x080	FCCU RF Status Register 0 (FCCU_RF_S0)	R/W always	*Section 89.6.6*
0x084	FCCU RF Status Register 1 (FCCU_RF_S1)		
0x088	FCCU RF Status Register 2 (FCCU_RF_S2)		
0x08C	FCCU RF Status Register 3 (FCCU_RF_S3)		

Table 2212. FCCU memory map (continued)

Address offset	Register	Access(1)	Location
0x090	FCCU RF Key Register (FCCU_RFK)	W always	*Section 89.6.7*
0x094	FCCU RF Enable Register 0 (FCCU_RF_E0)	R always; W in CONFIG state only	*Section 89.6.8*
0x098	FCCU RF Enable Register 1 (FCCU_RF_E1)		
0x09C	FCCU RF Enable Register 2 (FCCU_RF_E2)		
0x0A0	FCCU RF Enable Register 3 (FCCU_RF_E3)		
0x0A4	FCCU RF Timeout Enable Register 0 (FCCU_RF_TOE0)	R always; W in CONFIG state only	*Section 89.6.9*
0x0A8	FCCU RF Timeout Enable Register 1 (FCCU_RF_TOE1)		
0x0AC	FCCU RF Timeout Enable Register 2 (FCCU_RF_TOE2)		
0x0B0	FCCU RF Timeout Enable Register 3 (FCCU_RF_TOE3)		
0x0B4	FCCU RF Timeout Register (FCCU_RF_TO)	R always; W in CONFIG state only	*Section 89.6.10*
0x0B8	FCCU CFG Timeout Register (FCCU_CFG_TO)	R always; W in all states except CONFIG	*Section 89.6.11*
0x0BC	FCCU IO Control Register (FCCU_EINOUT)	R/W always	*Section 89.6.12*
0x0C0	FCCU Status Register (FCCU_STAT)	R only	*Section 89.6.13*
0x0C4	FCCU NA Freeze Status Register (N2AF_STATUS)	R/W always	*Section 89.6.14*
0x0C8	FCCU AF Freeze Status Register (A2FF_STATUS)	R/W always	*Section 89.6.15*
0x0CC	FCCU NF Freeze Status Register (N2FF_STATUS)	R/W always	*Section 89.6.16*
0x0D0	FCCU FA Freeze Status Register (F2A_STATUS)	R/W always	*Section 89.6.17*
0x0DC	FCCU RF Fake Register (FCCU_RFF)	R/W always	*Section 89.6.18*
0x0E0	FCCU IRQ Status Register (FCCU_IRQ_STAT)	R/W always	*Section 89.6.19*
0x0E4	FCCU IRQ Enable Register (FCCU_IRQ_EN)	R/W always	*Section 89.6.20*
0x0E8	FCCU XTMR Register (FCCU_XTMR)	R/W always	*Section 89.6.21*
0x0EC	FCCU MCS Register (FCCU_MCS)	R only	*Section 89.6.22*
0x0F0	FCCU Transient Lock Register (FCCU_TRANS_LOCK)	W always	*Section 89.6.23*
0x0F4	FCCU Permanent Lock Register (FCCU_PERMNT_LOCK)	W always	*Section 89.6.24*
0x0F8	FCCU Delta T Register (FCCU_DELTA_T)	R always; W in CONFIG state only	*Section 89.6.25*

Table 2212. FCCU memory map (continued)

Address offset	Register	Access[1]	Location
0x0FC	FCCU IRQ Alarm Enable Register 0 (FCCU_IRQ_ALARM_EN0)	R always; W in CONFIG state only	*Section 89.6.26*
0x100	FCCU IRQ Alarm Enable Register 1 (FCCU_IRQ_ALARM_EN1)		
0x104	FCCU IRQ Alarm Enable Register 2 (FCCU_IRQ_ALARM_EN2)		
0x108	FCCU IRQ Alarm Enable Register 3 (FCCU_IRQ_ALARM_EN3)		
0x10C	FCCU NMI Enable Register 0 (FCCU_NMI_EN0)	R always; W in CONFIG state only	*Section 89.6.27*
0x110	FCCU NMI Enable Register 1 (FCCU_NMI_EN1)		
0x114	FCCU NMI Enable Register 2 (FCCU_NMI_EN2)		
0x118	FCCU NMI Enable Register 3 (FCCU_NMI_EN3)		
0x11C	FCCU EOUT Signaling Enable Register 0 (FCCU_EOUT_SIG_EN0)	R always; W in CONFIG state only	*Section 89.6.28*
0x120	FCCU EOUT Signaling Enable Register 1 (FCCU_EOUT_SIG_EN1)		
0x124	FCCU EOUT Signaling Enable Register 2 (FCCU_EOUT_SIG_EN2)		
0x128	FCCU EOUT Signaling Enable Register 3 (FCCU_EOUT_SIG_EN3)		

1. In this column, R/W = read/write, R = read-only, and W = write-only.

89.6.1 FCCU Control Register (FCCU_CTRL)

The FCCU_CTRL register allows to execute the following operations:

- moving the FCCU state from the NORMAL state into the CONFIG state.
- moving the FCCU state from the CONFIG state into the NORMAL state.
- reading or to clear the RF status register.
- reading the FCCU FSM status register.
- reading or to clear the FCCU freeze registers.
- reading the ALARM timer.
- reading the Watchdog timer.
- configure the glitch filter present on EIN0 input (FCCU failure input #96).

Some critical operations require a key as defined in the FCCU_CTRLK register.

The Glitch filter, filters out pulses of widths up to 50/75/100 µs, as governed by FCCU_CTRL[FILTER_WIDTH]. It becomes active as soon as FCCU_CTRL[FILTER_BYPASS] is de-asserted (default value).

The implementation is as follows:

1. Fault line assertion - If a Fault is asserted, then the output of the filter is asserted after the delay, equal to the programmed glitch width time. If the fault is de-asserted before

the programmed glitch width time, then the output of the filter is not asserted and stays at the deactivated level (after having rejected the spurious glitch).

2. Fault line de-assertion - If a Fault, which was kept asserted for a time greater than the programmed pulse width is de-asserted (and glitch filter output is at asserted level now), then the output of the filter is de-asserted only after the delay equal to the programmed glitch width time. If the fault is again asserted before the programmed glitch width time, then the output of the filter is not de-asserted and stays at the active level (after having rejected the spurious glitch).

The output of the filter becomes the effective fault input fed to the FCCU error management engine. Since there is an inherent latency involved, SW must either wait for the programmed delay value before trying to clear the fault status flag, FCCU failure input #96 or first set FCCU_CTRL[FILTER_BYPASS] to one and then try to clear the fault status flag. This bit should not be made '1' again, till the filter width time elapses, else fault status will be latched again.

Offset: 0x000 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	FILTER_BYPASS	FILTER_WIDTH		0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	DEBUG	0	OPS		0	OPR				
W																
Reset	0	0	0	0	0	0	0	0	0/1(1)	0/1(1)	0	0	0	0	0	0

1. Refer to the Device Configuration chapter for chip-specific reset value.

Figure 2335. FCCU Control Register (FCCU_CTRL)

Table 2213. FCCU_CTRL field descriptions

Field	Description
0 FILTER_BYPASS	Filter bypass 0 glitch filter not bypassed 1 glitch filter bypassed
1:2 FILTER_WIDTH	Filter width 00 filters glitches up to 50 µs 01 filters glitches up to 75 µs 10 filters glitches up to 100 µs for 20 MHz IRC clock 11 filters glitches up to 100 µs for 20 MHz IRC clock Make sure that the filter width does not cross the FOSU timeout value else unintended reset may occur.

Table 2213. FCCU_CTRL field descriptions (continued)

Field	Description
22 DEBUG	Debug mode entry 0 Normal operation 1 Put FCCU into debug mode When Debug pin signal is asserted and this bit is set, FCCU moves into debug state and suspend the transition of the FSM by suspending timer elapse time effect.
24:25 OPS	Operation status 00 Idle 01 In progress 10 Aborted 11 Successful These bits can be read and cleared (via OP15 operation) by the software.

Table 2213. FCCU_CTRL field descriptions (continued)

Field	Description
27:31 OPR	Operation run 00000 No operation [OP0] 00001 Set the FCCU into the CONFIG state [OP1] 00010 Set the FCCU into the NORMAL state [OP2] 00011 Read the FCCU state (refer to the FCCU_STAT register) [OP3] 00100 Read the FCCU frozen status flags (refer to the N2AF_STATUS register) [OP4] 00101 Read the FCCU frozen status flags (refer to the A2FF_STATUS register) [OP5] 00110 Read the FCCU frozen status flags (refer to the N2FF_STATUS register) [OP6] 00111 Read the FCCU frozen status flags (refer to the F2A_STATUS register) [OP7] 01000 Reserved 01001 Reserved 01010 Read the RF status register (refer to the FCCU_RF_S register) [OP10] 01011 Reserved 01100 RF status clear operation in progress (refer to the FCCU_RF_S register) [OP12] 01101 Clear the freeze status registers (refer to the freeze registers) [OP13] 01110 CONFIG to NORMAL FCCU state (configuration timeout) in progress [OP14] 01111 Clear the operation status (OPS=Idle) [OP15] 10000 Reserved 10001 Read the ALARM timer (refer to the FCCU_XTMR register) [OP17] 10010 RESERVED 10011 Read the CFG timer (refer to the FCCU_XTMR register) [OP19] 10100 Read the Error Pin low counter value (refer to *Section 89.6.21: FCCU XTMR Register (FCCU_XTMR)*) [OP20] 10101 Reserved ... 11111 Reserved The SW application must not modify the OPR field (any write operation will be ignored) until the completion of the operation. Once the operation has been completed, the operation status (OPS) is set and the OPR field is automatically cleared (OPR = 000). The opcode OP11, OP12 must not be programmed. The OPR field is automatically set to OP11 or OP12 when the FCCU_RF_S registers are cleared by a write-clear operation into the related register. The opcode OP14 must not be programmed. The OPR field is automatically set to OP14 when the timeout occurs (FCCU_CFG_TO) during the configuration procedure => the FCCU state is automatically forced in NORMAL mode setting the default configuration. In this phase any write operation to the FCCU configuration registers is inhibited. The reserved operations (OP21 -OP30) are forbidden. In case of execution, they return an ABORT response without any side effect. The ABORT response occurs in the following cases: – wrong access (missing or wrong key) to the FCCU_RF_S register (clear operation OP12) – wrong access (missing or wrong key) to the FCCU_CTRL register (OP1, OP2 operation) – OP1 (CONFIG command) execution when FCCU state /= NORMAL or configuration locked – OP21-OP30 (RESERVED operations) execution The max timing needed for ABORT/SUCCESS response is 15 IRC cycles

89.6.2 FCCU CTRL Key Register (FCCU_CTRLK)

The FCCU_CTRLK register implements the key access for the operations OP1, OP2 according to the following sequence:

1. Write the key into the FCCU_CTRLK register.
2. Write the FCCU_CTRL register (operations OP1 or OP2).

Note: There shall not be any other operation in between the above steps and the 32 bits need to be written in one cycle. So, store FCCU_CTRLK, merge with OP1 or OP2 then send it.

The FCCU_CTRLK register is not readable, a 0x0000_0000 value is always returned in case of read operation. The key must be written by a word (32 bits) data write operation.

Offset: 0x004 Access: User write-only

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R																
W	CTRLK_VAL															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R																
W	CTRLK_VAL															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 2336. FCCU CTRL Key Register (FCCU_CTRLK)

Table 2214. FCCU_CTRLK field descriptions

Field	Description
0:31 CTRLK_VAL	Control register key 0x913756AF Key for the operation OP1 0x825A132B Key for the operation OP2

89.6.3 FCCU Configuration Register (FCCU_CFG)

The FCCU_CFG register defines the global configuration for the FCCU module.

This register is writable only in the CONFIG state.

Offset: 0x008 Access: User read/write(1)

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0							FCCU_SET_AFTER_RESET	FCCU_SET_CLEAR		0	0	Reserved			
W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	Reserved	Reserved		OD	0	SM	PS	FOM			Reserved					
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0(2)	0	0	0

1. Writable only in the CONFIG state.
2. After powerup, reset value will be 0. If configuration timeout occurs then default value will be changed to 1.

Figure 2337. FCCU Configuration Register (FCCU_CFG)

Table 2215. FCCU_CFG field descriptions

Field	Description
7 FCCU_SET_AFTER_RESET	This bit controls the enable of the o/p error pin after reset lifts 0 FCCU's Error indication stops functioning and error pins are in HI-Z state(1). 1 write to start FCCU functioning. After power-on the error out pins shall be in high impedance(1). They will go to normal state only on software request, this bit is set to 1. It is Software responsibility to write this bit to 1 else the error pin stays in High-Z state after reset lifts and FCCU is not functioning.
8:9 FCCU_SET_CLEAR	Error pin state can be controlled by these bits 01 Error Pin driven to logic 0. 11 Error Pin driven to logic 1 (write not possible when FCCU already in the T min timer phase or if FSM enters in FAULT state by capture of a fault). 00 FCCU acts independent of above SW control. 10 FCCU acts independent of above SW control. These bits clear(0) and set(1) the error pin. Higher priority is of the FCCU_SET_AFTER_RESET bit's capability to lead the error pins to Hi-Z(1). Switch to software configuration (01, 11) comes into effect when FCCU FSM leaves CONFIG state. After this is in effect, software maintains priority over FCCU FSM except for the case of '11' mentioned above. The switch back to FSM control is again done by writing into these fields with 00 or 10 by going to CONFIG state. This bit feature may not work with toggling protocols for example say EOUT pins are set high by SW using this bit followed by switching to a toggling protocol after returning control to FCCU FSM via setting this bit to 00 or 10, the user may find that EOUT pins do not toggle as was being expected.

Table 2215. FCCU_CFG field descriptions (continued)

Field	Description
19 OD	Open Drain Mechanism to select between Push-pull and Open drain (OD) mode for the error indicating pin(s). 0 push-pull 1 OD
21 SM	Switching mode 0 EOUT protocol (dual rail, time switching) slow switching mode. 1 EOUT protocol (dual rail, time switching) fast switching mode. SM has no effect on the bi-stable protocol.
22 PS	Polarity selection 0 FCCU Error output[1] active high, FCCU Error output[0] active low. 1 FCCU Error output[1] active low, FCCU Error output[0] active high. This bit can be read and written by the software. Active state here means a state indicating FAULT and applicable when pins are non toggling in Error phase.
23:25 FOM	Fault Output Mode selection 000 Dual Rail (default state) [FCCU Error output[1:0]= outputs] 001 Time Switching [FCCU Error output[1:0]= output to be used] 010 Bi-Stable 011 Reserved 100 Reserved 101 Test0 (controlled by FCCU_EOUT register) [FCCU Error output[0]= input, FCCU Error output[1]= output] 110 Test1 (controlled by FCCU_EOUT register) [FCCU Error output[0]= output, FCCU Error output[1]= output] 111 Test2 (controlled by FCCU_EOUT register) [FCCU Error output[0]= output, FCCU Error output[1]= input] **Note:** In Test mode, a simple double-stage resynchronization stage is used to resynchronize the EOUT input/outputs on the system/IRCOSC clock.

1. Actual value depends on the SOC settings at pad level.

89.6.4 FCCU RF Configuration Register *n* (FCCU_RF_CFG*n*)

The FCCU_RF_CFG*n* registers contain the configuration of each recoverable fault in terms of fault recovery management. The configuration depends on the type of signaling of a fault event. HW recoverable faults should be configured only if a previous latching stage captures and hold the physical fault otherwise the fault can be lost. All the other faults should be configured as SW fault.

These registers are writable only in the CONFIG state.

Offset: 0x01C + *n**0x4 (*n* = 0 to 3) Access: User read/write(1)

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	RFC31(2)	RFC30(2)	RFC29(2)	RFC28(2)	RFC27(2)	RFC26(2)	RFC25(2)	RFC24(2)	RFC23(2)	RFC22(2)	RFC21(2)	RFC20(2)	RFC19(2)	RFC18(2)	RFC17(2)	RFC16(2)
W																
Reset:	0/1(3)	0/1(3)	0/1(3)	0/1(3)	0/1(3)	0/1(3)	0/1(3)	0/1(3)	0/1(3)	0/1(3)	0/1(3)	0/1(3)	0/1(3)	0/1(3)	0/1(3)	0/1(3)

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	RFC15(2)	RFC14(2)	RFC13(2)	RFC12(2)	RFC11(2)	RFC10(2)	RFC9(2)	RFC8(2)	RFC7(2)	RFC6(2)	RFC5(2)	RFC4(2)	RFC3(2)	RFC2(2)	RFC1(2)	RFC0(2)
W																
Reset:	0/1(3)	0/1(3)	0/1(3)	0/1(3)	0/1(3)	0/1(3)	0/1(3)	0/1(3)	0/1(3)	0/1(3)	0/1(3)	0/1(3)	0/1(3)	0/1(3)	0/1(3)	0/1(3)

1. Writable only in the CONFIG state.
2. Refer to *Table 2217* for register offset to channel number relationship.
3. Refer to the Device Configuration chapter for chip-specific reset value.

Figure 2338. FCCU RF Configuration Register *n* (FCCU_RF_CFG*n*)

Table 2216. FCCU_RF_CFG*n* field descriptions

Field	Description
0:31 RFC[31:0]	Recoverable fault configuration 0 HW recoverable fault 1 SW recoverable fault The recoverable fault configuration defines the fault recovery mode. HW recoverable faults are self recovered (status flag clearing and related) if the root cause (input fault) has been removed. SW recoverable faults are recovered (status flag clearing) by SW clearing of the related status flag.

Table 2217 shows FCCU RF configuration register channels.

Table 2217. FCCU RF configuration register channels

Address offset	Register name	Channel range (x) (bit location [0:31])
0x01C	FCCU_RF_CFG0	RFC[31:0]
0x020	FCCU_RF_CFG1	RFC[63:32]
0x024	FCCU_RF_CFG2	RFC[95:64]
0x028	FCCU_RF_CFG3	RFC[127:96]

89.6.5 FCCU RFS Configuration Register *n* (FCCU_RFS_CFG*n*)

The FCCU_RFS_CFG*n* registers contain the configuration of each recoverable fault in terms of fault reaction (short or long functional reset request pulse) when it is the root cause for the FAULT state transition.

These registers are writable only in the CONFIG state.

Offset: 0x04C +*n**0x4 (*n* = 0 to 7) Access: User read/write(1)

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	RFSC15(2)		RFSC14(2)		RFSC13(2)		RFSC12(2)		RFSC11(2)		RFSC10(2)		RFSC9(2)		RFSC8(2)	
Reset:	0/1(3)	0/1(3)	0/1(3)	0/1(3)	0/1(3)	0/1(3)	0/1(3)	0/1(3)	0/1(3)	0/1(3)	0/1(3)	0/1(3)	0/1(3)	0/1(3)	0/1(3)	0/1(3)

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	RFSC7(2)		RFSC6(2)		RFSC5(2)		RFSC4(2)		RFSC3(2)		RFSC2(2)		RFSC1(2)		RFSC0(2)	
Reset:	0/1(3)	0/1(3)	0/1(3)	0/1(3)	0/1(3)	0/1(3)	0/1(3)	0/1(3)	0/1(3)	0/1(3)	0/1(3)	0/1(3)	0/1(3)	0/1(3)	0/1(3)	0/1(3)

1. Writable only in the CONFIG state.
2. Refer to *Table 2219* for register offset to channel number relationship.
3. Refer to the Device Configuration chapter for chip-specific reset value.

Figure 2339. FCCU RFS Configuration Register *n* (FCCU_RFS_CFG*n*)

Table 2218. FCCU_RFS_CFG*n* field descriptions

Field	Description
0:31 RFSC[15:0]	Recoverable fault state configuration 00 No reset reaction 01 Short functional reset request pulse (FAULT state reaction) 10 Long functional reset request pulse (FAULT state reaction) 11 No reset reaction

Table 2219 shows FCCU **RFS** configuration register channels.

Table 2219. FCCU RFS configuration register channels

Address offset	Register name	Channel range (x) (bit location [0:31])
0x04C	FCCU_RFS_CFG0	RFSC[15:0]
0x050	FCCU_RFS_CFG1	RFSC[31:16]
0x054	FCCU_RFS_CFG2	RFSC[47:32]
0x058	FCCU_RFS_CFG3	RFSC[63:48]
0x05C	FCCU_RFS_CFG4	RFSC[79:64]
0x060	FCCU_RFS_CFG5	RFSC[95:80]
0x064	FCCU_RFS_CFG6	RFSC[111:96]
0x068	FCCU_RFS_CFG7	RFSC[127:112]

89.6.6 FCCU RF Status register *n* (FCCU_RF_S*n*)

The FCCU_RF_S*n* register contains the latched fault indication collected from the recoverable fault sources. Faults are latched also in the CONFIG state and independently from the enabling or reactions programmed for the RF.

No reactions are executed until the FCCU moves in the NORMAL state.

FCCU reacts and moves from the NORMAL state into the ALARM state only if the respective enable bit for a fault is set in the FCCU_RF_E*n* register and the respective enable bit for the timeout is set in the FCCU_TOEx register.

FCCU reacts and moves from the NORMAL or ALARM state into the FAULT state if the respective enable bit for a fault is set in the FCCU_RF_E*n* register and the respective enable bit for the timeout is disabled in the FCCU_TOEx register.

FCCU reacts and moves from the ALARM state into the FAULT state if the timeout (FCCU_TO register) is elapsed before recovering the fault.

The timeout is stopped only when the FCCU returns in the NORMAL state.

The FCCU_RF_S*n* register is encoded respectively into the N2FF_STATUS or A2FF_STATUS register to freeze the entry condition in the FAULT state. The status bits of the FCCU_RF_S*n* register, configured as SW recoverable faults, can be cleared by the following locked sequence:

1. Write the proper key into the FCCU_RFK register
2. Clear the status (flag) bit RFSx => the opcode OP12 is automatically set into the FCCU_CTRL.OPR field
3. Wait for the completion of the operation (FCCU_CTRL.OPS field)
4. Read the FCCU_RF_S*n* register in order to verify the effective deletion and in case of failure to repeat the sequence

As a result of the above sequence, in addition the FAULT interface provides support to clear the external fake FAULT root (refer to *Section 89.5.7: FAULT interface*).

Note: There should not be any other operation in between the above steps.

The FCCU moves from the FAULT or ALARM state into the NORMAL state if all the source faults that caused the transition into the FAULT state have been removed (HW recoverable fault) or cleared via SW (SW recoverable fault). In case of nested faults that are not all recovered, the FCCU will remain in the FAULT or ALARM state.

The SW application executes the FCCU_RF_S*n* read operation by the following sequence:

1. Set the OP10 operation into the FCCU_CTRL.OPR field
2. Wait for the completion of the operation (FCCU_CTRL.OPS field)
3. Read the FCCU_RF_S*n* register

In case of reconfiguration of the FCCU (CONFIG state), before returning in NORMAL state the pending status bits into the FCCU_RF_S*n* must be cleared in order to avoid a false transition in ALARM/FAULT state.

The following errors are ignored:

- to write a wrong key into the FCCU_RFK register
- to attempt to clear a HW recoverable error

Offset: 0x080 + *n**0x4 (*n* = 0 to 3) Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	RFS31(1)	RFS30(1)	RFS29(1)	RFS28(1)	RFS27(1)	RFS26(1)	RFS25(1)	RFS24(1)	RFS23(1)	RFS22(1)	RFS21(1)	RFS20(1)	RFS19(1)	RFS18(1)	RFS17(1)	RFS16(1)
W	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
Reset:	0/1(2)	0/1(2)	0/1(2)	0/1(2)	0/1(2)	0/1(2)	0/1(2)	0/1(2)	0/1(2)	0/1(2)	0/1(2)	0/1(2)	0/1(2)	0/1(2)	0/1(2)	0/1(2)

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	RFS15(1)	RFS14(1)	RFS13(1)	RFS12(1)	RFS11(1)	RFS10(1)	RFS9(1)	RFS8(1)	RFS7(1)	RFS6(1)	RFS5(1)	RFS4(1)	RFS3(1)	RFS2(1)	RFS1(1)	RFS0(1)
W	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
Reset:	0/1(2)	0/1(2)	0/1(2)	0/1(2)	0/1(2)	0/1(2)	0/1(2)	0/1(2)	0/1(2)	0/1(2)	0/1(2)	0/1(2)	0/1(2)	0/1(2)	0/1(2)	0/1(2)

1. Refer to *Table 2221* for register offset to channel number relationship.
2. Refer to the Device Configuration chapter for chip-specific reset value.

Figure 2340. FCCU RF Status register *n* (FCCU_RF_S*n*)

Table 2220. FCCU_RF_S*n* field descriptions

Field	Description
0:31 RFS[31:0]	Recoverable fault status 0 No recoverable fault latched 1 Recoverable fault latched The status bits related to the recoverable fault configured as HW recoverable faults are read-only and the flag is self cleared when the fault source is removed.

Table 2221 shows FCCU **RF** status register channels.

Table 2221. FCCU RF status register channels

Address offset	Register name	Channel range (x) (bit location [0:31])
0x080	FCCU_RF_S0	RFS[31:0]
0x084	FCCU_RF_S1	RFS[63:32]
0x088	FCCU_RF_S2	RFS[95:64]
0x08C	FCCU_RF_S3	RFS[127:96]

89.6.7 FCCU RF Key register (FCCU_RFK)

The FCCU_RFK register implements the key access to clear the status flags of the FCCU_RF_S*n* register.

The status bits of the FCCU_RF_S*n* register, configured as SW recoverable faults, can be cleared by the following locked sequence:

1. Write the key into the FCCU_RFK register
2. Clear the status (flag) bit RFSx

The key must be written for each FCCU_RF_S*n* clear operation.

Note: There should not be any other operation in between the above steps.

The FCCU_RFK register is not readable; a 0x0000_0000 value is always returned in case of read operation.

Offset: 0x090 Access: User Write-only

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R																
W	RFK_VAL															
Reset:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R																
W	RFK_VAL															
Reset:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 2341. FCCU RF Key register (FCCU_RFK)

Table 2222. FCCU_RFK field descriptions

Field	Description
0:31 RFK_VAL	recoverable fault key = 0xAB34_98FE

89.6.8 FCCU RF Enable Register *n* (FCCU_RF_E*n*)

The FCCU_RF_E*n* registers enable the fault sources to allow a transition from the NORMAL into the FAULT or ALARM state. In case of fault masking, the respective status bit into the FCCU_RF_S*n* register is anyway set, only the reaction is masked.

These registers are writable only in the CONFIG state.

Offset: 0x094 + *n**0x4 (*n* = 0 to 3) Access: User read/write(1)

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	RFE31(2)	RFE30(2)	RFE29(2)	RFE28(2)	RFE27(2)	RFE26(2)	RFE25(2)	RFE24(2)	RFE23(2)	RFE22(2)	RFE21(2)	RFE20(2)	RFE19(2)	RFE18(2)	RFE17(2)	RFE16(2)
Reset:	0/1(3)	0/1(3)	0/1(3)	0/1(3)	0/1(3)	0/1(3)	0/1(3)	0/1(3)	0/1(3)	0/1(3)	0/1(3)	0/1(3)	0/1(3)	0/1(3)	0/1(3)	0/1(3)

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	RFE15(2)	RFE14(2)	RFE13(2)	RFE12(2)	RFE11(2)	RFE10(2)	RFE9(2)	RFE8(2)	RFE7(2)	RFE6(2)	RFE5(2)	RFE4(2)	RFE3(2)	RFE2(2)	RFE1(2)	RFE0(2)
Reset:	0/1(3)	0/1(3)	0/1(3)	0/1(3)	0/1(3)	0/1(3)	0/1(3)	0/1(3)	0/1(3)	0/1(3)	0/1(3)	0/1(3)	0/1(3)	0/1(3)	0/1(3)	0/1(3)

1. Writable only in the CONFIG state.
2. Refer to *Table 2224* for register offset to channel number relationship.
3. Refer to the Device Configuration chapter for chip-specific reset value.

Figure 2342. FCCU RF Enable Register *n* (FCCU_RF_E*n*)

Table 2223. FCCU_RF_E*n* field descriptions

Field	Description
0:31 RFE[31:0]	Recoverable fault enable 0 No actions following the respective recoverable fault assertion 1 FCCU moves to ALARM or FAULT state

Table 2224 shows FCCU **RF_E*n*** enable register channels.

Table 2224. FCCU RF_E*n* enable register channels

Address offset	Register name	Channel range (x) (bit location [0:31])
0x094	FCCU_RF_E0	RFE[31:0]
0x098	FCCU_RF_E1	RFE[63:32]
0x09C	FCCU_RF_E2	RFE[95:64]
0x0A0	FCCU_RF_E3	RFE[127:96]

89.6.9 FCCU RF Timeout Enable Register *n* (FCCU_RF_TOE*n*)

The FCCU_RF_TOE*n* registers enable a transition from the NORMAL state into the ALARM state if the respective recoverable fault is enabled (RFEx and RFTOEx are set). In case the respective timeout is disabled (RFTOEx is cleared) and the recoverable fault is enabled (RFEx is set) the FCCU moves into the FAULT state if the related recoverable fault is asserted. The timer (preset with the timeout value defined by FCCU_TO register) is started when the FCCU moves into the ALARM state. If the fault is not recovered within the timeout the FCCU moves from the ALARM state to the FAULT state.

These registers are writable only in the CONFIG state.

Offset: 0x0A4 + *n**0x4 (*n* = 0 to 3) Access: User read/write(1)

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	RFTOE31(2)	RFTOE30(2)	RFTOE29(2)	RFTOE28(2)	RFTOE27(2)	RFTOE26(2)	RFTOE25(2)	RFTOE24(2)	RFTOE23(2)	RFTOE22(2)	RFTOE21(2)	RFTOE20(2)	RFTOE19(2)	RFTOE18(2)	RFTOE17(2)	RFTOE16(2)
Reset:	0/1(3)	0/1(3)	0/1(3)	0/1(3)	0/1(3)	0/1(3)	0/1(3)	0/1(3)	0/1(3)	0/1(3)	0/1(3)	0/1(3)	0/1(3)	0/1(3)	0/1(3)	0/1(3)

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	RFTOE15(2)	RFTOE14(2)	RFTOE13(2)	RFTOE12(2)	RFTOE11(2)	RFTOE10(2)	RFTOE9(2)	RFTOE8(2)	RFTOE7(2)	RFTOE6(2)	RFTOE5(2)	RFTOE4(2)	RFTOE3(2)	RFTOE2(2)	RFTOE1(2)	RFTOE0(2)
Reset:	0/1(3)	0/1(3)	0/1(3)	0/1(3)	0/1(3)	0/1(3)	0/1(3)	0/1(3)	0/1(3)	0/1(3)	0/1(3)	0/1(3)	0/1(3)	0/1(3)	0/1(3)	0/1(3)

1. Writable only in the CONFIG state.
2. Refer to *Table 2226* for register offset to channel number relationship.
3. Refer to the Device Configuration chapter for chip-specific reset value.

Figure 2343. FCCU RF Timeout Enable Register *n* (FCCU_RF_TOE*n*)

Table 2225. FCCU_RF_TOE*n* field descriptions

Field	Description
0:31 RFTOE[31:0]	Fault timeout enable 0 FCCU moves into the FAULT state if the respective fault is enabled 1 FCCU moves into the ALARM state if the respective fault is enabled.

Table 2226 shows FCCU **RF** timeout enable register channels.

Table 2226. FCCU RF timeout enable register channels

Address offset	Register name	Channel range (x) (bit location [0:31])
0x0A4	FCCU_RF_TOE0	RFTOE[31:0]
0x0A8	FCCU_RF_TOE1	RFTOE[63:32]
0x0AC	FCCU_RF_TOE2	RFTOE[95:64]
0x0B0	FCCU_RF_TOE3	RFTOE[127:96]

89.6.10 FCCU RF Timeout Register (FCCU_RF_TO)

The FCCU_RF_TO register defines the preset value of the timer for the recovery of the recoverable faults (enabled). Once FCCU enters in ALARM state, following the assertion of a recoverable fault enabled (RFEx and RFTOEx are set), the timer starts the count down.

If the fault is not recovered within the timeout the FCCU moves from the ALARM state to the FAULT state. The alarm timeouts value should be programmed less than the FOSU_COUNT, else destructive resets may be generated by FOSU timeout.

This register is writable only in the CONFIG state.

Offset: 0x0B4 Access: User read/write(1)

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	TO															
Reset:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	TO															
Reset:	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

1. Writable only in the CONFIG state.

Figure 2344. FCCU RF Timeout Register (FCCU_RF_TO)

Table 2227. FCCU_RF_TO field descriptions

Field	Description
0:31 TO	Recoverable fault timeout Timeout = (TO) x $T_{RC16MHz}$

89.6.11 FCCU CFG Timeout Register (FCCU_CFG_TO)

The FCCU_CFG_TO register defines the preset value of the watchdog timer for the recovery from the CONFIG state. Once FCCU enters in CONFIG state, following a SW request (OP1 opcode), the watchdog timer is initialized and starts the countdown if the reset is not asserted.

If the configuration is not completed within the timeout, the FCCU moves automatically from the CONFIG state to the NORMAL state and the default values for all the configuration register is restored. Configuration registers are those registers which can be written only in CONFIG state. Refer to *Table 2212: FCCU memory map*. The watchdog timeout is clocked with the RC oscillator clock (16 MHz). The default timeout value is 4.096 ms. Longer activation of CONFIG state can lead to resets if a failure is indicated during the time the FCCU is in CONFIG state due to FOSU.

The FCCU_CFG_TO register is writable in any state excluded the CONFIG state as follows the execution of the OP1 opcode (NORMAL to CONFIG state) and until the completion of the OP2 opcode (CONFIG to NORMAL state).

In case of watchdog timeout the FCCU_CFG_TO register is not accessible until the OP14 operation (CONFIG to NORMAL) has been completed.

Offset: 0x0B8 Access: User read/write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	TO		
W																
Reset:	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0

Figure 2345. FCCU CFG Timeout Register (FCCU_CFG_TO)

Table 2228 shows FCCU_CFG_TO field descriptions.

Table 2228. FCCU_CFG_TO field descriptions

Field	Description
29:31 TO	Configuration timeout Timeout = $T_{RC16MHz}$ x $2^{(TO + 10)}$ 000 Timeout = 64 µs ... 111 Timeout = 8.192 ms

89.6.12 FCCU IO Control Register (FCCU_EINOUT)

The FCCU_EINOUT register allows the following operations typically in NORMAL state:

- to control the FCCU Error output pin[1] output level when the FCCU is configured in “Test1” or “Test0” fault output mode (FCCU_CFG.FOM)
- to control the FCCU Error output pin[0] output level when the FCCU is configured in “Test1” or “Test2” fault output mode (FCCU_CFG.FOM)
- to observe the FCCU Error inputs level

Table 2229 shows Bi-stable encoding.

Table 2229. Bi-stable encoding

Mode = FCCU_CFG.FOM	FCCU_EOUT[0]	FCCU_EOUT[1]
Test1	output	output
Test2	output	input
Test0	input	output

Note: Due to the resynchronization stage of the EOUT interface, there is a latency of a few IRCOSC clock cycles following a write/read operation of the FCCU_EINOUT register.

Offset: 0x0BC Access: User read/write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	EIN1(1)	EIN0(1)	0	0	EOUT1	EOUT0
W																
Reset:	0	0	0	0	0	0	0	0	0	0	0/1(2)	0/1(2)	0	0	0	0

1. FCCU Error input x depends on the PAD configuration, that is, internal pull-up enabled as default or pull-down and therefore EINx bit is 1 or 0 upon reset.
2. Refer to the Device Configuration chapter for chip-specific reset value.

Figure 2346. FCCU IO Control Register (FCCU_EINOUT)

Table 2230. FCCU_EINOUT field descriptions

Field	Description
26 EIN1	Error input 1 0 when FCCU Error input[1] = 0 1 when FCCU Error input[1] = 1 The EIN1 captures the respective FCCU Error input[1] signal.
27 EIN0	Error input 0 0 when FCCU Error input[0] = 0 1 when FCCU Error input[0] = 1 The EIN0 captures the FCCU Error input[0] signal.
30 EOUT1	Error out 1 (significant only if the FCCU_CFG.FOM = Test1 or Test0 => FCCU Error output[1] configured in output mode). 0 force FCCU Error output[1] = 0 1 force FCCU Error output[1] = 1 The EOUT1 set/clear the respective FCCU Error output[1] output signal if FCCU_CFG.FOM = 110 or 101, otherwise it is a “do not care” value.
31 EOUT0	Error out 0 (significant only if the FCCU_CFG.FOM = Test1 or Test2 => FCCU Error output[0] configured in output mode) 0 force FCCU Error output[0] = 0 1 force FCCU Error output[0] = 1 The EOUT0 set/clear the respective FCCU Error output[0] signal if FCCU_CFG.FOM = 110 or 111, otherwise it is a “do not care” value.

89.6.13 FCCU Status Register (FCCU_STAT)

The FCCU_STAT register includes the FCCU status. The FCCU finite state machine (FSM) operates by the RC oscillator clock asynchronous with the System clock. The FCCU status read operation requires a safe mechanism operated by a HW/SW synchronization

sequence. The SW application executes a FCCU status read operation by the following sequence:

1. Set the OP3 operation into the FCCU_CTRL.OPR field.
2. Wait for the completion of the operation (FCCU_CTRL.OPS field).
3. Read the FCCU status (FCCU_STAT register).

Offset: 0x0C0 Access: User read/write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	PhysicError Pin		ESTAT	STATUS		
W																
Reset:	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0

Figure 2347. FCCU Status Register (FCCU_STAT)

Table 2231. FCCU_STAT field descriptions

Field	Description
26:27 PhysicErrorPin	Physical Error Pin state PhysicErrorPin[26] corresponds to error pin 0, PhysicErrorPin[27] corresponds to error pin 1. For each of these, the bit values have the following meanings: 0 Error Pin is at logical 0 1 Error Pin is at logical 1 During Fault state, a static or toggling value is observed based on selected protocol. **Note:** This field indicates the real time status of the Error pin as driven by FCCU (at FCCU boundary) to the pad and not what is seen at/from the pad, which is based on different IPs (such as SIUL2) configurations of the SoC.
28 ESTAT	Current Error Pin Status SW can write this bit with a different value. The new value is valid for 1 clock cycle, thereafter it returns to indicate the actual value being driven from FCCU. For example in nonfaulty states, this bit reads 0 and can be set for one cycle to 1. Similarly in faulty state, this bit reads 1 and sw can write it to 0 and after one cycle the bit reads 1 again. 0 System is OPERATIONAL (OK). 1 System is in FAULT state.This state is taken from the FCCU Eout logic (That is, as late as possible).
29:31 STATUS	FCCU Status 000 NORMAL state 001 CONFIG state 010 ALARM state 011 FAULT state Other: UNKNOWN state

89.6.14 FCCU NA Freeze Status Register (N2AF_STATUS)

The N2AF_STATUS register contains a unique code to identify the recoverable source (FCCU recoverable fault x) that caused the state transition from the NORMAL state to the ALARM state. In case of multiple fault conditions the fault source cannot be identified.

The SW application executes the N2AF_STATUS read operation by the following sequence:

1. Set the OP4 operation into the FCCU_CTRL.OPR field.
2. Wait for the completion of the operation (FCCU_CTRL.OPS field).
3. Read the N2AF_STATUS register.

The SW application executes the N2AF_STATUS clear operation by the following sequence:

1. Set the OP13 operation into the FCCU_CTRL.OPR field.
2. Wait for the completion of the operation (FCCU_CTRL.OPS field).

All the freeze registers are cleared by this operation.

Offset: 0x0C4 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0(1)	0(1)	0(1)	0(1)	0(1)	0(1)	0(1)	0(1)	0(1)	0(1)	0(1)	0(1)	0(1)	0(1)	0(1)	0(1)
W																
Reset:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0(1)	0(1)	0(1)	0(1)	0(1)	0(1)	0(1)	0(1)	NAFS							
W									w1c							
Reset:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. These are reserved bits. These bits are always read as 0 and must always be written with 0.

Figure 2348. FCCU NA Freeze Status Register (N2AF_STATUS)

Table 2232. N2AF_STATUS field descriptions

Field	Description
24:31 NAFS	Normal to Alarm Frozen Status 0x00 No transition from NORMAL to ALARM state 0x01 NORMAL to ALARM state transition cause => FCCU recoverable fault[0] 0x02 NORMAL to ALARM state transition cause => FCCU recoverable fault[1] 0x03 NORMAL to ALARM state transition cause => FCCU recoverable fault[2] ... 0x80 NORMAL to ALARM state transition cause => FCCU recoverable fault[127] 0xFF NORMAL to ALARM state transition cause => multiple FCCU recoverable faults

89.6.15 FCCU AF Freeze Status Register (A2FF_STATUS)

The A2FF_STATUS register contains a unique code to identify the timeout trigger (recoverable fault) that caused the state transition from the ALARM state to the FAULT state. In case of multiple fault conditions the fault source cannot be identified.

The SW application executes the A2FF_STATUS read operation by the following sequence:

1. Set the OP5 operation into the FCCU_CTRL.OPR field.
2. Wait for the completion of the operation (FCCU_CTRL.OPS field).
3. Read the A2FF_STATUS register.

The SW application executes the FCCU_AFFS clear operation by the following sequence:

1. Set the OP13 operation into the FCCU_CTRL.OPR field.
2. Wait for the completion of the operation (FCCU_CTRL.OPS field).

All the freeze registers are cleared by this operation.

Offset: 0x0C8 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0(1)	0(1)	0(1)	0(1)	0(1)	0(1)	0(1)	0(1)	0(1)	0(1)	0(1)	0(1)	0(1)	0(1)	0(1)	0(1)
W																
Reset:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0(1)	0(1)	0(1)	0(1)	0(1)	0(1)	AF_SRC		AFFS							
W																
Reset:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. These are reserved bits. These bits are always read as 0 and must always be written with 0.

Figure 2349. FCCU AF Freeze Status Register (A2FF_STATUS)

Table 2233. A2FF_STATUS field descriptions

Field	Description
22:23 AF_SRC	Fault source 00 No fault 01 Reserved 10 Recoverable fault 11 Multiple or recoverable, or both, faults
24:31 AFFS	Alarm to Fault Frozen Status 0x00 No transition from ALARM to FAULT state 0x01 ALARM to FAULT state transition cause => FCCU recoverable fault[0] timeout 0x02 ALARM to FAULT state transition cause => FCCU recoverable fault[1] timeout 0x03 ALARM to FAULT state transition cause => FCCU recoverable fault[2] timeout ... 0x80 ALARM to FAULT state transition cause => FCCU recoverable fault[127] timeout 0xFF ALARM to FAULT state transition cause => multiple FCCU recoverable faults timeout

89.6.16 FCCU NF Freeze Status Register (N2FF_STATUS)

The N2FF_STATUS register contains a unique code to identify the recoverable fault that caused the state transition from the NORMAL state to the FAULT state. In case of multiple fault conditions the fault source cannot be identified. The SW application executes the N2FF_STATUS read operation by the following sequence:

1. Set the OP6 operation into the FCCU_CTRL.OPR field.
2. Wait for the completion of the operation (FCCU_CTRL.OPS field).
3. Read the N2FF_STATUS register.

The SW application executes the N2FF_STATUS clear operation by the following sequence:

1. Set the OP13 operation into the FCCU_CTRL.OPR field.
2. Wait for the completion of the operation (FCCU_CTRL.OPS field).

All the freeze registers are cleared by this operation.

Offset: 0x0CC Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0(1)	0(1)	0(1)	0(1)	0(1)	0(1)	0(1)	0(1)	0(1)	0(1)	0(1)	0(1)	0(1)	0(1)	0(1)	0(1)
W																
Reset:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0(1)	0(1)	0(1)	0(1)	0(1)	0(1)	NF_SRC		NFFS							
W																
Reset:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. These are reserved bits. These bits are always read as 0 and must always be written with 0.

Figure 2350. FCCU NF Freeze Status Register (N2FF_STATUS)

Table 2234. N2FF_STATUS field descriptions

Field	Description
22:23 NF_SRC	Fault source 00 No fault 01 Reserved 10 Recoverable fault 11 Multiple and recoverable faults
24:31 NFFS	Normal to Fault Frozen Status 0x00 No transition from NORMAL to FAULT state 0x01 NORMAL to FAULT state transition cause => FCCU recoverable fault[0] 0x02 NORMAL to FAULT state transition cause => FCCU recoverable fault[1] 0x03 NORMAL to FAULT state transition cause => FCCU recoverable fault[2] ... 0x80 NORMAL to FAULT state transition cause => FCCU recoverable fault[127] 0xFF NORMAL to FAULT state transition cause => multiple FCCU recoverable faults

89.6.17 FCCU FA Freeze Status Register (F2A_STATUS)

The F2A_STATUS register contains a unique code to identify the recoverable fault source (recoverable fault x) that caused the state transition from the FAULT state to the ALARM state. In case of multiple fault conditions the fault source cannot be identified.

The SW application executes the F2A_STATUS read operation by the following sequence:

1. Set the OP7 operation into the FCCU_CTRL.OPR field
2. Wait for the completion of the operation (FCCU_CTRL.OPS field)
3. Read the N2FF_STATUS register

The SW application executes the F2A_STATUS clear operation by the following sequence:

1. Set the OP13 operation into the FCCU_CTRL.OPR field
2. Wait for the completion of the operation (FCCU_CTRL.OPS field).

All the freeze registers are cleared by this operation.

Offset: 0x0D0 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0(1)	0(1)	0(1)	0(1)	0(1)	0(1)	0(1)	0(1)	0(1)	0(1)	0(1)	0(1)	0(1)	0(1)	0(1)	0(1)
W																
Reset:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0(1)	0(1)	0(1)	0(1)	0(1)	0(1)	0(1)	0(1)	FAFS							
W									w1c							
Reset:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. These are reserved bits. These bits are always read as 0 and must always be written with 0.

Figure 2351. FCCU FA Freeze Status Register (F2A_STATUS)

Table 2235. F2A_STATUS field descriptions

Field	Description
24:31 FAFS	Fault to Normal Frozen Status 0x00 No transition from FAULT to NORMAL state 0x01 FAULT to ALARM state transition cause => FCCU recoverable fault[0] 0x02 FAULT to ALARM state transition cause => FCCU recoverable fault[1] 0x03 FAULT to ALARM state transition cause => FCCU recoverable fault[2] ... 0x80 FAULT to ALARM state transition cause => FCCU recoverable fault[127] 0xFF FAULT to ALARM state transition cause => multiple FCCU recoverable faults

89.6.18 FCCU RF Fake Register (FCCU_RFF)

The FCCU_RFF register contains a unique code to set a recoverable fault in mutually exclusive mode by the external FAULT interface signal setting. It allows the SW emulation of the recoverable faults, by the injection of the fault directly in the FAULT root, in order to verify the entire path and reaction. The fault injection mechanism is optional and requires a proper management as described in *Section 89.5.7: FAULT interface*. The reaction following a fake recoverable fault cannot be masked. The FCCU_RFF is a write-only register with a set of codes corresponding to each recoverable fault injection.

Offset: 0x0DC Access: User Write-only

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0							
W										FRFC						
Reset:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 2352. FCCU RF Fake Register (FCCU_RFF)

Table 2236. FCCU_RFF field descriptions

Field	Description
25:31 FRFC	Fake recoverable fault code 0x00 Fake recoverable fault injection at recoverable fault source 0 0x01 Fake recoverable fault injectionat recoverable fault source 1 0x02 Fake recoverable fault injection at recoverable fault source 2 … 0x7F Fake recoverable fault injection at recoverable fault source 127 **Note:** Only writing to this register, fake fault injection occurs, writing 00 and default value being zero give different results.

89.6.19 FCCU IRQ Status Register (FCCU_IRQ_STAT)

The FCCU_IRQ_STAT register defines the FCCU interrupt status register related to the following events:

- Configuration timeout error
- Alarm interrupt
- NMI interrupt

The external interrupt line (Interrupt request) is asserted if any interrupt status bit of the FCCU_IRQ_STAT is set and the respective enable bit of the FCCU_IRQ_EN register is also set.

The NMI and ALARM interrupts are asserted and cleared according to the FCCU state. The status bits of the FCCU_IRQ_STAT trace the status of the related interrupt lines.

Offset: 0x0E0 Access: User read/write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	Reserved	Reserved	NMI_STAT	ALRM_STAT	CFG_TO_STAT
W																w1c
Reset:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 2353. FCCU IRQ Status Register (FCCU_IRQ_STAT)

Table 2237. FCCU_IRQ_STAT field descriptions

Field	Description
29 NMI_STAT	NMI Interrupt Status 0 NMI interrupt is OFF 1 NMI interrupt is ON
30 ALRM_STAT	Alarm Interrupt Status 0 Alarm interrupt is OFF 1 Alarm interrupt is ON
31 CFG_TO_STAT	Configuration Timeout Status 0 No configuration timeout error 1 Configuration timeout error

89.6.20 FCCU IRQ Enable Register (FCCU_IRQ_EN)

The FCCU_IRQ_EN register defines the FCCU interrupt enable register related to the following event:

- Configuration timeout error

The external interrupt line “Interrupt request (miscellaneous conditions)” is asserted if any interrupt status bit of the FCCU_IRQ_STAT is set and the respective enable bit of the FCCU_IRQ_EN register is also set.

Offset: 0x0E4 Access: User read/write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	CFG_TO_IEN
W																
Reset:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 2354. FCCU IRQ Enable Register (FCCU_IRQ_EN)

Table 2238. FCCU_IRQ_EN field descriptions

Field	Description
31 CFG_TO_IEN	Configuration Timeout Interrupt Enable 0 Configuration timeout interrupt disabled. 1 Configuration timeout interrupt enabled.

89.6.21 FCCU XTMR Register (FCCU_XTMR)

The FCCU_XTMR register contains the read values of the Alarm or Watchdog Timer or EOUT pin low counter. These timers are clocked on the IRCOSC clock.

EOUTMR is the EOUT pin in logic 0 counter. After EOUT has been set to low (by SW or by a real fault), it loads (a value corresponding to T_min = 250 µs + value programmed in FCCU_DELTA_T register) and then starts down-counting and becomes 0 on the expiry of T_min window. It reloads on occurrence of another fault event which has EOUT signaling enabled. T_min feature is available only in Bi-Stable

The SW application executes the timer read operation by the following sequence:

1. Set any of the following operations into the FCCU_CTRL.OPR field:
   a) OP17
   b) OP19
   c) OP20
2. Wait for the completion of the operation (FCCU_CTRL.OPS field)
3. Read the FCCU_XTMR register

Table 2240 shows Timer state/value.

Offset: 0x0E8 Access: User read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	XTMR_VAL															
W																
Reset:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	XTMR_VAL															
W																
Reset:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 2355. FCCU XTMR Register (FCCU_XTMR)

Table 2239. FCCU_XTMR field descriptions

Field	Description
0:31 XTMR_VAL	Alarm/Watchdog/Safe request timer The current timer value is measured in IRCOSC clock cycles. These bits can be read by the software.

Table 2240. Timer state/value

TIMER	CONFIG state	NORMAL state	ALARM state	FAULT state
ALARM	0x0000_0000	Initial value	Running	Idle/End of count
CFG	Running	0x0001_FFFF	0x0001_FFFF	0x0001_FFFF
EOUTMR	0x0000_0000	0x0000_0000	0x0000_0000	Running

89.6.22 FCCU MCS Register (FCCU_MCS)

The FCCU_MCS register contains a queue of the last four states of the MC. MCS0 is the latest one, while MCS3 is the oldest one. In addition a qualifier indicates if the FCCU is in the FAULT state when the MC state has been captured. The MC state is synchronous to the system clock and provided by a different module while the FCCU state is synchronous to the IRCOSC, therefore some uncertainty must be considered regarding the FAULT state indication.

Offset: 0x0EC Access: User read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	VL3	FS3	0	0	MCS3				VL2	FS2	0	0	MCS2			
W																
Reset:	0/1(1)	0/1(1)	0	0	0/1(1)	0/1(1)	0/1(1)	0/1(1)	0/1(1)	0/1(1)	0	0	0/1(1)	0/1(1)	0/1(1)	0/1(1)

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	VL1	FS1	0	0	MCS1				VL0	FS0	0	0	MCS0			
W																
Reset:	0/1(1)	0/1(1)	0	0	0/1(1)	0/1(1)	0/1(1)	0/1(1)	0/1(1)	0/1(1)	0	0	0/1(1)	0/1(1)	0/1(1)	0/1(1)

1. Refer to the Device Configuration chapter for chip-specific reset value.

Figure 2356. FCCU MCS Register (FCCU_MCS)

Table 2241. FCCU_MCS field descriptions

Field	Description
0 VL3	Valid It indicates that the correspondent MCSx and FSx fields are valid. 0 MCS3, FS3 fields are not significant. 1 MCS3, FS3 fields are significant.
1 FS3	Fault status It indicates that the correspondent MCSx field has been captured when the FCCU is in FAULT state. 0 MCS3 field captured in any state different from the FAULT state. 1 MCS3 field captured in FAULT state.
4:7 MCS3	State of MC modules The MCSx is the state of MC modules. MCS0 = latest state MCS3 = oldest state On any MC state change the previous MC states are shifted (MCS3 = MCS2, MCS2= MCS1, MCS1= MCS0) and the latest one is captured in MCS0. For a definition of these states, refer to the description of the MC modules in the chip reference manual.
8 VL2	Valid It indicates that the correspondent MCSx and FSx fields are valid. 0 MCS2, FS2 fields are not significant 1 MCS2, FS2 fields are significant
9 FS2	Fault status It indicates that the correspondent MCSx field has been captured when the FCCU is in FAULT state. 0 MCS2 field captured in any state different from the FAULT state 1 MCS2 field captured in FAULT state

Table 2241. FCCU_MCS field descriptions (continued)

Field	Description
12:15 MCS2	*State of MC modules* The MCSx is the state of MC modules. MCS0 = latest state MCS3 = oldest state On any MC state change the previous MC states are shifted (MCS3 = MCS2, MCS2= MCS1, MCS1= MCS0) and the latest one is captured in MCS0. For a definition of these states, refer to the description of the MC modules in the chip reference manual.
16 VL1	Valid It indicates that the correspondent MCSx and FSx fields are valid. 0 MCS1, FS1 fields are not significant. 1 MCS1, FS1 fields are significant.
17 FS1	Fault status It indicates that the correspondent MCSx field has been captured when the FCCU is in FAULT state. 0 MCS1 field captured in any state different from the FAULT state. 1 MCS field captured in FAULT state.
20:23 MCS1	*State of MC modules* The MCSx is the state of MC modules. MCS0 = latest state MCS3 = oldest state On any MC state change the previous MC states are shifted (MCS3 = MCS2, MCS2= MCS1, MCS1= MCS0) and the latest one is captured in MCS0. For a definition of these states, refer to the description of the MC modules in the chip reference manual.
24 VL0	Valid It indicates that the correspondent MCSx and FSx fields are valid. 0 MCS0, FS0 fields are not significant. 1 MCS0, FS0 fields are significant.
25 FS0	Fault status It indicates that the correspondent MCSx field has been captured when the FCCU is in FAULT state. 0 MCS0 field captured in any state different from the FAULT state. 1 MCS0 field captured in FAULT state.
28:31 MCS0	*State of MC modules* The MCSx is the state of MC modules. MCS0 = latest state MCS3 = oldest state On any MC state change the previous MC states are shifted (MCS3 = MCS2, MCS2= MCS1, MCS1= MCS0) and the latest one is captured in MCS0. For a definition of these states, refer to the description of the MC modules in the chip reference manual.

89.6.23 FCCU Transient Register (FCCU_TRANS_LOCK)

The FCCU_TRANS register is used for unlocking configuration by writing 0xBC. This register is available only in supervisor mode. This register is write only.

Offset: 0x0F0 Access: Supervisor Write-only

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0								
W									TRANSKEY							
Reset:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 2357. FCCU Transient Register (FCCU_TRANS_LOCK)

Table 2242. FCCU_TRANS_LOCK field descriptions

Field	Description
24:31 TRANSKEY	Transition Unlocking Key value 0x00 No change ... 0xBB No change 0xBC Configuration unlocked 0xBD No change ... 0xFF No change

89.6.24 FCCU Permanent Lock Register (FCCU_PERMNT_LOCK)

The FCCU_PERMNT_LOCK register is used for locking configuration permanently by writing 0xFF. This register is available only in supervisor mode. The permanent lock can only be removed by applying a reset (not necessarily power on reset). This register is write only.

Offset: 0x0F4 Access: Supervisor Write-only

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0								
W									PERMNTKEY							
Reset:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 2358. FCCU Permanent Lock Register (FCCU_PERMNT_LOCK)

Table 2243. FCCU_PERMNT_LOCK field descriptions

Field	Description
24:31 PERMNTKEY	Transition Locking Key value 0xFF Configuration Locked Any other value: No Change

89.6.25 FCCU Delta T Register (FCCU_DELTA_T)

The FCCU_DELTA_T register is used for programming the value of delta_T constant, in microseconds. This register can be written only in CONFIG state.

Offset: 0x0F8 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	Reserved													
W																
Reset:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	DELTA_T_VAL													
W																
Reset:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 2359. FCCU Delta T Register (FCCU_DELTA_T)

Table 2244. FCCU_DELTA_T field descriptions

Field	Description
18:31 DELTA_T_VAL	Value of Delta_T (in microseconds) Max. value can be 10,000 µs (10 ms)

89.6.26 FCCU IRQ Alarm Enable register *n* (FCCU_IRQ_ALARM_EN*n*)

These registers enable the corresponding IRQ alarm. This register is writable only in CONFIG state.

Offset: 0x0FC + *n**0x4 (*n* = 0 to 3) Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	IRQEN31(1)	IRQEN30(1)	IRQEN29(1)	IRQEN28(1)	IRQEN27(1)	IRQEN26(1)	IRQEN25(1)	IRQEN24(1)	IRQEN23(1)	IRQEN22(1)	IRQEN21(1)	IRQEN20(1)	IRQEN19(1)	IRQEN18(1)	IRQEN17(1)	IRQEN16(1)
Reset:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	IRQEN15(1)	IRQEN14(1)	IRQEN13(1)	IRQEN12(1)	IRQEN11(1)	IRQEN10(1)	IRQEN9(1)	IRQEN8(1)	IRQEN7(1)	IRQEN6(1)	IRQEN5(1)	IRQEN4(1)	IRQEN3(1)	IRQEN2(1)	IRQEN1(1)	IRQEN0(1)
Reset:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. Refer to *Table 2246* for register offset to channel number relationship.

Figure 2360. FCCU IRQ Alarm Enable register *n* (FCCU_IRQ_ALARM_EN*n*)

Table 2245. FCCU_IRQ_ALARM_EN*n* field descriptions

Field	Description
0:31 IRQEN[31:0]	IRQ alarm enable 0 Alarm is disabled for error source x. 1 Alarm is enabled for error source x.

Table 2246 shows FCCU IRQ alarm enable register channels.

Table 2246. FCCU IRQ alarm enable register channels

Address offset	Register name	Channel range (x) (bit location [0:31])
0x0FC	FCCU_IRQ_ALARM_EN0	IRQEN[31:0]
0x100	FCCU_IRQ_ALARM_EN1	IRQEN[63:32]
0x104	FCCU_IRQ_ALARM_EN2	IRQEN[95:64]
0x108	FCCU_IRQ_ALARM_EN3	IRQEN[127:96]

89.6.27 FCCU NMI Enable registers *n* (FCCU_NMI_EN*n*)

These registers enable the NMI. These registers are writable only in CONFIG state.

Offset: 0x10C + *n**0x4 (*n* = 0 to 3) Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	NMIEN31(1)	NMIEN30(1)	NMIEN29(1)	NMIEN28(1)	NMIEN27(1)	NMIEN26(1)	NMIEN25(1)	NMIEN24(1)	NMIEN23(1)	NMIEN22(1)	NMIEN21(1)	NMIEN20(1)	NMIEN19(1)	NMIEN18(1)	NMIEN17(1)	NMIEN16(1)
W																
Reset:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	NMIEN15(1)	NMIEN14(1)	NMIEN13(1)	NMIEN12(1)	NMIEN11(1)	NMIEN10(1)	NMIEN9(1)	NMIEN8(1)	NMIEN7(1)	NMIEN6(1)	NMIEN5(1)	NMIEN4(1)	NMIEN3(1)	NMIEN2(1)	NMIEN1(1)	NMIEN0(1)
W																
Reset:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. Refer to *Table 2248* for register offset to channel number relationship.

Figure 2361. FCCU NMI Enable register *n* (FCCU_NMI_EN*n*)

Table 2247. FCCU_NMI_EN*n* field descriptions

Field	Description
0:31 NMIEN[31:0]	NMI enable 0 NMI is disabled for error (RF) source x. 1 NMI is enabled for error (RF) source x.

Table 2248 shows FCCU NMI enable register channels.

Table 2248. FCCU NMI enable register channels

Address offset	Register name	Channel range (x) (bit location [0:31])
0x10C	FCCU_NMI_EN0	NMIEN[31:0]
0x110	FCCU_NMI_EN1	NMIEN[63:32]
0x114	FCCU_NMI_EN2	NMIEN[95:64]
0x118	FCCU_NMI_EN3	NMIEN[127:96]

89.6.28 FCCU EOUT Signaling Enable registers *n* (FCCU_EOUT_SIG_EN*n*)

These registers enable the error out signaling. Disabling of EOUT signaling is possible for Bi-stable protocol only. This register is writable only in CONFIG state.

Offset: 0x11C + n*0x4 (n = 0 to 3) Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	EOUTEN31(1)	EOUTEN30(1)	EOUTEN29(1)	EOUTEN28(1)	EOUTEN27(1)	EOUTEN26(1)	EOUTEN25(1)	EOUTEN24(1)	EOUTEN23(1)	EOUTEN22(1)	EOUTEN21(1)	EOUTEN20(1)	EOUTEN19(1)	EOUTEN18(1)	EOUTEN17(1)	EOUTEN16(1)
Reset:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	EOUTEN15(1)	EOUTEN14(1)	EOUTEN13(1)	EOUTEN12(1)	EOUTEN11(1)	EOUTEN10(1)	EOUTEN9(1)	EOUTEN8(1)	EOUTEN7(1)	EOUTEN6(1)	EOUTEN5(1)	EOUTEN4(1)	EOUTEN3(1)	EOUTEN2(1)	EOUTEN1(1)	EOUTEN0(1)
Reset:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. Refer to *Table 2250* for register offset to channel number relationship.

Figure 2362. FCCU EOUT Signaling Enable register *n* (FCCU_EOUT_SIG_EN*n*)

Table 2249. FCCU_EOUT_SIG_EN*n* field descriptions

Field	Description
0:31 EOUTEN[31:0]	EOUT signaling enable 0 EOUT signaling is disabled for error (RF) source x for bistable protocol. Error pins are as if in nonfaulty state.(1) 1 EOUT signaling is enabled for error (RF) source x.

1. In case of time switching and dual rail protocols, the error out signaling remains active even if the EOUT flag is 0b.

Table 2250 shows FCCU EOUT signaling enable register channels.

Table 2250. FCCU EOUT signaling enable register channels

Address offset	Register name	Channel range (x) (bit location [0:31])
0x11C	FCCU_EOUT_SIG_EN0	EOUTEN[31:0]
0x120	FCCU_EOUT_SIG_EN1	EOUTEN[63:32]
0x124	FCCU_EOUT_SIG_EN2	EOUTEN[95:64]
0x128	FCCU_EOUT_SIG_EN3	EOUTEN[127:96]

89.7 FCCU Output Supervision Unit

The FOSU provides a supervision of the primary error notification path by analyzing FCCU behavior for correctness. It waits for any reaction of the FCCU in a fixed time window after an error arrives.

The intention of the FOSU is to provide a secondary error reaction path in most cases the FCCU fails but not to needlessly propagate an error which is already handled by the FCCU into a full chip reset. Only a failed primary error reaction (that is FCCU's failure) is a reason for the secondary reaction to take over (and generate a destructive reset request).

There is a 'do nothing' input coming from the FCCU that indicates that the FCCU is programmed for no reaction. It is a “static” input in the sense that it does not change after FCCU configuration. This signal is automatically de-asserted whenever the FCCU moves to CONFIG state and hence FOSU shall not be able to monitor faults occurring in CONFIG state.The FOSU masks the incoming faults with the 'do nothing' control from the FCCU, meaning that a fault is not captured by the FOSU if the 'do nothing' signal is asserted (that is an enabled fault). There is no minimum pulse width requirement on the fault indication other than what is required by the technology, which is the same as that of the FCCU. FOSU does not monitor FCCU for the case of faults occurring during CONFIG state. Also, the case of a continuously incoming disabled fault being enabled later, is not monitored.

The FOSU contains a timer with a duration of FOSU_COUNT,driven by the IRCOSC.The timer is initialized and started on any captured, enabled fault. While the timer is running, any subsequent captured fault will neither restart nor reinitialize the timer. The timer is stopped when the FCCU shows any of the following reactions (the FCCU does not check whether the reaction is the configured one for the faults which occurred):

- Reset: Long or short functional reset.
- IRQ: NMI or Alarm.
- Error out triggered (by FCCU or by SW).

When the timer is stopped, the fault capture logic is cleared in order to ensure that the timer is not restarted due to faults still ‘stuck’ in the capture logic. The timer will then be restarted by the next new failure indication. When the timer expires, the FOSU's failure indicator output is asserted after it ensures that the fault is enabled and the static “fccu program to do nothing” signal is de-asserted. This is because FCCU uses settings after it exits CONFIG state, even if fault captured before the exit.
The FOSU's failure indicator output is connected to one of the MC_RGM's ‘destructive’ reset inputs, so its assertion will cause a reset sequence to be initiated starting at PHASE0. The FOSU module is reset with the same reset as it is used by the FCCU (System reset), which is asserted on power-on, ‘destructive’, and long external resets. When this reset is asserted, the FOSU's capture logic is cleared, its timer is kept stopped and in a non-expired state, and its failure indicator output is de-asserted.

The following conditions show a use case of the FOSU:

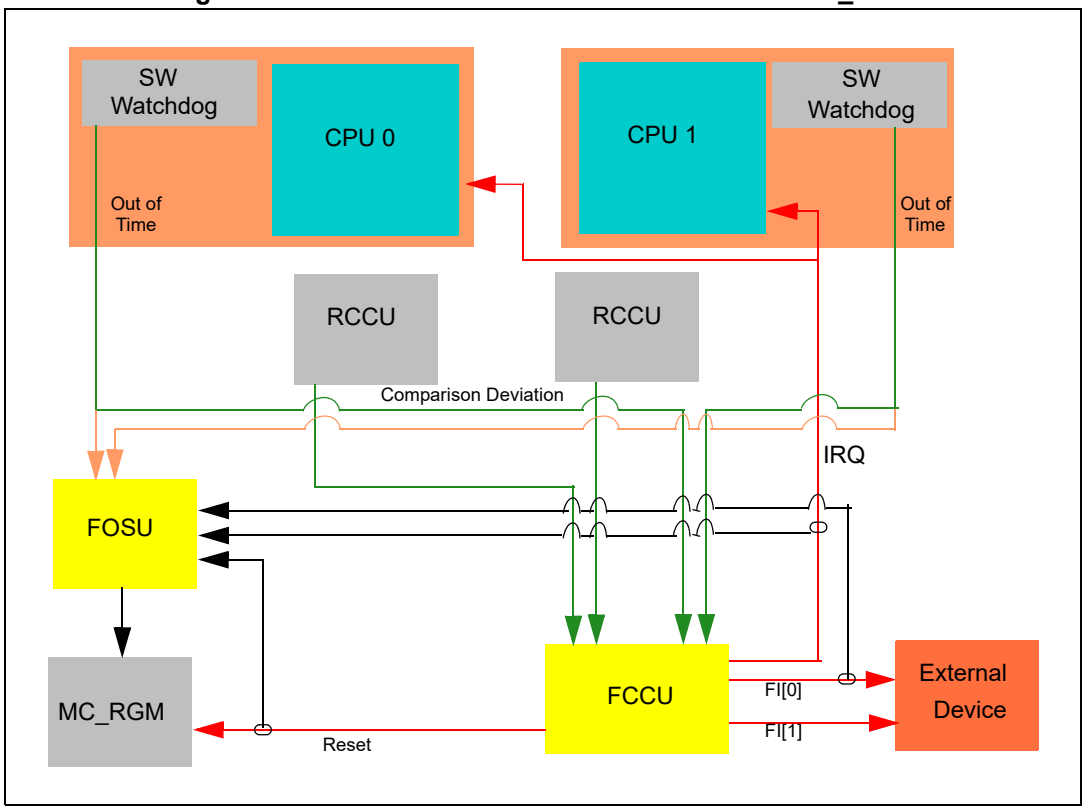
1. The user configures the channel x of the FCCU to react with a functional reset, either long or short.
2. The user sets the EOUTEN[x] flag of EOUT_SIG_EN register to 0.
3. The software does not clear the FCCU status flag related to the fault x after its first occurrence.
4. The fault x triggers again after the functional reset. Under these circumstances, the FOSU triggers a destructive reset after the FOSU timeout.

The rationale behind is that the functional reset is not able to clear the source of the fault. If the software can't react to the second occurrence of the fault before the FOSU timeout expires, the FOSU triggers a destructive reset.

If the users want to increase the availability of the system by avoiding the destructive reset, they can set the EOUTEN[x] flag of EOUT_SIG_EN register to 1. In such a case, the FOSU does not trigger the destructive reset.

Note: FOSU is triggered on assertion of enabled fault. In case the triggering fault is disabled, FOSU times-out without reaction. All intermediate faults are masked (not monitored) during this timeout cycle.

Figure 2363. FOSU connections to the FCCU and MC_RGM



It is important to remember that there will be no FCCU reaction to an error while the FCCU is in the CONFIG state. For this reason, the FCCU should not be kept in CONFIG for longer than the FOSU_COUNT duration. Otherwise, there is a risk that an incoming error report causing the FOSU to mistakenly see the FCCU as having failed, and then resets the MCU.

89.8 Use cases and limitations

89.8.1 Mis-configurations

The following configurations are appropriate:

1. If at least one reaction to fault is enabled by default then the corresponding fault should also be enabled by default.
2. Enable a fault but disabling all reactions is not a meaningful configuration from safety point of view.
3. Disable the fault and also its reactions. Applicable when a specific fault line is not of interest in a specific application scenario.

Example: FCCU goes into ALARM state due to a failure, but the IRQ is disabled for that failure.

4. Software reset is optional.
5. The user must program the FCCU for a higher timeout value than the value hard-coded for FOSU. In this way FCCU reacts later than FOSU expects and as a result, FOSU generates a destructive reset request on its timeout, if a fault occurred when the FCCU was in CONFIG state.

89.8.2 Recommendations to configure FCCU

1. After Power on reset, where both system and FCCU get reset, the following steps could be followed to configure FCCU:
   a) Check and clear any pending fault status.
   b) Verify FCCU is in NORMAL state, else repeat step (a) above.
   c) Configure FCCU.
2. After functional reset of the system, arising out of reset request from FCCU or other sources, the following steps could be followed to reconfigure FCCU:
   a) If active, wait for the Error out Tmin to expire.
   b) Check and clear fault status.
   c) Error pin moves to “non faulty” state, once fault status is cleared and Tmin expires.
   d) Reset FCCU by register bit in RGM. All FCCU registers and Error pin initialize to default state.
   e) Read and verify default value in FCCU_RF_Ex to ascertain reset of FCCU.
   f) Check and clear any new pending fault status until FCCU is in NORMAL state.
   g) Reconfigure FCCU.

Note: Software reset of FCCU by RGM_PRST5.FCCU_RST bit is optional since FCCU registers are not initialized during functional reset. RGM_PRST5.FCCU_RST has the same effect on FCCU as power on reset. If the bit is activated, it can change the state of the error pin, if the current state of the pin is different from default.

90 Self-Test Control Unit (STCU2)

90.1 Introduction

The self-test control unit (STCU2) is a comprehensive programmable hardware module that controls the self-test sequence applied both during the offline or online, or both, conditions. The hardware can manage the logic built-in self-test (LBIST) and the Memory (SRAM or ROM) built-in self-test (MBIST) blocks of the device. The STCU2 includes the SSCM DCF bus to load the self-test parameters (LBIST or MBIST scheduling activity, LBIST setup, unrecoverable faults (UF) or recoverable faults (RF) management, CRC and MISR expected values, PLL management, and so on) from Flash memory during the offline self-test phase. This interface is only able to write the configuration parameters and start the self-test execution once the STCU2 global reset has been applied. Register access by software is granted by an IPS interface to check the results of the offline self-test and to load or check the execution of the online self-test.

The STCU2 includes:

- A CRC block that can be optionally enabled to monitor the internal critical signals during the self-test run to increase the STCU2's self-checking capability.
- A programmable watchdog timer (WDG) to check the LBIST and MBIST self-test operations have completed within the assigned time slot or the STCU2 RUN or BYPASS bits have been programmed before watchdog timeout.
- Direct PLL control during the offline self-test sequence.
- Two interrupt lines enabled only during the online self-test to simplify the software handling of self-test execution end.
- Two sets of registers to separately collect the LBIST or MBIST results during offline and online self-test.

To increase the security of this module, a different security key sequence is also required during offline and online self-test to unlock write access to the STCU2 registers. The STCU2 also implements two switch off power-saving mechanisms:

- The global STCU2 clock after self-test sequence is completed.
- The register ITF clock when the WDG timeout is detected after a double security key sequence is applied during the application run (available only when IPS write access is enabled otherwise the clock is off).

90.2 Main features

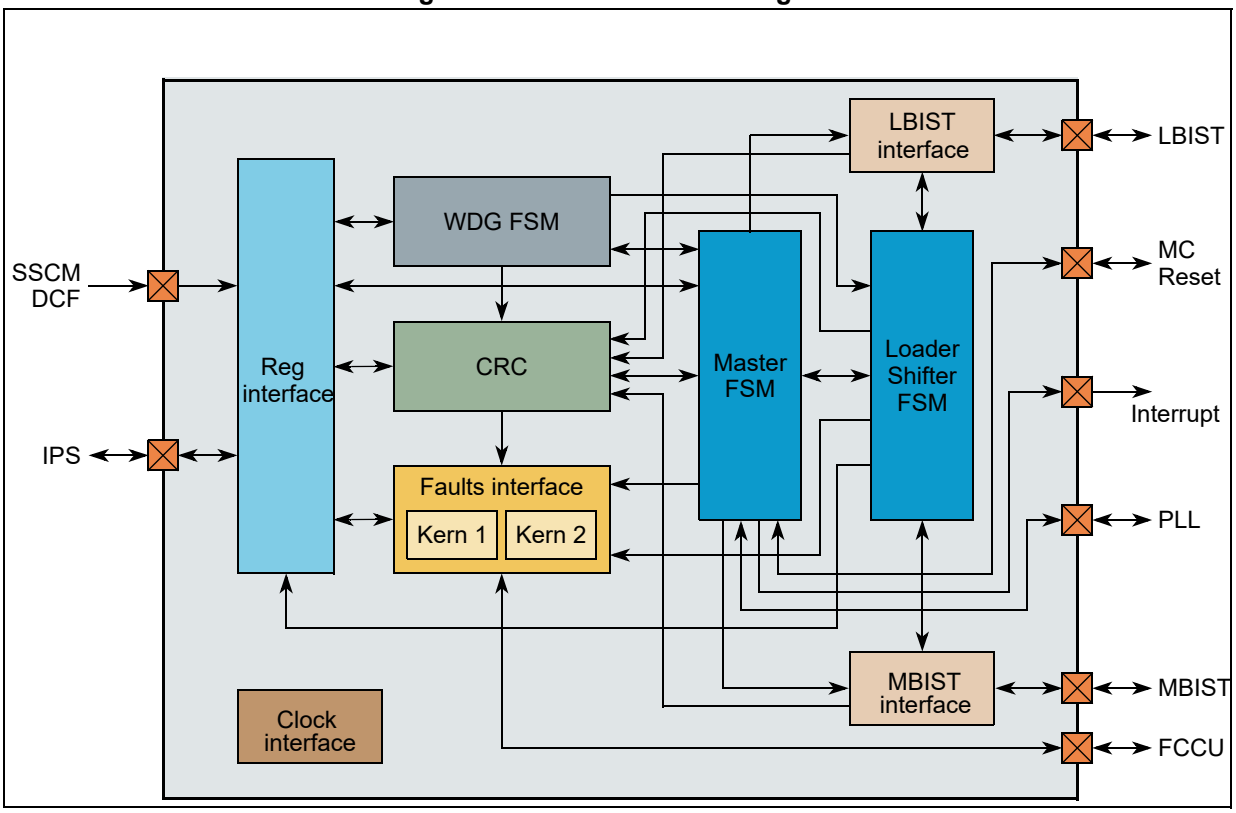
- Double registers interface (SSCM DCF with priority over IPS).
- SSCM DCF write once register interface.
- IPS read and write (depending on SSCM enable parameters and STCU_CFG[WRP] bit) register interface.
- Programmable scheduler for LBIST or MBIST execution.
- LBIST concurrent/sequential execution.
- MBIST concurrent/sequential execution.
- Programmable LBIST delayed concurrent start.

- 32-bit CRC for STCU2 self-checking capability (enabled by STCU_CFG[CRCEN] bit).
- Online hardware self-test sequence abort capability.
- Programmable internal clock prescaler to reduce internal and MBIST or LBIST TCK clocks.
- Programmable PLL direct control during offline self-test sequence.
- PLL lock signal monitoring during offline and online self-test.
- MBIST dedicated online interrupt source generation.
- Programmable WDG timer for self-check MBIST or LBIST time execution.
- Hard-coded WDG timeout to flag offline initialization errors.
- Hard-coded WDG timeout to auto-lock STCU2 write access after double key sequence.
- LBIST and SRAM or ROM MBIST offline status flags.
- STCU2 internal errors offline status flag.
- LBIST and SRAM or ROM MBIST online status flags.
- STCU2 internal errors online status flag.
- Online hardware abort status flag.
- LBIST and SRAM or ROM MBIST offline or online programmable failure severity (UF/RF).
- STCU2 internal errors offline or online programmable failure severity (UF/RF).
- STCU2 errors (UF/RF) lines for FCCU signaling.
- Redundant UF/RF generation logic to improve safety.
- FCCU UF/RF faults injection mechanism.
- CRCE register access to perform redundant check by software.
- LBIST MISRE registers access to perform redundant software checks.
- Self-test bypass capability.
- Protection mechanism of the global write register based on double security key.
- STCU2 global auto power-saving when:
  - Self-test is completed.
  - bypass mode is selected.
  - WDG timeout is detected.
- STCU2 ITF/WDG clock wake-up mechanism when software application writes double security key sequence (available when STCU_CFG[WRP] bit is active).
- STCU2 ITF/WDG auto power-saving when hard-coded WDG timeout is detected after software application writes double security key sequence (available when STCU_CFG[WRP] bit is active).

90.3 Block diagram

A top level diagram of the STCU2 module is given in *Figure 2364*.

Figure 2364. STCU2 block diagram



The STCU2 module includes the following submodules:

- **Reg interface**: This module includes the registers, the security key logic, the IPS and SSCM DCF bus interface.
- **CRC**: This module is the core of the self-checking architecture of the STCU2. It samples a set of selected internal signals when STCU2 is running.
- **Fault interface:** This module collects the error conditions respective to LBIST or MBIST execution and STCU2 internal failures and, depending on the UF/RF configuration of each of them, sets the global STCU2 UF or RF flag. It also manages the UF/RF fault lines and the set or clear injection mechanism provided by the FCCU. To improve the intrinsic safety of this critical logic, the generation logic is duplicated.
- **Clock Block:** This module manages the internal and the LBIST or MBIST TCK clock prescaler, the internal clock gating power saving and the wake-up clock feature.
- **WDG FSM:** This module includes two different watchdogs:
  - A hard-coded Watchdog used to auto-lock the STCU2 access, forcing a reset condition on the double security key registers.
  - A dual function watchdog:

    - After a reset event initializes the STCU2, it is used as a hard-coded watchdog timeout.

    - During LBIST or MBIST runs, it is used as a programmable watchdog timer to check the LBIST or MBIST have completed in the designated time slot.
- **STCU2 Master FSM**: This module includes the main FSM of the STCU2 used to coordinate and schedule all the operations performed during the self-test sequence.

- **STCU2 Load Shifter FSM:** This module includes the common shifter register and the respective state machine used to program the LBIST or MBIST registers and read back the data to be checked at the end of each self-test operation.
- **LBIST interface**: This module includes the interface between the LBIST engines and the STCU2 controllers of the device.
- **MBIST interface**: This module includes the interface between the MBIST controller and STCU2 controllers.

90.3.1 IPS bus interface

The IPS bus interface is a slave bus used for configuration purposes via CPU. The following bus read operations (contiguous byte enables) are supported:

- Word (32 bits) data read operations to any registers.
- Low and high half-words (16 bits, data[31:16] or data[15:0]) data read operations to any registers.
- Byte (8 bits, data[31:24] or data[23:16] or data[15:8] or data[7:0]) data read operations to any registers.
- Any other operation (free byte enables or other operations) has to be avoided.

The same operations are also supported in write mode under the following circumstances:

- The self-test sequence parameters are loaded before offline self-test is executed.
- The IPS access on some specific registers (see *Section 90.5: Register description*) is allowed by software only when the bit STCU_CFG[WRP] is cleared.
- Write operations are not supported and all STCU2 registers can only be read in all other circumstances except for the following R/W bits: STCU_CFG[WRP], STCU_ERR_STAT[UFSF] and STCU_ERR_STAT[RFSF]. See the respective register descriptions for additional details.

The STCU2 module generates a transfer error in the following cases:

- Any write/read access to the register addresses not mapped on the peripheral but included in the address space of the peripheral.
- Any write/read operation different from byte/halfword/word (free byte enables or other operations) on each register.
- Any write operation on double security key register applying a wrong sequence of keys (the two write operations cannot be interleaved with other accesses to STCU2 registers).
- Any write operation performed on a register when the double security keys have not been applied.
- Any write operation performed on registers when the STCU_CFG[WRP] bit is set and the access is performed through software (IPS interface).
- Any write operation performed through software (IPS interface) on row registers.
- Any write operation performed offline on registers where writes are only allowed during the online self-test phase when STCU_CFG[WRP] = 1.
- Any write operation performed on read only registers.

The registers of the STCU2 module are accessible (read/write) in each access mode: user, supervisor or test.

A transfer error is not generated for write operations to bits marked *Reserved*.

For more details regarding register protection features, see *Section 90.5: Register description*.

90.4 Functional description

90.4.1 FSM description

The module has three state machines that work together:

- Master State Machine: the core unit of the STCU2 module; it coordinates all the self-test operations and the other state machines.
- Loader/Shifter State Machine: used to program the MBIST and the LBIST parameters and to retrieve the respective results depending on the parameters stored in the STCU2 registers and under the control of the master state machine.
- Watchdog State Machine: evaluates all the schedule time for MBIST and LBIST and the timeouts in case of incorrect STCU2 programming.

90.4.2 Reset management

The asynchronous reset line initializes the STCU2 module status, forcing the offline self-test condition and the cleaning up of all the registers and state machines of the module.

The reset signals generated by the STCU2 module at the end of the self-test execution phase depends on the self-test conditions and on the content of the STCU_LBRMSW register reported in the following subsections.

In case of unrecoverable fault during offline self-test execution, the STCU2 generates a request of destructive reset to the associated MC_RGM module.

90.4.2.1 Offline reset generation

At the end of the offline self-test execution, the STCU2 raises the global functional reset. In case of unrecoverable fault during offline self-test execution, the STCU2 generates a request of destructive reset to the MC_RGM module.

90.4.2.2 Online reset generation (only MBIST)

The online execution of the MBIST does not generate any reset from STCU2 at the end of the execution. This is because the software application has to manage all the possible consequences respective to the write operations performed by the MBIST execution other than prevent any read/write operation during the self-test execution. If necessary, the software must also restore the memory content (only for RAM) at the end of the self-test execution.

90.4.2.3 Online Reset generation (LBIST enabled)

At the end of the execution, the online execution of the LBIST generates two possible types of reset depending on the status of the STCU_LBRMSW register. See *Section 90.5.16: STCU2 Online LBIST Reset Management (STCU_LBRMSW)*.

If all the selected LBISTs do not require the global functional reset, only the partitions subjected to LBIST are cleaned up applying a pulse on the dedicated reset line. If at least one of the selected LBISTs requires the global functional reset, the global functional reset is raised at the end of the online self-test execution.

Note: If an ABORT condition is detected by STCU2 during the online self-test execution, regardless of the value of the STCU_LBRMSW register, only the respective reset line is pulsed to prevent any potential dangerous system level operations

90.4.3 LBIST and MBIST scheduling

The STCU2 module is designed to be very flexible, allowing the programming of the parallel or serial execution of the MBIST or LBIST depending on the power, timing and coverage constraints. The limitation in the programming flexibility is described in *Section 90.6.4: Design implementation information*.

The mechanism used to provide this flexibility is a linked list where the starting pointer is the bitfield STCU_CFG[PTR]. The first LBIST is mapped on 0x00, the second on 0x01, and so on. The first MBIST is mapped at the address 0x10, the second on 0x11 and so on. The additional pointers are in the bitfield STCU_LB_CTRL*x*[PTR] for LBIST*x* (where *x* is the selected LBIST) and STCU_MB_CTRL*y*[PTR] for MBIST (where *y* is the selected MBIST), and have to be filled depending on the selected sequence of run. The additional bitfields STCU_LB_CTRL[CSM] and STCU_MB_CTRL[CSM] provide the flexibility to run the chosen set of the LBIST or MBIST concurrently or sequentially, or to close the linked list by setting the NIL pointer. For more details, see *Section 90.5.38: STCU2 LBIST Control n Register (STCU_LB_CTRLn)* and *Section 90.5.50: STCU2 MBIST Control k Register (STCU_MB_CTRLk)*.

90.4.4 ABORT management

The STCU2 module provides online self-test execution abort. When the abort is detected, the online self-test operation stops running and the status of the currently running MBIST or LBIST is saved in the respective registers to provide intermediate results that might be useful in case of debug or a concurrent run. For an LBIST run, see *Section 90.4.2: Reset management*, regarding reset specific management when abort is detected.

90.4.4.1 Hardware ABORT management

The STCU2 enters this abort condition as a consequence of a functional reset that requires the self-test execution to stop.

When hardware abort is detected, the STCU_ERR_STAT[ABORTHW] bit is set. This flag allows the software to diagnose what occurred during the online self-test execution run.

90.4.5 PLL interface

The STCU2 module is able to directly control the PLL during the offline self-test operations depending on the status of the STCU_RUN[MBPLLEN] and STCU_RUN[LBPLLEN] as described in *Section 90.5.2: STCU2 Run Software Register (STCU_RUNSW)*.

When STCU_RUN[RUN] is set to start the offline self-test operations, the STCU2 takes control of the PLL and changes the PLL clock configuration parameters according to the values programmed in the STCU_PLL_CFG register.

When the STCU_RUN[MBPLLEN] is active and the MBIST is currently selected, the STCU2 waits for the PLL lock. As soon as the lock occurs, the STCU forces the clock to switch to the PLL output clock, allowing the MBIST to proceed at the PLL output frequency. At the end of the MBIST run, the clock source reverts to the original one.

The same behavior applies when STCU_RUN[LBPLLEN] is active and LBIST is currently selected.

90.4.6 Interrupt interface

There are two interrupt sources:

- One triggered by MBIST (enabled by programming STCU_RUNSW[MBIE] = 1).
- One triggered by LBIST (enabled by programming STCU_RUNSW[LBIE] = 1).

As described in *Section 90.5.2: STCU2 Run Software Register (STCU_RUNSW)*, the interrupts are generated at the end of any concurrent run. This means that in case of multiple sequential runs in the same session, the number of interrupts generated is equivalent to the number of sequential sessions, thus providing fine granularity with respect to the completed run.

The MBIFLG bit in the *Section 90.5.7: STCU2 Interrupt Flag Register (STCU_INT_FLG)* registers flag interrupt condition and allows the clearing of interrupt requests. Interrupt bit to be cleared by writing the respective bit to 1 to enable the next interrupt request.

90.4.7 FCCU interface

The FCCU interface is the hardware flag mechanism to the system, indicating the occurrence of an unrecoverable failure (UF) or a recoverable failure (RF) during the self-test sequence.

To diagnose physical defects on the two fault signals, a fault injection mechanism is also provided. The FCCU interface allows the user application to check the integrity of the UF and RF connection lines between the STCU2 and the FCCU. Refer to the description of FCCU fault injection mechanism to understand how the UF/RF set and clear mechanism works.

90.4.8 Watchdogs

The STCU2 implements three different watchdogs to ensure that operations are finished on time.

90.4.8.1 Initialization sequence

The STCU2 has two initialization operating modes:

- Program and run the self-test sequence (safety mode).
- Bypass the self-test sequence by setting the STCU_RUN[BYP] bit. If Bypass bit is configured/set by DCF it cannot be anymore changed on the device even though STCU_RUN register itself is not Write once.

If there are faults during the initialization phase which prevent the selection of one of these two operating modes, a hard-coded watchdog timeout flags the incorrect behavior. See *Section 90.5.6: STCU2 Watchdog Register Granularity (STCU_WDG)* for more details.

90.4.8.2 LBIST and MBIST scheduling

The LBIST and MBIST execution time is configured as described in *Section 90.5.6: STCU2 Watchdog Register Granularity (STCU_WDG)* to account for the overall execution time of the self-test sequence. If the selected LBISTs or MBISTs are not yet completed:

- During an offline self-test assigned time, the current LBISTs or MBISTs execution is interrupted and a failure is flagged in STCU_ERR_STAT[WDTO] and STCU_MBE1/2/3/4 or STCU_LBE registers.
- During an online self-test, the current LBISTs or MBISTs execution is interrupted and a failure is flagged in STCU_ERR_STAT[WDTOSW] and STCU_MBE1SW/2SW/3SW/4SW or STCU_LBESW.

In case of multiple sequential runs in the same online or offline session and the timeout occurs in the middle of a sequential run, the next sequential run is skipped and the execution ends with the current updated status of the registers reported above.

In all cases, the STCU2 reset generation is managed in one of the modes described in *Section 90.4.2: Reset management*.

90.4.8.3 Write access timeout

The access on the STCU2 registers during the self-test configuration phase (both online or offline) is protected by a key mechanism to prevent any unwanted access, as described in the STCU_SKC register description. In addition to this safety feature, there is a hardware watchdog that, after 1024 STCU2 clock cycles, locks access and requires reinsertion of the STCU_SKC keys.

This feature is particularly useful if STCU_CFG[WRP] = 0 during the software self-test configuration as the software application may enable write access to the STCU2 registers.

90.4.9 CRC

The CRC block is used to increase the intrinsic coverage of the STCU2 module and implements a 32-bit ethernet protocol polynomy to evaluate the signature of the most important and critical internal signals. This feature is switched off by default but it can be turned on setting the STCU_CFG[CRCEN] bit. Once this bit is set, the expected signature has to be programmed in the STCU_CRCE register. At the end of the self-test execution, the STCU2 updates the content of the STCU_CRCR register with the evaluated signature and compares this value with the expected one previously programmed in STCU_CRCE. In case of mismatches, a UF/RF condition is flagged depending on the fault mapping selected in the STCU_ERR_FM register.

Note: The expected value of the CRC depends on multiple internal critical signals and can differ among different samples.

90.4.10 SSCM interface

The SSCM interface is used to program the STCU2's configuration parameters without CPU intervention after a reset trigger event initializes the STCU2. This bus interface has priority over the IPS.

90.5 Register description

The STCU2 registers are listed in *Table 2251*.

Note: n and k are variables representing the repeated register blocks of the multiple LBISTs and MBISTs: n ranges from 0 up to 6, k ranges from 0 up to 101.

Table 2251. STCU2 memory map

Address offset	Register Name	Location
0x0000	STCU2 Run Register (STCU_RUN)	*Section 90.5.1*
0x0004	STCU2 Run Software Register (STCU_RUNSW)	*Section 90.5.2*
0x0008	STCU2 SK Code Register (STCU_SKC)	*Section 90.5.3*
0x000C	STCU2 Configuration Register (STCU_CFG)	*Section 90.5.4*
0x0010	STCU2 PLL Configuration Register (STCU_PLL_CFG)	*Section 90.5.5*
0x0014	STCU2 Watchdog Register Granularity (STCU_WDG)	*Section 90.5.6*
0x0018	STCU2 Interrupt Flag Register (STCU_INT_FLG)	*Section 90.5.7*
0x001C	STCU2 CRC Expected Status Register (STCU_CRCE)	*Section 90.5.8*
0x0020	STCU2 CRC Read Status Register (STCU_CRCR)	*Section 90.5.9*
0x0024	STCU2 Error Register (STCU_ERR_STAT)	*Section 90.5.10*
0x0028	STCU2 Error FM Register (STCU_ERR_FM)	*Section 90.5.11*
0x002C	STCU2 offline LBIST Status Register (STCU_LBS)	*Section 90.5.12*
0x0030	STCU2 offline LBIST End Flag Register (STCU_LBE)	*Section 90.5.13*
0x0034	STCU2 online LBIST Status Register (STCU_LBSSW)	*Section 90.5.14*
0x0038	STCU2 online LBIST End Flag Register (STCU_LBESW)	*Section 90.5.15*
0x003C	STCU2 online LBIST Reset Management (STCU_LBRMSW)	*Section 90.5.16*
0x0040	STCU2 LBIST Unrecoverable FM Register (STCU_LBUFM)	*Section 90.5.17*
0x0044	STCU2 offline MBIST 1 Status Register (STCU_MBS1)	*Section 90.5.18*
0x0048	STCU2 offline MBIST 2 Status Register (STCU_MBS2)	*Section 90.5.19*
0x004C	STCU2 offline MBIST 3 Status Register (STCU_MBS3)	*Section 90.5.20*
0x0050	STCU2 offline MBIST 4 Status Register (STCU_MBS4)	*Section 90.5.21*
0x0054	STCU2 offline MBIST 1 End Flag Register (STCU_MBE1)	*Section 90.5.22*
0x0058	STCU2 offline MBIST 2 End Flag Register (STCU_MBE2)	*Section 90.5.23*
0x005C	STCU2 offline MBIST 3 End Flag Register (STCU_MBE3)	*Section 90.5.24*
0x0060	STCU2 offline MBIST 4 End Flag Register (STCU_MBE4)	*Section 90.5.25*
0x0064	STCU2 online MBIST 1 Status Register (STCU_MBS1SW)	*Section 90.5.26*
0x0068	STCU2 online MBIST 2 Status Register (STCU_MBS2SW)	*Section 90.5.27*
0x006C	STCU2 online MBIST 3 Status Register (STCU_MBS3SW)	*Section 90.5.28*
0x0070	STCU2 online MBIST 4 Status Register (STCU_MBS4SW)	*Section 90.5.29*
0x0074	STCU2 online MBIST 1 End Flag Register (STCU_MBE1SW)	*Section 90.5.30*

Table 2251. STCU2 memory map (continued)

Address offset	Register Name	Location
0x0078	STCU2 online MBIST 2 End Flag Register (STCU_MBE2SW)	*Section 90.5.31*
0x007C	STCU2 online MBIST 3 End Flag Register (STCU_MBE3SW)	*Section 90.5.32*
0x0080	STCU2 online MBIST 4 End Flag Register (STCU_MBE4SW)	*Section 90.5.33*
0x0084	STCU2 MBIST 1 Unrecoverable FM Register (STCU_MBUFM1)	*Section 90.5.34*
0x0088	STCU2 MBIST 2 Unrecoverable FM Register (STCU_MBUFM2)	*Section 90.5.35*
0x008C	STCU2 MBIST 3 Unrecoverable FM Register (STCU_MBUFM3)	*Section 90.5.36*
0x0090	STCU2 MBIST 4 Unrecoverable FM Register (STCU_MBUFM4)	*Section 90.5.37*
0x0094–0x00FF	Reserved	
LBIST registers (*n* = 0 to 6)		
0x0100 + *n* * 0x40	STCU2 LBIST Control *n* Register (STCU_LB_CTRL*n*)	*Section 90.5.38*
0x0104 + *n* * 0x40	STCU2 LBIST PC Stop *n* Register (STCU_LB_PCS*n*)	*Section 90.5.39*
0x0108 + *n* * 0x40	STCU2 LBIST PRPG Low *n* Register (STCU_LB_PRPGL*n*)	*Section 90.5.40*
0x010C + *n* * 0x40	STCU2 LBIST PRPG High *n* Register (STCU_LB_PRPGH*n*)	*Section 90.5.41*
0x0110 + *n* * 0x40	STCU2 offline LBIST MISR Expected Low *n* Register (STCU_LB_MISREL*n*)	*Section 90.5.42*
0x0114 + *n* * 0x40	STCU2 offline LBIST MISR Expected High *n* Register (STCU_LB_MISREH*n*)	*Section 90.5.43*
0x0118 + *n* * 0x40	STCU2 offline LBIST MISR Read Low *n* Register (STCU_LB_MISRRL*n*)	*Section 90.5.44*
0x011C + *n* * 0x40	STCU2 offline LBIST MISR Read High *n* Register (STCU_LB_MISRRH*n*)	*Section 90.5.45*
0x0120 + *n* * 0x40	STCU2 online LBIST MISR Expected Low *n* Register (STCU_LB_MISRELSW*n*)	*Section 90.5.46*
0x0124 + *n* * 0x40	STCU2 online LBIST MISR Expected High *n* Register (STCU_LB_MISREHSW*n*)	*Section 90.5.47*
0x0128 + *n* * 0x40	STCU2 online LBIST MISR Read Low *n* Register (STCU_LB_MISRRLSW*n*)	*Section 90.5.48*
0x012C + *n* * 0x40	STCU2 online LBIST MISR Read High *n* Register (STCU_LB_MISRRHSW*n*)	*Section 90.5.49*
MBIST registers (*k* = 0 to 101)		
0x0600 + *k* * 0x4	STCU2 MBIST Control Register (STCU_MB_CTRL*k*)	*Section 90.5.50*

90.5.1 STCU2 Run Register (STCU_RUN)

The STCU_RUN register defines the RUN bit to start the offline self-test procedure if some configuration bits are required *only* during this self-test phase.

The R/W fields in this register are readable at any time. However, you can write to these fields only when offline self-test phase is still active.

Offset: 0x0000 Access: User read/write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	BYP	MBPLLEN	LBPLLEN	0	0	0	0	0	0	0	RUN
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 2365. STCU2 Run Register (STCU_RUN)

Table 2252. STCU_RUN field descriptions

Field	Description
21 BYP	Bypass mode 0 Offline self-test is active. If the STCU_RUN[RUN] bit is not set before the hard-coded WDG timeout, the WDTO error is generated in the STCU_ERR_STAT register. 1 Offline self-test is bypassed and access to STCU2 is locked until the STCU_SCK register is written according to the request access mode.
22 MBPLLEN	Offline MBIST with PLL0 Enabled 0 Offline MBIST is executed without using the on-chip PLL0. 1 Offline MBIST is executed enabling the on-chip PLL0 control interface selecting the parameters defined in STCU_PLL_CFG register.
23 LBPLLEN	Offline LBIST with PLL0 Enabled 0 Offline LBIST is executed without using the on-chip PLL0. 1 Offline LBIST is executed enabling the on-chip PLL0 control interface selecting the parameters defined in STCU_PLL_CFG register.
31 RUN	RUN The RUN bit is automatically cleared by the STCU2 when the offline self-test procedure has been completed. 0 Idle 1 Offline self-test procedure is running

90.5.2 STCU2 Run Software Register (STCU_RUNSW)

The STCU_RUNSW register defines the RUN bit to start the online self-test procedure if some configuration bits are required *only* during this self-test phase.

The R/W fields in this register are readable at any time. However, you can write to these fields only when both of the following conditions are true:

- STCU_CFG[WRP] = 0.
- Online self-test phase is active. The online condition starts when the offline condition finishes. In this condition, the system is alive and the SW can program the STCU.

Offset: 0x0004 Access: User read/write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	MBIE	LBIE	MBSWPLLEN	LBSWPLLEN	0	0	0	0	0	0	0	RUNSW
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 2366. STCU2 Run Software Register (STCU_RUNSW)

Table 2253. STCU_RUNSW field descriptions

Field	Description
20 MBIE	MBIST Interrupt Enable 0 Interrupt is not generated at the end of the software MBIST execution phase of the selected memories. 1 At the end of the software MBIST execution phase of the concurrent selected memories, the STCU2 forces an interrupt request to notify MBISTs of that concurrent selected memories are concluded. If there is a sequence of concurrent runs in the same self-test run, the interrupt request is generated at the end of each concurrent run provided that the software clears the STCU_INT_FLG[MBIFG] bit in between.
21 LBIE	LBIST Interrupt Enable 0 Interrupt is not generated at the end of the software LBIST execution phase of the selected LBIST. 1 At the end of the software LBIST execution phase of the concurrent selected LBIST, the STCU2 forces an interrupt request to notify LBIST of the concurrent selected partitions are concluded. If there is a sequence of concurrent run in the same self-test run, the interrupt request is generated at the end of each concurrent run provided that the software clears the STCU_INT_FLG[LBIFG] bit in between.
22 MBSWPLLEN	Online MBIST with PLL1 Enabled 0 Online MBIST is executed without waiting for PLL1 lock. 1 Online MBIST is executed once PLL1 is locked. PLL1 configuration used is the one specified in the PLLDIG PLL1DV register. STCU2 does not take PLL1 control but monitors the PLL1 lock signal to check if PLL1 is working correctly.

Table 2253. STCU_RUNSW field descriptions (continued)

Field	Description
23 LBSWPLLEN	Online LBIST with PLL1 Enabled 0 Online LBIST is executed without waiting for PLL1 lock. 1 Online LBIST is executed once PLL1 is locked. PLL1 configuration used is the one specified in the PLLDIG PLL1DV register. STCU2 does not take the PLL1 control but monitors the PLL1 lock signal to check if PLL1 is working correctly.
31 RUNSW	RUNSW The RUNSW bit is automatically cleared by the STCU2 when the online self-test procedure has been completed. 0 Idle. 1 Online self-test procedure is running.

90.5.3 STCU2 SK Code Register (STCU_SKC)

The STCU_SKC register implements the security key code mechanism needed for write access to the other STCU2 registers. In order to unlock the STCU2 access after an STCU2 asynchronous reset or the end of the STCU2 run, the software (IPS bus) or the SSCM interfaces must apply the following sequence:

- Write Key1 in the STCU_SKC register.
- Write Key2 in the STCU_SKC register.

Depending on the offline or online test step, the two keys will be different. Byte write operation is not allowed as the full key has to be recognized as one unit.

Note: Note that after the self-test sequence has been completed or the bypass feature has been enabled (by setting the STCU_RUN[BYP] bit), the SSCM interface is no longer available.

In case of invalid access or sequence (Keys1 and 2 must be applied consecutively), a transfer error on the IPS or SSCM bus is asserted depending on the selected source. The STCU2 write access is locked until the correct sequence is applied.

If the STCU2 register access lasts more cycles than defined in the hard-coded WDG timeout, STCU2 write access is locked and the WDG and register ITF clocks are switched off. Also in this case the correct sequence must be applied in order to re-enable write access to the STCU2 and the WDG and register ITF clocks.

To extend the STCU2 register access cycles before the hard-coded WDG timeout expires, only Key2 needs to be applied. The effect of this write operation is to re-initialize the WDG timeout counter. Key1 must not be applied or a transfer error on the IPS or SSCM bus is asserted depending on the selected source. The STCU2 write access is locked until the correct sequence is applied.

The STCU_SKC register is not readable; any attempts to read this register always returns the value 0x00000000.

Offset: 0x0008 Access: User write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0															
W	SKC															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0															
W	SKC															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 2367. STCU2 SK Code Register (STCU_SKC)

Table 2254. STCU_SKC field descriptions

Field	Description
0:31 SKC	STCU2 security key code for offline test 0xD3FEA98B = Key1 to unlock the write access the STCU2 0x2C015674 = Key2 to unlock the write access the STCU2 STCU2 security key code for online test 0x753F924E = Key1 to unlock the write access the STCU2 0x8AC06DB1 = Key2 to unlock the write access the STCU2

90.5.4 STCU2 Configuration Register (STCU_CFG)

The STCU_CFG register includes the global configuration of the STCU2 and can be updated both in the offline and online test steps.

Access to this register is described in *Figure 2368*. It also depends on the state of the WRP bit as follows:

- When WRP = 0, the register can be written during the online self-test case without restrictions
- When WRP = 1:
  - The only bit that can be written without any restriction is WRP.
  - If software operations write to other register bits, a transfer error is raised if the value of the selected byte that is written differs from the current status of the register. This function is implemented to prevent potential compilation behavior that might invalidate this single bit clean capability.

Offset: 0x000C Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	PTR(1)								LB_DELAY(1)							
Reset	The reset value is chip-specific; see the chapter that describes how modules are configured and connected.															

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	WRP	0	0	CRCEN(1)	PMOSEN(1)	MBU(1)	CLK_CFG(1)		
W																
Reset	The reset value is chip-specific; see the chapter that describes how modules are configured and connected.															

1. You can always read this field. In the offline self-test phase, you can always write to this field. In the online self-test phase, you can write to this field only when WRP = 0.

Figure 2368. STCU2 Configuration Register (STCU_CFG)

Table 2255. STCU_CFG field descriptions

Field	Description
0:7 PTR	LBIST or MBIST pointer. PTR defines the logical pointer to the first LBIST or MBIST to be scheduled when the self-test procedure is enabled. PTR is the entry pointer to a linked list of BIST descriptors. If PTR = 0, then the LBIST0 is initially scheduled. (0x00 to LBIST - 1) – pointer to LBIST (0x10 to 0x10 + MBIST - 1) pointer to MBIST 0xFF pointer to NIL. No BIST execution.
8:15 LB_DELAY	Delay LBIST run. LB_DELAY defines the delay between the LBIST starts when more than a single LBIST is selected to be executed concurrently with the purpose of smoothing the power consumption transient. 0x00 No delay 0x01 1 × 16 STCU2 Core clock cycles 0x02 2 × 16 STCU2 Core clock cycles 0x03 3 × 16 STCU2 Core clock cycles … 0xFD 253 × 16 STCU2 Core clock cycles 0xFE 254 × 16 STCU2 Core clock cycles 0xFF 255 × 16 STCU2 Core clock cycles **Note:** STCU2 is being clocked from PBRIDGE Clock. Refer to Clocking chapter for PBRIDGE clock frequency value.
23 WRP	Write Protection. 0 Specific STCU2 registers can be written through IPS bus interface after offline self-test sequence has completed. 1 Specific STCU2 registers cannot be written though IPS even after offline self-test sequence has completed, thus preventing any user application write operation.

Table 2255. STCU_CFG field descriptions (continued)

Field	Description
26 CRCEN	CRC Enable comparison. 0 The CRC comparison is not performed and the STCU_ERR_STAT[CRCS][SW] and STCU_ERR_STAT[UF/RF] global flags are not updated in case of mismatches between STCU2[CRCE] and STCU2[CRCR] values. 1 The CRC comparison is performed and the status is updated in the respective flags of the STCU_ERR_STAT register
27 PMOSEN	MBIST PMOS Test Enable. 0 MBIST PMOS test is not enabled 1 MBIST PMOS test is enabled
28 MBU	MBIST MBU Test Enabled. 0 MBIST full algorithm or reduced one without PMOS test (PMOSEN set to 0) is applied. 1 MBIST simplified multi-bit upset algorithm is used to check the RAM. If a ROM is selected, the applied algorithm will be standard one without PMOS test. ***This bit overwrites the PMOSEN feature***.
29:31 CLK_CFG	Logic, memory BIST and STCU2 core clock configuration. CLK_CFG defines the ratio between the STCU2 core clock and the TCK used to program the LBIST, the MBIST and the STCU2 core clock. The punch-out mechanism is used to generate the derived clocks. The allowed configurations are: 000 PBRIDGE_CLK 001 PBRIDGE_CLK/2 010 PBRIDGE_CLK/3 011 PBRIDGE_CLK/4 100 PBRIDGE_CLK/5 101 PBRIDGE_CLK/6 110 PBRIDGE_CLK/7 111 PBRIDGE_CLK/8 **Note:** refer to Clocking chapter for PBRIDGE_CLK clock frequency value.

90.5.5 STCU2 PLL Configuration Register (STCU_PLL_CFG)

The STCU_PLL_CFG register defines the parameters used to program the PLL only when the offline LBISTs or MBISTs are performed and STCU_RUN[MBPLLEN] = 1 or STCU_RUN[LBPLLEN] = 1.

In the online state, these bits are not effective. It is also possible to set the PLL during active online mode to prevent any potential functionality issues.

The R/W fields in this register are readable at any time. However, you can write to these fields only when offline self-test phase is active.

Offset: 0x0010 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	PLLODF						0	0	0	0	0	PLLIDF		
W																
Reset	The reset value is chip-specific; see the chapter that describes how modules are configured and connected.															

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	PLLLDF						
W																
Reset	The reset value is chip-specific; see the chapter that describes how modules are configured and connected.															

Figure 2369. STCU2 PLL Configuration Register (STCU_PLL_CFG)

Note: f_{RC} is the frequency if the internal RC oscillator (that is 16 MHz). f_{PLLin} must be higher than 8 MHz.

Table 2256. STCU_PLL_CFG field descriptions

Field	Description
2:7 PLLODF	PLL Output Division Factor. The value of this field drives the output division factor of the PLL embedded in the device. Refer to the PLL specifications to define the exact mapping between the PLLODF value and the respective PLL's output division factor.
13:15 PLLIDF	PLL Input Division Factor. The value of this field drives the input division factor of the PLL embedded in the device. Refer to the PLL specifications to define the exact mapping between the PLLIDF value and the respective PLL's input division factor.
25:31 PLLLDF	PLL Loop Division Factor. The value of this field drives the loop division factor of the PLL embedded in the device. Refer to the PLL specifications to define the exact mapping between the PLLLDF value and the respective PLL's loop division factor.

90.5.6 STCU2 Watchdog Register Granularity (STCU_WDG)

The STCU_WDG register defines the time budget for LBIST and MBIST execution, providing protection against dead-locks or infinite loop conditions during the self-test procedure. When the offline self-test sequence is not run and the STCU_RUN[BYP] bit is not set, the bits 15 to 0 define the timeout before the STCU_ERR[WDTO] bit is set and the STCU2 core clock is switched off.

When the offline or online self-test sequence is run, it defines the timeout of the execution run.

The R/W fields in this register are readable at any time. You can write to these fields when either of the following conditions is true:

- Offline self-test phase is active
- Online self-test phase is active and STCU_CFG[WRP] = 0

Offset: 0x0014 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	WDGEOC															
Reset	The reset value is chip-specific; see the chapter that describes how modules are configured and connected.															

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	WDGEOC															
Reset	The reset value is chip-specific; see the chapter that describes how modules are configured and connected.															

Figure 2370. STCU2 Watchdog Register Granularity (STCU_WDG)

Table 2257. STCU_WDG field descriptions

Field	Description
0:31 WDGEOC	Watchdog End of Count Timer. This value must be set in order to define the time budget respective to the offline and online self-test execution and check that everything is working correctly in this timeslot. The allowed time delays are: 0x00000000 1*k*STCU2 core clock cycles 0x00000001 2*k*STCU2 core clock cycles 0x00000002 3*k*STCU2 core clock cycles … 0xFFFFFFFD 4294967294*k*STCU2 core clock cycles 0xFFFFFFFE 4294967295*k*STCU2 core clock cycles 0xFFFFFFFF 4294967296*k*16 STCU2 core clock cycles where k is: – ‘1’ for offline self-test – ‘16’ for online self-test **Note:** STCU2 core clock is specified through STCU_CFG[CLK_CFG] field.

90.5.7 STCU2 Interrupt Flag Register (STCU_INT_FLG)

The STCU_INT_FLG register includes the MBIST and LBIST interrupt pending bits. These bits are effective only during the online self-test phase and managed only when the respective control bit in the STCU_RUNSW register (MBIE and LBIE) enabled.

Offset: 0x0018 Access: User read/write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	MBIFLG	LBIFLG
W															w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 2371. STCU2 Interrupt Flag Register (STCU_INT_FLG)

Table 2258. STCU_INT_FLG field descriptions

Field	Description
30 MBIFLG	MBIST Interrupt Flag. 0 No interrupt is pending. 1 An interrupt highlighting that the online execution of the concurrent scheduled MBIST is completed is pending. Write 1 to clear the bit.
31 LBIFLG	LBIST Interrupt Flag. 0 No interrupt is pending. 1 An interrupt highlighting that the online execution of the concurrent scheduled LBIST is completed is pending. Write 1 to clear the bit.

90.5.8 STCU2 CRC Expected Status Register (STCU_CRCE)

The STCU_CRCE register includes the expected signature of the CRC-32 (ethernet protocol). The CRC-32 bit engine is initialized at STCU_CRCE_RES and computes the CRC signature of the STCU2 critical signals. When STCU_CFG[CRCEN] = 1, it provides the self check capability of the STCU2 managing the CRCS[SW] bits of the STCU_ERR_STAT register. The polynomial CRC used to evaluate the status of this register is the following:

CRC-32 (ethernet protocol)

$$x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^8 + x^7 + x^5 + x^4 + x^2 + x + 1$$

The R/W fields in this register are readable at any time. You can write to these fields when either of the following conditions are true:

- Offline self-test phase is active.
- Online self-test phase is active and STCU_CFG[WRP] = 0.

Note: The expected value of the CRC depends on multiple internal critical signals and can differ among different samples.

Offset: 0x001C Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	CRCE															
Reset	The reset value is chip-specific; see the chapter that describes how modules are configured and connected.															

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	CRCE															
Reset	The reset value is chip-specific; see the chapter that describes how modules are configured and connected.															

Figure 2372. STCU2 CRC Expected Status Register (STCU_CRCE)

Table 2259. STCU_CRCE field descriptions

Field	Description
0:31 CRCE	CRC expected signature

90.5.9 STCU2 CRC Read Status Register (STCU_CRCR)

The STCU_CRCR register reports the value obtained at the end of the offline and online self-test sequence. It may be used for diagnoses and also for additional checking with respect to the CRCS[SW] bit of the STCU_ERR_STAT register.

The content of this register is initialized to its reset value as soon as STCU_RUNSW[RUNSW] = 1.

Offset: 0x0020 Access: User read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	CRCR															
W																
Reset	The reset value is chip-specific; see the chapter that describes how modules are configured and connected.															

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	CRCR															
W																
Reset	The reset value is chip-specific; see the chapter that describes how modules are configured and connected.															

Figure 2373. STCU2 CRC Read Status Register (STCU_CRCR)

Table 2260. STCU_CRCR field descriptions

Field	Description
0:31 CRCR	Read CRC signature

90.5.10 STCU2 Error Register (STCU_ERR_STAT)

The STCU_ERR_STAT register includes the status flags respective to the STCU2 internal error conditions that occurred during configuration or the online and offline self-test execution.

The Unrecoverable Faults Status Flag (UFSF) and Recoverable Faults Status Flag (RFSF) can be set and cleared using the FCCU dedicated channels as described in *Section 90.6.4: Design implementation information*.

Access to this register is described in *Figure 2374* and as follows:

- If you select the byte write capability to write only the UFSF and RFSF, then there is no restriction in writing these bits.
- If your software performs the write operations on other bits besides UFSF/RFSF, then a transfer error is generated only if the value you are writing to the other bits differs from their value currently stored in the register. This functionality is implemented to prevent potential compilation behavior that might invalidate the UFSF/RFSF single bit set and clean capability.

Offset: 0x0024 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	ABORTHW(1)	0	0	0	0	LOCKESW(1)	WDTOSW(1)	CRCSSW(1)	ENGESW(1)	INVPSW(1)
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	UFSF	RFSF	0	0	0	LOCKE	WDTO	CRCS	ENGE	INVP
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. You can always read this field. The content of this field is initialized to its reset value as soon as STCU_RUNSW[RUNSW] = 1.

Figure 2374. STCU2 Error Register (STCU_ERR_STAT)

Table 2261. STCU_ERR_STAT field descriptions

Field	Description
6 ABORTHW	Online hardware abort flag. 0 No hardware abort was requested during the online self-test sequence 1 A hardware abort was detected during the online self-test sequence
11 LOCKESW	Online LOCK Error. 0 If PLL is enabled, it is correctly locked during the self-test sequence 1 When the PLL is enabled, this flag highlights an unexpected PLL unlock event during the online self-test sequence execution. The online self-test run is stopped and the status of the current running LBISTs or MBISTs is saved in the respective registers. The LOCK signal is monitored during the LBIST run when STCU_RUNSW[LBSWPLLEN] is set or during the MBIST run when STCU_RUNSW[MBSWPLLEN] = 1, or during both.
12 WDTOSW	Online Watchdog timeout. 0 LBIST and MBIST time slots have completed within the assigned watchdog time. 1 LBIST and MBIST time slots have not completed in the assigned watchdog time or there are internal mismatches in the end of execution signals. The conditions that flag the failures are the following: – MBIST BEND status flags that at least one of the selected MBIST run is not finished – lbist_done of the selected LBIST flags the LBIST run is/are not finished
13 CRCSSW	Online CRC Status. This flag is activated *only* when online self-test is performed and STCU_CFG[CRCEN] is set to 1, otherwise it is always forced to 0. This prevents the generation of the respective STCU_ERR_STAT[UFSF/RFSF] bits. In both the cases, the content of the STCU_CRCR register reports the current evaluated CRC. 0 Successful CRC comparison or comparison has been masked (STCU_CFG[CRCEN] set to 0) 1 Failed CRC comparison
14 ENGESW	Online Engine Error. 0 Valid Engine execution 1 Invalid Engine execution. The following conditions set this bit: – STCU2 state machines (watchdog, master and loader/shifter) selects an unused code – Serial pass/fail status of the selected MBIST and the global BBAD flag are not aligned Serial bend status of the selected MBIST and the global BEND flag are not aligned
15 INVPSW	Online Invalid Pointer. 0 Valid linked pointer list 1 Invalid linked pointer list. The following conditions set this bit: – Initial LBIST or MBIST pointer is out of range – LBIST is selected when MBIST is concurrently running or vice versa – Error in LBIST or MBIST linking (execution generates an infinite loop)
22 UFSF	Unrecoverable Faults Status Flag This flag reports the global status of the UF. 0 No errors that trigger the unrecoverable faults condition 1 There are errors that trigger the unrecoverable faults condition

Table 2261. STCU_ERR_STAT field descriptions (continued)

Field	Description
23 RFSF	Recoverable Faults Status Flag 0 No errors that trigger the recoverable faults condition 1 There are errors that trigger the recoverable faults condition
27 LOCKE	Offline LOCK Error 0 The PLL is correctly locked during the self-test sequence 1 When the PLL is enabled, this flag highlights an unexpected PLL unlock event during the offline self-test sequence execution. The online self-test run is stopped and the status of the current running LBISTs or MBISTs is saved in the respective registers. The LOCK signal is monitored during the LBIST run when STCU_RUN[LBPLLEN] = 1 or during the MBIST run when STCU_RUN[MBPLLEN] = 1, or during both.
28 WDTO	Offline Watchdog timeout. 0 LBIST and MBIST time slots have been completed within the assigned watchdog time. 1 LBIST and MBIST time slots have not been completed within the assigned watchdog time or there are internal mismatches among end of execution signals. The conditions that flag the failures are the following: – The STCU_RUN[RUN] bit or the STCU_RUN[BYP] bit are not set before the watchdog reach the end of count – MBIST BEND status flags that at least one of the selected MBIST run is not finished – lbist_done of the selected LBIST flags the LBIST run is/are not finished
29 CRCS	Offline CRC status. This flag is activated *only* when offline self-test is performed STCU_CFG[CRCEN] is set to 1, otherwise it is always forced to 0. This prevents the generation of the respective STCU_ERR_STAT[UFSF/RFSF] bits. In both the cases, the content of the STCU_CRCR register reports the current evaluated CRC. 0 Successful CRC comparison or comparison has been masked (STCU_CFG[CRCEN] programmed to 0) 1 Failed CRC comparison
30 ENGE	Offline Engine Error. 0 Valid engine execution 1 Invalid engine execution. The following conditions set this bit: – STCU2 state machines (watchdog, master and loader/shifter) selects an unused code – Serial pass/fail status of the selected MBIST and the global BBAD flag are not aligned – Serial bend status of the selected MBIST and the global BEND flag are not aligned
31 INVP	Offline Invalid pointer. 0 Valid linked pointer list 1 Invalid linked pointer list. The following conditions set this bit: – Initial LBIST or MBIST pointer is out of range – LBIST is selected when MBIST is concurrently running or vice versa – Error in LBIST or MBIST linking (execution generates an infinite loop)

90.5.11 STCU2 Error FM Register (STCU_ERR_FM)

The STCU_ERR_FM register defines the fault mapping of the STCU2 faults described in the STCU_ERR_STAT register in terms of Unrecoverable or recoverable faults. All sources of internal faults can be routed to UF and RF.

The R/W fields in this register are readable at any time. You can write to these fields when either of the following conditions are true:

- Offline self-test phase is active.
- Online self-test phase is active and STCU_CFG[WRP] = 0.

Offset: 0x0028 Access: User read/write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	LOCKEUFM	WDTOUFM	CRCSUFM	ENGEUFM	INVPUFM
W																
Reset	The reset value is chip-specific; see the chapter that describes how modules are configured and connected.															

Figure 2375. STCU2 Error FM Register (STCU_ERR_FM)

Table 2262. STCU_ERR_FM field descriptions

Field	Description
27 LOCKEUFM	PLL LOCK Unrecoverable Fault Mapping 0 Recoverable fault mapping 1 Unrecoverable fault mapping
28 WDTOUFM	WatchDog timeout Unrecoverable Fault Mapping 0 Recoverable fault mapping 1 Unrecoverable fault mapping
29 CRCSUFM	CRC Status Unrecoverable Fault Mapping 0 Recoverable fault mapping 1 Unrecoverable fault mapping
30 ENGEUFM	Engine Error Unrecoverable fault Mapping 0 Recoverable fault mapping 1 Unrecoverable fault mapping
31 INVPUFM	Invalid Pointer Unrecoverable Fault Mapping 0 Recoverable fault mapping 1 Unrecoverable mapping

90.5.12 STCU2 Offline LBIST Status Register (STCU_LBS)

The STCU_LBS register includes the results corresponding to the execution of the selected offline LBIST. The size of the register depends on the number of LBISTs.

Note: On this device, the LBISTs are 7.

Offset: 0x002C Access: User read

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	LBS7	LBS6	LBS5	LBS4	LBS3	LBS2	LBS1	LBS0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 2376. STCU2 Offline LBIST Status Register (STCU_LBS)

Table 2263. STCU_LBS field descriptions

Field	Description
0:31 LBS[7:0]	Offline status of the selected LBIST 0 Failed LBIST execution 1 Successful LBIST execution

90.5.13 STCU2 Offline LBIST End Flag Register (STCU_LBE)

The STCU_LBE register includes the end flag respective to the execution of the selected offline LBIST. The size of the register depends on the number of LBISTs.

Note: On this device, the LBISTs are 7.

Offset: 0x0030 Access: User read

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	LBE7	LBE6	LBE5	LBE4	LBE3	LBE2	LBE1	LBE0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 2377. STCU2 Offline LBIST End Flag Register (STCU_LBE)

Table 2264. STCU_LBE field descriptions

Field	Description
0:31 LBE[7:0]	Offline End status of the selected LBIST 0 LBIST execution not yet completed 1 LBIST execution finished

90.5.14 STCU2 Online LBIST Status Register (STCU_LBSSW)

The STCU_LBSSW register includes the results corresponding to the execution of the selected online LBIST. The size of the register depends on the number of LBISTs. The content of this register is initialized to its reset value as soon as STCU_RUNSW[RUNSW] = 1.

Note: On this device, the LBISTs are 7.

Offset: 0x0034 Access: User read

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	LBSSW7	LBSSW6	LBSSW5	LBSSW4	LBSSW3	LBSSW2	LBSSW1	LBSSW0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 2378. STCU2 Online LBIST Status Register (STCU_LBSSW)

Table 2265. STCU_LBSSW field descriptions

Field	Description
24:31 LBSSW[7:0]	LBSSW*x*: online status of the selected LBIST 0 Failed LBIST execution 1 Successful LBIST execution

90.5.15 STCU2 Online LBIST End Flag Register (STCU_LBESW)

The STCU_LBESW register includes the end flag respective to the execution of the selected online LBIST. The size of the register depends on the number of LBISTs. The content of this register is initialized to its reset value as soon as STCU_RUNSW[RUNSW] = 1.

Note: On this device, the LBISTs are 7.

Offset: 0x0038 Access: User read

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	LBESW7	LBESW6	LBESW5	LBESW4	LBESW3	LBESW2	LBESW1	LBESW0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 2379. STCU2 Online LBIST End Flag Register (STCU_LBESW)

Table 2266. STCU_LBESW field descriptions

Field	Description
16:31 LBESW[7:0]	LBESW*x*: online LBIST End status. 0 LBIST execution not yet completed 1 LBIST execution finished

90.5.16 STCU2 Online LBIST Reset Management (STCU_LBRMSW)

The STCU_LBRMSW register defines the online reset mapping for each LBIST. Global or specific functional resets can be programmed depending on the online Self Test requirements and top level integration architecture.

The size of the register depends on the number of LBISTs.

The R/W fields in this register are readable at any time. You can write to these fields when either of the following conditions is true:

- Offline self-test phase is active.
- Online self-test phase is active and STCU_CFG[WRP] = 0.

Offset: 0x003C Access: User read/write

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

Reset: The reset value is chip-specific; see the chapter that describes how modules are configured and connected.

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	LBRMSW7	LBRMSW6	LBRMSW5	LBRMSW4	LBRMSW3	LBRMSW2	LBRMSW1	LBRMSW0
W																

Reset: The reset value is chip-specific; see the chapter that describes how modules are configured and connected.

Figure 2380. STCU2 Online LBIST Reset Management (STCU_LBRMSW)

Table 2267. STCU_LBRMSW field descriptions

Field	Description
16:31 LBRMSW[7:0]	Online LBIST Reset Management. **Note:** If one of the selected LBIST has this bit set to '1', then only the Global functional reset will be pulsed. 0 Dedicated functional reset is pulsed at the end of LBIST run 1 Global functional reset is pulsed at the end of LBIST run

90.5.17 STCU2 LBIST Unrecoverable FM Register (STCU_LBUFM)

The STCU_LBUFM register defines the fault mapping of each LBIST in terms of unrecoverable or recoverable fault. The size of the register depends on the number of LBISTs.

Note: On this device, the LBISTs are 7.

The R/W fields in this register are readable at any time. You can write to these fields when either of the following conditions is true:

- Offline self-test phase is active.
- Online self-test phase is active and STCU_CFG[WRP] = 0.

Offset: 0x0040 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	The reset value is chip-specific; see the chapter that describes how modules are configured and connected.															

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	LBUFM7	LBUFM6	LBUFM5	LBUFM4	LBUFM3	LBUFM2	LBUFM1	LBUFM0
W																
Reset	The reset value is chip-specific; see the chapter that describes how modules are configured and connected.															

Figure 2381. STCU2 LBIST Unrecoverable FM Register (STCU_LBUFM)

Table 2268. STCU_LBUFM field descriptions

Field	Description
16:31 LBUFM[7:0]	LBIST Unrecoverable Fault Mapping 0 Recoverable Fault mapping 1 Unrecoverable Fault mapping

90.5.18 STCU2 Offline MBIST 1 Status Register (STCU_MBS1)

The STCU_MBS1 register includes the results corresponding to the execution of the selected offline MBIST in the range NMCUT = 0 to 31. The size of the register depends on the amount of RAM or ROM subject to BIST.

Note: Refer to Section 7.10.6: STCU2 configuration in Chapter 7: Device configuration for the number of NMCUT available in this device.

Offset: 0x0044 Access: User read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	MBS31(1)	MBS30(1)	MBS29(1)	MBS28(1)	MBS27(1)	MBS26(1)	MBS25(1)	MBS24(1)	MBS23(1)	MBS22(1)	MBS21(1)	MBS20(1)	MBS19(1)	MBS18(1)	MBS17(1)	MBS16(1)
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	MBS15(1)	MBS14(1)	MBS13(1)	MBS12(1)	MBS11(1)	MBS10(1)	MBS9(1)	MBS8(1)	MBS7(1)	MBS6(1)	MBS5(1)	MBS4(1)	MBS3(1)	MBS2(1)	MBS1(1)	MBS0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 2382. STCU2 Offline MBIST 1 Status Register (STCU_MBS1)

1. The availability of this field is based on the number of NMCUT available in this device. For details, refer to *Table 114: MBIST registers in STCU2 with available fields* in *Chapter 7: Device configuration*.

Table 2269. STCU_MBS1 field descriptions

Field	Description
0:31 MBS[31:0]	Offline status (NMCUT range = 0 to 31) of the selected MBIST **Note:** This bit is meaningful when the respective bit of the STCU_MBE1 register reports the MBIST is finished. 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution

90.5.19 STCU2 Offline MBIST 2 Status Register (STCU_MBS2)

The STCU_MBS2 register includes the results corresponding to the execution of the selected offline MBIST in the range NMCUT = 32 to 63. The size of the register depends on the amount of RAM or ROM subject to BIST.

Note: Refer to Section 7.10.6: STCU2 configuration in Chapter 7: Device configurationfor the number of NMCUT available in this device.

Offset: 0x0048 Access: User read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	MBS63(1)	MBS62(1)	MBS61(1)	MBS60(1)	MBS59(1)	MBS58(1)	MBS57(1)	MBS56(1)	MBS55(1)	MBS54(1)	MBS53(1)	MBS52(1)	MBS51(1)	MBS50(1)	MBS49(1)	MBS48(1)
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	MBS47(1)	MBS46(1)	MBS45(1)	MBS44(1)	MBS43(1)	MBS42(1)	MBS41(1)	MBS40(1)	MBS39(1)	MBS38(1)	MBS37(1)	MBS36(1)	MBS35(1)	MBS34(1)	MBS33(1)	MBS32
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 2383. STCU2 Offline MBIST 2 Status Register (STCU_MBS2)

1. The availability of this field is based on the number of NMCUT available in this device. For details, refer to *Table 114: MBIST registers in STCU2 with available fields* in *Chapter 7: Device configuration*.

Table 2270. STCU_MBS2 field descriptions

Field	Description
0:31 MBS[63:32]	MBS*x*: Offline status (NMCUT range = 32 to 63) of the selected MBIST **Note:** This bit is meaningful when the respective bit of the STCU_MBE2 register reports the MBIST is finished). 0 Failed NMCUT BIST execution 1 No fault detected during the NMCUT BIST execution

90.5.20 STCU2 Offline MBIST 3 Status Register (STCU_MBS3)

The STCU_MBS3 register includes the results corresponding to the execution of the selected offline MBIST in the range NMCUT = 64 to 95. The size of the register depends on the amount of RAM or ROM subject to BIST.

Note: Refer to Section 7.10.6: STCU2 configuration in Chapter 7: Device configurationfor the number of NMCUT available in this device.

Offset: 0x004C Access: User read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	MBS95(1)	MBS94(1)	MBS93(1)	MBS92(1)	MBS91(1)	MBS90(1)	MBS89(1)	MBS88(1)	MBS87(1)	MBS86(1)	MBS85(1)	MBS84(1)	MBS83(1)	MBS82(1)	MBS81(1)	MBS80(1)
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	MBS79(1)	MBS78(1)	MBS77(1)	MBS76(1)	MBS75(1)	MBS74(1)	MBS73(1)	MBS72(1)	MBS71(1)	MBS70(1)	MBS69(1)	MBS68(1)	MBS67(1)	MBS66(1)	MBS65(1)	MBS64
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 2384. STCU2 Offline MBIST 3 Status Register (STCU_MBS3)

1. The availability of this field is based on the number of NMCUT available in this device. For details, refer to *Table 114: MBIST registers in STCU2 with available fields* in *Chapter 7: Device configuration*.

Table 2271. STCU_MBS3 field descriptions

Field	Description
0:31 MBS[95:64]	Offline status (NMCUT range = 64 to 95) of the selected MBIST. **Note:** This bit is meaningful when the respective bit of the STCU_MBE3 register reports the MBIST is finished). 0 Failed NMCUT BIST execution 1 No fault detected during the NMCUT BIST execution

90.5.21 STCU2 Offline MBIST 4 Status Register (STCU_MBS4)

The STCU_MBS4 register includes the results corresponding to the execution of the selected offline MBIST in the range NMCUT = 96 to 127. The size of the register depends on the amount of RAM or ROM subject to BIST.

Note: On this device, the selected offline MBIST is in the range 96 to 101.

Refer to Section 7.10.6: STCU2 configuration in Chapter 7: Device configuration for the number of NMCUT available in this device.

Offset: 0x0050 Access: User read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	MBS127(1)	MBS126(1)	MBS125(1)	MBS124(1)	MBS123(1)	MBS122(1)	MBS121(1)	MBS120(1)	MBS119(1)	MBS118(1)	MBS117(1)	MBS116(1)	MBS115(1)	MBS114(1)	MBS113(1)	MBS112(1)
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	MBS111(1)	MBS110(1)	MBS109(1)	MBS108(1)	MBS107(1)	MBS106(1)	MBS105(1)	MBS104(1)	MBS103(1)	MBS102(1)	MBS101(1)	MBS100(1)	MBS99(1)	MBS98(1)	MBS97(1)	MBS96
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 2385. STCU2 Offline MBIST 4 Status Register (STCU_MBS4)

1. The availability of this field is based on the number of NMCUT available in this device. For details, refer to *Table 114: MBIST registers in STCU2 with available fields* in *Chapter 7: Device configuration*.

Table 2272. STCU_MBS4 field descriptions

Field	Description
0:31 MBS[127:96]	Offline status (NMCUT range = 96 to 127) of the selected MBIST. **Note:** This bit is meaningful when the respective bit of the STCU_MBE4 register reports the MBIST is finished). 0 Failed NMCUT BIST execution 1 No fault detected during the NMCUT BIST execution

90.5.22 STCU2 Offline MBIST 1 End Flag Register (STCU_MBE1)

The STCU_MBE1 register includes the end flag respective to the execution of the selected offline MBIST in the range NMCUT = 0 to 31. The size of the register depends on the amount of RAM or ROM subject to BIST.

Note: Refer to Section 7.10.6: STCU2 configuration in Chapter 7: Device configuration for the number of NMCUT available in this device.

Offset: 0x0054 Access: User read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	MBE31(1)	MBE30(1)	MBE29(1)	MBE28(1)	MBE27(1)	MBE26(1)	MBE25(1)	MBE24(1)	MBE23(1)	MBE22(1)	MBE21(1)	MBE20(1)	MBE19(1)	MBE18(1)	MBE17(1)	MBE16(1)
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	MBE15(1)	MBE14(1)	MBE13(1)	MBE12(1)	MBE11(1)	MBE10(1)	MBE9(1)	MBE8(1)	MBE7(1)	MBE6(1)	MBE5(1)	MBE4(1)	MBE3(1)	MBE2(1)	MBE1(1)	MBE0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 2386. STCU2 Offline MBIST 1 End Flag Register (STCU_MBE1)

1. The availability of this field is based on the number of NMCUT available in this device. For details, refer to *Table 114: MBIST registers in STCU2 with available fields* in *Chapter 7: Device configuration*.

Table 2273. STCU_MBE1 field descriptions

Field	Description
0:31 MBE[31:0]	Offline End status (0 to 31) of the selected MBIST. 0 MBIST execution still ongoing 1 MBIST execution finished

90.5.23 STCU2 Offline MBIST 2 End Flag Register (STCU_MBE2)

The STCU_MBE2 register includes the end flag respective to the execution of the selected offline MBIST in the range NMCUT = 32 to 63. The size of the register depends on the amount of RAM or ROM subject to BIST.

Note: Refer to Section 7.10.6: STCU2 configuration in Chapter 7: Device configuration for the number of NMCUT available in this device.

Offset: 0x0058 Access: User read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	MBE63(1)	MBE62(1)	MBE61(1)	MBE60(1)	MBE59(1)	MBE58(1)	MBE57(1)	MBE56(1)	MBE55(1)	MBE54(1)	MBE53(1)	MBE52(1)	MBE51(1)	MBE50(1)	MBE49(1)	MBE48(1)
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	MBE47(1)	MBE46(1)	MBE45(1)	MBE44(1)	MBE43(1)	MBE42(1)	MBE41(1)	MBE40(1)	MBE39(1)	MBE38(1)	MBE37(1)	MBE36(1)	MBE35(1)	MBE34(1)	MBE33(1)	MBE32
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 2387. STCU2 Offline MBIST 2 End Flag Register (STCU_MBE2)

1. The availability of this field is based on the number of NMCUT available in this device. For details, refer to *Table 114: MBIST registers in STCU2 with available fields* in *Chapter 7: Device configuration*.

Table 2274. STCU_MBE2 field descriptions

Field	Description
0:31 MBE[63:32]	Offline End status (32 to 63) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished

90.5.24 STCU2 Offline MBIST 3 End Flag Register (STCU_MBE3)

The STCU_MBE3 register includes the end flag respective to the execution of the selected offline MBIST in the range NMCUT = 64 to 95. The size of the register depends on the amount of RAM or ROM subject to BIST.

Note: Refer to Section 7.10.6: STCU2 configuration in Chapter 7: Device configuration for the number of NMCUT available in this device.

Offset: 0x005C Access: User read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	MBE95(1)	MBE94(1)	MBE93(1)	MBE92(1)	MBE91(1)	MBE90(1)	MBE89(1)	MBE88(1)	MBE87(1)	MBE86(1)	MBE85(1)	MBE84(1)	MBE83(1)	MBE82(1)	MBE81(1)	MBE80(1)
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	MBE79(1)	MBE78(1)	MBE77(1)	MBE76(1)	MBE75(1)	MBE74(1)	MBE73(1)	MBE72(1)	MBE71(1)	MBE70(1)	MBE69(1)	MBE68(1)	MBE67(1)	MBE66(1)	MBE65(1)	MBE64
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 2388. STCU2 Offline MBIST 3 End Flag Register (STCU_MBE3)

1. The availability of this field is based on the number of NMCUT available in this device. For details, refer to *Table 114: MBIST registers in STCU2 with available fields* in *Chapter 7: Device configuration*.

Table 2275. STCU_MBE3 field descriptions

Field	Description
0:31 MBE[95:64]	MBE*x*: offline End status (64 to 95) of the **selected** MBIST 0 MBIST execution still ongoing 1 MBIST execution finished

90.5.25 STCU2 Offline MBIST 4 End Flag Register (STCU_MBE4)

The STCU_MBE4 register includes the end flag respective to the execution of the selected offline MBIST in the range NMCUT = 96 to 127. The size of the register depends on the amount of RAM or ROM subject to BIST.

Note: On this device, the selected offline MBIST is in the range 96 to 101.

Note: Refer to Section 7.10.6: STCU2 configuration in Chapter 7: Device configuration for the number of NMCUT available in this device.

Offset: 0x0060 Access: User read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	MBE127(1)	MBE126(1)	MBE125(1)	MBE124(1)	MBE123(1)	MBE122(1)	MBE121(1)	MBE120(1)	MBE119(1)	MBE118(1)	MBE117(1)	MBE116(1)	MBE115(1)	MBE114(1)	MBE113(1)	MBE112(1)
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	MBE111(1)	MBE110(1)	MBE109(1)	MBE108(1)	MBE107(1)	MBE106(1)	MBE105(1)	MBE104(1)	MBE103(1)	MBE102(1)	MBE101(1)	MBE100(1)	MBE99(1)	MBE98(1)	MBE97(1)	MBE96
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 2389. STCU2 Offline MBIST 4 End Flag Register (STCU_MBE4)

1. The availability of this field is based on the number of NMCUT available in this device. For details, refer to *Table 114: MBIST registers in STCU2 with available fields* in *Chapter 7: Device configuration*.

Table 2276. STCU_MBE4 field descriptions

Field	Description
0:31 MBE[127:96]	Offline End status (96 to 127) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished

90.5.26 STCU2 Online MBIST 1 Status Register (STCU_MBS1SW)

The STCU_MBS1SW register includes the results corresponding to the execution of the selected online MBIST in the range NMCUT = 0 to 31. The size of the register depends on the amount of RAM or ROM subject to BIST. The content of this register is initialized to its reset value as soon as STCU_RUNSW[RUNSW] = 1.

Note: Refer to Section 7.10.6: STCU2 configuration in Chapter 7: Device configuration for the number of NMCUT available in this device.

Offset: 0x0064 Access: User read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	MBSSW31(1)	MBSSW30(1)	MBSSW29(1)	MBSSW28(1)	MBSSW27(1)	MBSSW26(1)	MBSSW25(1)	MBSSW24(1)	MBSSW23(1)	MBSSW22(1)	MBSSW21(1)	MBSSW20(1)	MBSSW19(1)	MBSSW18(1)	MBSSW17(1)	MBSSW16(1)
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	MBSSW15(1)	MBSSW14(1)	MBSSW13(1)	MBSSW12(1)	MBSSW11(1)	MBSSW10(1)	MBSSW9(1)	MBSSW8(1)	MBSSW7(1)	MBSSW6(1)	MBSSW5(1)	MBSSW4(1)	MBSSW3(1)	MBSSW2(1)	MBSSW1(1)	MBSSW0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 2390. STCU2 Online MBIST 1 Status Register (STCU_MBS1SW)

1. The availability of this field is based on the number of NMCUT available in this device. For details, refer to *Table 114: MBIST registers in STCU2 with available fields* in *Chapter 7: Device configuration*.

Table 2277. STCU_MBS1SW field descriptions

Field	Description
0:31 MBSSW[31:0]	Online status (NMCUT range = 0 to 31) of the selected MBIST 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution

90.5.27 STCU2 Online MBIST 2 Status Register (STCU_MBS2SW)

The STCU_MBS2SW register includes the results corresponding to the execution of the selected online MBIST in the range NMCUT = 32 to 63. The size of the register depends on the amount of RAM or ROM subject to BIST. The content of this register is initialized to its reset value as soon as STCU_RUNSW[RUNSW] = 1.

Note: Refer to Section 7.10.6: STCU2 configuration in Chapter 7: Device configuration for the number of NMCUT available in this device.

Offset: 0x0068 Access: User read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	MBSSW63(1)	MBSSW62(1)	MBSSW61(1)	MBSSW60(1)	MBSSW59(1)	MBSSW58(1)	MBSSW57(1)	MBSSW56(1)	MBSSW55(1)	MBSSW54(1)	MBSSW53(1)	MBSSW52(1)	MBSSW51(1)	MBSSW50(1)	MBSSW49(1)	MBSSW48(1)
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	MBSSW47(1)	MBSSW46(1)	MBSSW45(1)	MBSSW44(1)	MBSSW43(1)	MBSSW42(1)	MBSSW41(1)	MBSSW40(1)	MBSSW39(1)	MBSSW38(1)	MBSSW37(1)	MBSSW36(1)	MBSSW35(1)	MBSSW34(1)	MBSSW33(1)	MBSSW32
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 2391. STCU2 Online MBIST 2 Status Register (STCU_MBS2SW)

1. The availability of this field is based on the number of NMCUT available in this device. For details, refer to *Table 114: MBIST registers in STCU2 with available fields* in *Chapter 7: Device configuration*.

Table 2278. STCU_MBS2SW field descriptions

Field	Description
0:31 MBSSW[63:32]	MBSSW*x*: online status (NMCUT range = 32 to 63) of the selected MBIST. 0 Failed NMCUT BIST execution 1 No fault detected during the NMCUT BIST execution

90.5.28 STCU2 Online MBIST 3 Status Register (STCU_MBS3SW)

The STCU_MBS3SW register includes the results corresponding to the execution of the selected online MBIST in the range NMCUT = 64 to 95. The size of the register depends on the amount of RAM or ROM subject to BIST. The content of this register is initialized to its reset value as soon as STCU_RUNSW[RUNSW] = 1.

Note: Refer to Section 7.10.6: STCU2 configuration in Chapter 7: Device configuration for the number of NMCUT available in this device.

Offset: 0x006C Access: User read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	MBSSW95(1)	MBSSW94(1)	MBSSW93(1)	MBSSW92(1)	MBSSW91(1)	MBSSW90(1)	MBSSW89(1)	MBSSW88(1)	MBSSW87(1)	MBSSW86(1)	MBSSW85(1)	MBSSW84(1)	MBSSW83(1)	MBSSW82(1)	MBSSW81(1)	MBSSW80(1)
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	MBSSW79(1)	MBSSW78(1)	MBSSW77(1)	MBSSW76(1)	MBSSW75(1)	MBSSW74(1)	MBSSW73(1)	MBSSW72(1)	MBSSW71(1)	MBSSW70(1)	MBSSW69(1)	MBSSW68(1)	MBSSW67(1)	MBSSW66(1)	MBSSW65(1)	MBSSW64
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 2392. STCU2 Online MBIST 3 Status Register (STCU_MBS3SW)

1. The availability of this field is based on the number of NMCUT available in this device. For details, refer to *Table 114: MBIST registers in STCU2 with available fields* in *Chapter 7: Device configuration*.

Table 2279. STCU_MBS3SW field descriptions

Field	Description
0:31 MBSSW[95:64]	MBSSW*x*: online status (NMCUT range = 64 to 95) of the selected MBIST. 0 Failed NMCUT BIST execution 1 No fault detected during the NMCUT BIST execution

90.5.29 STCU2 Online MBIST 4 Status Register (STCU_MBS4SW)

The STCU_MBS4SW register includes the results corresponding to the execution of the selected online MBIST in the range NMCUT = 96 to 127. The size of the register depends on the amount of RAM or ROM subject to BIST. The content of this register is initialized to its reset value as soon as STCU_RUNSW[RUNSW] = 1.

Note: On this device, the selected offline MBIST is in the range 96 to 101.

Note: Refer to Section 7.10.6: STCU2 configuration in Chapter 7: Device configuration for the number of NMCUT available in this device.

Offset: 0x0070 Access: User read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	MBSSW127(1)	MBSSW126(1)	MBSSW125(1)	MBSSW124(1)	MBSSW123(1)	MBSSW122(1)	MBSSW121(1)	MBSSW120(1)	MBSSW119(1)	MBSSW118(1)	MBSSW117(1)	MBSSW116(1)	MBSSW115(1)	MBSSW114(1)	MBSSW113(1)	MBSSW112(1)
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	MBSSW111(1)	MBSSW110(1)	MBSSW109(1)	MBSSW108(1)	MBSSW107(1)	MBSSW106(1)	MBSSW105(1)	MBSSW104(1)	MBSSW103(1)	MBSSW102(1)	MBSSW101(1)	MBSSW100(1)	MBSSW99(1)	MBSSW98(1)	MBSSW97(1)	MBSSW96
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 2393. STCU2 Online MBIST 4 Status Register (STCU_MBS4SW)

1. The availability of this field is based on the number of NMCUT available in this device. For details, refer to *Table 114: MBIST registers in STCU2 with available fields* in *Chapter 7: Device configuration*.

Table 2280. STCU_MBS4SW field descriptions

Field	Description
0:31 MBSSW[127:96]	MBSSW*x*: online status (NMCUT range = 96 to 127) of the selected MBIST. 0 Failed NMCUT BIST execution 1 No fault detected during the NMCUT BIST execution

90.5.30 STCU2 Online MBIST 1 End Flag Register (STCU_MBE1SW)

The STCU_MBE1SW register includes the end flag respective to the execution of the selected online MBIST in the range NMCUT = 0 to 31. The size of the register depends on the amount of RAM or ROM subject to BIST. The content of this register is initialized to its reset value as soon as STCU_RUNSW[RUNSW] = 1.

Note: Refer to Section 7.10.6: STCU2 configuration in Chapter 7: Device configuration for the number of NMCUT available in this device.

Offset: 0x0074 Access: User read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	MBESW31(1)	MBESW30(1)	MBESW29(1)	MBESW28(1)	MBESW27(1)	MBESW26(1)	MBESW25(1)	MBESW24(1)	MBESW23(1)	MBESW22(1)	MBESW21(1)	MBESW20(1)	MBESW19(1)	MBESW18(1)	MBESW17(1)	MBESW16(1)
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	MBESW15(1)	MBESW14(1)	MBESW13(1)	MBESW12(1)	MBESW11(1)	MBESW10(1)	MBESW9(1)	MBESW8(1)	MBESW7(1)	MBESW6(1)	MBESW5(1)	MBESW4(1)	MBESW3(1)	MBESW2(1)	MBESW1(1)	MBESW0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 2394. STCU2 Online MBIST 1 End Flag Register (STCU_MBE1SW)

1. The availability of this field is based on the number of NMCUT available in this device. For details, refer to *Table 114: MBIST registers in STCU2 with available fields* in *Chapter 7: Device configuration*.

Table 2281. STCU_MBE1SW field descriptions

Field	Description
0:31 MBESW[31:0]	MBESW*x*: online End status (0 to 31) of the selected MBIST. 0 MBIST execution still ongoing 1 MBIST execution finished

90.5.31 STCU2 Online MBIST 2 End Flag Register (STCU_MBE2SW)

The STCU_MBE2SW register includes the end flag respective to the execution of the selected online MBIST in the range NMCUT = 32 to 63. The size of the register depends on the amount of RAM or ROM subject to BIST. The content of this register is initialized to its reset value as soon as STCU_RUNSW[RUNSW] = 1.

Note: Refer to Section 7.10.6: STCU2 configuration in Chapter 7: Device configuration for the number of NMCUT available in this device.

Offset: 0x0078 Access: User read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	MBESW63(1)	MBESW62(1)	MBESW61(1)	MBESW60(1)	MBESW59(1)	MBESW58(1)	MBESW57(1)	MBESW56(1)	MBESW55(1)	MBESW54(1)	MBESW53(1)	MBESW52(1)	MBESW51(1)	MBESW50(1)	MBESW49(1)	MBESW48(1)
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	MBESW47(1)	MBESW46(1)	MBESW45(1)	MBESW44(1)	MBESW43(1)	MBESW42(1)	MBESW41(1)	MBESW40(1)	MBESW39(1)	MBESW38(1)	MBESW37(1)	MBESW36(1)	MBESW35(1)	MBESW34(1)	MBESW33(1)	MBESW32
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 2395. STCU2 Online MBIST 2 End Flag Register (STCU_MBE2SW)

1. The availability of this field is based on the number of NMCUT available in this device. For details, refer to *Table 114: MBIST registers in STCU2 with available fields* in *Chapter 7: Device configuration*.

Table 2282. STCU_MBE2SW field descriptions

Field	Description
0:31 MBESW[63:32]	Online End status (32 to 63) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished

90.5.32 STCU2 Online MBIST 3 End Flag Register (STCU_MBE3SW)

The STCU_MBE3SW register includes the end flag respective to the execution of the selected online MBIST in the range NMCUT = 64 to 95. The size of the register depends on the amount of RAM or ROM subject to BIST. The content of this register is initialized to its reset value as soon as STCU_RUNSW[RUNSW] = 1.

Note: Refer to Section 7.10.6: STCU2 configuration in Chapter 7: Device configuration for the number of NMCUT available in this device.

Offset: 0x007C Access: User read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	MBESW95(1)	MBESW94(1)	MBESW93(1)	MBESW92(1)	MBESW91(1)	MBESW90(1)	MBESW89(1)	MBESW88(1)	MBESW87(1)	MBESW86(1)	MBESW85(1)	MBESW84(1)	MBESW83(1)	MBESW82(1)	MBESW81(1)	MBESW80(1)
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	MBESW79(1)	MBESW78(1)	MBESW77(1)	MBESW76(1)	MBESW75(1)	MBESW74(1)	MBESW73(1)	MBESW72(1)	MBESW71(1)	MBESW70(1)	MBESW69(1)	MBESW68(1)	MBESW67(1)	MBESW66(1)	MBESW65(1)	MBESW64
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 2396. STCU2 Online MBIST 3 End Flag Register (STCU_MBE3SW)

1. The availability of this field is based on the number of NMCUT available in this device. For details, refer to *Table 114: MBIST registers in STCU2 with available fields* in *Chapter 7: Device configuration*.

Table 2283. STCU_MBE3SW field descriptions

Field	Description
0:31 MBESW[95:64]	MBESW*x*: online End status (64 to 95) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished

90.5.33 STCU2 Online MBIST 4 End Flag Register (STCU_MBE4SW)

The STCU_MBE4SW register includes the end flag respective to the execution of the selected online MBIST in the range NMCUT = 96 to 127. The size of the register depends on the amount of RAM or ROM subject to BIST. The content of this register is initialized to its reset value as soon as STCU_RUNSW[RUNSW] = 1.

Note: On this device, the selected offline MBIST is in the range 96 to 101.

Refer to Section 7.10.6: STCU2 configuration in Chapter 7: Device configuration for the number of NMCUT available in this device.

Offset: 0x0080 Access: User read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	MBESW127(1)	MBESW126(1)	MBESW125(1)	MBESW124(1)	MBESW123(1)	MBESW122(1)	MBESW121(1)	MBESW120(1)	MBESW119(1)	MBESW118(1)	MBESW117(1)	MBESW116(1)	MBESW115(1)	MBESW114(1)	MBESW113(1)	MBESW112(1)
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	MBESW111(1)	MBESW110(1)	MBESW109(1)	MBESW108(1)	MBESW107(1)	MBESW106(1)	MBESW105(1)	MBESW104(1)	MBESW103(1)	MBESW102(1)	MBESW101(1)	MBESW100(1)	MBESW99(1)	MBESW98(1)	MBESW97(1)	MBESW96
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 2397. STCU2 Online MBIST 4 End Flag Register (STCU_MBE4SW)

1. The availability of this field is based on the number of NMCUT available in this device. For details, refer to *Table 114: MBIST registers in STCU2 with available fields* in *Chapter 7: Device configuration*.

Table 2284. STCU_MBE4SW field descriptions

Field	Description
0:31 MBESW[127:96]	MBESWx: online End status (96 to 127) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished

90.5.34 STCU2 MBIST 1 Unrecoverable FM Register (STCU_MBUFM1)

The STCU_MBUFM1 register defines the fault mapping, in terms of unrecoverable or recoverable fault, of the MBIST in the range NMCUT = 0 to 31. The size of the register depends on the amount of RAM or ROM subject to BIST.

Note: Refer to Section 7.10.6: STCU2 configuration in Chapter 7: Device configuration for the number of NMCUT available in this device.

The R/W fields in this register are readable at any time. You can write to these fields when either of the following conditions is true:

- Offline self-test phase is active.
- Online self-test phase is active and STCU_CFG[WRP] = 0.

Offset: 0x0084 Access: User read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	MBUFM31(1)	MBUFM30(1)	MBUFM29(1)	MBUFM28(1)	MBUFM27(1)	MBUFM26(1)	MBUFM25(1)	MBUFM24(1)	MBUFM23(1)	MBUFM22(1)	MBUFM21(1)	MBUFM20(1)	MBUFM19(1)	MBUFM18(1)	MBUFM17(1)	MBUFM16(1)
W																
Reset	The reset value is chip-specific; see the chapter that describes how modules are configured and connected.															

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	MBUFM15(1)	MBUFM14(1)	MBUFM13(1)	MBUFM12(1)	MBUFM11(1)	MBUFM10(1)	MBUFM9(1)	MBUFM8(1)	MBUFM7(1)	MBUFM6(1)	MBUFM5(1)	MBUFM4(1)	MBUFM3(1)	MBUFM2(1)	MBUFM1(1)	MBUFM0
W																
Reset	The reset value is chip-specific; see the chapter that describes how modules are configured and connected.															

Figure 2398. STCU2 MBIST 1 Unrecoverable FM Register (STCU_MBUFM1)

1. The availability of this field is based on the number of NMCUT available in this device. For details, refer to *Table 114: MBIST registers in STCU2 with available fields* in *Chapter 7: Device configuration*.

Table 2285. STCU_MBUFM1 field descriptions

Field	Description
0:31 MBUFM[31:0]	MBIST Unrecoverable Fault Mapping. 0 Recoverable fault mapping 1 Unrecoverable fault mapping

90.5.35 STCU2 MBIST 2 Unrecoverable FM Register (STCU_MBUFM2)

The STCU_MBUFM2 register defines the fault mapping, in terms of unrecoverable or recoverable fault, of the MBIST in the range NMCUT = 32 to 63. The size of the register depends on the amount of RAM or ROM subject to BIST.

Note: Refer to Section 7.10.6: STCU2 configuration in Chapter 7: Device configuration for the number of NMCUT available in this device.

The R/W fields in this register are readable at any time. You can write to these fields when either of the following conditions is true:

- Offline self-test phase is active.
- Online self-test phase is active and STCU_CFG[WRP] = 0.

Offset: 0x0088 Access: User read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	MBUFM63(1)	MBUFM62(1)	MBUFM61(1)	MBUFM60(1)	MBUFM59(1)	MBUFM58(1)	MBUFM57(1)	MBUFM56(1)	MBUFM55(1)	MBUFM54(1)	MBUFM53(1)	MBUFM52(1)	MBUFM51(1)	MBUFM50(1)	MBUFM49(1)	MBUFM48(1)
W																
Reset	The reset value is chip-specific; see the chapter that describes how modules are configured and connected.															

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	MBUFM47(1)	MBUFM46(1)	MBUFM45(1)	MBUFM44(1)	MBUFM43(1)	MBUFM42(1)	MBUFM41(1)	MBUFM40(1)	MBUFM39(1)	MBUFM38(1)	MBUFM37(1)	MBUFM36(1)	MBUFM35(1)	MBUFM34(1)	MBUFM33(1)	MBUFM32
W																
Reset	The reset value is chip-specific; see the chapter that describes how modules are configured and connected.															

Figure 2399. STCU2 MBIST 2 Unrecoverable FM Register (STCU_MBUFM2)

1. The availability of this field is based on the number of NMCUT available in this device. For details, refer to *Table 114: MBIST registers in STCU2 with available fields* in *Chapter 7: Device configuration*.

Table 2286. STCU_MBUFM2 field descriptions

Field	Description
0:31 MBUFM[63:32]	MBIST Unrecoverable Fault Mapping 0 Recoverable fault mapping 1 Unrecoverable fault mapping

90.5.36 STCU2 MBIST 3 Unrecoverable FM Register (STCU_MBUFM3)

The STCU_MBUFM3 register defines the fault mapping, in terms of unrecoverable or recoverable fault, of the MBIST in the range NMCUT = 64 to 95. The size of the register depends on the amount of RAM or ROM subject to BIST.

Note: Refer to Section 7.10.6: STCU2 configuration in Chapter 7: Device configuration for the number of NMCUT available in this device.

The R/W fields in this register are readable at any time. You can write to these fields when either of the following conditions is true:

- Offline self-test phase is active.
- Online self-test phase is active and STCU_CFG[WRP] = 0.

Offset: 0x008C Access: User read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	MBUFM95(1)	MBUFM94(1)	MBUFM93(1)	MBUFM92(1)	MBUFM91(1)	MBUFM90(1)	MBUFM89(1)	MBUFM88(1)	MBUFM87(1)	MBUFM86(1)	MBUFM85(1)	MBUFM84(1)	MBUFM83(1)	MBUFM82(1)	MBUFM81(1)	MBUFM80(1)
W																
Reset	The reset value is chip-specific; see the chapter that describes how modules are configured and connected.															

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	MBUFM79(1)	MBUFM78(1)	MBUFM77(1)	MBUFM76(1)	MBUFM75(1)	MBUFM74(1)	MBUFM73(1)	MBUFM72(1)	MBUFM71(1)	MBUFM70(1)	MBUFM69(1)	MBUFM68(1)	MBUFM67(1)	MBUFM66(1)	MBUFM65(1)	MBUFM64(1)
W																
Reset	The reset value is chip-specific; see the chapter that describes how modules are configured and connected.															

Figure 2400. STCU2 MBIST 3 Unrecoverable FM Register (STCU_MBUFM3)

1. The availability of this field is based on the number of NMCUT available in this device. For details, refer to *Table 114: MBIST registers in STCU2 with available fields* in *Chapter 7: Device configuration*.

Table 2287. STCU_MBUFM3 field descriptions

Field	Description
0:31 MBUFM[95:64]	MBIST Unrecoverable Fault Mapping 0 Recoverable fault mapping 1 Unrecoverable fault mapping

90.5.37 STCU2 MBIST 4 Unrecoverable FM Register (STCU_MBUFM4)

The STCU_MBUFM4 register defines the fault mapping, in terms of unrecoverable or recoverable fault, of the MBIST in the range NMCUT = 96 to 127. The size of the register depends on the amount of RAM or ROM subject to BIST.

Note: On this device, the selected offline MBIST is in the range 96 to 101.

Refer to Section 7.10.6: STCU2 configuration in Chapter 7: Device configuration for the number of NMCUT available in this device.

The R/W fields in this register are readable at any time. You can write to these fields when either of the following conditions is true:

- Offline self-test phase is active.
- Online self-test phase is active and STCU_CFG[WRP] = 0.

Offset: 0x0090 Access: User read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	MBUFM127(1)	MBUFM126(1)	MBUFM125(1)	MBUFM124(1)	MBUFM123(1)	MBUFM122(1)	MBUFM121(1)	MBUFM120(1)	MBUFM119(1)	MBUFM118(1)	MBUFM117(1)	MBUFM116(1)	MBUFM115(1)	MBUFM114(1)	MBUFM113(1)	MBUFM112(1)
W																
Reset	The reset value is chip-specific; see the chapter that describes how modules are configured and connected.															

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	MBUFM111(1)	MBUFM110(1)	MBUFM109(1)	MBUFM108(1)	MBUFM107(1)	MBUFM106(1)	MBUFM105(1)	MBUFM104(1)	MBUFM103(1)	MBUFM102(1)	MBUFM101(1)	MBUFM100(1)	MBUFM99(1)	MBUFM98(1)	MBUFM97(1)	MBUFM96(1)
W																
Reset	The reset value is chip-specific; see the chapter that describes how modules are configured and connected.															

Figure 2401. STCU2 MBIST 4 Unrecoverable FM Register (STCU_MBUFM4)

1. The availability of this field is based on the number of NMCUT available in this device. For details, refer to *Table 114: MBIST registers in STCU2 with available fields* in *Chapter 7: Device configuration*.

Table 2288. STCU_MBUFM4 field descriptions

Field	Description
0:31 MBUFM[127:96]	MBIST Unrecoverable Fault Mapping 0 Recoverable fault mapping 1 Unrecoverable fault mapping

90.5.38 STCU2 LBIST Control *n* Register (STCU_LB_CTRL*n*)

The STCU_LB_CTRL register defines the control setting of each LBIST controller.

The R/W fields in this register are readable at any time. You can write to these fields when either of the following conditions is true:

- Offline self-test phase is active.
- Online self-test phase is active and STCU_CFG[WRP] = 0.

Offset: 0x0100 + *n* * 0x40 (*n* = 0 to 6) Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	PTR								CSM	0	0	0	PRPGEN	SHS		
W																
Reset	The reset value is chip-specific; see the chapter that describes how modules are configured and connected.															

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	SCEN_OFF				SCEN_ON				0	0	PFT	CWS				
W																
Reset	The reset value is chip-specific; see the chapter that describes how modules are configured and connected.															

Figure 2402. STCU2 LBIST Control *n* Register (STCU_LB_CTRL*n*)

Table 2289. STCU_LB_CTRL*n* field descriptions

Field	Description
0:7 PTR	Next LBIST or MBIST pointer PTR defines the logical pointer to the next LBIST or MBIST to be scheduled. The next LBIST or MBIST is scheduled concurrently to the current one if the CSM bit is set to 1, otherwise it is scheduled sequentially to the completion of the current LBIST execution. In case of NIL pointer, the CSM bit has to be set Sequential (0) to define this is the end of the list. The self-test procedure is stopped when the current LBIST has been completed. (0x0 to NLBIST - 1) pointer to LBIST (0x10 to 0x10 + MBIST - 1) pointer to MBIST 0xFF pointer to NIL. No BIST execution. Any other value is considered as invalid pointer and an error is set in the STCU_ERR register.
8 CSM	Concurrent/sequential mode 0 Sequential mode 1 Concurrent mode
12 PRPGEN	Enable PRPG Loading 0 Default LBIST value of the PRPG is used during LBIST run 1 LBIST PRPG value is initialized with the STCU_LB_PRPGL/H
13:15 SHS	Shift Speed 000 Shift at full rate (STCU core clock) 001 Shift at 1/2 rate (STCU core clock) 010 Shift at 1/3 rate (STCU core clock) 011 Shift at 1/4 rate (STCU core clock) 100 Shift at 1/5 rate (STCU core clock) 101 Shift at 1/6 rate (STCU core clock) 110 Shift at 1/7 rate (STCU core clock) 111 Shift at 1/8 rate (STCU core clock)

Table 2289. STCU_LB_CTRL*n* field descriptions (continued)

Field	Description
16:19 SCEN_OFF	Scan enable OFF SCEN_OFF defines the number of clock cycles OFF following the falling transition on the SCEN 0000 0 delay cycles 0001 1 delay cycles … 1111 15 delay cycles
20:23 SCEN_ON	Scan enable ON SCEN_ON defines the number of clock cycles OFF following the rising transition on the SCEN 0000 0 delay cycles 0001 1 delay cycles … 1111 15 delay cycles
26 PFT	Past Flush Test The LBIST controller, by default apply 32 flush test patterns. This bit allows to skip/apply the flush test pattern sequence. 0 Apply flush test patterns 1 Skip flush test patterns
27:31 CWS	Capture window size 00000 Controller waits 16 shift cycle for capture to finish. 00001 Controller waits 1 shift cycle for capture to finish. 00010 Controller waits 2 shift cycles for capture to finish. … 00111 Controller waits 7 shift cycles for capture to finish.

90.5.39 STCU2 LBIST PC Stop *n* Register (STCU_LB_PCS*n*)

The STCU_LB_PCS register defines the pattern counter stop of each LBIST controller.

The R/W fields in this register are readable at any time. You can write to these fields when either of the following conditions is true:

- Offline self-test phase is active.
- Online self-test phase is active and STCU_CFG[WRP] = 0.

Offset: 0x0104 + n * 0x40 (n = 0 to 6) Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	PCS									
W																
Reset	The reset value is chip-specific; see the chapter that describes how modules are configured and connected.															

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	PCS															
W																
Reset	The reset value is chip-specific; see the chapter that describes how modules are configured and connected.															

Figure 2403. STCU2 LBIST PC Stop *n* Register (STCU_LB_PCS*n*)

Table 2290. STCU_LB_PCS*n* field descriptions

Field	Description
6:31 PCS	Pattern counter stop PCS defines the pattern counter stop value.

90.5.40 STCU2 LBIST PRPG Low *n* Register (STCU_LB_PRPGL*n*)

The STCU_LB_PRPGL register defines the LSB part (31 to 0) of the PRPG start value of the LBIST controller. The size of the register depends on the number of PRPG bits of the respective LBIST.

The R/W fields in this register are readable at any time. You can write to these fields when either of the following conditions is true:

- Offline self-test phase is active.
- Online self-test phase is active and STCU_CFG[WRP] = 0.

Offset: 0x0108 + n * 0x40 (n = 0 to 6) Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	PRPG31	PRPG30	PRPG29	PRPG28	PRPG27	PRPG26	PRPG25	PRPG24	PRPG23	PRPG22	PRPG21	PRPG20	PRPG19	PRPG18	PRPG17	PRPG16
W																
Reset	The reset value is chip-specific; see the chapter that describes how modules are configured and connected.															

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	PRPG15	PRPG14	PRPG13	PRPG12	PRPG11	PRPG10	PRPG9	PRPG8	PRPG7	PRPG6	PRPG5	PRPG4	PRPG3	PRPG2	PRPG1	PRPG0
W																
Reset	The reset value is chip-specific; see the chapter that describes how modules are configured and connected.															

Figure 2404. STCU2 LBIST PRPG Low *n* Register (STCU_LB_PRPGL*n*)

Table 2291. STCU_LB_PRPGL*n* field descriptions

Field	Description
0:31 PRPG[31:0]	PRPG LBIST low bits This field defines the low part (31 to 0) of the LBIST PRPG start value.

90.5.41 STCU2 LBIST PRPG High *n* Register (STCU_LB_PRPGH*n*)

The STCU_LB_PRPGH register defines the MSB part (63 to 32) of the PRPG start value of the LBIST controller. The size of the register depends on the number of PRPG bits of the respective LBIST.

The R/W fields in this register are readable at any time. You can write to these fields when either of the following conditions is true:

- Offline self-test phase is active.
- Online self-test phase is active and STCU_CFG[WRP] = 0.

Offset: 0x010C + n * 0x40 (n = 0 to 6) Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	PRPG63	PRPG62	PRPG61	PRPG60	PRPG59	PRPG58	PRPG57	PRPG56	PRPG55	PRPG54	PRPG53	PRPG52	PRPG51	PRPG50	PRPG49	PRPG48
Reset	The reset value is chip-specific; see the chapter that describes how modules are configured and connected.															

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	PRPG47	PRPG46	PRPG45	PRPG44	PRPG43	PRPG42	PRPG41	PRPG40	PRPG39	PRPG38	PRPG37	PRPG36	PRPG35	PRPG34	PRPG33	PRPG32
Reset	The reset value is chip-specific; see the chapter that describes how modules are configured and connected.															

Figure 2405. STCU2 LBIST PRPG High *n* Register (STCU_LB_PRPGH*n*)

Table 2292. STCU_LB_PRPGH*n* field descriptions

Field	Description
0:31 PRPG[63:32]	PRPG LBIST High Bits This field is the high part (63 to 32) of the LBIST PRPG start value.

90.5.42 STCU2 Offline LBIST MISR Expected Low *n* Register (STCU_LB_MISREL*n*)

The STCU_LB_MISREL register defines the LSB part (31 to 0) of the expected MISR of the offline LBIST controller. The size of the register depends on the number of MISR bits of the respective LBIST.

The R/W fields in this register are readable at any time. However, you can write to these fields only when offline self-test phase is still active.

Offset: 0x0110 + n * 0x40 (n = 0 to 6) Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	MISRE31	MISRE30	MISRE29	MISRE28	MISRE27	MISRE26	MISRE25	MISRE24	MISRE23	MISRE22	MISRE21	MISRE20	MISRE19	MISRE18	MISRE17	MISRE16
W																
Reset	The reset value is chip-specific; see the chapter that describes how modules are configured and connected.															

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	MISRE15	MISRE14	MISRE13	MISRE12	MISRE11	MISRE10	MISRE9	MISRE8	MISRE7	MISRE6	MISRE5	MISRE4	MISRE3	MISRE2	MISRE1	MISRE0
W																
Reset	The reset value is chip-specific; see the chapter that describes how modules are configured and connected.															

Figure 2406. STCU2 Offline LBIST MISR Expected Low *n* Register (STCU_LB_MISREL*n*)

Table 2293. STCU_LB_MISREL*n* field descriptions

Field	Description
0:31 MISRE[31:0]	Offline MISR Expected low Bits. This field defines the low part (31 to 0) of the expected MISR.

90.5.43 STCU2 Offline LBIST MISR Expected High *n* Register (STCU_LB_MISREH*n*)

The STCU_LB_MISREH register defines the MSB part (63 to 32) of the expected MISR of the offline LBIST controller. The size of the register depends on the number of MISR bits of the respective LBIST.

The R/W fields in this register are readable at any time. However, you can write to these fields only when offline self-test phase is still active.

Offset: 0x0114 + n * 0x40 (n = 0 to 6) Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	MISRE63	MISRE62	MISRE61	MISRE60	MISRE59	MISRE58	MISRE57	MISRE56	MISRE55	MISRE54	MISRE53	MISRE52	MISRE51	MISRE50	MISRE49	MISRE48
W																
Reset	The reset value is chip-specific; see the chapter that describes how modules are configured and connected.															

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	MISRE47	MISRE46	MISRE45	MISRE44	MISRE43	MISRE42	MISRE41	MISRE40	MISRE39	MISRE38	MISRE37	MISRE36	MISRE35	MISRE34	MISRE33	MISRE32
W																
Reset	The reset value is chip-specific; see the chapter that describes how modules are configured and connected.															

Figure 2407. STCU2 LBIST MISR Expected High *n* Register (STCU_LB_MISREH*n*)

Table 2294. STCU_LB_MISREH*n* field descriptions

Field	Description
0:31 MISRE[63:32]	Offline MISR Expected high Bits This field defines the high part (63 to 32) of the expected MISR.

90.5.44 STCU2 Offline LBIST MISR Read Low *n* Register (STCU_LB_MISRRL*n*)

The STCU_LB_MISRRL register report the LSB part (31 to 0) of the MISR obtained at the end of the offline LBIST controller execution. The size of the register depends on the number of MISR bits of the respective LBIST.

Offset: 0x0118 + n * 0x40 (n = 0 to 6) Access: User read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	MISRR31	MISRR30	MISRR29	MISRR28	MISRR27	MISRR26	MISRR25	MISRR24	MISRR23	MISRR22	MISRR21	MISRR20	MISRR19	MISRR18	MISRR17	MISRR16
W																
Reset	The reset value is chip-specific; see the chapter that describes how modules are configured and connected.															

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	MISRR15	MISRR14	MISRR13	MISRR12	MISRR11	MISRR10	MISRR9	MISRR8	MISRR7	MISRR6	MISRR5	MISRR4	MISRR3	MISRR2	MISRR1	MISRR0
W																
Reset	The reset value is chip-specific; see the chapter that describes how modules are configured and connected.															

Figure 2408. STCU2 Offline LBIST MISR Read Low *n* Register (STCU_LB_MISRRL*n*)

Table 2295. STCU_LB_MISRRL*n* field descriptions

Field	Description
0:31 MISRR[31:0]	Offline MISR Read Low Bits. This field is equivalent to the low bits (31 to 0) of the MISR obtained at the end of the offline LBIST execution

90.5.45 STCU2 Offline LBIST MISR Read High *n* Register (STCU_LB_MISRRH*n*)

The STCU_LB_MISRRH register report the MSB part (63 to 32) of the MISR obtained at the end of each offline LBIST controller execution. The size of the register depends on the number of MISR bits of the respective LBIST.

Offset: 0x011C + n * 0x40 (n = 0 to 6) Access: User read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	MISRR63	MISRR62	MISRR61	MISRR60	MISRR59	MISRR58	MISRR57	MISRR56	MISRR55	MISRR54	MISRR53	MISRR52	MISRR51	MISRR50	MISRR49	MISRR48
W																
Reset	The reset value is chip-specific; see the chapter that describes how modules are configured and connected.															

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	MISRR47	MISRR46	MISRR45	MISRR44	MISRR43	MISRR42	MISRR41	MISRR40	MISRR39	MISRR38	MISRR37	MISRR36	MISRR35	MISRR34	MISRR33	MISRR32
W																
Reset	The reset value is chip-specific; see the chapter that describes how modules are configured and connected.															

Figure 2409. STCU2 Offline LBIST MISR Read High *n* Register (STCU_LB_MISRRH*n*)

Table 2296. STCU_LB_MISRRH*n* field descriptions

Field	Description
0:31 MISRR[63:32]	Offline MISR Read high bits. This field is equivalent to the high bits (63 to 32) of the MISR obtained at the end of the offline LBIST execution.

90.5.46 STCU2 Online LBIST MISR Expected Low *n* Register (STCU_LB_MISRELSW*n*)

The STCU_LB_MISRELSW register defines the LSB part (31 to 0) of the expected MISR of the online LBIST controller. The size of the register depends on the number of MISR bits of the respective LBIST.

The R/W fields in this register are readable at any time. You can write to these fields when either of the following conditions is true:

- Offline self-test phase is active.
- Online self-test phase is active and STCU_CFG[WRP] = 0.

Offset: 0x0120 + n * 0x40 (n = 0 to 6) Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	MISRESW31	MISRESW30	MISRESW29	MISRESW28	MISRESW27	MISRESW26	MISRESW25	MISRESW24	MISRESW23	MISRESW22	MISRESW21	MISRESW20	MISRESW19	MISRESW18	MISRESW17	MISRESW16
W																
Reset	The reset value is chip-specific; see the chapter that describes how modules are configured and connected.															

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	MISRESW15	MISRESW14	MISRESW13	MISRESW12	MISRESW11	MISRESW10	MISRESW9	MISRESW8	MISRESW7	MISRESW6	MISRESW5	MISRESW4	MISRESW3	MISRESW2	MISRESW1	MISRESW0
W																
Reset	The reset value is chip-specific; see the chapter that describes how modules are configured and connected.															

Figure 2410. STCU2 Online LBIST MISR Expected Low *n* Register (STCU_LB_MISRELSW*n*)

Table 2297. STCU_LB_MISRELSW*n* field descriptions

Field	Description
0:31 MISRESW[31:0]	Online MISR expected low bits. This field defines the low part (31 to 0) of the expected MISR.

90.5.47 STCU2 Online LBIST MISR Expected High *n* Register (STCU_LB_MISREHSW*n*)

The STCU_LB_MISREHSW register defines the MSB part (63 to 32) of the expected MISR of the online LBIST controller. The size of the register depends on the number of MISR bits of the respective LBIST.

The R/W fields in this register are readable at any time. You can write to these fields when either of the following conditions is true:

- Offline self-test phase is active.
- Online self-test phase is active and STCU_CFG[WRP] = 0.

Offset: 0x0124 + n * 0x40 (n = 0 to 6) Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	MISRESW63	MISRESW62	MISRESW61	MISRESW60	MISRESW59	MISRESW58	MISRESW57	MISRESW56	MISRESW55	MISRESW54	MISRESW53	MISRESW52	MISRESW51	MISRESW50	MISRESW49	MISRESW48
W																
Reset	The reset value is chip-specific; see the chapter that describes how modules are configured and connected.															

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	MISRESW47	MISRESW46	MISRESW45	MISRESW44	MISRESW43	MISRESW42	MISRESW41	MISRESW40	MISRESW39	MISRESW38	MISRESW37	MISRESW36	MISRESW35	MISRESW34	MISRESW33	MISRESW32
W																
Reset	The reset value is chip-specific; see the chapter that describes how modules are configured and connected.															

Figure 2411. STCU2 Online LBIST MISR Expected High *n* Register (STCU_LB_MISREHSW*n*)

Table 2298. STCU_LB_MISREHSW*n* field descriptions

Field	Description
0:31 MISRESW[63:32]	Online MISR expected high bits. This field defines the high part (63 to 32) of the expected MISR.

90.5.48 STCU2 Online LBIST MISR Read Low *n* Register (STCU_LB_MISRRLSW*n*)

The STCU_LB_MISRRLSW register report the LSB part (31 to 0) of the MISR obtained at the end of the online LBIST controller execution. The size of the register depends on the number of MISR bits of the respective LBIST.

The content of this register is initialized to its reset value as soon as STCU_RUNSW[RUNSW] = 1.

Offset: 0x0128 + n * 0x40 (n = 0 to 6) Access: User read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	MISRRSW31	MISRRSW30	MISRRSW29	MISRRSW28	MISRRSW27	MISRRSW26	MISRRSW25	MISRRSW24	MISRRSW23	MISRRSW22	MISRRSW21	MISRRSW20	MISRRSW19	MISRRSW18	MISRRSW17	MISRRSW16
W																
Reset	The reset value is chip-specific; see the chapter that describes how modules are configured and connected.															

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	MISRRSW15	MISRRSW14	MISRRSW13	MISRRSW12	MISRRSW11	MISRRSW10	MISRRSW9	MISRRSW8	MISRRSW7	MISRRSW6	MISRRSW5	MISRRSW4	MISRRSW3	MISRRSW2	MISRRSW1	MISRRSW0
W																
Reset	The reset value is chip-specific; see the chapter that describes how modules are configured and connected.															

Figure 2412. STCU2 Online LBIST MISR Read Low *n* Register (STCU_LB_MISRRLSW*n*)

Table 2299. STCU_LB_MISRRLSW*n* field descriptions

Field	Description
0:31 MISRRSW[31:0]	Online MISR read low bits. This field is equivalent to the low bits (31 to 0) of the MISR obtained at the end of the offline LBIST execution.

90.5.49 STCU2 Online LBIST MISR Read High *n* Register (STCU_LB_MISRRHSW*n*)

The STCU_LB_MISRRHSW register report the MSB part (63 to 32) of the MISR obtained at the end of each online LBIST controller execution. The size of the register depends on the number of MISR bits of the respective LBIST.

The content of this register is initialized to its reset value as soon as STCU_RUNSW[RUNSW] = 1.

Offset: 0x012C + n * 0x40 (n = 0 to 6) Access: User read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	MISRRSW63	MISRRSW62	MISRRSW61	MISRRSW60	MISRRSW59	MISRRSW58	MISRRSW57	MISRRSW56	MISRRSW55	MISRRSW54	MISRRSW53	MISRRSW52	MISRRSW51	MISRRSW50	MISRRSW49	MISRRSW48
W																
Reset	The reset value is chip-specific; see the chapter that describes how modules are configured and connected.															

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	MISRRSW47	MISRRSW46	MISRRSW45	MISRRSW44	MISRRSW43	MISRRSW42	MISRRSW41	MISRRSW40	MISRRSW39	MISRRSW38	MISRRSW37	MISRRSW36	MISRRSW35	MISRRSW34	MISRRSW33	MISRRSW32
W																
Reset	The reset value is chip-specific; see the chapter that describes how modules are configured and connected.															

Figure 2413. STCU2 Online LBIST MISR Read High *n* Register (STCU_LB_MISRRHSW*n*)

Table 2300. STCU_LB_MISRRHSW*n* field descriptions

Field	Description
0:31 MISRRSW[63:32]	Online MISR Read high bits. This field is equivalent to the high bits (63 to 32) of the MISR obtained at the end of the offline LBIST execution.

90.5.50 STCU2 MBIST Control *k* Register (STCU_MB_CTRL*k*)

The STCU_MB_CTRL register defines the control setting of MBIST controller.

The R/W fields in this register are readable at any time. You can write to these fields when either of the following conditions is true:

- Offline self-test phase is active.
- Online self-test phase is active and STCU_CFG[WRP] = 0.

Offset: 0x0600 + k*0x4 (k = 0 to 101) Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	PTR								CSM	0	0	0	0	0	0	0
W																
Reset	The reset value is chip-specific; see the chapter that describes how modules are configured and connected.															

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	The reset value is chip-specific; see the chapter that describes how modules are configured and connected.															

Figure 2414. STCU2 MBIST Control *k* Register (STCU_MB_CTRL*k*)

Table 2301. STCU_MB_CTRL*k* field descriptions

Field	Description
0:7 PTR	Next LBIST or MBIST pointer PTR defines the logical pointer to the next LBIST or MBIST to be scheduled. The next LBIST or MBIST is scheduled concurrently to the current one if the CSM bit is set to 1, otherwise it is scheduled sequentially to the completion of the current MBIST execution. In case of NIL pointer, the CSM bit has to be set sequential (0) to define this is the end of the list. The self-test procedure is stopped when the current MBIST has been completed. (0x00 to LBIST - 1) — pointer to LBIST PTR = 0x10 + $MBIST_i$ (Pointer to MBIST) for i = 0 to XX: please refer to the MBIST partitions table for the $MBIST_i$ code. PTR = 0xFF: Pointer to NIL. No additional MBIST execution. Any other value is considered as invalid pointer and an error is set in the STCU_ERR register.
8 CSM	Concurrent/sequential mode 0 Sequential mode 1 Concurrent mode

90.6 Use cases and limitations

The STCU2 module is designed to give high flexibility during the online and offline self-test. This section shows the programming operations that have to be performed to correctly use this module in three typical use cases:

- Offline self-test sequence after a reset trigger event
- Bypass mode of the self-test sequence after a reset trigger event
- Online self-test sequence

The STCU setup has to be defined based on the application-specific safety, boot time and consumption requirements. Contact ST representative for support.

90.6.1 Offline self-test sequence

This is the typical mode of using the STCU2 module after reset trigger event is applied to STCU2. The SSCM DCF bus is used to retrieve the STCU2 schedule and LBIST or MBIST execution parameters stored in the NVM memory. The target is to cover the amount of physical defects in the digital logic and in the system RAM or ROM according to the system specifications. The sequence is the following:

- Unlock the offline STCU2 access writing the Key1/Key2 sequence in the STCU_SKC register.
- Program the LBIST fault reaction conditions (UF/RF) setting the STCU_LBUFM register.
- Program the MBIST fault reaction conditions (UF/RF) setting, depending on the number of memories, the STCU_MBUFM1/2/3/4 registers.
- Program the STCU2 internal fault reaction conditions (UF/RF) setting the STCU_ERR_FM register.
- Program the STCU_LB_CTRL, STCU_LB_PCS, STCU_MISREL/H registers of each LBIST to be executed. Optionally, the STCU_LB_PRPGL/H can also be programmed to load a specific seed.
- Program the STCU_MB_CTRL registers of each NMCUT to be executed.
- Program the STCU_WDG register to define the time budget assigned for LBIST or MBIST execution.
- Program the STCU_CFG register to define:
  - the core and LBIST or MBIST TCK clock prescaling factor setting the CLK_CFG bits,
  - the LB_DELAY field to delay the LBIST start when many LBIST are run concurrently to mitigate the potential transient power issue during the start-up phase,
  - the PMOSEN or CHKBRD to define the algorithm to be used during MBIST run,
  - the CRCREN bit to enable the self-checking feature,
  - set the pointer to the first LBIST/NMCUT to be executed.
- If the internal CRC comparison has been enabled, program the CRCE value expected at the end of the offline self-test sequence in the STCU_CRCE register.
- If PLL usage IS enabled, program the STCU_PLL_CFG value to set the required system frequency.
- Program the RUN bit in the STCU_RUN register to enable the offline self-test execution and the LM/BPLLEN bits to enable the PLL during LBIST or MBIST execution according to STCU_PLL_CFG register content.
- Start the offline self-test depending on PLL management and LBIST or MBIST execution order.
- Wait the WDGEOC execution run time specified in the STCU_WDG register.
- The STCU2 switches-off the core clock at the end of the self-test run and releases the PLL control if this feature is enabled.
- The STCU2 resets the STCU_RUN[RUN] bit and the device exits the boot sequence. If there are failures, the STCU2 flags these failures to the FCCU by raising one or both of the stcu_ncf and stcu_cf lines.
- The software application read the STCU_LBS flag register to check the failed LBIST when UF or RF conditions are detected.

- The software application read the STCU_LBE flag register to check the LBIST still on-going when UF or RF are detected.
- The software application read the STCU_MBS1/2/3/4 flag registers, depending on the number of memories, to check the failed RAM or ROM memory BIST when UF or RF are detected.
- The software application read the STCU_MBE1/2/3/4 flag registers, depending on the number of memories, to check the RAM or ROM memory BIST still running when UF or RF are detected.
- The software application read the bit INV, ENGE, CRCS, WDTO, LOCKE of the STCU_ERR_STAT register to check whether there has been an internal STCU2 engine/parameters failure when UF or RF are detected.
- If the CRCEN bit is enabled, the software application reads the CRCE and CRCR registers to check the coherence with the bit CRCS of the STCU_ERR register.
- For each LBIST, the software application reads the STCU_LBMISREL/H and STCU_LBIST_MISRRL/H registers to check the coherence with the STCU_LBS bits.

Note: The software application read operations on the STCU2 CRC and LBIST MISR registers after offline self-test execution has successfully completed are an additional safety layer added to improve the already implemented auto self-test feature included in STCU2.

90.6.2 Online self-test Sequence

This is the typical mode of using the STCU2 module during the application run. The IPS interface is used to program the STCU2 registers and to schedule the LBIST or MBIST execution. Since the online operations clean up the content of the respective digital logic (LBIST) or RAM (MBIST), activation of this mode requires caution. Follow this sequence:

- Unlock the online STCU2 access writing the Key1/Key2 sequence in the STCU_SKC register.
- Check and clear the STCU_CFG[WRP] bit to open the IPS access on the STCU2 online self-test registers.
- The LBIST, MBIST and STCU2 internal fault reaction should already be programmed during the offline self-test sequence. If they have not been programmed or there is a need to change them, they can be overwritten by setting the registers: STCU_LBUFM, STCU_MBUFM1/2/3/4 and STCU_ERR_FM.
- Program the Reset management during online LBIST execution setting the STCU_LBRMSW register.
- The STCU_LB_CTRL, STCU_LB_PCS should already be programmed during offline self-test sequence. If they have not been programmed or there is a need to change them, they can be overwritten. The registers STCU_LB_MISRELSW/HSW have to be programmed according to the expected signature.
- Overwrite/Program the STCU_MB_CTRL registers of each NMCUT to be executed.
- Overwrite/Program the STCU_WDG register to define the time budget assigned for LBIST or MBIST execution.

- Overwrite/Program the STCU_CFG register to define:
  - the core and LBIST or MBIST TCK clock prescaling factor setting the CLK_CFG bits,
  - the LB_DELAY field to delay the LBIST start when many LBISTs are running simultaneously to mitigate the potential transient power issue during the start-up phase,
  - the PMOSEN or CHKBRD to define the algorithm to be used during MBIST run,
  - the CRCREN bit to enable the self-checking feature
  - set the pointer to the first LBIST/NMCUT to be executed.
- If the internal CRC comparison is enabled, overwrite/program the CRCE value expected at the end of the online self-test sequence in the STCU_CRCE register.
- Program the STCU_RUNSW[RUNSW] bit to enable the online self-test execution, the L/MBIE to enable the Interrupt requests, and the L/MBSWPLLEN bits to enable the PLL lock signal monitor during LBIST or MBIST execution.
- Start the online self-test depending on PLL management and LBIST or MBIST execution order.
- Wait the WDGEOC execution run time specified in the STCU_WDG register.
- If the L/MBIE are enabled, an interrupt is generated at the end of all the LBIST or MBIST runs. The software application has to manage these interrupt signals and clean up the respective flags (L/MBIFLG) in the STCU_INT_FLG register.
- The STCU2 switches off the core clock at the end of the self-test run.
- The activated functional reset is managed by the STCU_LBRMSW.
- If at least one of the selected LBISTs enables the global functional reset, the STCU2 requests a functional reset which restarts the system. If all the selected LBISTs enable their own dedicated functional reset, only these local reset lines are applied and only the respective LBIST partitions are cleaned up.
- The STCU2 resets the STCU_RUNSW[RUNSW] bit. In both the reset cases and in case of errors, the STCU2 flags the failure to the FCCU by rising one or both of the stcu_ncf and stcu_cf lines.
- The software application reads the STCU_LBSSW flag register to check the failed LBIST when UF or RF conditions are detected.
- The software application reads the STCU_LBESW flag register to check the LBIST is still running when UFs or RFs are detected.
- The software application reads the STCU_MBS1SW/2SW/3S+W/4SW flag registers, depending on the number of memories, to check the failed RAM/ROM memory BIST when UFs or RFs are detected.
- The software application reads the STCU_MBE1SW/2SW/3SW/4SW flag registers, depending on the number of memories, to check the RAM/ROM memory BIST is still running when UFs or RFs are detected.
- The software application reads the bit INVPSW, ENGESW, CRCSSW, WDTOSW, LOCKESW of the STCU_ERR_STAT register to check whether there has been an internal STCU2 engine/parameters failure when UFs or RFs are detected.
- If the CRCEN bit is enabled, the software application reads the CRCE and CRCR registers to check the coherence with the STCU_ERR[CRCSSW] bit.

- For each LBIST, the software application reads the STCU_LBMISRELSW/HSW and STCU_LBIST_MISRRLSW/HSW registers to check the coherence with the STCU_LBS bits

Note: The software application read operations on the STCU2 CRC and LBIST MISR registers after online self-test execution has successfully completed are an additional safety layer added to improve the already implemented auto self-test feature included in STCU2.

90.6.3 Bypass USER Mode

In this mode, the user application parameters stored in Flash and read by SSCM are written to skip the self-test procedure after a reset trigger event is applied. It might be useful if the device is not configured for Safety applications. Follow this sequence:

1. Unlock the STCU2 access writing the offline Key1/Key2 sequence in the STCU_SKC register.
2. Set the BYP bit in the STCU_RUN register.

After the BYP bit is set, the core clock is switched off, the self-test sequence is not applied and the system can wake up and start the user application.

90.6.4 Design implementation information

The following list reports the limitations regarding the current implementation:

- Every NMCUT MBIST or LBIST can be run one time during the on/offline self-test procedure.
- It is not possible to run LBIST and MBIST concurrently.
- As the software has full control of the system while applications are running during the online self-test, the STCU2 may not take control of the PLL, but only monitor the lock signal to flag incorrect unlock events.

91 Register Protection (REG_PROT)

91.1 Introduction

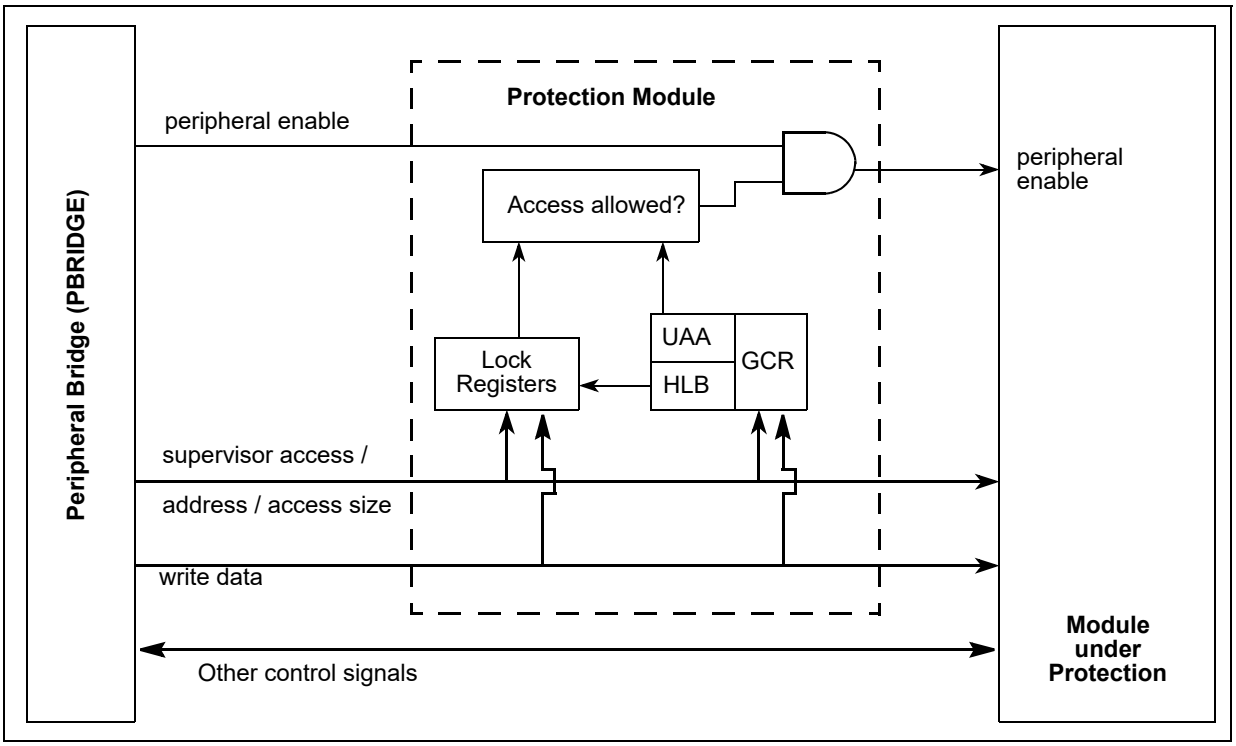
91.1.1 Overview

The Register Protection (REG_PROT) module offers a mechanism to protect defined memory-mapped address locations in a module under protection from being written. The address locations that can be protected are module-specific.

Note: Refer to the Chapter 7: Device configuration for device-specific details of the protected registers and bytes.

The protection module is located between the module under protection and the peripheral bridge (PBRIDGE). This is shown in *Figure 2415*.

Figure 2415. REG_PROT block diagram



91.1.2 Features

Register Protection includes these distinctive features:

- Restrict write accesses for the module under protection to supervisor mode only
- Lock registers for first 6 KB of memory-mapped address space
- Write to address mirror automatically sets corresponding lock bit
- Once configured lock bits can be protected from changes

Also, hard lock bit protection can only be cleared by a system reset once set.

91.1.3 Modes of operation

The Register Protection module is operable when the module under protection is operable.

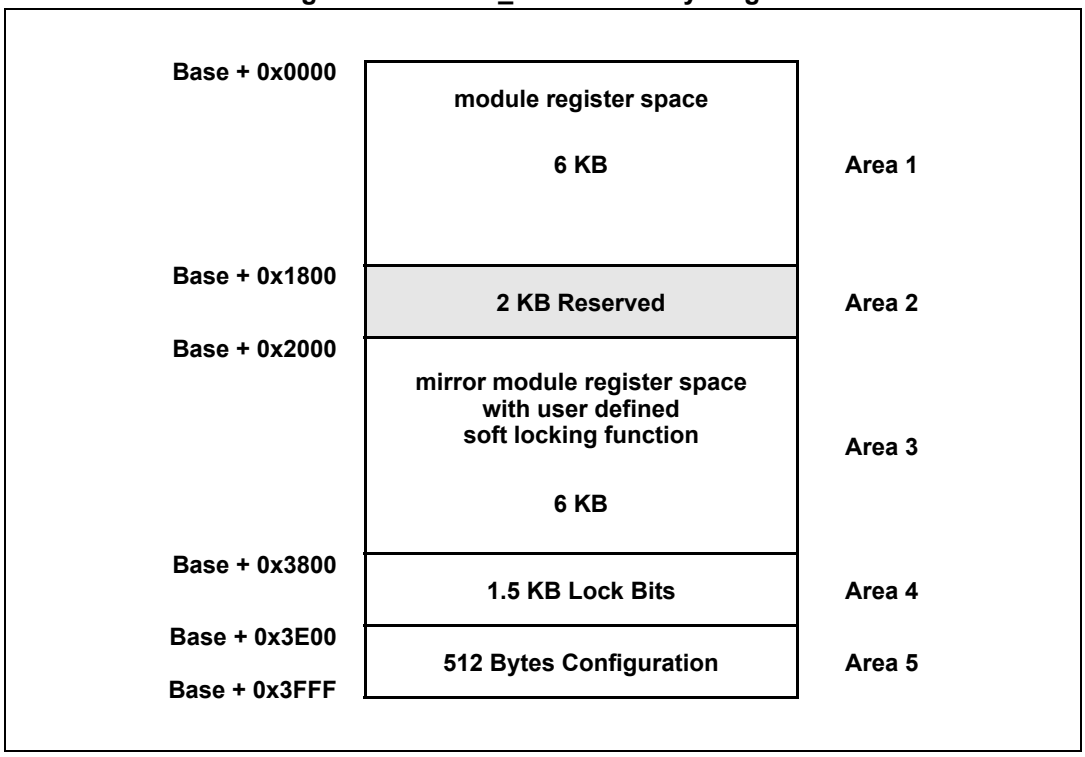
91.2 External signal description

There are no external signals.

91.3 Memory map and register definition

This section provides a detailed description of the memory map of a module with register protection. The original 16 KB module memory space is divided into five areas as shown in *Figure 2416*.

Figure 2416. REG_PROT memory diagram



- Area 1 is 6 KB and holds the normal functional module registers and is transparent for all read/write operations.
- Area 2 is 2 KB starting at offset 0x1800 is a reserved area, which shall not be accessed.
- Area 3 is 6 KB, starting at offset 0x2000 and is a mirror of area 1. A read/write access to an offset 0x2000+X will read/write the register at offset X. However a write access to offset 0x2000+X will additionally set the optional Soft Lock Bits for this offset X in the same cycle as the register at offset X is written. This provides for an atomic write and lock operation. Not all registers in area 1 need to have protection defined by associated Soft Lock Bits. For unprotected registers at offset Y, accesses to offset 0x2000+Y will be identical to accesses at offset Y.

- Area 4 is 1.5 KB and holds the Soft Lock Bits, one bit per byte in area 1. The four Soft Lock Bits associated with one module register word are arranged at byte boundaries in the memory map. The Soft Lock Bit registers can be directly written using a bit mask.
- Area 5 is 512 bytes large and holds the configuration bits of the protection mode. There is one configuration hard lock bit per module that prevents all further modifications to the Soft Lock Bits and can only be cleared by a system reset once set. The other bits, if set, will allow user access to the protected module.

If any locked byte is accessed with a write transaction, a transfer error will be issued to the system and the write transaction will not be executed. This is true even if not all accessed bytes are locked.

Accessing unimplemented 32-bit registers in Areas 4 and 5 will result in a transfer error.

91.3.1 Memory map

Table 2302 shows the memory map for a module which is protected via a REG_PROT module.

Table 2302. REG_PROT memory map

Offset	Use	Location
0x0000	Module Register 0 (MR0)	*Section 91.3.2.1*
0x0001	Module Register 1 (MR1)	*Section 91.3.2.1*
0x0002	Module Register 2 (MR2)	*Section 91.3.2.1*
0x0003 - 0x17FF	Module Register 3 (MR3) - Module Register 6143(MR6143)	*Section 91.3.2.1*
0x1800 - 0x1FFF	Reserved	
0x2000	Module Register 0 (MR0) + Set Soft Lock Bit 0 (LMR0)	*Section 91.3.2.2*
0x2001	Module Register 1 (MR1) + Set Soft Lock Bit 1 (LMR1)	*Section 91.3.2.2*
0x2002 - 0x37FF	Module Register 2 (MR2) + Set Soft Lock Bit 2 (LMR2) - Module Register 6143 (MR6143) + Set Soft Lock Bit 6143 (LMR6143)	*Section 91.3.2.2*
0x3800	Soft Lock Bit Register 0 (SLBR0): Soft Lock Bits 0-3	*Section 91.3.2.3*
0x3801	Soft Lock Bit Register 1 (SLBR1): Soft Lock Bits 4-7	*Section 91.3.2.3*
0x3802 - 0x3DFF	Soft Lock Bit Register 2 (SLBR2): Soft Lock Bits 8-11 - Soft Lock Bit Register 1535 (SLBR1535): Soft Lock Bits 6140-6143	*Section 91.3.2.3*
0x3E00 - 0x3FFB	Reserved	
0x3FFC	Global Configuration Register (GCR)	*Section 91.3.2.4*

Note: Reserved registers in area #1 will be handled as specified in the protected module's documentation.

91.3.2 Register descriptions

This section describes in address order all the REG_PROT registers. Each description includes a standard register diagram with an associated figure number. Details of register bit and field function follow the register diagrams, in bit order.

91.3.2.1 Module Registers (MR0-6143)

This is the lower 6 KB module memory space which holds all the functional registers of the module that is protected by the REG_PROT module.

91.3.2.2 Module Register and Set Soft Lock Bit (LMR0-6143)

This is memory area #3 that provides mirrored access to the MR0-6143 registers with the side effect of setting Soft Lock Bits in case of a write access to a MR that is defined as protectable by the locking mechanism. Each MR is protectable by one associated bit in a SLBR*n*.SLB*m*, according to the mapping described in *Table 2304*.

91.3.2.3 Soft Lock Bit Register (SLBR0-1535)

The Soft Lock Bit Registers hold the Soft Lock Bits for the protected registers in memory area #1, which is the normal register address space of the protected module. Each SLB register has a four Soft Lock Bits (SLB0-SLB3), each of which controls write access to a byte in memory area #1. Each Soft Lock Bit also has a corresponding Write Enable bit in the same register that controls whether the Soft Lock Bit can be written. The following table shows the mapping between the Soft Lock Bits to the bytes in memory area

Table 2303. Soft lock bits vs. protected address

Soft Lock Bit	Protected address
SLBR0.SLB0	MR0
SLBR0.SLB1	MR1
SLBR0.SLB2	MR2
SLBR0.SLB3	MR3
SLBR1.SLB0	MR4
SLBR1.SLB1	MR5
SLBR1.SLB2	MR6
SLBR1.SLB3	MR7
SLBR2.SLB0	MR8
...	...

Note: As many as 1536 Soft Lock Bit Registers (SLBRn) may be implemented, depending on the number of protected module register bytes.

Access in User Mode is read-only; in Supervisor Mode access is read/write.

Offset: 0x3800-0x3DFF

Access: Read always
Supervisor write

	0	1	2	3	4	5	6	7
R	0	0	0	0	SLB0	SLB1	SLB2	SLB3
W	WE0	WE1	WE2	WE3				
Reset	0	0	0	0	0	0	0	0

Figure 2417. Soft Lock Bit Register (SLBR*n*)

Table 2304. SLBR*n* field descriptions

Field	Description
0 WE0	Write Enable Bits for Soft Lock Bits (SLB): WE0 enables writing to SLB0 1 Value is written to SLB 0 SLB is not modified
1 WE1	Write Enable Bits for Soft Lock Bits (SLB): WE1 enables writing to SLB1 1 Value is written to SLB 0 SLB is not modified
2 WE2	Write Enable Bits for Soft Lock Bits (SLB): WE2 enables writing to SLB2 1 Value is written to SLB 0 SLB is not modified
3 WE3	Write Enable Bits for Soft Lock Bits (SLB): WE3 enables writing to SLB3 1 Value is written to SLB 0 SLB is not modified
4 SLB0	Soft Lock Bits for one MR*n* register: SLB0 can block accesses to MR[*n* *4 + 0] 1 Associated MR*n* byte is locked against write accesses 0 Associated MR*n* byte is unprotected and writable
5 SLB1	Soft Lock Bits for one MR*n* register: SLB1 can block accesses to MR[*n* *4 + 1] 1 Associated MR*n* byte is locked against write accesses 0 Associated MR*n* byte is unprotected and writable
6 SLB2	Soft Lock Bits for one MR*n* register: SLB2 can block accesses to MR[*n* *4 + 2] 1 Associated MR*n* byte is locked against write accesses 0 Associated MR*n* byte is unprotected and writable
7 SLB3	Soft Lock Bits for one MR*n* register: SLB3 can block accesses to MR[*n* *4 + 3] 1 Associated MR*n* byte is locked against write accesses 0 Associated MR*n* byte is unprotected and writable

91.3.2.4 Global Configuration Register (GCR)

This register controls the configuration of the locking of the SLB's and the level of user access allowed to the protected registers.

Note: Access in User Mode is read-only; in Supervisor Mode access is read/write.

Offset: 0x3FFC

Access: Read Always
Supervisor write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	HLB	0	0	0	0	0	0	0	UAA	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

Figure 2418. Global Configuration Register (GCR)

Table 2305. GCR field descriptions

Field	Description
0 HLB	Hard Lock Bit. This register cannot be cleared once it is set by software. It can only be cleared by a system reset. 1 All SLB bits are write protected and cannot be modified 0 All SLB bits are accessible and can be modified.
8 UAA	User Access Allowed. 1 The registers in the module under protection can be accessed in the mode defined for the module registers without any additional restrictions. 0 The registers in the module under protection can only be written in supervisor mode. All write accesses in non-supervisor mode are not executed and a transfer error is issued. This access restriction is in addition to any access restrictions imposed by the protected IP module.

Note: The GCR.UAA bit has no effect on the allowed access modes for the registers in the REG_PROT module.

91.4 Functional description

The following sections describe the functional characteristics of the Register Protection module.

91.4.1 General

This module provides a generic register (address) write-protection mechanism. The protection size can be:

- 32-bit (address == multiples of 4)
- 16-bit (address == multiples of 2)
- 8-bit (address == multiples of 1)

Which addresses are protected and the protection size is device and module specific, refer to the *Chapter 7: Device configuration for details.*

For all addresses that are protected there are SLBR*n*.SLB*m* bits that specify whether the address is locked. When an address is locked it can only be read but not written in any mode (supervisor/normal). If an address is unprotected the corresponding SLBR*n*.SLB*m* bit is always 0b0 no matter what software is writing to.

91.4.2 Change lock settings

To change the setting whether an address is locked or unlocked the corresponding SLBR*n*.SLB*m* bit needs to be changed. This can be done using the following methods:

- Modify the SLBR*n*.SLB*m* directly by writing to area #4
- Set the SLBR*n*.SLB*m* bit(s) by writing to the mirror module space (area #3)

Both methods are explained in the following sections.

91.4.2.1 Change lock settings directly via area #4

In memory area #4 the lock bits are located. They can be modified by writing to them. Each SLBR*n*.SLB*m* bit has a mask bit SLBR*n*.WE*m* which protects it from being modified. This masking makes clear-modify-write operations unnecessary.

Figure 2419 shows two modification examples. In the left example there is a write access to the SLBR*n* register specifying a mask value which allows modification of all SLBR*n*.SLB*m* bits. The example on the right specifies a mask which only allows modification of the bits SLBR*n*.SLB[3:1].

Figure 2419. Change lock settings directly via Area #4

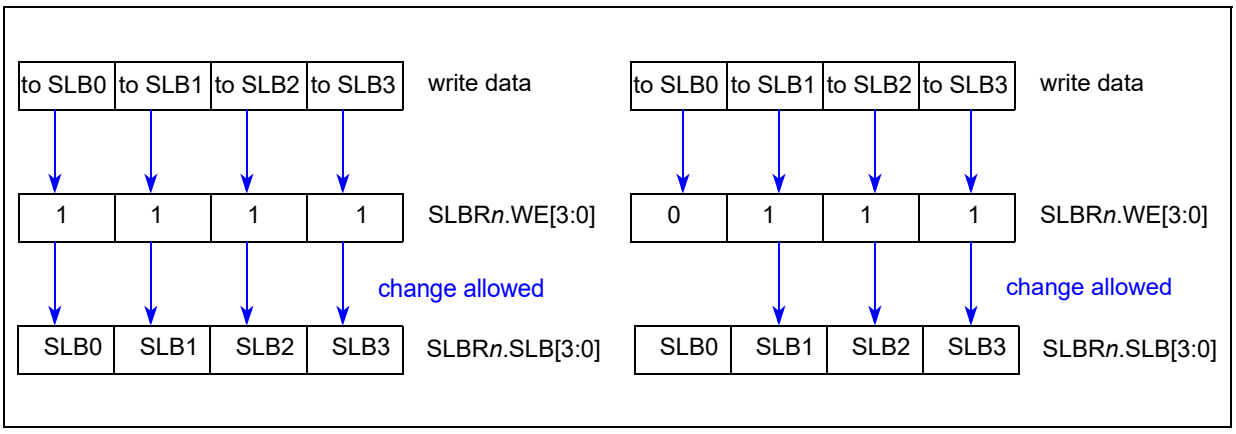
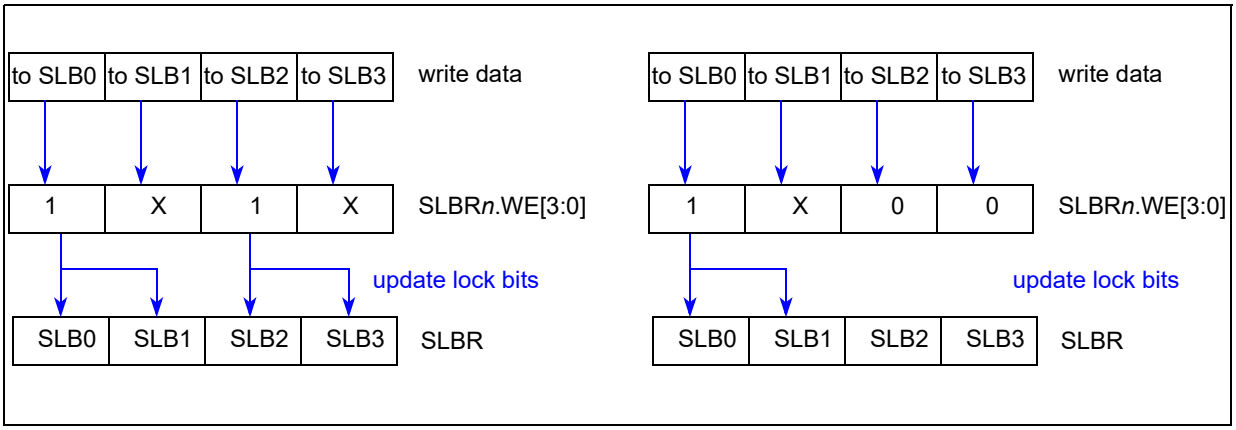


Figure 2419 shows four registers that can be protected 8-bit wise. In *Figure 2420* registers with 16-bit protection and in *Figure 2421* registers with 32-bit protection are shown:

Figure 2420. Change lock settings for 16-bit protected addresses

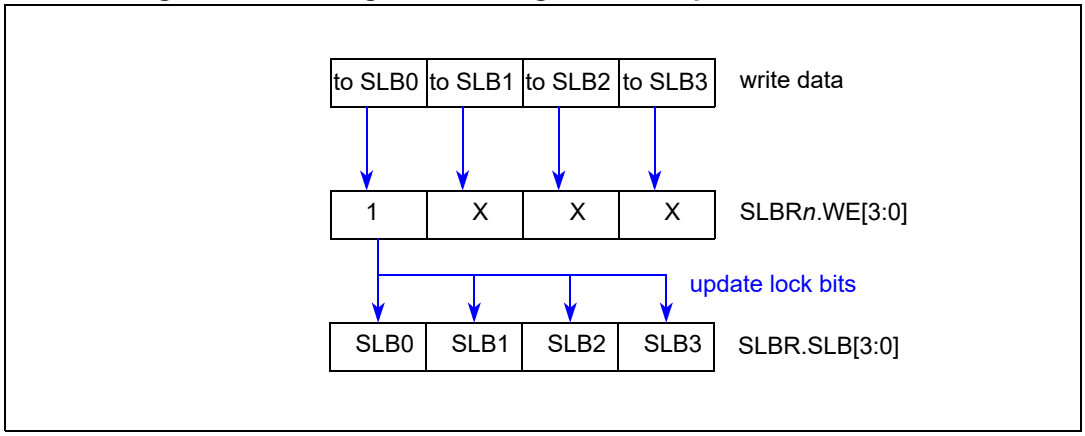


On the right side of *Figure 2420* it is shown that the data written to SLBR*n*.SLB[0] is automatically written to SLBR*n*.SLB[1] also. This is done as the address reflected by SLBR*n*.SLB[0] is protected 16-bit wise. Note that in this case the write enable SLBR*n*.WE[0] must be set while SLBR*n*.WE[1] does not matter. As the enable bits SLBR*n*.WE[3:2] are cleared the lock bits SLBR*n*.SLB[3:2] remain unchanged.

In the example on the left side of *Figure 2420* the data written to SLBR*n*.SLB[0] is mirrored to SLBR*n*.SLB[1] and the data written to SLBR*n*.SLB[2] is mirrored to SLBR*n*.SLB[3] as for both registers the write enables are set.

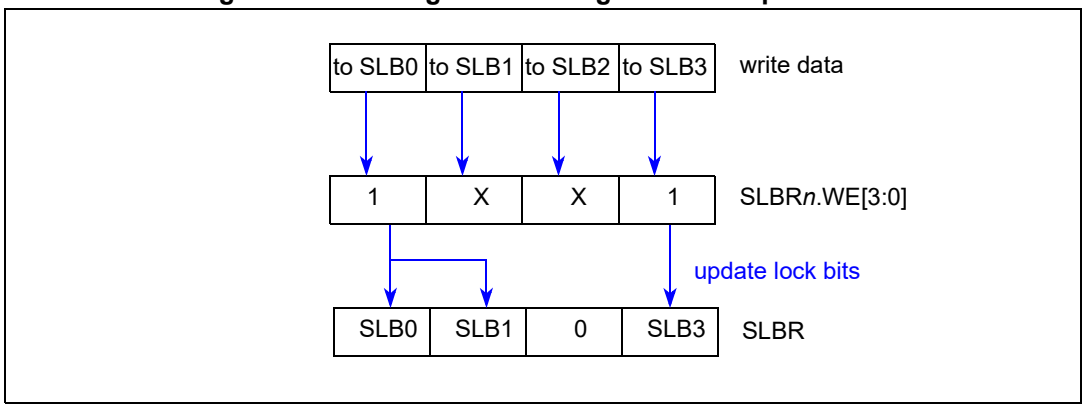
In *Figure 2421* a 32-bit wise protected register is shown. When SLBR*n*.WE[0] is set the data written to SLBR*n*.SLB[0] is automatically written to SLBR*n*.SLB[3:1] also. Otherwise SLBR*n*.SLB[3:0] remains unchanged.

Figure 2421. Change lock settings for 32-bit protected addresses



In *Figure 2422* an example is shown which has a mixed protection size configuration:

Figure 2422. Change lock settings for mixed protection

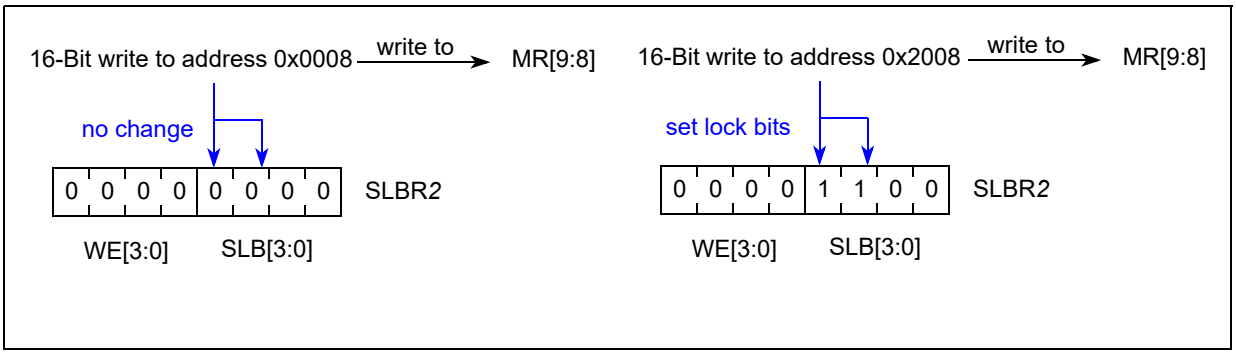


The data written to SLBR*n*.SLB[0] is mirrored to SLBR*n*.SLB[1] as the corresponding register is 16-bit protected. The data written to SLBR*n*.SLB[2] is blocked as the corresponding register is unprotected. The data written to SLBR*n*.SLB[3] is written to SLBR*n*.SLB[3].

91.4.2.2 Enable locking via mirror module space (Area #3)

It is possible to enable locking for a register after writing to it. To do so the mirrored module address space must be used. *Figure 2423* shows one example:

Figure 2423. Enable locking via mirror module space (Area #3)

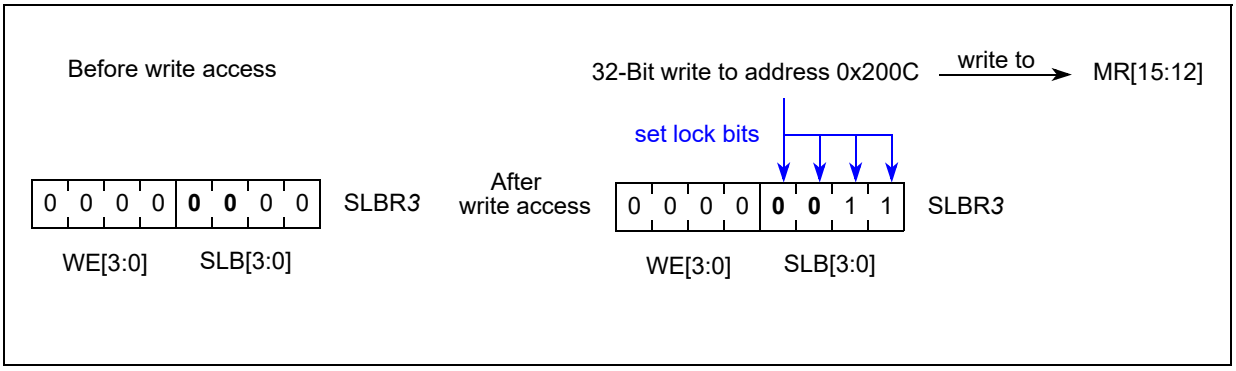


When writing to address 0x0008 the registers MR9 and MR8 in the protected module are updated. The corresponding lock bits remain unchanged (left part of *Figure 2420*).

When writing to address 0x2008 the registers MR9 and MR8 in the protected module are updated. The corresponding lock bits SLBR2.SLB[1:0] are set while the lock bits SLBR2.SLB[3:2] remain unchanged (right part of *Figure 2420*).

Figure 2424 shows an example where some addresses are protected and some are not:

Figure 2424. Enable locking for protected and unprotected addresses



In the example in *Figure 2424* addresses 0x0C and 0x0D are unprotected. Therefore their corresponding lock bits SLBR3.SLB[1:0] are always 0b0 (shown in bold). When doing a 32-bit write access to address 0x200C only lock bits SLBR3.SLB[3:2] are set while bits SLBR3.SLB[1:0] stay 0b0.

Note: Lock bits can only be set via writes to the mirror module space. Reads from the mirror module space will not change the lock bits.

91.4.2.3 Write protection for locking bits

Changing the locking bits through any of the procedures mentioned in *Section 91.4.2.1: Change lock settings directly via area #4* and *Section 91.4.2.2: Enable locking via mirror module space (Area #3)* is only possible as long as the bit GCR.HLB is cleared. Once this bit is set the locking bits can no longer be modified until there is a system reset.

91.4.3 Access errors

The protection module generates transfer errors under several circumstances. For the area definition refer to *Figure 2416*

1. If accessing area #1 or area #3, the protection module will pass on any access error from the underlying Module under Protection.
2. If user mode is not allowed, user writes to all areas will assert a transfer error and the writes will be blocked.
3. If accessing the reserved area #2, a transfer error will be asserted.
4. If accessing unimplemented 32-bit registers in area #4 and area #5 a transfer error will be asserted.
5. If writing to a register in area #1 and area #3 with Soft Lock Bit set for any of the affected bytes a transfer error is asserted and the write will be blocked. Also the complete write operation to non-protected bytes in this word is ignored.
6. If writing to a Soft Lock Register in area #4 with the Hard Lock Bit being set a transfer error is asserted.
7. Any write operation in any access mode to area #3 while Hard Lock Bit GCR is set.

91.5 Initialization/application information

91.5.1 Reset

The reset state of each individual bit is shown within the Register Description section (refer to *Section 91.3.2: Register descriptions*). In summary, after reset, locking for all MR*n* registers is disabled. The registers can be accessed in Supervisor Mode only.

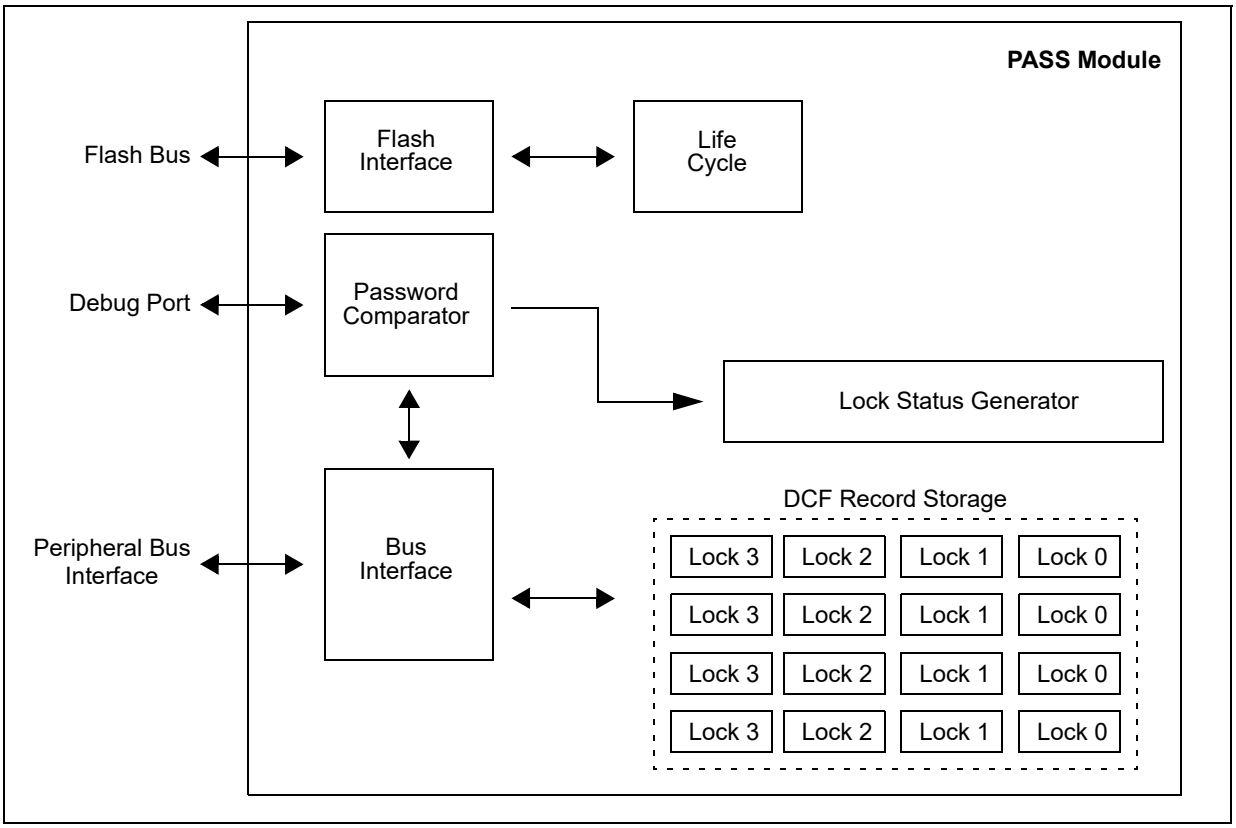
92 Password and Device Security Module (PASS)

92.1 Introduction

92.1.1 Overview

The Password and Device Security (PASS) module shown in *Figure 2425* receives password challenges and determines their validity. It also maintains the device security and access states.

Figure 2425. PASS module block diagram



92.1.2 Features

The PASS includes these distinctive features:

- Lifecycle status
- Password comparison
- JTAG password comparison
- Production Disable control
- Fuse Bypass Configuration and Password Comparison

92.1.3 Modes of operation

The PASS operates identically in all system modes.

92.2 External signal description

The PASS has 1 external pin that is used for Production Disable control (refer to *Section 92.4.2: Production Disable*).

92.3 Memory map and register definition

This section provides a detailed description of all memory-mapped registers in the PASS. (DCF clients are described in *Section 92.4: DCF clients*.)

Table 2306 shows the memory map for the PASS. Note that all addresses are offsets: the absolute address for registers may be calculated by adding the specified offset to the base address of the PASS. The address of the DCF client is local to the PASS module and must be qualified with the appropriate Client Select (CS[14:0]) for the PASS module, as defined in the DCF Record Section.

Do not write to address 0x00D0 with bit value 0.

Table 2306. PASS memory map

Offset	Register	Access	Reset Value	DCF Client Address(1) (ADDR[14:0])	Section
0x0000	Lifecycle Status Register (LCSTAT)	R	0xX0000007	None	*Section 92.3.1.1*
0x0008	Challenge Selector Register (CHSEL)	R/W	0x00000000	None	*Section 92.3.1.2*
0x0010	Challenge Status Register (CSTAT)	R	0x00000000	None	*Section 92.3.1.3*
0x0014-0x001F	Reserved				
0x0020	Challenge Input Register 0 (CIN0)	W	0x00000000	None	*Section 92.3.1.4*
0x0024	Challenge Input Register 1 (CIN1)	W	0x00000000	None	*Section 92.3.1.4*
0x0028	Challenge Input Register 2 (CIN2)	W	0x00000000	None	*Section 92.3.1.4*
0x002C	Challenge Input Register 3 (CIN3)	W	0x00000000	None	*Section 92.3.1.4*
0x0030	Challenge Input Register 4 (CIN4)	W	0x00000000	None	*Section 92.3.1.4*
0x0034	Challenge Input Register 5 (CIN5)	W	0x00000000	None	*Section 92.3.1.4*
0x0038	Challenge Input Register 6 (CIN6)	W	0x00000000	None	*Section 92.3.1.4*
0x003C	Challenge Input Register 7 (CIN7)	W	0x00000000	None	*Section 92.3.1.4*
0x0040-0x00AF	Reserved				
None	Censorship	None	0x00000000	0x00B0	*Section 92.4.1*
0x00B4-0x00BF	Reserved				
None	Production Disable	None	0x00000000	0x00C0	*Section 92.4.2*
0x00C4-0x00CF	Reserved				

Table 2306. PASS memory map (continued)

Offset	Register	Access	Reset Value	DCF Client Address[1] (ADDR[14:0])	Section
0x00D0	Clock Jitter Enable (CJE)	R/W	0xFFFFFFFF	None	*Section 92.3.1.5*
0x00D4-0x00DF	Reserved				
None	Fuse Bypass	None	0x00000001	0x00E0	*Section 92.4.3*
0x00E4-0x00FF	Reserved				
Password Group 0					
0x0100	Lock 0 Status Register (LOCK0_PG0)	R/W	0xXXXXXXXX[2]	0x0100	*Section 92.3.1.6*
0x0104	Lock 1 Status Register (LOCK1_PG0)	R/W	0x0000XXXX[2]	0x0104	*Section 92.3.1.7*
0x0108	Lock 2 Status Register (LOCK2_PG0)	R/W	0xXXXXXXXX[2]	0x0108	*Section 92.3.1.8*
0x010C	Lock 3 Status Register (LOCK3_PG0)	R/W	0xXFXXXXXX[2]	0x010C	*Section 92.3.1.9*
Password Group 1					
0x0110	Lock 0 Status Register (LOCK0_PG1)	R/W	0xXXXXXXXX[2]	0x0110	*Section 92.3.1.6*
0x0114	Lock 1 Status Register (LOCK1_PG1)	R/W	0x0000XXXX[2]	0x0114	*Section 92.3.1.7*
0x0118	Lock 2 Status Register (LOCK2_PG1)	R/W	0xXXXXXXXX[2]	0x0118	*Section 92.3.1.8*
0x011C	Lock 3 Status Register (LOCK3_PG1)	R/W	0xXFXXXXXX[2]	0x011C	*Section 92.3.1.9*
Password Group 2					
0x0120	Lock 0 Status Register (LOCK0_PG2)	R/W	0xXXXXXXXX[2]	0x0120	*Section 92.3.1.6*
0x0124	Lock 1 Status Register (LOCK1_PG2)	R/W	0x0000XXXX[2]	0x0124	*Section 92.3.1.7*
0x0128	Lock 2 Status Register (LOCK2_PG2)	R/W	0xXXXXXXXX[2]	0x0128	*Section 92.3.1.8*
0x012C	Lock 3 Status Register (LOCK3_PG2)	R/W	0xXFXXXXXX[2]	0x012C	*Section 92.3.1.9*
Password Group 3					
0x0130	Lock 0 Status Register (LOCK0_PG3)	R/W	0xXXXXXXXX[2]	0x0130	*Section 92.3.1.6*
0x0134	Lock 1 Status Register (LOCK1_PG3)	R/W	0x0000XXXX[2]	0x0134	*Section 92.3.1.7*
0x0138	Lock 2 Status Register (LOCK2_PG3)	R/W	0xXXXXXXXX[2]	0x0138	*Section 92.3.1.8*
0x013C	Lock 3 Status Register (LOCK3_PG3)	R/W	0xXFXXXXXX[2]	0x013C	*Section 92.3.1.9*

1. DCF Client Address column shows what is the DCF used to preload the register value during reset. If the column value is “None”, it means that the register value at reset is fixed in hardware.
2. Reset value depends on password group lock state.

Note: Non-implemented registers generate bus aborts if enabled by the SOC.

92.3.1 Register descriptions

The following registers are available in the PASS. The shaded bits are reserved for future use. For future compatibility, only write ‘0’ to these bits; they always read ‘0’.

92.3.1.1 Lifecycle Status register (LCSTAT)

The Lifecycle Status Register reflects the production status and censorship status of the device. It is possible to mature the device (move the device lifecycle forward), but never to revert the lifecycle to an earlier state.

The reset value of the LCSTAT register LIFE field derives from an external lifecycle management module (like SSCM). It is a 3-bit lifecycle. This register simply displays the lifecycle input from external blocks.

Offset: 0x0000 Access: Software Read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	CNS	JUN	FBE	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	x(1)	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	LIFE		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1

1. Reset value depends on the Censorship DCF client.

Figure 2426. Lifecycle Status register (LCSTAT)

PASS_LCSTAT register functions are as shown in *Table 2307*.

Table 2307. LCSTAT register field descriptions

Field	Description
0 CNS	This field indicates the state of the Censorship DCF Client. The bit can be set (or cleared) via DCF Record in UTEST Flash. Refer to *Section 92.6.3: Censoring and uncensoring the device*. 0 Device is uncensored 1 Device is censored
1 JUN	This field indicates whether the device has been temporarily uncensored. Successful transmission of the JTAG password sets this bit to 1, otherwise it is 0. Unsuccessful transmission clears this bit. **Note:** Refer to the SPC58xEx/SPC58xGx Microcontroller Security Reference Manual for details.

Table 2307. LCSTAT register field descriptions (continued)

Field	Description
2 FBE	Fuse Bypass enable This field indicates whether the Fuse Bypass DCF Client is set to Fuse Bypass Enable. 0 No Fuse Bypass 1 Fuse Bypass Enable **Note:** Refer to *Section 92.4.3* for details.
29:31 LIFE	The lifecycle setting is manipulated externally (SSCM). Pass displays a 3-bit input lifecycle. 000 Failure Analysis mode 001 Reserved 010 OEM Production 011 Customer Delivery 110 ST Production 111 In Field

92.3.1.2 Challenge Selector register (CHSEL)

This register determines which password group is challenged via the Challenge Input Register.

When a master accesses this register, the PASS module stores the number of the master in the LOCK3_PGn register of the selected passgroup and sets the PGL bit in that register.

The ID of the accessing master sets the CMST field of the Challenge Status register.

Offset: 0x0008 Access: Software Read/Write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	GRP	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 2427. CHSEL register

CHSEL register functions are as shown in *Table 2308*.

Table 2308. CHSEL register field descriptions

Field	Description
30:31 GRP	Password Group to unlock 00 Password Group 0 01 Password Group 1 10 Password Group 2 11 Password Group 3

92.3.1.3 Challenge Status register (CSTAT)

This register provides status information for the password challenge.

Offset: 0x0010 Access: Software Read

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	CMST			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 2428. Challenge Status register (CSTAT)

CSTAT register functions are as shown in *Table 2309*.

Table 2309. CSTAT register field descriptions

Field	Description
28:31 CMST	ID of the master which has last written the CHSEL register

Only the master shown in this register can access the CINn registers; access by other masters result in a comparison fail regardless of the values written.

92.3.1.4 Challenge Input register (CIN*n*)

This register is used to provide the Challenge word *n* for a password group as selected by the Challenge Selector Register.

Valid password criteria is that it should not be all Zeros or all Ones

Offset: 0x0020 + *n**0x4 (*n* = 0 to 7) Access: Software Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0															
W	PW32															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0															
W	PW32															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 2429. Challenge Input register (CIN*n*)

CIN register functions are as shown in *Table 2310*.

Table 2310. CIN*n* register field descriptions

Field	Description
0:31 PW32	32 bits of the 256-bit password challenge. Only 32-bit writes may be used.

After programming the Challenge Selector Register, the password challenge needs to be written with eight 32-bit writes from CIN0 to CIN7—in that order. The most significant 32 bits of the password challenge must be written to CIN0. Once CIN7 is written, the password is compared. The write access only completes after the comparison has been executed.

All accesses to these registers need to be made by the bus master which wrote the CHSEL register previously. If another master writes any of the CIN registers before the original master has written CIN7, the comparison fails.

92.3.1.5 Clock Jitter Enable (CJE)

Software accessible one time writable register. This register stores the jitter enable status for the Pseudo Random Clock Divider Module (PRCD). The default value of this register if '1'. When this bit is cleared, the PRCD is enabled and the baud rate clock to several serial communications modules includes large amounts of pseudo random clock edge jitter, making the serial communications modules unusable. For details about which serial communications modules are affected, refer to the clocking section of the Reference Manual.

For more information on Production disable, refer to *Section 92.5.5: Production Disable*

Debug access is blocked when Production disable bit[0] (Gated Debug Enable) is asserted, the Chip is *In Field* and Clock Jitter is not enabled.

Offset: 0x00D0 Access: Software Read/Write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	CJE
W																
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Figure 2430. Clock Jitter Enable (CJE)

Table 2311. CJE field descriptions

Field	Description
31 CJE	Clock Jitter Enable 0 Clock Jitter is enabled. The baud rate clock to Serial communications modules includes large amounts of pseudo random clock edge jitter. 1 Clock Jitter is disabled. The baud rate clocks function normally.

92.3.1.6 Password Group *n* - Lock 0 Status register (LOCK0_PG*n*)

Each Password Group has one Lock 0 Status register. This register shows the current lock state for the Flash blocks. Each time a password group is unlocked the register becomes writable. For additional details of the LOCK bits, refer to the Flash memory description.

The LOCK bits do not come into effect before the lifecycle has matured to *OEM Production* or older.

Offset: 0x0100 + *n**0x10 (*n* = 0 to 3) Access: Software Read/Write & DCF Record Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	TSLOCK	ATSL	LOWLOCK													
Reset	x(1)	x(1)	x(1)	x(1)	x(1)	x(1)	x(1)	x(1)	x(1)	x(1)	x(1)	x(1)	x(1)	x(1)	x(1)	x(1)

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	MIDLOCK															
Reset	x(1)	x(1)	x(1)	x(1)	x(1)	x(1)	x(1)	x(1)	x(1)	x(1)	x(1)	x(1)	x(1)	x(1)	x(1)	x(1)

1. If no DCF records have been written to change the lock state, the bit will reset to '1'.

Figure 2431. Password Group *n* - Lock 0 Status register (LOCK0_PG*n*)

LOCK0_PG*n* register functions are as shown in *Table 2312*.

Table 2312. LOCK0_PG*n* register field descriptions

Field	Description
0 TSLOCK	UTEST NVM Lock This bit is used to lock the UTEST NVM block from programs (erase protection not needed since UTEST NVM is OTP and not erasable).
1 ATSL	Alternate Interface UTEST NVM Lock. This bit is used to lock the UTEST NVM block from programs on the alternate interface (erase protection not needed since UTEST NVM is OTP and not erasable).
2:15 LOWLOCK	Low Block Lock A value of 1 in a bit of the lock register signifies that the corresponding block is locked for program and erase.
16:31 MIDLOCK	Mid Block Lock A value of 1 in a bit of the lock register signifies that the corresponding block is locked for program and erase.

Note: Lock registers can only be accessed with 32-bit operations.

The initial value of the lock bits is set via DCF records stored in UTEST Flash block and copied over during reset. The default value before any DCF record is written is '1', indicating that a block is locked.

Warning: **When Password Slot Access bits are set, failure to program valid passwords or DCF records for all password groups before lifecycle matures to In-Field renders the part unusable.**
When Password Slot Access bits are not set, failure to program valid passwords or DCF records for all password groups before lifecycle matures to OEM Production renders the part unusable.

Once a password group is unlocked, software can set or clear any of the lock bits in the group, by writing to the registers.

The resulting lock status of a Flash block is determined by the combination of the lock bits of all password groups. If a block is locked in multiple groups, then all lock bits for the block need to be cleared before program and erase is possible. However, unless the lifecycle is older than *OEM Production*, the Flash blocks are always unlocked regardless of the values in the password groups.

When Production Disable register bit[1](BAF PLock Hardware Interlock enable) is asserted and lifecycle is *In Field*, then the external signal is ORed with BAF_plock(low_lock[0]) of LOCK0_PGx.

92.3.1.7 Password Group *n* - Lock 1 status register (LOCK1_PG*n*)

Each Password Group has one Lock 1 Status Register. This register shows the current lock state for the Flash blocks. Each time a password group is unlocked, the register becomes writable. For additional details of the LOCK bits, refer to the Flash description.

The LOCK bits do not come into effect before the lifecycle matures to *OEM Production* or older.

Offset: 0x0104 + *n**0x10 (*n* = 0 to 3) Access: Software Read/Write & DCF Record Write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	HIGHLOCK															
W																
Reset	x(1)	x(1)	x(1)	x(1)	x(1)	x(1)	x(1)	x(1)	x(1)	x(1)	x(1)	x(1)	x(1)	x(1)	x(1)	x(1)

1. If no DCF records have been written to change the lock state, the bit will reset to '1'.

Figure 2432. Password Group *n* - Lock 1 status register (LOCK1_PG*n*)

LOCK1_PG*n* register functions are as shown in *Table 2313*.

Table 2313. LOCK1_PG*n* register field descriptions

Field	Description
16:31 HIGHLOCK	High Block Lock A value of 1 in a bit of the lock register signifies that the corresponding block is locked for program and erase.

Note: Lock registers can only be accessed with 32-bit operations.

The initial value of the lock bits is set via DCF records stored in UTEST Flash block and copied over during reset. The default value before any DCF record is written, is '1', which indicates a block is locked.

Warning: When Password Slot Access bits are set, failure to program valid passwords or DCF records for all password groups before lifecycle matures to In-Field renders the part unusable.
When Password Slot Access bits are not set, failure to program valid passwords or DCF records for all password groups before lifecycle matures to OEM Production renders the part unusable.
However, additional DCF Records can be written to UTEST to change the initial value of this register.

Once a password group is unlocked, software can set or clear any of the lock bits in the group by writing to the registers.

The resulting lock status of a Flash block is determined by the combination of the lock bits of all password groups. If a block is locked in multiple groups, then all lock bits for the block need to be cleared before program and erase is possible. However, unless the lifecycle is older than *OEM Production*, the Flash blocks are always unlocked regardless of the values in the password groups.

92.3.1.8 Password Group *n* - Lock 2 status register (LOCK2_PG*n*)

Each Password Group has one Lock 2 Status Register. This register shows the current lock state for the Flash blocks. Each time a password group is unlocked, the register becomes writable. For additional details of the LOCK bits, refer to the Flash description.

The LOCK bits do not come into effect before the lifecycle is matured to *OEM Production* or older.

Offset: 0x0108 + *n**0x10 (*n* = 0 to 3) Access: Software Read/Write & DCF Record Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	A256KLOCK															
Reset	x(1)	x(1)	x(1)	x(1)	x(1)	x(1)	x(1)	x(1)	x(1)	x(1)	x(1)	x(1)	x(1)	x(1)	x(1)	x(1)

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	A256KLOCK															
Reset	x(1)	x(1)	x(1)	x(1)	x(1)	x(1)	x(1)	x(1)	x(1)	x(1)	x(1)	x(1)	x(1)	x(1)	x(1)	x(1)

1. If no DCF records have been written to change the lock state, the bit resets to ‘1’.

Figure 2433. Password Group *n* - Lock 2 status register (LOCK2_PG*n*)

Note: Lock registers can only be accessed with 32-bit operations.

LOCK2_PG*n* register functions are as shown in *Table 2314*.

Table 2314. LOCK2_PG*n* register field descriptions

Field	Description
0:31 A256KLOCK	256K address space block lock A256KLOCK[31:0] A value of 1 in a bit of the lock register signifies that the corresponding block is locked for program and erase.

The initial value of the lock bits is set via DCF records stored in UTEST Flash block and copied over during reset. The default value before any DCF record is written is ‘1’, which indicates a block is locked.

Warning: When Password Slot Access bits are set, failure to program valid passwords or DCF records for all password groups before lifecycle matures to In-Field renders the part unusable.
When Password Slot Access bits are not set, failure to program valid passwords or DCF records for all password groups before lifecycle matures to OEM Production renders the part unusable.
However, additional DCF Records can be written to UTEST to change the initial value of this register.

Once a password group is unlocked, software can set or clear any of the lock bits in the group, by writing to the registers.

The resulting lock status of a Flash block is determined by the combination of the lock bits of all password groups. If a block is locked in multiple groups, then all lock bits for the block need to be cleared before program and erase are possible. However, unless the lifecycle is older than *OEM Production*, the Flash blocks are always unlocked regardless of the values in the password groups.

92.3.1.9 Password Group *n* - Lock 3 status register (LOCK3_PG*n*)

Each Password Group has one Lock 3 Status Register. This register shows the current lock state for the Flash blocks. Each time a password group is unlocked, the PGL bit is cleared and the Lock registers become writable.

The LOCK bits do not come into effect before the lifecycle matures to *OEM Production* or older.

Offset: 0x010C + *n**0x10 (*n* = 0 to 3) Access: Software Read/Write & DCF Record Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	PGL	DBL	MO	0	MSTR				0	0	0	RL4(1)	RL3(1)	RL2(1)	RL1(1)	RL0(1)
W																
Reset	x(2)	x(2)	x(2)	0	1	1	1	1	0	0	0	x(2)	x(2)	x(2)	x(2)	x(2)

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	A256KLOCK(1)															
W																
Reset	x(3)	x(3)	x(3)	x(3)	x(3)	x(3)	x(3)	x(3)	x(3)	x(3)	x(3)	x(3)	x(3)	x(3)	x(3)	x(3)

1. Not all devices include this field. Refer to the device reference manual's chip configuration chapter for details.
2. Reset value depends on lifecycle. If lifecycle is *Customer delivery* or earlier, the value will be 0, otherwise 1.
3. If no DCF records have been written to change the lock state, the bit will reset to '1'.

Figure 2434. Password Group *n* - Lock 3 status register (LOCK3_PG*n*)

LOCK3_PG*n* register functions are as shown in *Table 2315*.

Table 2315. LOCK3_PG*n* register field descriptions

Field	Description
0 PGL	Password Group Lock This bit may be set in software to lock the password group, when the device is older than *Customer Delivery*. Before that the value of the bit is always 0. This bit is only updatable from SW path. It cannot be updated from DCF loading from Flash. This bit can be cleared by writing a valid password to the Password Challenge Input Registers. 0 Password group registers are unlocked. All four registers in the Password group may be read and written without restriction. 1 Password group registers are locked. Writes to the password group registers have no effect. Read accesses work normally (if lifecycle is *Customer Delivery* or earlier, this bit has no effect and the registers are always unlocked).
1 DBL	Debug Interface Lock The default value of this bit would be '1' if not updated from flash. 0 Debug interface is unlocked. Accessed to all JTAG clients is allowed (Except HSM). 1 Debug interface is locked. Only access to JTAG password challenge register is possible. Access to all other JTAG clients is blocked.

Table 2315. LOCK3_PG*n* register field descriptions (continued)

Field	Description
2 MO	Master Only 0 All masters can modify the passgroup settings if PGL is cleared. 1 Only the Master which unlocked the passgroup can change the passgroup settings.
4:7 MSTR	Master Access Only the Master with this matching master ID can change the passgroup settings. This field is loaded with the ID of the master which unlocked the passgroup once the correct password is provided.
11 RL4	Read Lock Region 4 lock bit The default value of this bit would be ‘1’ if not updated from flash. 0 Region 4 is not protected. 1 Region 4 is protected. **Note:** The region lock plays an important role on protecting the flash from programming and reading operations.
12 RL3	Read Lock Region 3 lock bit The default value of this bit would be ‘1’ if not updated from flash. 0 Region 3 is not protected. 1 Region 3 is protected. **Note:** The region lock plays an important role on protecting the flash from programming and reading operations.
13 RL2	Read Lock Region 2 lock bit The default value of this bit would be ‘1’ if not updated from flash. 0 Region 2 is not protected. 1 Region 2 is protected. **Note:** The region lock plays an important role on protecting the flash from programming and reading operations.
14 RL1	Read Lock Region 1 lock bit The default value of this bit would be ‘1’ if not updated from flash. 0 Region 1 is not protected. 1 Region 1 is protected. **Note:** The region lock plays an important role on protecting the flash from programming and reading operations.
15 RL0	Read Lock Region 0 lock bit The default value of this bit would be ‘1’ if not updated from flash. 0 Region 0 is not protected. 1 Region 0 is protected. **Note:** The region lock plays an important role on protecting the flash from programming and reading operations.
16:31 A256KLOCK	256K address space block lock A256KLOCK[47:32] A value of 1 in a bit of the lock register signifies that the corresponding block is locked for program and erase.

Note: Lock registers can only be accessed with 32-bit operations.

The initial value of the lock bits is set via DCF records stored in UTEST Flash block and copied over during reset. The default value before any DCF record is written is ‘1’, which indicates a block is locked.

Warning: **When Password Slot Access bits are set, failure to program valid passwords or DCF records for all password groups before lifecycle matures to In-Field renders the part unusable.**
When Password Slot Access bits are not set, failure to program valid passwords or DCF records for all password groups before lifecycle matures to OEM Production renders the part unusable.
However, additional DCF Records can be written to UTEST to change the initial value of this register.

Once a password group is unlocked, software can set or clear any of the lock bits in the group, by writing to the registers.

The resulting lock status of a Flash block is determined by the combination of the lock bits of all password groups. If a block is locked in multiple groups, then all lock bits for the block need to be cleared before program and erase are possible. However, unless the lifecycle is older than *OEM Production*, the Flash blocks are always unlocked regardless of the values in the password groups.

For operation of the region read protect feature, refer to *Section 92.5.3: Flash Memory Read Protection*.

Once a censored device enters the *Failure Analysis* lifecycle, read protection is active regardless whether the system is in debug mode or not. It is still possible to unlock Flash regions with the correct passwords, however. Flash UTest region controlled by LOCK3[0]is read unlocked in Failure Analysis lifecycle.

92.4 DCF clients

Where a DCF client is listed, the reset value of that register may automatically be preloaded during reset based on the data stored in DCF Records in Flash UTEST region.

The address of the DCF client is local to the PASS module and must be qualified with the appropriate Client Select (CS[14:0]) for the PASS module, as defined in the DCF Record Section.

92.4.1 Censorship

Censorship is a DCF client that is *not* accessible by software. It is automatically preloaded during reset with data stored in DCF Record format in the Flash UTEST region. The Censorship DCF client defines whether the MCU is censored or not (Refer to *Table 2306*).

DCF Client Address: 0x00B0 Censorship Access: DCF Record Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R																
W																
Reset	x(1)	x(1)	x(1)	x(1)	x(1)	x(1)	x(1)	x(1)	x(1)	x(1)	x(1)	x(1)	x(1)	x(1)	x(1)	x(1)

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R																
W	x(2)	x(2)	x(2)	x(2)	x(2)	x(2)	x(2)	x(2)	x(2)	x(2)	x(2)	x(2)	x(2)	x(2)	x(2)	x(2)
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. This bit is not defined.
2. This value is loaded by DCF Record during reset.

Figure 2435. Censorship register (not accessible)

Table 2316. Censorship register field descriptions

Field	Description
16:31	Censorship If this value is 0x55AA the MCU is NOT censored. If this value is anything other that 0x55AA, the MCU is censored.

92.4.2 Production Disable

Note: Production Disable is a DCF client that is not accessible by software. It is automatically preloaded during reset with data stored in DCF Record format in the Flash UTEST region. The Production disable DCF client defines the Debug Disable and Hardware Interlock aspects of the Production Disable feature (refer to Table 2306).

Note: The DCF client implements 2 bits with a default value of '0'. The DCF client can only be written once and writes from subsequent DCF records are ignored.

For more information on Production disable, refer to *Section 92.5.5: Production Disable*

DCF Client Address: 0x00C0 Production Disable — Access: DCF Record Write Once Only

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R																
W															HI(1)	DD(1)
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. This value is loaded by DCF Record during reset.

Figure 2436. Production Disable DCF client

Table 2317. Production Disable DCF Client field descriptions

Field	Description
HI	Hardware Interlock 0 BAF Flash block programming is controlled by PASS_LOCK0[BAF] bits. 1 BAF Flash block is locked for programming while external hardware pin is asserted '0'.
DD	Debug disable 0 Debug Interface is Enabled and Disabled using Censorship, JTAG password & PASS_LOCK3[DBL] bit as described elsewhere. 1 Debug Interface is disabled while Clock Jitter is NOT enabled.

92.4.2.1 Debug Disable (DD)

Note: Debug Disable is an optional feature that, once the Lifecycle is In Field or beyond, overrides all other debug controls and disables the debug interface (refer to Table 2320).

92.4.2.2 Hardware Interlock (HI)

Hardware Interlock is an optional feature that, once the lifecycle is In Field or beyond, prevents programming of the BAF Flash block while an external hardware pin is asserted low. This feature has higher priority than PASS_LOCK0[BAF] bits. While the pin is asserted, no software action anywhere in the MCU can cause the BAF to be programmed.

Once the hardware pin is asserted high, programming of BAF is controlled by PASS_LOCK0[BAF] bits.

92.4.3 Fuse Bypass DCF Client

This DCF client is used to configure the password protected Fuse bypass feature. Refer to *Section 92.5.6: Fuse Bypass* for details.

The Fuse Bypass DCF Client is not directly readable in software, but the state of the Fuse Bypass bit can be read in the LCSTAT register (refer to *Section 92.3.1.1: Lifecycle Status register (LCSTAT)*).

The DCF client is written during Destructive Reset Phase 3 and compared during Functional Reset Phase 3; if this compare does not match the data already written, the logic will decode No fuse bypass.

DCF Client Address: 0x00E0 Fuse Bypass — Access: DCF Record Write Once Only

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R																
W																
Reset	x(1)	x(1)	x(1)	x(1)	x(1)	x(1)	x(1)	x(1)	x(1)	x(1)	x(1)	x(1)	x(1)	x(1)	x(1)	x(1)

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R																
W																FB(2)
Reset	x(1)	x(1)	x(1)	x(1)	x(1)	x(1)	x(1)	x(1)	x(1)	x(1)	x(1)	x(1)	x(1)	x(1)	x(1)	1

1. This bit is not defined.
2. This value is loaded by DCF Record during reset.

Figure 2437. Fuse Bypass DCF Client

Table 2318. Fuse Bypass register field descriptions

Field	Description
31 FB	Fuse Bypass enable 0 The Fuse can NOT be Bypassed 1 Bypass of the fuse is possible with a password

92.4.4 Password Slot Access

Password Slot Access is a write-once DCF client. The DCF controls access for both read and write starting from Lifecycle = OEM Production. If the bit is 1, the slot is still accessible when Life Cycle = OEM Production (and blocked starting from Life Cycle = In Field). Otherwise if the bit is set to 0, the slot is blocked starting from Life Cycle = OEM Production.

DCF Client Address: : 0x00B4 Password Slot Access — Access: DCF Record Write Once

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R																
W													Password Group 3 Access	Password Group 2 Access	Password Group 1 Access	Password Group 0 Access
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 2438. Password Slot Access DCF Client

Table 2319. Password Slot Access DCF Client field descriptions

Field	Description
28:31	Password Slot Access per password group

92.5 Functional description

The primary purpose of the PASS is to provide fine-grained access control to the Flash and debugger interface. It utilizes the DCF mechanism, as described in the SSCM section.

92.5.1 Censorship

Censorship is the overall control that enables and disables two independent security features: Debug Interface access; and flash memory read protection.

The MCU can be Censored by writing a DCF Record to flash that writes to the Censorship DCF client (refer to *Section 92.4.1: Censorship*).

92.5.2 Debug Interface Access

The debug interface in fully enabled in Customer_Delivery Lifecycle. But from OEM_Production lifecycle, it can be blocked using censorship and selectively by either entering a 256 bits password via JTAG or by software clearing the DBL bit the PASS_LOCK3 register (refer to *92.3.1.9: Password Group n - Lock 3 status register (LOCK3_PGn)*). Both of these debug enabling options are further conditioned by lifecycle and Production Disable.

The following table describes all options for enabling/blocking Debug Interface access.

Table 2320. Debug Interface Enable Truth table

Input Conditions						Outcome
Lifecycle	**Production Disable mode**	**Clock Jitter Enable**	**Censorship**	**Debug Enable (LOCK3[DBL] bit)**	**JTAG Password**	**Debug Interface Enable**
Customer Delivery	Don't Care	Don't Care	Don't Care	Don't Care	Don't Care	Enabled
OEM Production	Don't Care	Don't Care	Not censored	Don't Care	Don't Care	Enabled
			Censored	Unlocked (DBL=0)	Don't Care	Enabled
				Locked (DBL=1)	Matched	Enabled
					Not Matched	Blocked
In Field	Disabled	Don't Care	Not censored	Don't Care	Don't Care	Enabled
			Censored	Unlocked (DBL=0)	Don't Care	Enabled
				Locked (DBL=1)	Matched	Enabled
					Not Matched	Disabled
	Enabled	Jitter enabled	Not censored	Don't Care	Don't Care	Enabled
			Censored	Unlocked (DBL=0)	Don't Care	Enabled
				Locked (DBL=1)	Matched	Enabled
					Not matched	Disabled
		Jitter disabled		Don't Care	Don't Care	Disabled
Failure Analysis	Don't Care	Don't Care	Don't Care	Don't Care	Don't Care	Enabled

The HSM can also block Debug Interface Access, even if it enabled by PASS. This allows HSM to granted Debug Interface Access after an encrypted secure challenge/response protocol with an external tool (refer to the SPC58xEx/SPC58xGx Microcontroller Security Reference Manual).

92.5.2.1 JTAG Password Challenge

To enable the Debug Interface Access, the JTAG password challenge register in the JTAGC, can be written by an external tool. If that password matches the JTAG password stored in UTEST flash, Debug Interface Access is enabled (according to the table above) until the next destructive reset.

92.5.3 Flash Memory Read Protection

Flash Memory Read Protection is a security feature that can selectively make regions of the flash memory unreadable. Any attempt to read a region of flash that is read protected results in the read access terminating with a transfer error. This is consistent for all bus masters including debug bus masters.

The assertion of flash memory region read protection is controlled by Lifecycle, Censorship and whether the Debug Interface access is enabled or blocked. *Table 2321: Flash Memory Read Protection truth table* describes the combination of conditions that drive Flash Memory Read Protection.

The intention of the feature is to protect the contents of the flash from being read. The flash memory is readable in normal operation, allowing the application to run, but becomes read protected when a Debug tool is attached and enabled (JTAG password or LOCK3[DBL] bit).

The Flash memory is also read protected when the lifecycle status is *Failure_Analysis*. This lifecycle is used for a module field failure, where the device is returned for failure analysis. This protection prevents the contents of the flash memory from being read.

The flash memory is divided into a number of Regions (refer to *Chapter 7: Device configuration*, *Section 7.11.1: Password and Device Security Module (PASS) configuration*).

Table 2321. Flash Memory Read Protection truth table

INPUTS				OUTPUTS	
Lifecycle	**Censorship**	**Debug Interface Enable from HSM**	**Lock3[RLx]**	**Read Protected RL[0] (UTEST)**	**Read Protected RL[4:1] (Other Regions)**
STMicroelectronics Production / Customer Delivery	Don't Care	Don't Care	Don't Care	Readable	Readable
OEM Production / IN Field	Uncensored	Don't Care	Don't Care	Readable	Readable
	Censored	Blocked	Don't Care	Readable	Readable
		Enabled	Locked (1)	Read Blocked	Read Blocked
			Unlocked (0)	Readable	Readable
Failure Analysis	Uncensored	Don't Care	Don't Care	Readable	Readable

92.5.4 UTEST protection

During the *Customer Delivery* lifecycle, the user must program a valid password or a DCF record that writes LOCKm_PGn registers to unprotect the UTest for each password group in order to prevent permanent and irreversible locking of Flash blocks.

92.5.5 Production Disable

Production disable activation protocol and implementation is described in *Chapter 65: Boot Assist Flash (BAF)*, *Section 65.3.6: Production disable activation protocol and implementation*.

92.5.5.1 Overview and rationale

Production Disable is an optional feature that, once enabled, allows the MCU to function completely normally without any constraints, except the Debug interface is disabled.

Note: Once the customer has decided that the Debug Interface should be re-enabled, the process to re-enable it makes the MCU unusable in a production environment by adding clock jitter to the baud rate clocks of several serial communications modules (CAN and FlexRay). This jitter means those serial communications modules are not able to communicate on the network.
As this Production Disable feature renders the MCU unusable in the application, inadvertent activation would be disastrous. Therefore a feature is included that prevents this Jitter activation unless an external hardware pin is asserted (refer to Section 92.4.2: Production Disable for details).

92.5.5.2 Enabling

Note: The feature is enabled via the Production Disable DCF client in the PASS module. Bit 0 enables the Debug Interface Disable. Bit 1 enables the Hardware Interlock pin.

This information can be written by adding a DCF Record to the DCF record list in Flash UTEST. The DCF Client is configured to accept only one write. Subsequent DCF records that write to this DCF client are ignored.

Note: If no DCF Record writes to this client, the default state of both features is disabled. If, for a particular application, there is no intention of ever using this feature, it would be good practice to add a DCF Record that writes this client to 0x0000_0000. This is not essential, but ensures the feature cannot be Enabled accidentally.

92.5.5.3 Clock jitter activation

The BAF code copies the ‘Clock Jitter Activation’ constant value from a location in the BAF code space, to the Clock Jitter Register in the PASS module (refer to *Section 92.3.1.5: Clock Jitter Enable (CJE)* for details).

This register is write once. The default value for the Clock Jitter Activation constant in BAF Flash is 0xFFFF_FFFF (erased state).

To enable the Clock Jitter feature, the value of the Clock Jitter Activation constant in BAF Flash should be programmed to make the most significant bit ‘0’. After the next reset, the BAF copies this updated value to the Clock Jitter Register and the feature is activated.

92.5.5.4 Clock Jitter Truth table

Table 2322 shows the relationship between Lifecycle, Production Disable & Clock Jitter enable input conditions versus the state of the Baud Rate Clock.

Table 2322. Clock Jitter Truth Table

Lifecycle	Production Disable [Debug Disable]	Clock Jitter Enable	Baud Rate Clock State
STMicroelectronics_ Production Customer_Delivery In_Field	Don’t Care	0b1XXXXXXX_(24 more X)	Normal - No Jitter
		0b0XXXXXXX_(24 more X)	Jitter
Failure_Analysis	Off[0]	0b1XXXXXXX_(24 more X)	Normal - No Jitter
		0b0XXXXXXX_(24 more X)	Jitter
	On[1]	Don’t Care	Jitter

92.5.6 Fuse Bypass

92.5.6.1 Fuse

Some devices include an additional security feature, whereby a silicon fuse is used to gate access to Flash Test Mode when the device is in Failure_Analysis lifecycle. The fuse must be blown, creating physical damage to the silicon, before access is granted to Flash Test Mode. The PASS implements a configurable feature to bypass the requirement to blow the fuse to enter flash test mode, by entering a password instead. This feature is called Fuse Bypass.

92.5.6.1.1 Use Case: No Fuse Bypass

Customers who want no alternative to the Fuse to enter Flash Test Mode in FA lifecycle:

- During customer assembly, customer programs DCF Record in UTEST to write this DCF client with 0.
- And as a back-up, during customer assembly, the customer could program the Fuse Bypass Password to a random value.

92.5.6.1.2 Use Case: Fuse Bypass Password

Customers who require a password to enter Flash Test Mode in FA lifecycle, but do not require the added sophistication of blowing a fuse:

- During customer assembly, customer programs DCF Record in UTEST to write this DCF client with 1.
- During customer assembly, the customer programs the Fuse Bypass Password into Flash.
- The customer maintains a system for retrieving these passwords for the life of the module.
- If the part is returned to STMicroelectronics for FA, the customer must provide the Fuse Bypass password that was originally programmed into the device.
- If the password is lost or irretrievable, the customer must accept that FA can only be performed if the die is removed from the package and the fuse blown.

92.5.6.1.3 Use Case: No Fuse. No Password

Customers who are less concerned with security and will accept neither fuse nor password to enter Flash Test Mode in FA lifecycle:

- Customer need do nothing special for this mode in customer assembly.
- When the part is returned to STMicroelectronics for FA, The DCF Client is still in the default state of 1. This is Fuse Bypass.
- The Fuse Bypass password should still be in the erased state (all 1). The tester can enter all 1 as the password to gain access to Flash Test Mode.

92.5.6.2 Fuse Bypass Password

The Fuse Bypass Password is a password entered by the STMicroelectronics via JTAG interface, during device FA, to gain access to Flash Test Mode. Flash Test Mode is required to perform Vt Distribution analysis and other tests on an MCU. This password is only functional during FA lifecycle. Flash blocks that are assigned as sealed during production

are locked against any form of Flash test, including reading, erasing and Vt Distribution analysis.

92.5.6.2.1 Fuse Bypass Password Generation

The Fuse Bypass password is created by the customer during their module assembly process. It must be programmed into UTest Flash while in CUSTOMER_DELIVERY lifecycle.

When an MCU is returned to STMicroelectronics for FA, the MCU must be in FA lifecycle and this password must also be provided to STMicroelectronics to allow access to Flash Test Mode without blowing the fuse. Therefore the customer should develop a process for managing Fuse Bypass passwords for the life of the module.

If the password is not available when a device is returned for FA, STMicroelectronics must remove the die from the package and blow the fuse before entry is possible to Flash Test Mode and Flash FA can be undertaken.

92.5.6.2.2 Fuse Bypass Password storage

The Fuse Bypass Password is 224 bits (28 bytes) wide and is to be stored in UTest Flash at a fixed location. Refer to the device reference manual for the address of the Fuse Bypass password.

The password region of UTest Flash is readable only when in CUSTOMER_DELIVERY lifecycle. In any lifecycle after this (including FA lifecycle), this region is read protected.

92.5.6.2.3 Fuse Bypass Password Challenge entry

The Fuse Bypass password challenge is entered by an external tool to the password register in JTAGC.

As the Fuse Bypass Password is 224 bits; and the JTAGC password register is 256, only the least significant 224 bits of the register are used.

92.5.6.2.4 Fuse Bypass Password compare

The Fuse Bypass Password challenged, is compared against the password stored in UTest Flash. Only the least significant 224 bits are compared.

If the password matches, Flash Test Mode access is allowed until the next POR reset.

There are no passwords that the hardware will detect as illegal. All 0 and all 1 are valid passwords.

92.6 Initialization/application information

92.6.1 Reset

The reset state of each individual bit is shown within *Section 92.3.1: Register descriptions*. However, many bits receive their values based on DCF records stored in the UTest Flash memory block.

92.6.2 Setting lock bits in a password group

There is no restriction on setting and clearing lock registers in the UTEST Flash, apart from the available space for DCF records. It is possible to set and clear them multiple times.

Table 2323. CS/ADDR values for Lock registers

Register	DCF CS/ADDR
PW Group 0	
Lock0	0xXXXX0100(1)
Lock1	0xXXXX0104(1)
Lock2	0xXXXX0108(1)
Lock3	0xXXXX010C(1)
PW Group 1	
Lock0	0xXXXX0110(1)
Lock1	0xXXXX0114(1)
Lock2	0xXXXX0118(1)
Lock3	0xXXXX011C(1)
PW Group 2	
Lock0	0xXXXX0120(1)
Lock1	0xXXXX0124(1)
Lock2	0xXXXX0128(1)
Lock3	0xXXXX012C(1)
PW Group3	
Lock0	0xXXXX0130(1)
Lock1	0xXXXX0134(1)
Lock2	0xXXXX0138(1)
Lock3	0xXXXX013C(1)

1. Value depends on the DCF Client Select assigned to the PASS module in the device.

In order to set the MIDLOCK field in the Lock0 register of password group 1 to the value 0x00FF and other fields to 0, the DCF record has to be added as shown in *Table 2324*.

Table 2324. Setting some bits in the MIDLOCK field

ADDR offset	DATA	Description
0x00	0x05AA55AF	Start Record
0x04	0x00000000	
0x08	0x000000FF	PW Group 1, Lock0 Set MIDLOCK to 0xFF
0x0C	0xXXXX0110(1)	

Table 2324. Setting some bits in the MIDLOCK field (continued)

ADDR offset	DATA	Description
0x10	0xFFFFFFFF	Stop Record
0x14	0xFFFFFFFF	
...	...	

1. Value depends on the DCF Client Select assigned to the PASS module in the device.

92.6.3 Censoring and uncensoring the device

There is no restriction on setting and clearing the censorship status in the UTest Flash, apart from the available space for DCF records. It is possible to censor and un-censor multiple times.

The DCF Client Select / Address for the Censorship DCF client, is a concatenation of the DCF Client Select (assigned outside of PASS) and the address 0x00B0. Therefore 0xXXXX00B0. In order to un-censor the device, the data field of the record needs to be written with 0x----55AA (upper 16 bits are not compared). Any other value censors the device.

On delivery the device is censored.

93 Tamper Detection Module (TDM)

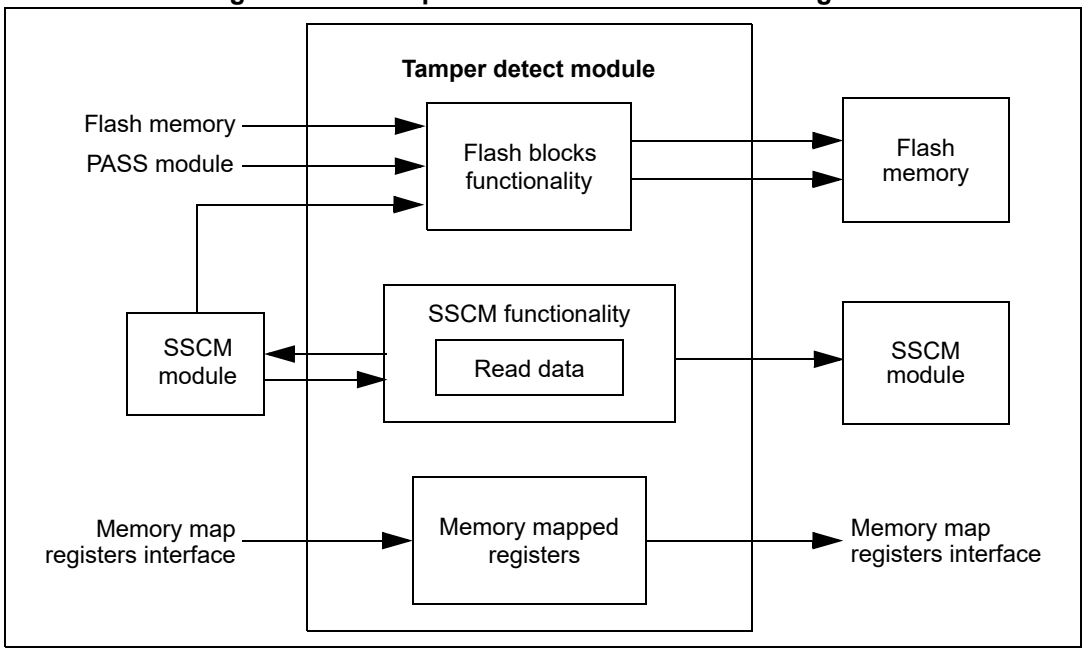
93.1 Overview

The Tamper Detection Module provides a type of flash memory erase protection mechanism that forces software to write a record associated with one or more blocks in a Tamper Detection Region (TDR) before the block(s) can be erased. The mechanism does not *prevent* an erase operation on any block within a TDR: it requires a record to be written before the operation can execute. Collectively, the records are referred to as a “diary”. At minimum, the diary serves as an erase counter. Because the diary record content is customer-defined, the diary can also serve as a history.

Up to 6 TDRs can be defined via DCF records.

Figure 2439 shows the block diagram of the TDM and its interface to other system components.

Figure 2439. Tamper Detection Module block diagram



Security information is passed to the TDM by the SSCM module. Among the data are:

- OTP (One-Time Programmable) settings for each of the flash blocks
- Base address for the diary in the flash
- Block assignments for the TDRs
- Tamper region override information (via the DCF bus)
- Software tamper region override

The TDM module controls the flash blocks for erase operations and sets them as OTP

The two flash buses are used to control the flash blocks for erase operation and set them as OTP. OTP means that flash erase of the entire block is disabled and the only words that are already erased can be programed. Over-programming is not possible.

Note: The PASS module provides the control for erase operation for each flash block as well as for each region of the flash memory, and the PASS settings have priority over the settings of the TDM.

The flash block indicates to the TDM the address of the programed word. This address is internally decoded and provides a means to verify whether the block is part of any TDRs. The memory map registers interface provides a means to the user to read information of the TDM registers.

93.2 Features

The Tamper Detection Module supports these distinctive features:

- Erase counter (diary) for each TDR to record up to 256 write attempts on the flash blocks
  - The diary is divided into 6 parts (TDRs), each consisting of 256 128-bit records
  - The diary block should be assigned as OTP region in flash
  - The diary area can store any type of relevant information
- Tamper Detection Region Block Assignment
  - Any flash block can be assigned to any TDR and be controlled
- 32-bit Erase Lock DCF clients for each flash memory TDR defined
- 32-bit OTP Enable DCF clients to disable the erase operation for an entire flash block
- Override TDR via DCF records(35)
  - Allows block assigned to a TDR to be unlocked for erase even if the TDR is locked for this operation
- Memory Map Registers Interface for reading via software of specific registers

93.3 External signal description

There are no external signals driving or being driven by the TDM.

93.4 DCF client

TDM-related DCF records perform the following functions:

- Establish a permanent diary base address
- Define Tamper Detect Regions (TDRs) to monitor on-chip flash memory program/erase activity
- Permanently disable one or more Tamper Regions
- Permanently disable ability for software to permanently override one or more TDRs
- Configure flash memory as One Time Programmable (OTP) on a per-block basis

The following table gives an overview on the TDM DCF clients implemented.

35. The Tamper Region Override DCF record is a “write one only” DCF record. If an override is applied to a TDR it is permanent and cannot be reversed.

Table 2325. DCF client address map

DCF client name	Description	Strategy
Diary Base Address	Stores the base address of the tamper detection diary	write-once, write 1 to 0 only
Tamper Region Override	Stores information of which TDR is to be overridden	write 0 to 1 only
OTPEN[nb] (OTP Enable)	Stores information of which flash blocks is to be assigned as OTP (One Time Programmable) - nb means one register for each Lock register of the flash memory	write 0 to 1 only
TDR[n]_LOCK[m] Region Assignment	Stores information of which flash block is assigned to a TDR – "*n*" specifies a TDR – "*m*" specifies the flash blocks as per the LOCK registers of the flash memory	write 0 to 1 only

93.4.1 Diary Base Address (DBA) DCF client

This DCF client holds base address information of the diary. *Figure 2440* shows the diagram for this register.

Caution: This DCF client is implemented as Write Once and is writable from 1 to 0 only. After one DCF record has written to this client, all subsequent writes are ignored.

DCF Client Address: 0x0000 Access: DCF write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R																
W	DBA															
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R																
W	DBA															
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Figure 2440. Diary Base Address (DBA) DCF client

Table 2326. DBA field descriptions

Field	Description
0:31 DBA	Diary Base Address This field holds the base address of the diary. This information is then used to verify if the diary being programed matches any TDR region. **Note:** Diary Base Address must be on a 16KB boundary.

93.4.2 Tamper Region Override (TO) DCF client

This DCF client holds overriden information for each TDR. *Figure 2441* shows the diagram for this register.

Caution: This DCF client is writable from 0 to 1 only. Attempted writes of bits from 1 to 0 are ignored, making the override of a TDR permanent.

DCF Client Address: 0x0001 Access: DCF write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R																
W	0	0	0	0	0	0	0	0	0	0	TOE5	TOE4	TOE3	TOE2	TOE1	TOE0
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 2441. Tamper Region Override (TO) DCF client

Table 2327. TO field descriptions

Field	Description
26:31 TOE[5:0]	TDR Override Enable Each TOE field, where *n* represents the TDRs, holds information of the Override Enable status for each TDR. 0 Normal operation. Blocks assigned to this TDR can NOT be erased until the diary is written 1 Diary override. The diary for this TDR can be erased WITHOUT writing to the diary

93.4.3 One Time Programmable Enable (OTPEN0) DCF client

This DCF client holds OTP information for particular blocks of flash memory. This DCF client mirrors the LOCK0 register of the flash memory. *Figure 2442* depicts this client.

Caution: This DCF client is writable from 0 to 1 only. Attempted writes of bits from 1 to 0 are ignored, making the assignment of a flash block to OTP irreversible.

DCF Client Address: 0x0008 Access: DCF write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R																
W	0	0	0	0	LOWOTPEN											
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R																
W	MIDOTPEN															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 2442. One Time Programmable Enable (OTPEN0) DCF client

Table 2328. OTPEN0 field descriptions

Field	Description
4:15 LOWOTPEN	Low Block OTP Enable This field affects flash memory blocks in Low address space. See the chip-specific information for the mapping between field bits and flash blocks. The same mapping also applies to control similar functions in the embedded flash memory module. 0 Corresponding flash blocks are not assigned as OTP 1 Corresponding flash blocks are assigned as OTP
16:31 MIDOTPEN	Mid Block OTP Enable This field affects flash memory blocks in Mid address space. See the chip-specific information for the mapping between field bits and flash blocks. The same mapping also applies to control similar functions in the embedded flash memory module. 0 Corresponding flash blocks are not assigned as OTP 1 Corresponding flash blocks are assigned as OTP

93.4.4 One Time Programmable Enable (OTPEN1) DCF client

This DCF client holds OTP information for particular blocks of flash memory. This DCF client mirrors the LOCK1 register of the flash memory. *Figure 2443* depicts the client.

Caution: This DCF client is writable from 0 to 1 only. Attempted writes of bits from 1 to 0 will be ignored, making the assignment of a flash block to OTP irreversible.

DCF client Address: 0x0009 Access: DCF write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R																
W	HIGHOTPEN															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 2443. One Time Programmable Enable (OTPEN1) DCF client

Table 2329. OTPEN1 field descriptions

Field	Description
16:31 HIGHOTPEN	High Block OTP Enable This field affects flash memory blocks in High address space. See the chip-specific information for the mapping between field bits and flash blocks. (Refer to the embedded memories chapter). 0 Corresponding flash blocks are not assigned as OTP 1 Corresponding flash blocks are assigned as OTP

93.4.5 One Time Programmable Enable (OTPEN2) DCF client

This DCF client holds OTP information for particular blocks of flash memory. This DCF client mirrors the LOCK2 register of the flash memory. *Figure 2444* depicts the client.

Caution: This DCF client is writable from 0 to 1 only. Attempted writes of bits from 1 to 0 will be ignored, making the assignment of a flash block to OTP irreversible.

DCF Client Address: 0x000A Access: DCF write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R																
W	256KBOTPEN[31:16]															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R																
W	256KBOTPEN[15:0]															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 2444. One Time Programmable Enable (OTPEN2) DCF client

Table 2330. OTPEN2 field descriptions

Field	Description
0–31 256KBOTPEN	256 KB Block OTP Enable (256KBOTPEN[31:0]) This field affects flash memory blocks in 256KB address space. See the chip-specific information for the mapping between field bits and flash blocks. The same mapping also applies to control similar functions in the embedded flash memory module. 0 Corresponding flash blocks are not assigned as OTP 1 Corresponding flash blocks are assigned as OTP

93.4.6 One Time Programmable Enable (OTPEN3) DCF client

This DCF client holds OTP information for particular blocks of flash memory. This DCF Client mirrors the LOCK3 register of the flash memory. *Figure 2445* depicts the client.

Caution: This DCF client is writable from 0 to 1 only. Attempted writes of bits from 1 to 0 will be ignored, making the assignment of a flash block to OTP irreversible.

DCF client Address: 0x000B Access: DCF write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R																
W	256KBOTPEN[47:32]															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 2445. One Time Programmable Enable (OTPEN3) DCF client

Table 2331. OTPEN3 field descriptions

Field	Description
16–31 256KBOTPEN	256 KB Block OTP Enable (256KBOTPEN[47:32]) This field affects flash memory blocks in the upper range of 256 KB address space. See the chip-specific information for the mapping between field bits and flash blocks. The same mapping also applies to control similar functions in the embedded flash memory module. 0 Corresponding flash blocks are not assigned as OTP 1 Corresponding flash blocks are assigned as OTP

93.4.7 Tamper Region Assignment DCF client (TDR*n*_LOCK0)

This DCF client holds the assignment information of the flash blocks to a specific TDR. Each flash block can be assigned to any of the TDR (*n* varies from zero to 5). This DCF client however holds the assignment information for only those blocks specified in the LOCK0 register in the flash memory. Therefore, the diagram of this register mirrors the LOCK0 register in the flash memory. *Figure 2446* depicts the TDR*n*_LOCK0 DCF client.

Caution: This DCF client is writable from 0 to 1 only. Attempted writes of bits from 1 to 0 will be ignored, making the assignment of a flash block to a TDR irreversible unless the entire TDR is overridden.

DCF client Address: 0x14 + ((n * 4) + 0) Access: DCF write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R																
W	0	0	0	0	LOWTDRAGN											
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R																
W	MIDTDRAGN															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 2446. Tamper Region Assignment DCF client (TDR*n*_LOCK0)

Table 2332. TDRn_LOCK0 field descriptions

Field	Description
4:15 LOWTDRAGN	Low Block Tamper Region Assignment This field affects flash memory blocks in Low address space. See the chip-specific information for the mapping between field bits and flash blocks. The same mapping also applies to control similar functions in the embedded flash memory module. 0 Corresponding flash blocks are not assigned to the TDR*n* 1 Corresponding flash blocks are assigned to the TDR*n*
16:31 MIDTDRAGN	Mid Block Tamper Region Assignment This field affects flash memory blocks in Mid address space. See the chip-specific information for the mapping between field bits and flash blocks. The same mapping also applies to control similar functions in the embedded flash memory module. 0 Corresponding flash blocks are not assigned to the TDR*n* 1 Corresponding flash blocks are assigned to the TDR*n*

93.4.8 Tamper Region Assignment DCF client (TDR*n* _LOCK1)

This DCF client holds the assignment information of the flash blocks to a specific TDR. Each flash block can be assigned to any of the TDR (*n* varies from zero to 5). This DCF client however holds the assignment information for only those blocks specified in the LOCK1 register in the flash memory. Therefore, the diagram of this DCF client mirrors the LOCK1 register in the flash memory. *Figure 2447* depicts the TDR*n*_LOCK1 DCF client.

Caution: This DCF client is writable from 0 to 1 only. Attempted writes of bits from 1 to 0 will be ignored

DCF client Address: 0x14 + ((n * 4) + 1) Access: DCF write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R																
W	HIGHTDRAGN															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 2447. Tamper Region Assignment DCF client (TDR*n* _LOCK1)

Table 2333. TDR*n* _LOCK1 field descriptions

Field	Description
16:31 HIGHTDRAGN	High Block Tamper Region Assignment This field affects flash memory blocks in High address space. See the chip-specific information for the mapping between field bits and flash blocks. The same mapping also applies to control similar functions in the embedded flash memory module. 0 Corresponding flash blocks are not assigned to the TDR*n* 1 Corresponding flash blocks are assigned to the TDR*n*

93.4.9 Tamper Region Assignment DCF client (TDR*n*_LOCK2)

This DCF client holds the assignment information of the flash blocks to a specific TDR. Each flash block can be assigned to any of the TDR (*n* varies from zero to 5). This DCF Client however holds the assignment information for only those blocks specified in the LOCK2 register in the flash memory. Therefore, the diagram of this DCF Client mirrors the LOCK2 register in the flash memory. *Figure 2448* depicts the TDR*n*_LOCK2 DCF client.

Caution: This DCF client is writable from 0 to 1 only. Attempted writes of bits from 1 to 0 will be ignored, making the assignment of a flash block to a TDR irreversible unless the entire TDR is overridden.

DCF client Address: 0x14 + ((n * 4) + 2) Access: DCF write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R																
W	256KBTDRAGN[31:16]															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R																
W	256KBTDRAGN[15:0]															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 2448. Tamper Region Assignment DCF client (TDR*n*_LOCK2)

Table 2334. TDR*n*_LOCK2 field descriptions

Field	Description
0:31 256KBTDRAGN	256 KB Block Tamper Region Assignment (256KBTDRAGN[31:0]) This field affects flash memory blocks in 256 KB address space. See the chip-specific information for the mapping between field bits and flash blocks. The same mapping also applies to control similar functions in the embedded flash memory module. 0 Corresponding flash blocks are not assigned to the TDR*n* 1 Corresponding flash blocks are assigned to the TDR*n*

93.4.10 Tamper Region Assignment DCF client (TDR*n*_LOCK3)

This DCF Clients holds the assignment information of the flash blocks to a specific TDR. Each flash block can be assigned to any of the TDR (*n* varies from zero to 5). This DCF Clients however holds the assignment information for only those blocks specified in the LOCK3 register in the flash memory. Therefore, the diagram of this DCF Clients mirrors the LOCK3 register in the flash memory. *Figure 2449* depicts the TDRn_LOCK3 DCF client.

Caution: This DCF client is writable from 0 to 1 only. Attempted writes of bits from 1 to 0 will be ignored, making the assignment of a flash block to a TDR irreversible unless the entire TDR is overridden.

DCF client Address: 0x14 + ((n * 4) + 3) Access: DCF write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R																
W	256KBTDRAGN[47:32]															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 2449. Tamper Region Assignment DCF client (TDR*n*_LOCK3)

Table 2335. TDR*n*_LOCK3 field descriptions

Field	Description
16:31 256KBTDRAGN	256 KB Block Tamper Region Assignment (256KBTDRAGN[47:32]) This field affects flash memory blocks in 256 KB address space. See the chip-specific information for the mapping between field bits and flash blocks. The same mapping also applies to control similar functions in the embedded flash memory module. 0 Corresponding flash blocks are not assigned to the TDR*n* 1 Corresponding flash blocks are assigned to the TDR*n*

93.5 Memory map and registers

Note: The Tamper Detection Module's memory-mapped registers are read-only. Any attempted write generates a transfer error.

Table 2336. TDM memory map

Address offset (hex)	Register name	Location
0x00	TDR Status Register (TDM_TDRSR)	*Section 93.5.1*
0x04	Last Flash Programed Address Register (TDM_LFPAR)	*Section 93.5.2*
0x08	Diary Base Address (TDM_DBA)	*Section 93.5.3*
0x10–0x24	Software Tamper Override Key Region (TDM_STO_KEY0–TDM_STO_KEY5)	*Section 93.5.4*

93.5.1 TDR Status Register (TDM_TDRSR)

This register contains the current status, such as locked or unlocked, of each tamper region, such as TDR.

- If a tamper region is locked, the blocks assigned to that tamper region cannot be erased.
- If a region is unlocked, the blocks within the tamper region can be erased.

The register contains a status bit for each TDR. Status is determined as follows:

1. The default status for each TDR is ‘1’, which indicates that the erase operation is disabled for all flash blocks associated with the TDR.
2. The status bit for a TDR is set to ‘0’ (indicating that the erase operation for all flash blocks associated with the TDR is enabled) if either a successful programming operation to the TDR diary region has been performed or the TDR Override bit for that TDR is set.

Note: TDRSR is a read-only register and writes have no effect. Write operations result in a transfer error.

If the reset value for this register is determined by a DCF record, it will be noted in the chip-specific information at the beginning of this chapter. Otherwise, the reset value is as shown.

Offset: 0x00 Access: User read

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	TDRSR5	TDRSR4	TDRSR3	TDRSR2	TDRSR1	TDRSR0
W																
Reset	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1

Figure 2450. TDR Status and Diary Override Register (TDM_TDRSR)

Table 2337. TDM_TDRSR field descriptions

Field	Description
26 TDRSR5	Represents the status of TDR 5 in conjunction with the Diary Override Enable status. 0 TDR is unlocked. Blocks assigned to this TDR are NOT blocked for erasing by the TDM module. This unlock state is achieved by a successful programming operation to the correct diary region or this TDR has been permanently unlocked by a DCF record that writes to the Tamper Region Override DCF register or a successful software override operation. 1 TDR is locked. Erase of blocks assigned to this TDR is not possible.
27 TDRSR4	Represents the status of TDR 4 in conjunction with the Diary Override Enable status. 0 TDR is unlocked. Blocks assigned to this TDR are NOT blocked for erasing by the TDM module. This unlock state is achieved by a successful programming operation to the correct diary region or this TDR has been permanently unlocked by a DCF record that writes to the Tamper Region Override DCF register or a successful software override operation. 1 TDR is locked. Erase of blocks assigned to this TDR is not possible.
28 TDRSR3	Represents the status of TDR 3 in conjunction with the Diary Override Enable status. 0 TDR is unlocked. Blocks assigned to this TDR are NOT blocked for erasing by the TDM module. This unlock state is achieved by a successful programming operation to the correct diary region or this TDR has been permanently unlocked by a DCF record that writes to the Tamper Region Override DCF register or a successful software override operation. 1 TDR is locked. Erase of blocks assigned to this TDR is not possible.

Table 2337. TDM_TDRSR field descriptions (continued)

Field	Description
29 TDRSR2	Represents the status of TDR 2 in conjunction with the Diary Override Enable status. 0 TDR is unlocked. Blocks assigned to this TDR are NOT blocked for erasing by the TDM module. This unlock state is achieved by a successful programming operation to the correct diary region or this TDR has been permanently unlocked by a DCF record that writes to the Tamper Region Override DCF register or a successful software override operation. 1 TDR is locked. Erase of blocks assigned to this TDR is not possible.
30 TDRSR1	Represents the status of TDR 1 in conjunction with the Diary Override Enable status. 0 TDR is unlocked. Blocks assigned to this TDR are NOT blocked for erasing by the TDM module. This unlock state is achieved by a successful programming operation to the correct diary region or this TDR has been permanently unlocked by a DCF record that writes to the Tamper Region Override DCF register or a successful software override operation. 1 TDR is locked. Erase of blocks assigned to this TDR is not possible.
31 TDRSR0	Represents the status of TDR 0 in conjunction with the Diary Override Enable status. 0 TDR is unlocked. Blocks assigned to this TDR are NOT blocked for erasing by the TDM module. This unlock state is achieved by a successful programming operation to the correct diary region or this TDR has been permanently unlocked by a DCF record that writes to the Tamper Region Override DCF register or a successful software override operation. 1 TDR is locked. Erase of blocks assigned to this TDR is not possible.

93.5.2 Last Flash Programed Address Register (TDM_LFPAR)

This register holds information of the address of the last successful programming operation provided from the flash memory to the TDM. Writes to this register are ignored and will generate a transfer error.

Offset: 0x04 Access: User read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	LFPAR							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	LFPAR															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 2451. Last Flash Programed Address Register (TDM_LFPAR)

Table 2338. TDM_LFPAR field descriptions

Field	Description
8:31 LFPAR	Last Flash Programed Address These bits represent the address of the last successful programming operation.

93.5.3 Diary Base Address (TDM_DBA)

This register holds base address information of the diary. Any write to this register is ignored and generates a transfer error.

Offset: 0x08 Access: User read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	DBA															
W																
Reset	1(1)	1(1)	1(1)	1(1)	1(1)	1(1)	1(1)	1(1)	1(1)	1(1)	1(1)	1(1)	1(1)	1(1)	1(1)	1(1)

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	DBA															
W																
Reset	1(1)	1(1)	1(1)	1(1)	1(1)	1(1)	1(1)	1(1)	1(1)	1(1)	1(1)	1(1)	1(1)	1(1)	1(1)	1(1)

Figure 2452. Diary Base Address Register (TDM_DBA)

1. Reset value depends on the Diary Base Address (DBA) DCF client

Table 2339. TDM_DBA field descriptions

Field	Description
0:31 DBA	Diary Base Address These bits represent the address of the diary location in the flash. **Note:** The diary base address is established via the DBA DCF record. The reset value of this register is determined by that record.

93.5.4 Software Tamper Override Key Region (TDM_STO_KEY*n*)

The STO_KEYn register holds the service key for overriding the Tamper Detect Region *n*, where *n* denotes the Tamper Detect Region number.

Caution: The STO_KEY*n* registers can only be written once.
The associated DCF client is writable from 0 to 1 only. Attempted writes of bits from 1 to 0 are ignored.

Offset: 0x10 + *n* * 0x4 (*n* = 0 to 5) Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	STO_KEY															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	STO_KEY															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 2453. Software Tamper Override Key Region (TDM_STO_KEY*n*)

Table 2340. TDM_STO_KEY*n* field descriptions

Field	Description
0:31 STO_KEY	Software Tamper Override Key Diary *n* This field holds the service key value for overriding Tamper Detect Region n. By loading STO_KEYn with the correct signature of 0x55AA5A5A, Tamper Detect Region n is overridden, and the flash blocks assigned to Tamper Detect Region n are open for erase. This field has no effect when STO_DISn field of STO_DIS DCF client is set. **Note:** The STO_KEYn registers can only be written once. **Note:** Refer to *Section 65.3.7: TDM Diary Operation* for programming details

93.6 Functional description

The primary purpose of the TDM is to provide a Tamper Detection mechanism by enabling an Erase Counter (or diary) for regions within the flash. These regions are called Tamper Detection Regions. The following sections cover the entire functionality of this module.

93.6.1 Flash erase counter and tamper detection

A flash region block or part of it is assigned for implementing the diary. The base address for the Dairy is defined by a DCF Record .

The diary is divided into 6 parts where each part consists of 256 × 128-bit records, to record until 256 write attempts on the flash blocks assigned to each Tamper Detection Region. For correct operation, the Diary should reside in a flash block that is assigned as OTP.

Each flash block can be assigned to any of the TDRs. Related to each TDR is a group of 4 × 32-bit internal hardware registers that are initialized by DCF records that define the tamper region . These registers mirror exactly the LOCK registers bits from the flash module.

Each TDRn_LOCK*m* register is loaded at boot time by the SSCM , which defines if the associated flash block is Locked for Erase in a particular TDR:

- ’1’ means the flash block is locked for erase
- ’0’ means the flash block is not locked for erase

Before performing an erase in one of the blocks locked for erase and associated to a TDR, a program operation has to be performed on the diary part for that TDR.As the diary should be OTP, successive writes are allowed on unprogramed double words only. This ensure that diary records can only increase and that data recorded in the diary is not modifiable by software.

At the end of a successful double word program operation, the flash module indicates to the TDM the address of the programed double word. TDM determines whether the programed data was written to the correct TDR Diary Location and then unlocks the blocks for erase operation for the corresponding TDR. TDM ensures that this erase operation on unlocked blocks is allowed until the next program operation on any block, other than Diary, is performed with program completion interrupt enabled or till next hard reset.

Note: All blocks associated to a TDR where an erase operation is allowed can be erased all together.

TDM has no control of how much data a diary has loaded. It is a responsibility for the user to verify if the diary is full. There is also no restriction to what type of data can be programed in

the diary. If diary is detected to be full, TDM provides a means to disable the tamper detect mechanism so that erase operations can be performed to a TDR without any diary entry. In this case, the user is required to program the Tamper Region Override Register for the corresponding TDR.

The data programed in the diary location is flexible, and may include count information or other important customer information.

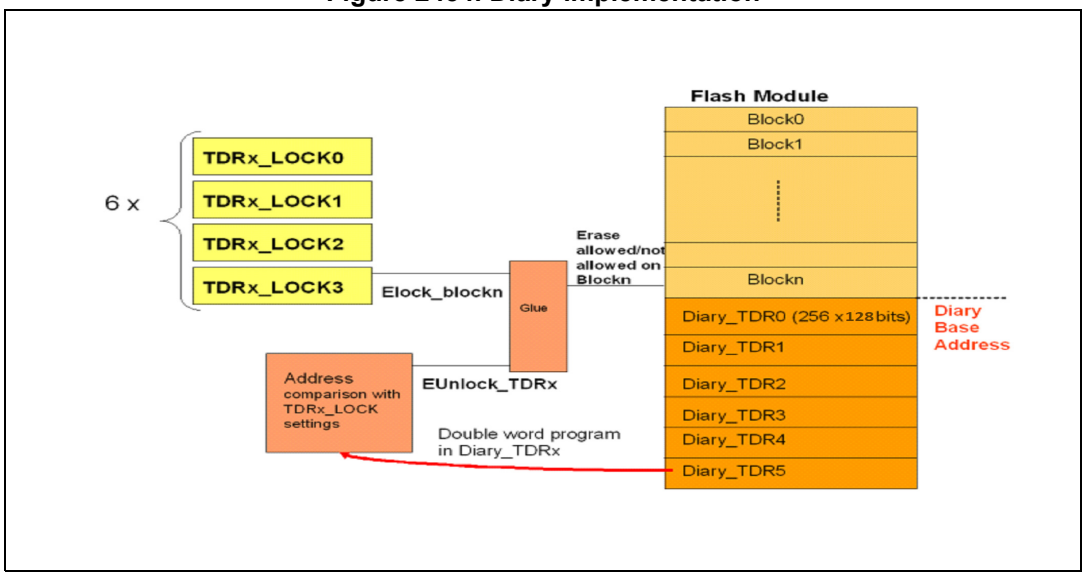
Software has to check whether the diary is full. Once the diary area for a TDR is full, the software has to program a DCF record for overriding blocking of erase from TDR, disabling tamper detect, so that erases can be done to that TDR without any diary entry

The setting from “override tamper region lock” DCF record (refer to *Section 93.6.3: Diary*) has priority over the settings in the TDRn_LOCK0-3 registers. In other words, if a TDR is unlocked for erase by the override tamper region lock, the TDR*n*_LOCK*m* bits have no effect on the TDR that has been overridden, but the tamper detection remains active for all TDRs not having the override bit set.

Moreover, the settings from PASS_LOCK*x*_PG*n* registers have priority over TDRn_LOCK*m* settings. For example, if a block is password-protected for write operations, the TDR lock setting is ignored.

A schematic representation of the diary is depicted in *Figure 2454*.

Figure 2454. Diary implementation



Note: The settings in the Tamper Region Override Register have priority over the TDRn_LOCKm Registers. The SSCM module also has priority over the settings done in the TDM. For instance, if blocks in a TDR are unlocked for erase operation via TDM, but the same blocks are programed to be locked for erase operation in SSCM, the blocks end up being locked for erase operation.

93.6.2 Assigning blocks to Tamper Detect Regions (TDR)

Each TDR is defined by a group of 4 × 32-bit DCF records and clients, with each containing bits that map to specific blocks in a flash region. These mappings mirror exactly the LOCK registers bits from the flash module. Hence, the block assignment registers are named as

TDRn_LOCKm where n is the number of a tamper region, that is, 0-5 and m varies from 0-3 as flash contains four Lock registers.

In these DCF records:

- ’1’ means that the corresponding flash block is assigned to this TDR
- ’0’ means that the corresponding flash block is not assigned to this TDR (Default)

DCF clients are not visible to software. They cannot be read nor written. They can only be initialized by DCF record within the UTEST flash block during reset. The value of these registers can be evaluated by reading all the UTest DCF records and searching for records that correspond to these clients.

Note: No flash block should be assigned to more than one TDR. Each block should correspond to a single TDR. For example, if a single block is assigned to TDR0 and TDR1, then even if the Diary is programed for both TDR0 and TDR1, this flash block will never be unlocked for erase operation.

93.6.3 Diary

The diary is a region of the flash memory where records of block erases for each TDR are stored. The format and size of the records are not fixed. The only requirements are that each diary has a maximum size of 4 KB, and the minimum size of any programming operation depends on the Flash sector, which is 8 bytes for Data sector and 16 bytes for the remaining sector to honor OTP restrictions. Therefore, each diary can hold a maximum of two-hundred and fifty-six 128-bit entries.

A flash region block or part of it must be assigned for implementing the diary. The base address for the diary is determined by a value written in a DCF record. The diary can be placed within any flash block in flash array, and history records in it can be read by any core.

Note: To maintain the security of the TDR, the block where the diary is placed must be assigned as OTP within the OTP registers.

The base address of the diary is implemented using a DCF Client. The DCF client is only writable once during reset and the default bit state is 1.

The DCF clients cannot be read nor written by software. They can only be initialized by DCF record from within UTEST block during reset. The value of these registers can be evaluated by reading all the UTEST DCF records and searching for records that write these DCF clients.

93.6.4 Specifying the diary base address

The diary is a region of flash memory where records of block erases for each TDR are stored. The format and size of these records is defined by the customer. The only hardware constraints are that each diary has a maximum size of 4 KB and the minimum size of any programming operation depends on the Flash sector, which is 8 bytes for Data sector and 16 bytes for the remaining sector.

The diary can be placed within any flash block in the flash array. To maintain the security of the TDR, the block where the diary is placed, must be assigned as OTP within the OTP registers, while in customer delivery or OEM production life cycle states.

The base address of the diary is implemented with a 20-bit DCF client that can be initialized by the SSCM during reset. The DCF client is writable once during reset and the default bit state is ‘0’.

The base address register is not visible to software. It cannot be read nor written. It can only be initialized by DCF record within UTest flash during reset. The value of this register can be evaluated by reading all the UTest flash DCF Records and searching for record that writes this register.

Each TDR is assigned a 4 KB section of the diary. With 6 TDRs, 24 KB of diary space is required.

The diary base address must be at a 16 KB boundary. Therefore the least significant 14 bits of the base address must all be ‘0’.

The Base address of the diary is represented by:

Diary_Base_Address = 0bxxxx_xxxx_xxxx_xxxx_xxxx_0000_0000_0000

The base address of each TDR is shown below:

- TDR0 = Diary_Base_Address + 0b000_0000_0000_0000
- TDR1 = Diary_Base_Address + 0b001_0000_0000_0000
- TDR2 = Diary_Base_Address + 0b010_0000_0000_0000
- TDR3 = Diary_Base_Address + 0b011_1000_0000_0000
- TDR4 = Diary_Base_Address + 0b100_0000_0000_0000
- TDR5 = Diary_Base_Address + 0b101_1000_0000_0000
- End of the diary = Diary_Base_Address + 0b110_0000_0000_0000

93.6.5 Diary base address verification

The status of whether a TDR is locked for erase is maintained by the TDM. This is accomplished by implementing a register with as many bits as the number of TDRs. The default state of these bits is 1, which is defined as “TDR Erase Blocked”. Refer to *Section 93.5.1: TDR Status Register (TDM_TDRSR)* for further details about this register.

When the flash memory module has completed a programming operation, the address of the location programed is output from the flash module. The TDM in turn compares that address to determine if it falls within the range of any TDR diary regions. If it is within any TDR, the corresponding status bit is set to 0 meaning “TDR Erase Enabled”.

These status bits are all reset to 1 when the flash module signals an erase complete by asserting the PEC (Program Erase Complete) signal.

The state of the 6 status bits can be read through the TDRSR. Refer to *Section 93.5.1: TDR Status Register (TDM_TDRSR)*. The address of the last successful programming operation provided from the flash memory to the TDM can be read through the LFPAR. Refer to *Section 93.5.2: Last Flash Programed Address Register (TDM_LFPAR)*.

93.6.5.1 Diary management

Before a block protected by a tamper detect region can be erased, a record must be written to the associated diary. Software must search through the diary to find the end of the records, where the next available diary location can be written. After a maximum of 256 records are programed to the diary, a TDR diary is full. Full means that at least 1 bit has been programed to ‘0’ in each record within the 4 KB diary.

If software determines that a particular TDR diary is full it may override the diary to allow erasing to continue or it may choose to stop further erases and to flag an error.

Diary Override bits are implemented with a DCF client consisting of 6 bits, one for each TDR. See the device configuration information at the beginning of this chapter for TDM DCF client details.

The Diary Override bits are by default set to ‘0’; which means the diary must be written before an erase can occur. The DCF client can be written many times but the bits can only be written to ‘1’. Writes to ‘0’ are ignored. In this way, by default, all Tamper Detect Regions are active. The Tamper detect region diaries can be individually overridden by writing a DCF record that sets that corresponding bit to ‘1’. In this mode, the TDR status bit is permanently set to 0 to allow flash erase.

93.6.6 One Time Programmable (OTP)

93.6.6.1 Overview

Any flash block within the flash array can be assigned, at any time, to be OTP. Once a flash block is assigned OTP, it cannot be changed back.

OTP means that flash erase of the entire block is disabled and that only 64-bit double words that are already erased (that is, flash content is 0xFFFF_FFFF_FFFF_FFFF) can be programed. Over-programming is not possible.

Flash blocks are assigned as OTP by writing a DCF record. The block will become OTP after the next reset.

94 ASC Modeling Unit version 2 (AMU2)

94.1 Introduction

The Advanced Simulation for Calibration (ASC) Modeling Unit (AMU) is a high performance floating-point arithmetic calculation engine implementing a specific pre-defined algorithm (see *Section 94.3.1: AMU2 algorithm* for details). This is a second generation implementation.

This computation unit is used to describe unknown systems using a model that is based on measured data. The model is based on mathematical approximations such that no physical description is necessary. Additionally, this approach minimizes the required number of measurements of the system through the use of a statistically optimal distribution.

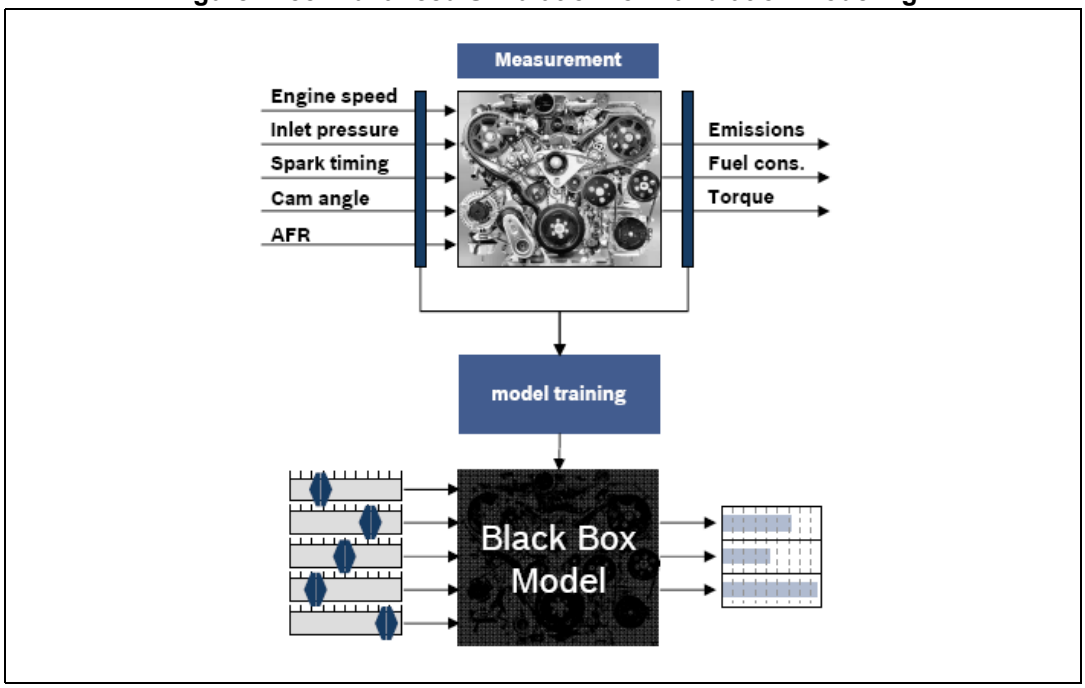
There are various modeling algorithms typically used for this type of system analysis. Examples include classic polynomial descriptions and neural networks. The statistical machine learning method implemented by the AMU offers a number of significant improvements versus previous approaches, including:

- Ability to describe complex behaviors with very high accuracy
- No model parameterization is necessary
- Robust against measurement errors and data “outliers”
- Provides best trade-off between measurement number and model accuracy

In the past, a significant drawback to this approach was the high computational power necessary for this type of modeling. The AMU, which is a compact hardware implementation of a specific algorithm, provides the needed calculation capabilities within the context of a general purpose microcontroller.

See *Figure 2455* for a simplified diagram of the intended use of the ASC Modeling Unit.

Figure 2455. Advanced Simulation for Calibration Modeling



Note the module names AMU and AMU2 are meant to be completely synonymous and are used here interchangeably.

94.1.1 Overview

The AMU2 implementation includes the pre-defined floating-point algorithm that includes:

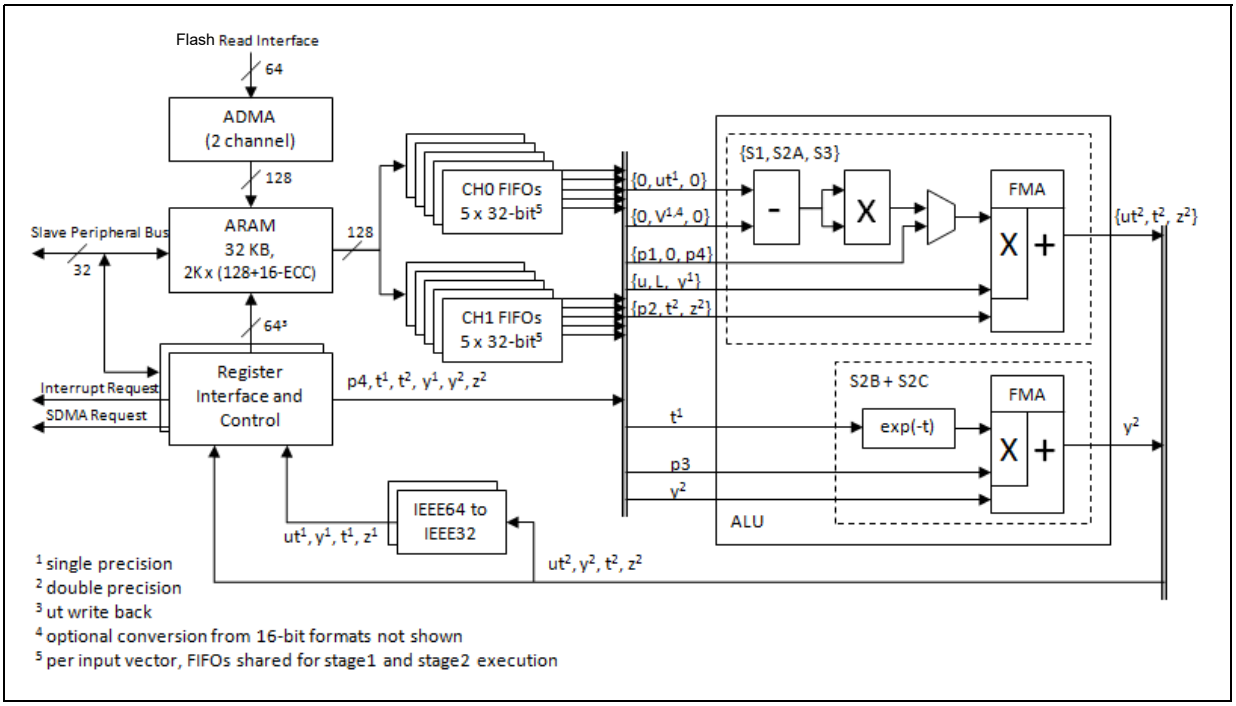
- Multiple loop constructs with extensive vectored multiply-accumulate operations
- Single-precision exponential function, exp(x)
- 32 Kbytes of memory-mapped local RAM for parameter and coefficient storage
- Local DMA (ADMA) with private connection into system Flash for fetching data into the local memory
- Local RAM (ARAM) accessible via local DMA and system bus interface

The AMU contains two separate channels, which are timing-independent and register-independent. Each channel logically includes a calculation engine and a DMA channel. The calculation engines both share the same arithmetic logic using a time multiplexed scheme. Likewise, the DMA channels use a time multiplexed scheme to share common interfaces for fetching from the system Flash memory (or the calibration remapped destination memories, or both) and writing into the local RAM.

Within each calculation engine, there are two pipelined floating-point engines: one performs the stage 1, 2a and 3 calculations using a subtracter, a multiplier and an FMA (fused multiply-add), while the second performs the stage 2b and 2c operations using the exp(x) function and an FMA. This structure was selected to allow stage2a to be computed continuously in one pipeline, while stages 2b and 2c are computed in parallel in the other pipeline to maximize overall performance. Operations from the two channels are interleaved on the calculation engines and also include a number of blocks to feed these arithmetic blocks with data. The interleaving of the channels is such that operations on one channel do not influence the timing on the other channel. This includes a 128-bit read interface from the ARAM to dedicated input data FIFOs to source data to the ALU.

A block diagram of the AMU2 is shown in *Figure 2456*. The module consists of a local 32 KB RAM with a SEC-DED Error Correcting Code (ECC) organized on 64-bit doublewords, that is, 8 check bits per 64 bits of data, 2 DMA channels to support data transfers from Flash to the AMU local RAM, dedicated data FIFOs for input data and the hardwired data path to calculate the AMU2 function. The AMU local RAM is also visible over the peripheral bus to the system.

Figure 2456. AMU2 block diagram(36)



The AMU2's system interfaces and local RAM, along with the register interface and control, are shown on the left side, and the two floating-point ALU pipelines on the right side of *Figure 2456*. The top pipeline calculates the results from stages 1, 2a and 3 using a generic function of the form:

result = ((in1 - in2) * (in1 - in2) + in3) * in4 + in5

where the inputs are simply labeled in1, in2, …, in5 from top to bottom. As shown in *Table 2341*, a different input operand set is sent into this pipeline structure for each stage, producing the required result calculations.

Table 2341. Stage 1, 2a, 3 Input Operands and Results

stage	in1	in2	in3	in4	in5	result, ((in1 - in2)*(in1 - in2) + in3) * in4 + in5
1	0	0	p1[*]	u[*]	p2[*]	p1[*] * u[*] + p2[*]
2a	ut[*]	V[*]	0	L[*]	t	((ut[*] - V[*]) * (ut[*] - V[*])) * L[*] + t
3	0	0	p4	y	z	p4 * y + z

For the stage 2b + 2c pipeline, the single input operand set produces the required calculation:

(exp(-t) * p3[*]) + y

36. Refer to the device block diagram for the AMU2 connection to the Flash Controler

94.1.2 Features

- High performance floating-point calculation of pre-defined algorithm with extensive vectored multiply accumulate operations and an exp(x) evaluation
- Time multiplexed hardware structure for the calculation engine to minimize die size while still maintaining execution speed
- Two independent calculation channels
  - The two independent channels allow two concurrent processes to use the AMU to perform calculations; no interference between the two processes is guaranteed by the hardware.
- Two independent AMU DMA (ADMA) channels
  - The ADMA channels are high-bandwidth 64-bit direct memory access channels, supporting transfer of Flash (or remapped calibration) data to local AMU RAM
- 32 KB of local AMU RAM (ARAM) memory storage
  - Logically dual ported
  - Connected to system peripheral bus, ADMA channels and internal AMU calculation engines
- Programming model located in peripheral address space of system memory map
  - Includes ARAM memory space and registers
  - Optimized for system DMA result & save, data reload and calculation initiation
- Support for multiple use cases involving system DMA or local ADMA, or both

94.1.3 Modes of operation

The AMU2 module does *not* support any special modes of operation. As a memory-mapped device located on the peripheral bus as a slave module, the AMU2 responds strictly based on accesses to its programming model via the peripheral bridge bus controller. The module generates a number of control output signals that can be configured to generate processor core interrupts (to either the main core(s) or I/O Processor) or system DMA requests in response to changes in the AMU2's calculation state.

The module optionally supports the ability to run or stall the calculation engine if a processor core is halted due to a debug event; for the AMU2, this function is programmable. By default, the AMU2 stalls in response to a debug event as signaled by a module input signal. See the AMU2_CCR1CH*n*[DDSL] description in *Section 94.2.3.21: AMU Control & Configuration 1 Channel n Register (AMU2_CCR1CHn)* for more details.

The functionality associated with the AMU2 resides in two clock domains: the module operates at the frequency of the main cores while its peripheral bus interface to the programming model operates at the same speed as that domain (typically 0.5x the speed of the main cores).

94.2 Memory map and register definition

The AMU module supports its programming model mapped to standard 16 KB on-platform peripheral slots. The local AMU RAM is accessible for read and write via the slave peripheral bus interface and is mapped to four contiguous and properly aligned 16 KB peripheral slots for a maximum size of 64 KB. The programming model, including the ARAM, can only be referenced in supervisor mode using 32-bit (word) accesses, either as single or burst transfers. It implements a big-endian memory organization. Attempted

references while in user mode, using different access sizes, writing to a read-only address, or to *reserved* addresses generate an error termination. It should also be noted that byte and halfword-sized accesses can be emulated using the Decorated Storage Memory Controller to perform bit field inserts for writes, and simple processor loads with shift or masking operations for reads, or both. Unless noted otherwise, *unused* addresses are treated as RAZ/WI: read-as zero, writes ignored. There are specific unused addresses that are located in read-only space and are therefore marked as “RAZ”; attempted writes to these locations generate an error termination.All programming model register reads and writes complete in a single peripheral bus clock cycle.

94.2.1 AMU memory map

Table 2342 shows the memory map for the AMU module.This address space is partitioned into 3 regions: local ARAM, channel 0 registers and channel 1 registers.

Table 2342. AMU memory map

Address Offset	Register	Access	Reset Value	Section
Local ARAM Address Space				
0x00000–0x07BFF	Direct access to Local RAM (ARAM)	RW	0x0000_0000	—
0x07C00–0x07FFF	ARAM storage reserved for ALU (UT[*], streaming P3[*] and V[*])	RW	0x0000_0000	*Section 94.2.2*
0x08000–0x0FFFF	Reserved	None	—	—
Channel 0 Register Address Space				
0x10000	AMU ID & Version Channel 0 Register (IDVCH0)	R	0x0000_0002	*Section 94.2.3.1*
0x1000C	AMU N crossing Channel 0 Register (NCH0)	RW	0x0000_0000	*Section 94.2.3.2*
0x10010	AMU P1 Pointer Channel 0 Register (P1PTRCH0)	RW	0x0000_0000	*Section 94.2.3.3*
0x10014	AMU P2 Pointer Channel 0 Register (P2PTRCH0)	RW	0x0000_0000	*Section 94.2.3.4*
0x10018	AMU P3 Pointer Channel 0 Register (P3PTRCH0)	RW	0x0000_0000	*Section 94.2.3.5*
0x1001C	AMU P4 Data Channel 0 Register (P4CH0)	RW	0x0000_0000	*Section 94.2.3.6*
0x10020	AMU P5 Data Channel 0 Register (P5CH0)	RW	0x0000_0000	*Section 94.2.3.7*
0x10024	AMU P6 Data Channel 0 Register (P6CH0)	RW	0x0000_0000	*Section 94.2.3.8*
0x10028	AMU P7 Data Channel 0 Register (P7CH0)	RW	0x0000_0000	*Section 94.2.3.9*
0x1002C	AMU P8 Data Channel 0 Register (P8CH0)	RW	0x0000_0000	*Section 94.2.3.10*
0x10030	AMU P9 Most Significant Data Channel 0 Register (P9MCH0)	RW	0x0000_0000	*Section 94.2.3.11*
0x10034	AMU P9 Least Significant Data Channel 0 Register (P9LCH0)	RW	0x0000_0000	*Section 94.2.3.12*
0x10038	AMU V Pointer Channel 0 Register (VPTRCH0)	RW	0x0000_0000	*Section 94.2.3.13*
0x1003C	AMU L Pointer Channel 0 Register (LPTRCH0)	RW	0x0000_0000	*Section 94.2.3.14*
0x10040	AMU U Pointer Channel 0 Register (UPTRCH0)	RW	0x0000_0000	*Section 94.2.3.15*

Table 2342. AMU memory map (continued)

Address Offset	Register	Access	Reset Value	Section
0x10044	AMU DMA Source Address Channel 0 Register (DSACH0)	RW	0x0000_0000	*Section 94.2.3.16*
0x10048	AMU DMA Destination Address Channel 0 Register (DDACH0)	RW	0x0000_0000	*Section 94.2.3.17*
0x1004C	AMU DMA Transfer Count Channel 0 Register (DCNTCH0)	RW	0x0000_0000	*Section 94.2.3.18*
0x10050	AMU DMA Expected Checksum Channel 0 Register (DCKSCH0)	RW	0x0000_0000	*Section 94.2.3.19*
0x10054	AMU DMA Control & Configuration Channel 0 Register (DCCRCH0)	RW	0x0000_0000	*Section 94.2.3.21*
0x10058	AMU Control & Configuration 1 Channel 0 Register (CCR1CH0)	RW	0x0000_0000	*Section 94.2.3.21*
0x1005C	AMU Control & Configuration 2 Channel 0 Register (CCR2CH0)	RW	0x0000_0000	*Section 94.2.3.22*
0x10080	AMU Result Most Significant Data Channel 0 Register (RES64MCH0)	R	0x0000_0000	*Section 94.2.3.23*
0x10084	AMU Result Least Significant Data Channel 0 Register (RES64LCH0)	R	0x0000_0000	*Section 94.2.3.24*
0x10088	Status, Error and Interrupt Channel 0 Register (SEIRCH0)	R	0x0000_0000	*Section 94.2.3.25*
0x1008C	Actual DMA Checksum Channel 0 Register (ACKSCH0)	R	0x0000_0000	*Section 94.2.3.26*
0x10090	32-bit Result Channel 0 Register (RES32CH0)	R	0x0000_0000	*Section 94.2.3.27*
0x10094	Aliased to SEIRCH0 Status, Error and Interrupt Channel 0 Register (ASEIRCH0)	R	0x0000_0000	*Section 94.2.3.28*
0x10098	Aliased to ACKSCH0 Actual DMA Checksum Channel 0 Register (AACKSCH0)	R	0x0000_0000	*Section 94.2.3.29*
0x100A0	Y Result Most Significant Data Channel 0 Register (Y64MCH0)	R	0x0000_0000	*Section 94.2.3.30*
0x100A4	Y Result Least Significant Data Channel 0 Register (Y64LCH0)	R	0x0000_0000	*Section 94.2.3.31*
0x100A8	Aliased to P8CH0 AMU P8 Data Channel 0 Register (AP8CH0)	RW	0x0000_0000	*Section 94.2.3.32*
0x100AC	Status, Error and Interrupt (No Clear) Channel 0 Register (SEINCCH0)	R	0x0000_0000	*Section 94.2.3.33*
0x100B0	32-bit Y Result Channel 0 Register (Y32CH0)	R	0x0000_0000	*Section 94.2.3.34*
Channel 1 Register Address Space				
0x10100	AMU ID & Version Channel 1 Register (IDVCH1)	R	0x0001_0002	*Section 94.2.3.1*
0x1010C	AMU N crossing Channel 1 Register (NCH1)	RW	0x0000_0000	*Section 94.2.3.2*
0x10110	AMU P1 Pointer Channel 1 Register (P1PTRCH1)	RW	0x0000_0000	*Section 94.2.3.3*

Table 2342. AMU memory map (continued)

Address Offset	Register	Access	Reset Value	Section
0x10114	AMU P2 Pointer Channel 1 Register (P2PTRCH1)	RW	0x0000_0000	*Section 94.2.3.4*
0x10118	AMU P3 Pointer Channel 1 Register (P3PTRCH1)	RW	0x0000_0000	*Section 94.2.3.5*
0x1011C	AMU P4 Data Channel 1 Register (P4CH1)	RW	0x0000_0000	*Section 94.2.3.6*
0x10120	AMU P5 Data Channel 1 Register (P5CH1)	RW	0x0000_0000	*Section 94.2.3.7*
0x10124	AMU P6 Data Channel 1 Register (P6CH1)	RW	0x0000_0000	*Section 94.2.3.8*
0x10128	AMU P7 Data Channel 1 Register (P7CH1)	RW	0x0000_0000	*Section 94.2.3.9*
0x1012C	AMU P8 Data Channel 1 Register (P8CH1)	RW	0x0000_0000	*Section 94.2.3.10*
0x10130	AMU P9 Most Significant Data Channel 1 Register (P9MCH1)	RW	0x0000_0000	*Section 94.2.3.11*
0x10134	AMU P9 Least Significant Data Channel 1 Register (P9LCH1)	RW	0x0000_0000	*Section 94.2.3.12*
0x10138	AMU V Pointer Channel 1 Register (VPTRCH1)	RW	0x0000_0000	*Section 94.2.3.13*
0x1013C	AMU L Pointer Channel 1 Register (LPTRCH1)	RW	0x0000_0000	*Section 94.2.3.14*
0x10140	AMU U Pointer Channel 1 Register (UPTRCH1)	RW	0x0000_0000	*Section 94.2.3.15*
0x10144	AMU DMA Source Address Channel 1 Register (DSACH1)	RW	0x0000_0000	*Section 94.2.3.16*
0x10148	AMU DMA Destination Address Channel 1 Register (DDACH1)	RW	0x0000_0000	*Section 94.2.3.17*
0x1014C	AMU DMA Transfer Count Channel 1 Register (DCNTCH1)	RW	0x0000_0000	*Section 94.2.3.18*
0x10150	AMU DMA Expected Checksum Channel 1 Register (DCKSCH1)	RW	0x0000_0000	*Section 94.2.3.19*
0x10154	AMU DMA Control & Configuration Channel 1 Register (DCCRCH1)	RW	0x0000_0000	*Section 94.2.3.20*
0x10158	AMU Control & Configuration 1 Channel 1 Register (CCR1CH1)	RW	0x0000_0000	*Section 94.2.3.21*
0x1015C	AMU Control & Configuration 2 Channel 1 Register (CCR2CH1)	RW	0x0000_0000	*Section 94.2.3.22*
0x10180	AMU Result Most Significant Data Channel 1 Register (RES64MCH1)	R	0x0000_0000	*Section 94.2.3.23*
0x10184	AMU Result Least Significant Data Channel 1 Register (RES64LCH1)	R	0x0000_0000	*Section 94.2.3.24*
0x10188	Status, Error and Interrupt Channel 1 Register (SEIRCH1)	R	0x0000_0000	*Section 94.2.3.25*
0x1018C	Actual DMA Checksum Channel 1 Register (ACKSCH1)	R	0x0000_0000	*Section 94.2.3.26*
0x10190	32-bit Result Channel 1 Register (RES32CH1)	R	0x0000_0000	*Section 94.2.3.27*
0x10194	Aliased to SEIRCH1 Status, Error and Interrupt Channel 1 Register (ASEIRCH1)	R	0x0000_0000	*Section 94.2.3.28*

Table 2342. AMU memory map (continued)

Address Offset	Register	Access	Reset Value	Section
0x10198	Aliased to ACKSCH1 Actual DMA Checksum Channel 1 Register (AACKSCH1)	R	0x0000_0000	*Section 94.2.3.29*
0x101A0	Y Result Most Significant Data Channel 1 Register (Y64MCH1)	R	0x0000_0000	*Section 94.2.3.30*
0x101A4	Y Result Least Significant Data Channel 1 Register (Y64LCH1)	R	0x0000_0000	*Section 94.2.3.31*
0x101A8	Aliased to P8CH1 AMU P8 Data Channel 1 Register (AP8CH1)	RW	0x0000_0000	*Section 94.2.3.32*
0x101AC	Status, Error and Interrupt (No Clear) Channel 1 Register (SEINCCH1)	R	0x0000_0000	*Section 94.2.3.33*
0x101B0	32-bit Y Result Channel 1 Register (Y32CH1)	R	0x0000_0000	*Section 94.2.3.34*

94.2.2 ARAM memory map details

The 32 KB physical address space allocated to the AMU's local RAM (ARAM) is segmented into 2 regions. The first, a 31 KB space (offset addresses = 0x0_0000 - 0x0_7BFF), is intended for user-defined storage of the AMU input data. The second, a 1 KB space (offset addresses = 0x0_7C00 - 0x0_7FFF) is used by the AMU2 for temporary storage. This RAM space is used by the calculation engines to temporarily store the UT[*] vector generated as the stage1 output; additionally, it implements small FIFO memories to store "streamed data" associated with the V[*] and P3[*] vectors. The specific memory allocation within this 1 KB space is shown in *Table 2343*. Note if the streaming feature for the V[*] and P3[*] vectors is not enabled, the first 768 bytes of this 1 KB space can be used for general purpose ARAM data storage.

Table 2343. 1 KB ARAM Space Reserved for the Calculation Engines

Offset Start Address	Offset End Address	Size	Purpose
0x0_7C00	0x0_7CFF	256 bytes	Channel 0 Streaming v[*] FIFO
0x0_7D00	0x0_7DFF	256 bytes	Channel 1 Streaming v[*] FIFO
0x0_7E00	0x0_7E7F	128 bytes	Channel 0 Streaming p3[*] FIFO
0x0_7E80	0x0_7EFF	128 bytes	Channel 1 Streaming p3[*] FIFO
0x0_7F00	0x0_7F7F	128 bytes	Channel 0 ut[*] Storage
0x0_7F80	0x0_7FFF	128 bytes	Channel 1 ut[*] Storage

Given the 32 KB physical address space associated with the ARAM, all the pointers for this memory are implemented as 15-bit byte addresses that define the offset into the local memory. These local memory pointers are restricted to reference data items on their natural size boundaries, for example, with a 0-modulo-2 byte addresses (2-byte aligned) for 16-bit data, 0-modulo-4 byte addresses (4-byte aligned) for 32-bit data and 0-modulo-8 byte addresses (8-byte aligned) for 64-bit data. Additionally, software must manage the pointer values near the end of the address ranges as the hardware pointer calculations do *not* perform any special handling near the 31 KB or 32 KB boundaries.

Concerning the alignment of streamed V[*] or P3[*] data in their FIFOs, it should be noted that if the streamed data starts at offset 8 or 24 within the 32-byte Flash page, the first doubleword of data is stored at offset 8 in the associated FIFO, else it is stored at offset 0. During normal usage, this is completely transparent, but may be of interest during AMU debug if the contents of the FIFOs are examined.

If the ARAM is protected from single bit errors by enabling the ECC logic, it is the system's responsibility to initialize its contents with correct ECC before any AMU calculations are performed. This initialization can be performed by the system DMA performing 16-byte burst transfers to write the entire contents of the ARAM, using the memory initialization feature of the ADMA, or using the ADMA to load the entire memory from Flash. If the system DMA is used for ARAM initialization, both AMU2_CCR1CH{0,1}[EECC] configuration bits must be cleared until the initialization is complete.

Finally, it should be noted that the two 32-bit Control & Configuration Registers are treated in a special manner when configured as the last destination of an ADMA transfer. The AMU2_DCCRCH*n*[DARU] provides this specific configuration. For example, when configured with DARU = 0b01, the last 64-bit ADMA transfer associated with channel 0 writes to the CCR{1,2}CH0 registers to initiate an ALU calculation. Likewise, the last 64-bit ADMA transfer associated with channel 1 could write to the CCR{1,2}CH1 registers. This mechanism allows the basic data block transfer controlled by the ADMA to load new input data into the ARAM and then initiate a calculation by writing to the CCR{1,2}CH*n* registers.

94.2.3 Register descriptions

94.2.3.1 AMU ID and Version Channel *n* Register (AMU2_IDVCH*n*)

This read-only register provides information on the AMU channel number and revision.

Offset: 0x10000 + n * 0x100 (n = 0 to 1) Access: Supervisor Read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	CH
W																
Reset:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	*n*

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	VER			
W																
Reset:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0

Figure 2457. AMU ID and Version Channel *n* Register (AMU2_IDVCH*n*)

Table 2344. AMU2_IDVCH*n* field descriptions

Field	Description
15 CH	Channel number, *n* (*n* = 0 to 1) 0 Channel 0 1 Channel 1
28:31 VER	AMU version number This implementation is Version 2.

94.2.3.2 AMU N crossing Channel *n* Register (AMU2_NCH*n*)

This register provides a debug capability for the algorithm's stage2 calculation. If properly enabled (AMU2_CCR2CH*n*[ENCRS] = 1), when the outer loop in the stage2 calculation *completes* the iteration count defined by this register, the calculation engine stops (if AMU2_CCR2CH*n*[SNCRS] = 1) and optionally generates an interrupt request or system DMA request (based on AMU2_CCR1CH*n*[NCICFG]). If the calculation engine stops, the p8 outer loop count is NCRS+1.

The register also provides an optional bit masking capability so that multiple "N crossing" events can occur as the result of a single register write. As a limit, this mechanism could be used to stop the calculation after every iteration of the stage2 outer loop.

Offset: 0x1000C + n * 0x100 (n = 0 to 1) Access: Supervisor Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	NMSK															
Reset:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	NCRS															
Reset:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 2458. AMU N crossing Channel *n* Register (AMU2_NCH*n*)

Table 2345. AMU2_NCH*n* field descriptions

Field	Description
0:15 NMSK	N crossing Mask. This field provides the ability to mask individual bits in the N crossing comparison. If set, the bit is ignored in the equality comparison comparing the N crossing value and the outer loop iteration count from stage2 of the algorithm. if NMSK[i] == 0, then the corresponding bit in NCRS[i] is included in the comparison if NMSK[i] == 1, then the corresponding bit in NCRS[i] is excluded (masked) in the comparison
16:31 NCRS	N crossing value (unsigned integer). When enabled by AMU2_CCR2CH*n*[ENCRS], the stage2 calculation is stopped *after* the masked NCRS iterations of the outer loop. After the calculation is stopped, the p8 outer loop count contains an NCRS+1 value in this register. The intermediate value contained in the y64 register is consistent with this p8/NCRS value. Additionally, if the AMU2_CCR1CH*n*[NCICFG] is set to an appropriate value, an interrupt or a DMA request is triggered when the outer loop count reaches the masked NCRS value. The interrupt enable/disable control is independent of the AMU2_CCR2CH*n*[ENCRS] setting. The equality comparison is defined as: N crossing_event = ((p8[15:0] | NMSK[15:0]) == (NCRS[15:0] | NMSK[15:0])) where "|" is the boolean OR operator.

94.2.3.3 AMU P1 Pointer Channel *n* Register (AMU2_P1PTRCH*n*)

This register is a 4-byte aligned pointer into the ARAM defining the starting location for the 32-bit P1[*] input vector. This input vector is used in the stage1 calculation.

Offset: 0x10010 + n * 0x100 (n = 0 to 1) Access: Supervisor Read/Write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	PTR													0	0
W																
Reset:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 2459. AMU P1 Pointer Channel *n* Register (AMU2_P1PTRCH*n*)

Table 2346. AMU2_P1PTRCH*n* field descriptions

Field	Description
17:29 PTR	Pointer into ARAM for the p1[*] vector. PTR[14:0] is a byte address. Since this array contains aligned 32-bit data elements, the two least significant address bits are fixed at zero.

94.2.3.4 AMU P2 Pointer Channel *n* Register (AMU2_P2PTRCH*n*)

This register is a 4-byte aligned pointer into the ARAM defining the starting location for the 32-bit P2[*] input vector. This input vector is used in the stage1 calculation.

Offset: 0x10014 + n * 0x100 (n = 0 to 1) Access: Supervisor Read/Write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	PTR													0	0
W																
Reset:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 2460. AMU P2 Pointer Channel *n* Register (AMU2_P2PTRCH*n*)

Table 2347. AMU2_P2PTRCH*n* field descriptions

Field	Description
17:29 PTR	Pointer into ARAM for the p2[*] vector. PTR[14:0] is a byte address. Since this array contains aligned 32-bit data elements, the two least significant address bits are fixed at zero.

94.2.3.5 AMU P3 Pointer Channel *n* Register (AMU2_P3PTRCH*n*)

In the normal operating mode, this register is a 4-byte aligned pointer into the ARAM defining the starting location for the 32-bit P3[*] input vector. This input vector is used in the

stage2 outer loop calculation as a scale factor in the accumulation of results from the exp(x) evaluation.

Due to its size, there are two methods for accessing this vector. In normal mode (AMU2_DCCRCH*n*[STMP3] = 0), the entire contents of the vector are written into the ARAM before the stage2 calculation begins; for this mode, the standard PTR[14:2] field defines the offset pointer into the ARAM. In the data streaming mode (AMU2_DCCRCH*n*[STMP3] = 1), PTR[31:2] defines a starting system Flash address for the P3[*] input. Note this mode requires the system Flash address be 8-byte aligned like all the other ADMA source data addresses. As data values are fetched, they are temporarily stored in the last 1 KB region of the ARAM, an area managed by the AMU2 for special data handling. For both modes of operation, the pointer value remains unchanged, even when data streaming. Finally, the pointer must reference the 0th element in the data array, even when starting or resuming a calculation with a non-zero P8 value. To summarize:

- if AMU2_DCCRCH*n*[STMP3] = 0, then PTR[14:2] defines P3 offset pointer into the ARAM.
- if AMU2_DCCRCH*n*[STMP3] = 1, then PTR[31:3] defines P3 pointer as a system Flash address.

Offset: 0x10018 + n * 0x100 (n = 0 to 1) Access: Supervisor Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	PTR															
W																
Reset:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	PTR														0	0
W																
Reset:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 2461. AMU P3 Pointer Channel *n* Register (AMU2_P3PTRCH*n*)

Table 2348. AMU2_P3PTRCH*n* field descriptions

Field	Description
0:29 PTR	Pointer for the p3[*] vector. Depending on the data mode for the p3[*] vector, this field is used in two different ways: if AMU2_DCCRCH*n*[STMP3] = 0, then PTR[14:2] defines offset pointer into ARAM if AMU2_DCCRCH*n*[STMP3] = 1, then PTR[31:3] defines system Flash address PTR[31:0] is a byte address. The two least significant address bits are fixed at zero to enforce 4-byte (32 bit) alignment for the non-streaming configuration and the low-order three bits are forced to zero for the streaming configuration.

94.2.3.6 AMU P4 Channel *n* Register (AMU2_P4CH*n*)

This is the P4 data register. It is a 32-bit single-precision value used as a scale factor at the stage3 output normalization calculation.

Offset: 0x1001C + n * 0x100 (n = 0 to 1) Access: Supervisor Read/Write

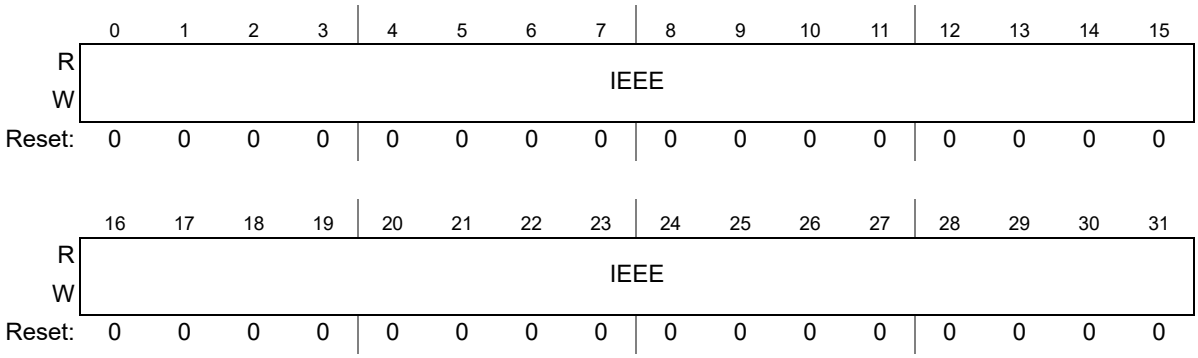


Figure 2462. AMU P4 Channel *n* Register (AMU2_P4CH*n*)

Table 2349. AMU2_P4CH*n* field descriptions

Field	Description
0:31 IEEE	Parameter p4 value in IEEE 32-bit floating-point format. This value remains static once written.

94.2.3.7 AMU P5 Channel *n* Register (AMU2_P5CH*n*)

This is the P5 data register. It is a 32-bit single-precision value optionally used as the initial result at the stage3 output normalization calculation.

Offset: 0x10020 + n * 0x100 (n = 0 to 1) Access: Supervisor Read/Write

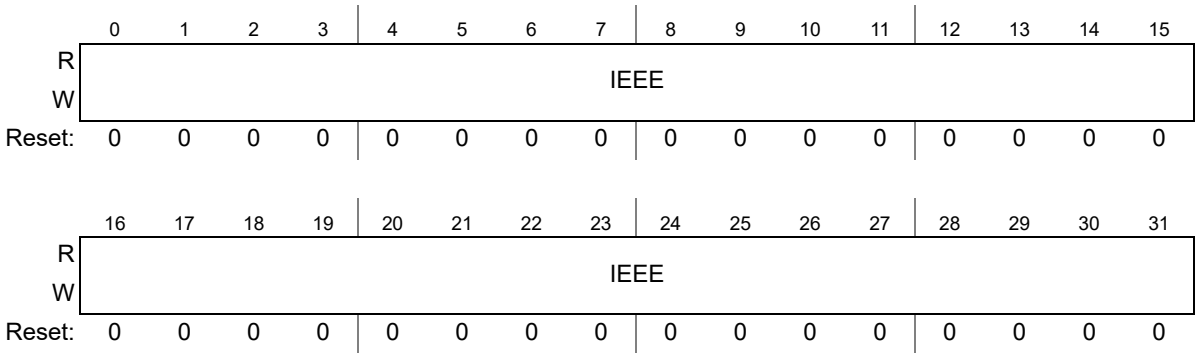


Figure 2463. AMU P5 Channel *n* Register (AMU2_P5CH*n*)

Table 2350. AMU2_P5CH*n* field descriptions

Field	Description
0:31 IEEE	Parameter p5 value in IEEE 32-bit floating-point format. This value remains static once written.

94.2.3.8 AMU P6 Channel *n* Register (AMU2_P6CH*n*)

The P6 register is a 16-bit unsigned integer defining the maximum iteration count for the stage2 outer loop calculation. This value defines the size of the ASC function.

Offset: 0x10024 + n * 0x100 (n = 0 to 1) Access: Supervisor Read/Write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	UINT															
W																
Reset:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 2464. AMU P6 Channel *n* Register (AMU2_P6CH*n*)

Table 2351. AMU2_P6CH*n* field descriptions

Field	Description
16:31 UINT	Parameter p6 value in a 16-bit unsigned integer format.

94.2.3.9 AMU P7 Channel *n* Register (AMU2_P7CH*n*)

The P7 register is a 5-bit unsigned integer defining the iteration count for the stage1 loop and the stage2 inner loop calculations. The encoding of this field has one special value, if the 5-bit value is zero, then P7 is decimal 32. This value defines the dimension of the ASC function.

Offset: 0x10028 + n * 0x100 (n = 0 to 1) Access: Supervisor Read/Write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	UINT				
W																
Reset:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 2465. AMU P7 Channel *n* Register (AMU2_P7CH*n*)

Table 2352. AMU2_P7CH*n* field descriptions

Field	Description
27:31 UINT	Parameter P7 value in a 5-bit unsigned integer format. A 0b00000 in this field means P7 = 32.

94.2.3.10 AMU P8 Channel *n* Register (AMU2_P8CH*n*)

The P8 register contains two 16-bit unsigned integers related to the outer loop at the stage2 calculation. The upper halfword contains the threshold count for the ALU's speed control (see *Section 94.3.5.2: ALU speed control*), while the lower halfword is the starting value for the stage2 outer loop calculation.

As the stage2 outer loop iterates, the lower halfword of the AMU2_P8CH*n* register is incremented. If the calculation is stopped or halted, the register reflects the incremented loop count value. When the outer loop completes execution, it is restored to its initial value. This provides hardware support for the loop control such that this register does *not* have to be explicitly re-initialized for every calculation.

There is also an aliased read-only copy of this register available so that intermediate results associated with Y64 and P8 can be retrieved by the system DMA (SDMA) in a simple, sequential block transfer.

Offset: 0x1002C + n * 0x100 (n = 0 to 1) Access: Supervisor Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	THRHC															
Reset:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	UINT															
Reset:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 2466. AMU P8 Channel *n* Register (AMU2_P8CH*n*)

Table 2353. AMU2_P8CH*n* field descriptions

Field	Description
0:15 THRHC	Threshold Count. This 16-bit unsigned integer defines the threshold count used in the ALU speed control function. There are additional control fields associated with this function in the AMU2_CCR1CH*n* register. There is a single ALU speed control function shared by the calculation engine and accessible from either the CH0 or CH1 programming models.
16:31 UINT	Parameter p8 value in a 16-bit unsigned integer format.

94.2.3.11 AMU P9M Channel *n* Register (AMU2_P9MCH*n*)

This is a 64-bit double-precision data register. It is the initial value for the stage2 outer loop accumulation. It is described here as the concatenation of two 32-bit words with P9{M,L} defining the most significant and least significant words.

Offset: 0x10030 + n * 0x100 (n = 0 to 1) Access: Supervisor Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	IEEE64															
Reset:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	IEEE64															
Reset:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 2467. AMU P9M Channel *n* Register (AMU2_P9MCH*n*)

Table 2354. AMU2_P9MCH*n* field descriptions

Field	Description
0:31 IEEE64	Most significant word of parameter p9 value in IEEE 64-bit floating-point format. This is the P9MCH*n* register.

94.2.3.12 AMU P9L Channel *n* Register (AMU2_P9LCH*n*)

This is a 64-bit double-precision data register. It is the initial value for the stage2 outer loop accumulation. It is described here as the concatenation of two 32-bit words with P9{M,L} defining the most significant and least significant words.

Offset: 0x10034 + n * 0x100 (n = 0 to 1) Access: Supervisor Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	IEEE64															
Reset:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	IEEE64															
Reset:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 2468. AMU P9L Channel *n* Register (AMU2_P9LCH*n*)

Table 2355. AMU2_P9LCH*n* field descriptions

Field	Description
0:31 IEEE64	Least significant word of parameter p9 value in IEEE 64-bit floating-point format. This is the P9LCH*n* register.

94.2.3.13 AMU V Pointer Channel *n* Register (AMU2_VPTRCH*n*)

In the normal operating mode, this register is a 4-byte aligned pointer into the ARAM defining the starting location for the 16-bit or 32-bit V[*] input array. It is used in the inner loop of the stage2 calculation in a difference evaluation.

The V[*] array supports three data formats as defined by the AMU2_CCR1CH*n*[V16, VFH, CMEXP] fields. The formats include a 32-bit single-precision floating-point, a 16-bit half-precision floating-point, and a 16-bit fixed-point format with a common 7-bit biased exponent. See *Section 94.2.3.21: AMU Control & Configuration 1 Channel n Register (AMU2_CCR1CHn)* for details on the selection of the formats and *Section 94.3.3.1: Floating-point data formats* and *Section 94.3.3.2: 16-bit fixed-point with common exponent* for information on these number formats.

The V[*] array is the only 2-dimensional data array used in the algorithm and is defined as V[p6, p7]. Its memory organization follows the standard C definition with a row-major ordering, that is, the sequential data elements as [0,0], [0,1], [0,2], …, [p6-1, p7-2], [p6-1, p7-1].

Due to its size, there are two methods for accessing this array. In normal mode (AMU2_DCCRCH*n*[STMV{32,16}] == 00), the entire contents of the array are written into the ARAM before the stage2 calculation begins; for this mode, the standard PTR[14:2] field defines the offset pointer into the ARAM. In the data streaming mode (AMU2_DCCRCH*n*[STMV{32,16}] != 00), PTR[31:3] defines a starting system Flash address for the V[*] input. Note this mode requires the system Flash address be 8-byte aligned like all the other ADMA source data addresses. As data values are fetched, they are temporarily stored in the last 1 KB region of the ARAM, an area managed by the AMU2 for special data handling. For both modes of operation, the pointer value remains unchanged, even when data streaming. Finally, the pointer must reference the [0,0] element in the data array, even when starting or resuming a calculation with a non-zero P8 value. To summarize:

- if AMU2_DCCRCH*n*[STMV{32,16}] = 00, then PTR[14:2] is V offset pointer into the ARAM.
- if AMU2_DCCRCH*n*[STMV{32,16}]!= 00, then PTR[31:3] is V pointer as Flash address.

Offset: 0x10038 + n * 0x100 (n = 0 to 1) Access: Supervisor Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	PTR															
Reset:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	PTR														0	0
Reset:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 2469. AMU V Pointer Channel *n* Register (AMU2_VPTRCH*n*)

Table 2356. AMU2_VPTRCH*n* field descriptions

Field	Description
0:29 PTR	Pointer for the v[*] array. Depending on the data mode for the v[*] array, this field is used in two different ways: – if AMU2_DCCRCH*n*[STMV{32,16}] == 00, then PTR[14:2] is offset pointer into ARAM. – if AMU2_DCCRCH*n*[STMV{32,16}] != 00, then PTR[31:3] is system Flash address. PTR[31:0] is a byte address. The two least significant address bits are fixed at zero to enforce 4-byte (32 bit) alignment for the non-streaming configuration and the low-order three bits are forced to zero for the streaming configuration.

94.2.3.14 AMU L Pointer Channel *n* Register (AMU2_LPTRCH*n*)

This register is a 4-byte aligned pointer into the ARAM defining the starting location for the 32-bit L[*] input vector. It is used in the inner loop of the stage2 calculation as a scale factor in the accumulation of squared differences.

Offset: 0x1003C + n * 0x100 (n = 0 to 1) Access: Supervisor Read/Write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	PTR													0	0
W																
Reset:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 2470. AMU L Pointer Channel *n* Register (AMU2_LPTRCH*n*)

Table 2357. AMU2_LPTRCH*n* field descriptions

Field	Description
17:29 PTR	Pointer into ARAM for the L[*] vector. PTR[14:0] is a byte address. Since this array contains aligned 32-bit data elements, the two least significant address bits are fixed at zero.

94.2.3.15 AMU U Pointer Channel *n* Register (AMU2_UPTRCH*n*)

This register is a 4-byte aligned pointer into the ARAM defining the starting location for the 32-bit single-precision U[*] input vector. It is used in the stage1 calculation.

Offset: 0x10040 + n * 0x100 (n = 0 to 1) Access: Supervisor Read/Write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	PTR													0	0
W																
Reset:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 2471. AMU U Pointer Channel *n* Register (AMU2_UPTRCH*n*)

Table 2358. AMU2_UPTRCH*n* field descriptions

Field	Description
17:29 PTR	Pointer into ARAM for the u[*] vector. PTR[14:0] is a byte address. Since this array contains aligned 32-bit data elements, the two least significant address bits are fixed at zero.

94.2.3.16 AMU DMA Source Address Channel *n* Register (AMU2_DSACH*n*)

This is the ADMA's source 8-byte aligned pointer into the system Flash region (including remapped accesses into on- and off-chip calibration memories or system RAM, or both). Depending on configuration information in the AMU2_DCCRCH*n* register, this register may be automatically updated during a data transfer, or left unchanged.

Offset: 0x10044 + n * 0x100 (n = 0 to 1) Access: Supervisor Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	PTR															
W																
Reset:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	PTR													0	0	0
W																
Reset:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 2472. AMU DMA Source Address Channel *n* Register (AMU2_DSACH*n*)

Table 2359. AMU2_DSACH*n* field descriptions

Field	Description
0:28 PTR	ADMA source address in Flash region of the system memory map. For best performance and efficiency, this byte address is forced to doubleword (8-byte aligned) alignment, matching the AMU's private interface to the Flash controller.

94.2.3.17 AMU DMA Destination Address Channel *n* Register (AMU2_DDACH*n*)

This is the ADMA's destination 8-byte aligned offset pointer into the ARAM. Depending on configuration information in the AMU2_DCCRCH*n* register, this register may be automatically updated during a data transfer, or left unchanged.

The ADMA also supports a mode where the last 8 bytes are written directly into the AMU2_CCR{1,2}CH*n* registers so that a data transfer can be coupled with the initiation of a calculation in the context of a single transfer. This mode is enabled by the AMU2_DCCRCH*n*[DARU] register field and affects the final value of the AMU2_DDACH*n* register.

Offset: 0x10048 + n * 0x100 (n = 0 to 1) Access: Supervisor Read/Write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	PTR												0	0	0
W																
Reset:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 2473. AMU DMA Destination Address Channel *n* Register (AMU2_DDACH*n*)

Table 2360. AMU2_DDACH*n* field descriptions

Field	Description
17:28 PTR	ADMA destination address in the ARAM. For best performance and efficiency, this byte address is enforced to a doubleword (8-byte aligned) alignment, matching the AMU's private interface to the Flash controller.

94.2.3.18 AMU DMA Transfer Count Channel *n* Register (AMU2_DCNTCH*n*)

This register contains two unsigned 16-bit byte counts associated with the current ADMA transfer operation. The upper halfword contains the threshold byte count associated with the ADMA's speed control function (see *Section 94.3.6.5: ADMA speed control*). The lower halfword defines the number of bytes to be transferred in the current ADMA operation. Based on ADMA address alignments, both counts are forced to be a multiple of 8 bytes.

Offset: 0x1004C + n * 0x100 (n = 0 to 1) Access: Supervisor Read/Write

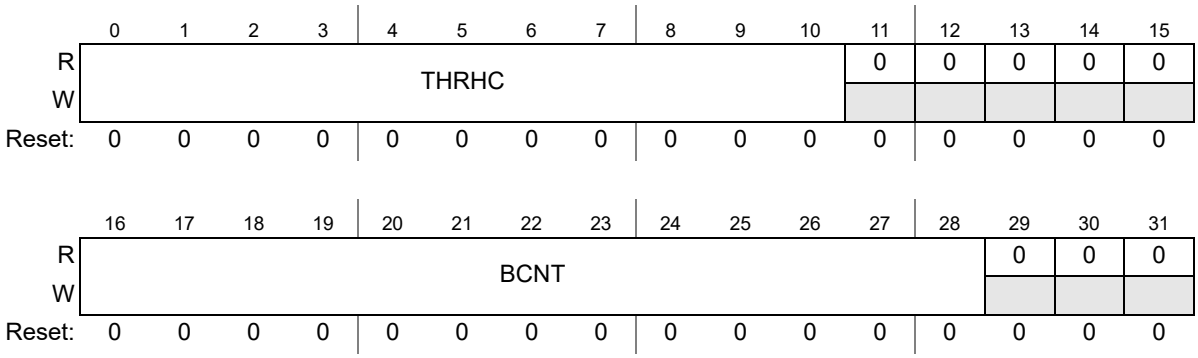


Figure 2474. AMU DMA Transfer Count Channel *n* Register (AMU2_DCNTCH*n*)

Table 2361. AMU2_DCNTCH*n* field descriptions

Field	Description
0:10 THRHC	Threshold Count. This 16-bit unsigned integer defines the threshold count used in the ADMA speed control function. There are additional control fields associated with this function in the AMU2_DCCRCH*n* register. There is a single ADMA speed control function shared by both channels and accessible from either the CH0 or CH1 programming models. The count is forced to be 32-byte aligned to match the ADMA's fetch size from the system Flash private connection.
16:28 BCNT	ADMA transfer byte count in a 16-bit unsigned integer format and forced to be an 8-byte aligned count.

94.2.3.19 AMU DMA Checksum Channel *n* Register (AMU2_DCKSCH*n*)

This is the expected ADMA checksum data register. The ADMA supports an optional calculation of a 32-bit checksum on the data transferred into the ARAM. This register defines the expected checksum result. If enabled, the checksum accumulated by the ADMA is compared to this value at the completion of the data transfer. If the actual and expected checksums are equivalent, the AMU continues, else a data transmission error has been detected (AMU2_SEIRCH*n*[CKSERR] = 1) and the AMU responds according to the programmed configuration (AMU2_DCCRCH*n*[ADICFG]).

Offset: 0x10050 + n * 0x100 (n = 0 to 1) Access: Supervisor Read/Write

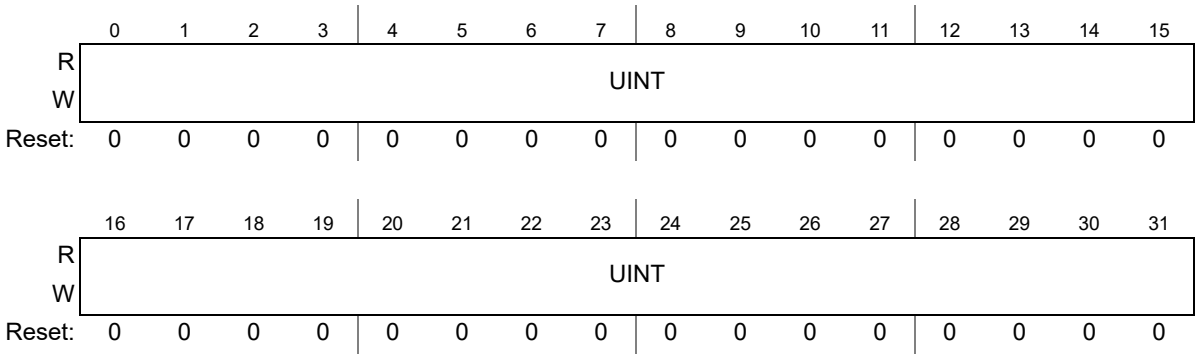


Figure 2475. AMU DMA Checksum Channel *n* Register (AMU2_DCKSCH*n*)

Table 2362. AMU2_DCKSCH*n* field descriptions

Field	Description
0:31 UINT	Expected ADMA checksum in a 32-bit unsigned integer format.

94.2.3.20 AMU DMA Control & Configuration Channel *n* Register (AMU2_DCCRCH*n*)

The AMU2_DCCRCH*n* register defines the operating configuration for the ADMA. It includes a speed control function, fields that configure the operation of the source and destination ADMA addresses, CRC configuration controls, fields to configure data streaming for the P3 and V vectors, an enable for an ARAM initialization function, conditional and unconditional start flags, plus stop and resume controls.

The conditional start flags (AMU2_DCCRCH*n*[WFS1D, WFS23D, WFDNI]) can be used to trigger ADMA transfers after the ALU completes execution of specific parts of the algorithm.

Offset: 0x10054 + n * 0x100 (n = 0 to 1) Access: Supervisor Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	SRTSP			0	MAXSP			ADICFG		SARU		DARU		HCRCE	ECRC
W																
Reset:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	DECC	STMP3	STMV32	STMV16	EHP	DSAS	STP	DEICFG	WFDNI	WFS23D	WFS1D	RSM	0	FARZ	SRT
W	CADS															
Reset:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 2476. AMU DMA Control & Configuration Channel *n* Register (AMU2_DCCRCH*n*)

Table 2363. AMU2_DCCRCH*n* field descriptions

Field	Description
1:3 SRTSP	Start Speed This 3-bit encoded value defines the start speed for the ADMA channel. It is used with the MAXSP field and AMU2_DCNTCH*n*[THRHC] with the current speed of the ADMA is reported in AMU2_SEIRCH*n*[DCURSP]. The ADMA speed control provides a mechanism to control the operation efficiency of the submodule for coarse power control of the AMU. See *Section 94.3.6.5: ADMA speed control* for more details on the ADMA speed control mechanism. There is a single ADMA speed control function shared by both channels and accessible from either the CH0 or CH1 programming models. 000 Maximum speed; speed control disabled 001 1/8 speed 010 2/8 speed 011 3/8 speed 100 4/8 speed 101 5/8 speed 110 6/8 speed 111 7/8 speed
5:7 MAXSP	Maximum Speed This 3-bit encoded value defines the maximum speed for the ADMA channel. It is used with the SRTSP field and AMU2_DCNTCH*n*[THRHC] with the current speed of the ADMA is reported in AMU2_SEIRCH*n*[DCURSP]. The ADMA speed control provides a mechanism to control the operation efficiency of the submodule for coarse power control of the AMU. See *Section 94.3.6.5: ADMA speed control* for more details on the ADMA speed control mechanism. There is a single ADMA speed control function shared by both channels and accessible from either the CH0 or CH1 programming models. This field uses the same definition as SRTSP above.
8:9 ADICFG	ADma Interrupt ConFiGuration This 2-bit field provides the reporting mechanism associated with the ADMA complete indicator flag (AMU2_SEIRCH*n*[ADIRQ]). 00 Set indicator flag; the flag is explicitly cleared by software. This mode can be used by software to poll for ADMA completion 01 Set indicator flag, assert core interrupt request. The flag is explicitly cleared by software 10 Set indicator flag, assert SDMA request. The flag is cleared by an SDMA acknowledgement 11 Set indicator flag, assert SDMA request. The flag is cleared by an SDMA acknowledgement
10:11 SARU	Source Address Register Update This 2-bit field controls the updating of the AMU2_DSACH*n* register. 00 ADMA does not modify the original starting value of AMU2_DSACH*n* 01 ADMA does not modify the original starting value of AMU2_DSACH*n* 10 ADMA increments the AMU2_DSACH*n* as the data transfer occurs. The final value points to the next 8-byte aligned system Flash address 11 ADMA increments the AMU2_DSACH*n* as the data transfer occurs and, after the final transfer, “rounds up” to the next 32-byte aligned system Flash page address

Table 2363. AMU2_DCCRCH*n* field descriptions (continued)

Field	Description
12:13 DARU	Destination Address Register Update This 2-bit field controls the updating of the AMU2_DDACH*n* register. 00 ADMA does not modify the original starting value of AMU2_DDACH*n* 01 ADMA uses AMU2_DDACH*n* + the current byte counter as the ARAM write address as each data transfer occurs *except for the final 8-byte transfer*. Both the AMU2_DDACH*n* and AMU2_DCNTCH*n* register values are unchanged. This value enables the special ADMA transfer where the last 8 bytes are *not* written into the ARAM, but instead are written into the AMU2_CCR{1,2}CH*n* registers. For this mode, the contents of AMU2_DCNTCH*n* defines the total number of data bytes to be transferred + 8 (for the final 8-byte transfer to the AMU2_CCR{1,2}CH*n*). Additionally, if DCCRCH*n*[DECC] = 0, any required single-bit corrections on the fetched data, including that destined for AMU2_CCR{1,2}CH*n*, are performed before being forwarded to the destination (either the ARAM or the control & configuration registers). If a non-correctable ECC event is detected on the last 64-bit data fetch, the update of the AMU2_CCR{1,2}CH*n* registers is not performed to avoid any inadvertent ALU activity 10 ADMA increments the AMU2_DDACH*n* as each data transfer occurs *including* the last transfer. The final value points to the next 8-byte aligned ARAM address 11 ADMA increments the AMU2_DDACH*n* as each data transfer occurs *including* the last transfer. The final value points to the next 8-byte aligned ARAM address
14 HCRCE	Halt on Cyclic Redundancy Check Error This bit determines the AMU2's response to a properly-enabled (ECRC == 1) CRC checksum error. 0 AMU2 continues calculation regardless of the error status associated with a CRC checksum 1 AMU2 halts execution if a CRC checksum error is detected when DCCRCH*n*[ECRC] = 1
15 ECRC	Enable Cyclic Redundancy Check This bit enables the ADMA's calculation of a CRC checksum as data is written into the ARAM. The expected 32-bit checksum result is defined in the AMU2_DCKSCH*n* register. 0 ADMA checksum is disabled 1 ADMA checksum is enabled
16 CADS	Clear ADMA State Setting this bit resets the DMA status state machine (AMU2_SEIRCH*n*[ADMAS]) to idle. The use of this bit is intended for situations where halted or stopped data transfers are *not* continued. This bit always reads as a zero. See *Section 94.3.6.2: ADMA execution states* for more information. 0 Do not reset the ADMA status state machine to idle 1 Reset the ADMA status state machines to idle
17 DECC	Disable ADMA ECC single bit corrections Setting this bit disables the ADMA's ECC logic from performing single-bit corrections and double-bit detections on data to be written into the ARAM and optionally sent to the CRC logic for checksum generation. This bit is intended for test and debug purposes. 0 Enable single-bit corrections and double-bit detections on corrupted data fetched from the system Flash address space 1 Disable single-bit corrections and double-bit detections on corrupted data fetched from the system Flash address space

Table 2363. AMU2_DCCRCH*n* field descriptions (continued)

Field	Description
18 STMP3	Stream the P3 vector into a 128-byte ARAM circular FIFO When enabled, the P3 vector is fetched from the system Flash using the AMU2_P3PTRCH*n* register (which is incremented) and written into a dedicated 128-byte ARAM FIFO. The transfers are paced by the ALU's execution speed. Due to the inherent pipelining of the AMU2, speculative prefetches beyond the actual P3 vector in Flash are likely - the AMU2_P3PTRCH*n* register must be reloaded before the next calculation. When operating in this mode, the checksum function is *not* supported. The streaming of P3 data is initiated when STMP3 = 1 and SRT = 1. Concurrent streaming of the V array can be supported by also setting STMV{32,16} != 00. When a non-streamed ADMA data transfer is required, this flag must be cleared. The ADMA supports either normal data transfers or a streaming operation. Writes to AMU2_CCR2CH*n*[ADCC18] may affect this indicator, that is, if AMU2_CCR2CH*n*[ADCCRE] == 1, AMU2_DCCRCH*n*[STMP3] = AMU2_CCR2CH*n*[ADCCR18]. 0 P3 streaming mode is disabled 1 P3 streaming mode is enabled
19 STMV32	Stream the 32-bit V array into a 256-byte ARAM circular FIFO When enabled, the 32-bit V array is fetched from the system Flash using the AMU2_VPTRCH*n* register (which is incremented) and written into a dedicated 256-byte ARAM FIFO. The transfers are paced by the ALU's execution speed. Due to the inherent pipelining of the AMU2, speculative prefetches beyond the actual V array in Flash are likely - the AMU2_VPTRCH*n* register must be reloaded before the next calculation. When operating in this mode, the checksum function is *not* supported. The separate STMV{32,16} are required so the ADMA can begin the appropriate data transfer *before* the AMU2_CCR1CH*n*[V16] is configured. It is software's responsibility to insure the STMV{32,16} and AMU2_CCR1CH*n*[V16] settings are consistent. The definition of the 2-bit {STMV32, STMV16} configuration bits is: (00) All V streaming is disabled (01) 16-bit V streaming is enabled (10) 32-bit V streaming is enabled (11) 32-bit V streaming is enabled The streaming of V data is initiated when STMV{32, 16} != 00 and SRT = 1. Concurrent streaming of the P3 vector can be supported by also setting STMP3 = 1. When a non-streamed ADMA data transfer is required, this flag must be cleared. The ADMA supports *either* normal data transfers or a streaming operation. Writes to AMU2_CCR2CH*n*[ADCC19] may affect this indicator, that is, if AMU2_CCR2CH*n*[ADCCRE] == 1, AMU2_DCCRCH*n*[STMV32] = AMU2_CCR2CH*n*[ADCCR19]. 0 V32 streaming mode is disabled 1 V32 streaming mode is enabled

Table 2363. AMU2_DCCRCH*n* field descriptions (continued)

Field	Description
20 STMV16	Stream the 16-bit V array into a 256-byte ARAM circular FIFO When enabled, the 16-bit V array is fetched from the system Flash using the AMU2_VPTRCH*n* register (which is incremented) and written into a dedicated 256-byte ARAM FIFO. The transfers are paced by the ALU's execution speed. Due to the inherent pipelining of the AMU2, speculative prefetches beyond the actual V array in Flash are likely - the AMU2_VPTRCH*n* register must be reloaded before the next calculation. When operating in this mode, the checksum function is not supported. The separate STMV{32,16} are required so the ADMA can begin the appropriate data transfer *before* the AMU2_CCR1CH*n*[V16] is configured. It is software's responsibility to insure the STMV{32,16} and AMU2_CCR1CH*n*[V16] settings are consistent. 0 V16 streaming mode is disabled 1 V16 streaming mode is enabled The definition of the 2-bit {STMV32, STMV16} configuration bits is: (00) All V streaming is disabled (01) 16-bit V streaming is enabled (10) 32-bit V streaming is enabled (11) 32-bit V streaming is enabled The streaming of V data is initiated when STMV{32,16} != 00 and SRT = 1. Concurrent streaming of the P3 vector can be supported by also setting STMP3 = 1. When a non-streamed ADMA data transfer is required, this flag must be cleared. The ADMA supports *either* normal data transfers or a streaming operation. Writes to AMU2_CCR2CH*n*[ADCC20] may affect this indicator, that is, if AMU2_CCR2CH*n*[ADCCRE] == 1, AMU2_DCCRCH*n*[STMV16] = AMU2_CCR2CH*n*[ADCCR20].
21 EHP	Enable High Priority The ADMA's private connection with the platform Flash controller includes a signal that elevates the priority of a data request versus requests from the other system bus masters. By default, the ADMA's priority is lower than the other system bus masters. However, when this control signal is asserted, the ADMA's request is given highest priority within the platform Flash controller. The assertion of this high priority signal is controlled by three mechanisms: (1) For non-streamed ADMA transfers, the high priority control signal is asserted based on the setting of the ADMA speed control and the platform Flash controller's response time. If the responses are too slow, the ADMA asserts the high priority signal. (2) For streamed ADMA transfers, the high priority control signal is asserted based on the setting of the ALU speed control and the platform Flash controller's response time. Again, if the responses are too slow based on the ALU's need for new data, the ADMA asserts the high priority signal. (3) If this EHP bit is set, the high priority control signal is unconditionally asserted. 0 ADMA does not unconditionally assert the high priority signal to the platform Flash controller 1 ADMA does unconditionally assert the high priority signal to the platform Flash controller

Table 2363. AMU2_DCCRCH*n* field descriptions (continued)

Field	Description
22 DSAS	ADMA Stop if Alu Stops. This bit provides a mechanism to automatically stop the ADMA channel transfer requests if the ALU stops for any reason. This bit applies to non-streamed ADMA transfers. When streaming P3 or V, or both, the ADMA automatically stops streaming if/when the ALU is stopped or halted, and automatically resumes streaming when the calculation is resumed. 0 ALU stops do not affect the ADMA 1 ALU stops force the ADMA to also stop
23 STP	ADMA Stop request The setting of this indicator bit signals the ADMA to stop at the next appropriate point. The ADMA run status is available in AMU2_SEIRCH*n*[ADMAS]. If the P3 vector or V array, or both, are being streamed, the ADMA data transfers are paced by the ALU calculation speed, so any stop or halt condition to the ALU automatically stalls the ADMA. Note the simultaneous assertion of the STP and any conditional (WFS1D, WFS23D, WFDNI) or unconditional (SRT) start indicators causes the attempted ADMA initiation to *effectively be ignored* with no change in the AMU2_SEIRCH*n*[ADMAS] status. 0 No ADMA stop request; it operates normally 1 ADMA stop request; stop data transfer at the next appropriate point. Once the ADMA stops, this indicator is cleared.
24 DEICFG	ADMA Error Interrupt Conflguration This field controls the reporting mechanism associated with the ADMA error indicator flag (AMU2_SEIRCH*n*[DEIRQ]). 0 If an ADMA error is detected, set the indicator flag; the flag is explicitly cleared by software. 1 If an ADMA error is detected, set the indicator flag and assert core interrupt request; the flag is explicitly cleared by software.
25 WFDNI	Wait For ALU Done Interrupt request This flag is a conditional ADMA start indicator. If asserted, the ADMA does not begin any data transfers until the ALU's stage2 + 3 calculations complete and set the AMU2_SEIRCH*n*[DNIRQ] indicator. This condition is signaled by (AMU2_SEIRCH*n*[DNIRQ] = 1'b1). If the ALU is idle when the AMU2_DCCRCH*n* register is loaded and a conditional ADMA start after the next ALU calculation completes is desired, the WFDNI indicator should be used. Once the ADMA transfer begins, this indicator bit is automatically cleared. Writes to AMU2_CCR2CH*n*[ADCC25] may affect this indicator, that is, if AMU2_CCR2CH*n*[ADCCRE] == 1, AMU2_DCCRCH*n*[WFDNI] = AMU2_CCR2CH*n*[ADCCR25]. 0 No ADMA start condition dependency 1 ADMA does not start until the ALU's stage2 + 3 calculations complete, and set AMU2_SEIRCH*n*[DNIRQ]

Table 2363. AMU2_DCCRCH*n* field descriptions (continued)

Field	Description
26 WFS23D	Wait For Stage23 Done This flag is a conditional ADMA start indicator. If asserted, the ADMA does not begin any data transfers until the ALU's stage2 + 3 calculations are complete. This condition is signaled by (AMU2_SEIRCH*n*[S23XS] == 2'b00). Once the ADMA transfer begins, this indicator bit is automatically cleared. If the ADMA's data transfer must be delayed until the entire ALU calculation is complete, then both WFS1D and WFS23D must be asserted. Alternatively, the WFDNI flag can be used as the conditional ADMA start condition. Writes to AMU2_CCR2CH*n*[ADCC26] may affect this indicator, that is, if AMU2_CCR2CH*n*[ADCCRE] == 1, AMU2_DCCRCH*n*[WFS23D] = AMU2_CCR2CH*n*[ADCCR26]. 0 No ADMA start condition dependency 1 ADMA does not start until the ALU's stage2 + 3 calculations are complete
27 WFS1D	Wait For Stage1 Done This flag is a conditional ADMA start indicator. If asserted, the ADMA does not begin any data transfers until the ALU's stage1 calculation is complete. This condition is signaled by (AMU2_SEIRCH*n*[S1XS] == 2'b00). Once the ADMA transfer begins, this indicator bit is automatically cleared. If the ADMA's data transfer must be delayed until the entire ALU calculation is complete, then both WFS1D and WFS23D must be asserted. Alternatively, the WFDNI flag can be used as the conditional ADMA start condition. Writes to AMU2_CCR2CH*n*[ADCC27] may affect this indicator, that is, if AMU2_CCR2CH*n*[ADCCRE] == 1, AMU2_DCCRCH*n*[WFS1D] = AMU2_CCR2CH*n*[ADCCR27]. 0 No ADMA start condition dependency 1 ADMA does not start until the ALU's stage1 calculation is complete
28 RSM	ADMA ReSuMe The assertion of a properly-qualified resume condition ((RSM == 1) && (STP == 0)) resumes the data transfer defined by the contents of the ADMA's registers, exiting from the stopped state. This bit applies to non-streamed ADMA transfers. When streaming P3 or V, or both, the ADMA automatically stops streaming if/when the ALU is stopped or halted, and automatically resumes streaming when the calculation is resumed. Once the ADMA resumes execution, this indicator bit is automatically cleared. 0 Do not resume the ADMA 1 Resume the ADMA's data transfer

Table 2363. AMU2_DCCRCH*n* field descriptions (continued)

Field	Description
30 FARZ	Force ARAM to Zero The assertion of this indicator flag enables a special ADMA mode of operation where up to 16 bytes of zero data (with the appropriate ECC check bits) are written into the ARAM destination defined by AMU2_DDACH*n* on alternating cycles. This ARAM initialization mode functions in a manner similar to a normal ADMA data transfer. Instead of fetching data from the Flash (or one of its remapped destination memories), the ADMA forces all zero data with correct ECC to be written into the ARAM. The AMU2_DSACH*n* register is *not* used, but the starting destination address is defined by AMU2_DDACH*n* and the size of the memory to be cleared by AMU2_DCNTCH*n*[BCNT]. The CRC is not supported in this mode, and AMU2_DCCRCH*n*[DARU] must be 0b00. The ARAM initialization is initiated by setting AMU2_DCCRCH*n*[SRT] = 1. All the ADMA status reporting operates like a normal data transfer. It is recommended that the other ADMA channel be idle during this initialization function. 0 The zeroing of the ARAM by the ADMA is disabled 1 The zeroing of the ARAM by the ADMA is enabled
31 SRT	ADMA Start request. This assertion of a properly-qualified start condition ((SRT == 1) && (STP == 0)) initiates the data transfer defined by the contents of the ADMA's registers. Once ADMA starts execution, this indicator bit is automatically cleared. Writes to AMU2_CCR2CH*n*[ADCC31] may affect this indicator, that is, if AMU2_CCR2CH*n*[ADCCRE] == 1, AMU2_DCCRCH*n*[SRT] = AMU2_CCR2CH*n*[ADCCR31]. 0 Do not unconditionally start the ADMA 1 Unconditionally start the ADMA

94.2.3.21 AMU Control & Configuration 1 Channel *n* Register (AMU2_CCR1CH*n*)

AMU2_CCR1CH*n* is the first of two control and configuration registers defining the operation of the ALU.

Offset: 0x10058 + n * 0x100 (n = 0 to 1) Access: Supervisor Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	SRTSP			0	MAXSP			AEICFG		OVICFG		NCICFG		DNICFG	
W																
Reset:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	BPRI	EECC	DDSL	SETZ	ESTZC	0	0	V16	VFH	CMEXP						
W																
Reset:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 2477. AMU Control & Configuration 1 Channel *n* Register (AMU2_CCR1CH*n*)

Table 2364. AMU2_CCR1CH*n* field descriptions

Field	Description
1:3 SRTSP	Start Speed This 3-bit encoded value defines the start speed for the ALU engine. It is used with the MAXSP field and AMU2_P8CH*n*[THRHC] with the current speed of the ALU is reported in AMU2_SEIRCH*n*[ACURSP]. The ALU speed control provides a mechanism to control the operation efficiency of the engine for coarse power control of the AMU. See *Section 94.3.5.2: ALU speed control* for more details on the ALU speed control mechanism. There is a single ALU speed control function shared by the calculation engine and accessible from either the CH0 or CH1 programming models. 000 Maximum speed; speed control disabled 001 1/8 speed 010 2/8 speed 011 3/8 speed 100 4/8 speed 101 5/8 speed 110 6/8 speed 111 7/8 speed
5:7 MAXSP	Maximum Speed This 3-bit encoded value defines the maximum speed for the ALU engine. It is used with the SRTSP field and AMU2_P8CH*n*[THRHC] with the current speed of the ALU is reported in AMU2_SEIRCH*n*[ACURSP]. The ALU speed control provides a mechanism to control the operation efficiency of the engine for coarse power control of the AMU. See *Section 94.3.5.2: ALU speed control* for more details on the ALU speed control mechanism. There is a single ALU speed control function shared by the calculation engine and accessible from either the CH0 or CH1 programming models. 000 Maximum speed 001 1/8 speed 010 2/8 speed 011 3/8 speed 100 4/8 speed 101 5/8 speed 110 6/8 speed 111 7/8 speed
8:9 AEICFG	ALU Error Interrupt ConFiGuration This 2-bit field provides the reporting mechanism associated with the ALU error indicator flag (AMU2_SEIRCH*n*[ERIRQ]). 00 Set indicator flag; the flag is explicitly cleared by software. This mode can be used by software to poll for ALU error condition 01 Set indicator flag, assert core interrupt request. The flag is explicitly cleared by software 10 Set indicator flag; the flag is explicitly cleared by software. This mode can be used by software to poll for ALU error condition 11 Set indicator flag, assert core interrupt request. The flag is explicitly cleared by software

Table 2364. AMU2_CCR1CH*n* field descriptions (continued)

Field	Description
10:11 OVICFG	Overflow Interrupt Configuration This 2-bit field provides the reporting mechanism associated with the ALU overflow indicator flag (AMU2_SEIRCH*n*[OVIRQ]). 00 Set indicator flag; the flag is explicitly cleared by software. This mode can be used by software to poll for ALU overflow condition. 01 Set indicator flag, assert core interrupt request. The flag is explicitly cleared by software. 10 Set indicator flag; the flag is explicitly cleared by software. This mode can be used by software to poll for ALU overflow condition. 11 Set indicator flag, assert core interrupt request. The flag is explicitly cleared by software.
12:13 NCICFG	N Crossing Interrupt Configuration This 2-bit field provides the reporting mechanism associated with the ALU N crossing indicator flag (AMU2_SEIRCH*n*[NCIRQ]). 00 Set indicator flag; the flag is explicitly cleared by software. This mode can be used by software to poll for ALU N crossing condition 01 Set indicator flag, assert core interrupt request. The flag is explicitly cleared by software 10 Set indicator flag, assert SDMA request. The flag is cleared by an SDMA acknowledgement.The AMU2_CCR2CH*n*[SNCRS] indicator should be set so the ALU stops its calculation so the correct intermediate result is available for the SDMA 11 Set indicator flag, assert SDMA request. The flag is cleared by an SDMA acknowledgement.The AMU2_CCR2CH*n*[SNCRS] indicator should be set so the ALU stops its calculation so the correct intermediate result is available for the SDMA
14:15 DNICFG	ALU Done Interrupt Configuration This 2-bit field provides the reporting mechanism associated with the ALU done, that is, a calculation has successfully completed, indicator flag (AMU2_SEIRCH*n*[DNIRQ]). 00 Set indicator flag; the flag is explicitly cleared by software. This mode can be used by software to poll for ALU done condition. 01 Set indicator flag, assert core interrupt request. The flag is explicitly cleared by software. 10 Set indicator flag, assert SDMA request. The flag is cleared by an SDMA acknowledgement 11 Set indicator flag, initiate an ADMA transfer if one of the conditional start flags (AMU2_DCCRCH*n*[WFS1D, WFS23D, WFDNI]) is also asserted. The AMU2_SEIRCH*n*[DNIRQ] flag is cleared by the AMU
16 BPRI	Bus Priority This indicator bit controls the bus arbitration priority associated with the ARAM and the programming model registers. Recall from *Figure 2456: AMU2 block diagram*, there are multiple master devices that can access the ARAM and programming model registers, specifically, the ADMA and ALU as well as system bus transfers routed via slave peripheral bus. This indicator bit controls the arbitration policy for selecting one master's request when concurrent accesses occur; idle cycles from the selected master provide opportunities for stalled accesses from other masters to be granted and complete. 0 ADMA has higher arbitration priority than the ALU which has higher priority than slave peripheral bus accesses (ADMA > ALU > slave peripheral bus) 1 ADMA has higher arbitration priority than slave peripheral bus accesses which have higher priority than the ALU (ADMA > slave peripheral bus > ALU)

Table 2364. AMU2_CCR1CH*n* field descriptions (continued)

Field	Description
17 EECC	Enable Error Correcting Code (ECC) in the ARAM This bit controls the checking of data read from the ARAM by the ALU and the slave peripheral bus. Note, ECC is generated on all writes to the ARAM from either the ADMA or the slave peripheral bus. If a single-bit error is detected by the ECC logic on an ARAM read, the corrected data is written back into the ARAM *only* in response to the read-modify-write sequence needed on all slave peripheral write accesses. 0 ECC checking of ARAM reads is disabled 1 ECC checking of ARAM reads is enabled If the system DMA is used for ARAM initialization, the EECC bits for *both* channels must be cleared until the initialization is complete.
18 DDSL	Disable Debug Stall This bit disables the AMU from stalling when an MCU debug event is underway. 0 All debug events, as signaled by an input signal, stall the AMU. The stall globally affects both channels of the ADMA and ALU 1 Debug events do not affect the AMU

19
SETZ
	Set Z result
	This bit controls the first operation performed in the ALU's stage3 output normalization. As the name implies, if this bit is asserted, the final result (z64) is set to the P5 data value, else it is unaffected. The second operation in stage3 involves a final accumulation with the result plus the product of y64 and p4.
	The stage3 output normalization pseudo-code is:
	`/* stage 3: output normalization */`
	`if (setz) {`
	`z64 = (ieee64) p5;`
	`}`
	`z64 = z64 + (ieee32) y64 * p4; // 32 x 32 -> 64 FMA`
	0 z64 is unaffected
	1 z64 is set to the 64-bit double-precision value of p5

20 ESTZC	Enable SETZ Clear If set, this bit forces the SETZ flag to be automatically cleared after a stage3 calculation as indicated by the assertion of AMU2_SEIRCH*n*[DNIRQ]. 0 SETZ flag is not cleared by the hardware. 1 SETZ flag is automatically cleared by the hardware after the stage3 calculation completes.
23 V16	V is a 16-bit data format This bit, along with VFH, defines the format for the V array: if V16 = 0, then V[*] is 32-bit single-precision format (ieee32) if V16 = 1 && VFH = 0, then V[*] is 16-bit half-precision format (ieee16) if V16 = 1 && VFH = 1, then V[*] is 16-bit fixed-point format with a common 7-bit biased exponent This bit cannot be changed while the ADMA is streaming V[*] data. See *Section 94.3.3.2: 16-bit fixed-point with common exponent* for details on the 16-bit fixed point format.

Table 2364. AMU2_CCR1CH*n* field descriptions (continued)

Field	Description
24 VFH	V is Fixed Halfword format This bit, along with V16, defines the format for the V array: if V16 = 0, then V[*] is 32-bit single-precision format (ieee32) if V16 = 1 && VFH = 0, then V[*] is 16-bit half-precision format (ieee16) if V16 = 1 && VFH = 1, then V[*] is 16-bit fixed-point format with a common 7-bit biased exponent See *Section 94.3.3.2: 16-bit fixed-point with common exponent* for details on the 16-bit fixed point format.
25:31 CMEXP	Common Exponent This 7-bit field defines the common exponent using an excess-63 format for the V array's 16-bit fixed-point format. See *Section 94.3.3.2: 16-bit fixed-point with common exponent* for details on the 16-bit fixed point format.

94.2.3.22 AMU Control & Configuration 2 Channel *n* Register (AMU2_CCR2CH*n*)

AMU2_CCR2CH*n* is the second of two control and configuration registers defining the operation of the ALU.

Offset: 0x1005C + n * 0x100 (n = 0 to 1) Access: Supervisor Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	ADCCR18	ADCCR19	ADCCR20	0	0	0	0	ADCCR25	ADCCR26	ADCCR27	ADCCR31	0	ENCRS	SNCRS
W		ADCCRE														
Reset:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	S23STP	0	WFSS23	WFDS23	WFDS1	S23RSM	0	S23SRT	S1SRT
W	CSXXS															
Reset:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 2478. AMU Control & Configuration 2 Channel *n* Register (AMU2_CCR2CH*n*)

Table 2365. AMU2_CCR2CH*n* field descriptions

Field	Description
1 ADCCRE	Alternate DCCR write Enable This bit enables all the aliased AMU2_DCCRCH*n* indicators (AMU2_CCR2CH*n*[ADCR18, ADCCR19, ADCCR20, ADCCR25, ADCCR26, ADCCR27]) to be written when this register is updated. Clearing this bit allows software to write the AMU2_CCR2CH*n* without affecting the aliased control bits in the AMU2_DCCRCH*n*. The ADCCRE indicator always reads as a zero. 0 Contents of the write data for AMU2_CCR2CH*n*[ADCCR{18,19,20,25,26,27}] do not affect the AMU2_DcCRCH*n* 1 Contents of the write data for AMU2_CCR2CHn[ADCCR{18,19,20,25,26,27}] are loaded into AMU2_DCCRCH*n*
2 ADCCR18	Alternate AMU2_DCCRCH*n*[STMP3] If a register write with AMU2_CCR2CHn[ADCCRE] == 1, then writes to this indicator flag directly affect AMU2_DCCRCH*n*[STMP3], that is, AMU2_DCCRCH*n*[STMP3] = AMU2_CCR2CH*n*[ADCCR18], else the AMU2_DCCRCH*n* indicator is unaffected. This alternate copy of the STMP3 indicator allows it to be written by the special ADMA mode that loads the AMU2_CCR{1,2}CH*n* registers as the last 8 bytes transferred.
3 ADCCR19	Alternate AMU2_DCCRCH*n*[STMV32] If a register write with AMU2_CCR2CHn[ADCCRE] == 1, then writes to this indicator flag directly affect AMU2_DCCRCH*n*[STMV32], that is, AMU2_DCCRCH*n*[STMV32] = AMU2_CCR2CH*n*[ADCCR19], else the AMU2_DCCRCH*n* indicator is unaffected. This alternate copy of the STMV32 indicator allows it to be written by the special ADMA mode that loads the AMU2_CCR{1,2}CH*n* registers as the last 8 bytes transferred.
4 ADCCR20	Alternate AMU2_DCCRCH*n*[STMV16] If a register write with AMU2_CCR2CHn[ADCCRE] == 1, then writes to this indicator flag directly affect AMU2_DCCRCH*n*[STMV16], that is, AMU2_DCCRCH*n*[STMV16] = AMU2_CCR2CH*n*[ADCCR20], else the AMU2_DCCRCH*n* indicator is unaffected. This alternate copy of the STMV16 indicator allows it to be written by the special ADMA mode that loads the AMU2_CCR{1,2}CH*n* registers as the last 8 bytes transferred.
9 ADCCR25	Alternate AMU2_DCCRCH*n*[WFDNI] If a register write with AMU2_CCR2CHn[ADCCRE] == 1, then writes to this indicator flag directly affect AMU2_DCCRCH*n*[WFDNI], that is, AMU2_DCCRCH*n*[WFDNI] = AMU2_CCR2CH*n*[ADCCR25], else the AMU2_DCCRCH*n* indicator is unaffected. This alternate copy of the WFDNI indicator allows it to be written by the special ADMA mode that loads the AMU2_CCR{1,2}CH*n* registers as the last 8 bytes transferred.
10 ADCCR26	Alternate AMU2_DCCRCH*n*[WFS23D] If a register write with AMU2_CCR2CHn[ADCCRE] == 1, then writes to this indicator flag directly affect AMU2_DCCRCH*n*[WFS23D], that is, AMU2_DCCRCH*n*[WFS23D] = AMU2_CCR2CH*n*[ADCCR26], else the AMU2_DCCRCH*n* indicator is unaffected. This alternate copy of the WFS23D indicator allows it to be written by the special ADMA mode that loads the AMU2_CCR{1,2}CH*n* registers as the last 8 bytes transferred.
11 ADCCR27	Alternate AMU2_DCCRCH*n*[WFS1D] If a register write with AMU2_CCR2CHn[ADCCRE] == 1, then writes to this indicator flag directly affect AMU2_DCCRCH*n*[WFS1D], that is, AMU2_DCCRCH*n*[WFS1D] = AMU2_CCR2CH*n*[ADCCR27], else the AMU2_DCCRCH*n* indicator is unaffected. This alternate copy of the WFS1D indicator allows it to be written by the special ADMA mode that loads the AMU2_CCR{1,2}CH*n* registers as the last 8 bytes transferred.

Table 2365. AMU2_CCR2CH*n* field descriptions (continued)

Field	Description
12 ADCCR31	Alternate AMU2_DCCRCH*n*[SRT] If a register write with AMU2_CCR2CHn[ADCCRE] == 1, then writes to this indicator flag directly affect AMU2_DCCRCH*n*[SRT], that is, AMU2_DCCRCH*n*[SRT] = AMU2_CCR2CH*n*[ADCCR31], else the AMU2_DCCRCH*n* indicator is unaffected. This alternate copy of the SRT indicator allows it to be written by the special ADMA mode that loads the AMU2_CCR{1,2}CH*n* registers as the last 8 bytes transferred.
14 ENCRS	Enable N CRoSsing event This bit enables the occurrence of an N crossing event. 0 The ALU's N crossing logic is disabled 1 The ALU's N crossing logic is enabled
15 SNCRS	Stop on N CRoSsing event This bit forces the ALU to stop execution on the occurrence of an N crossing event. 0 The ALU does not stop execution when an N crossing event occurs 1 The ALU stops execution when an N crossing event occurs
16 CSXXS	Clear Stage "X" eXecution State Setting this bit resets both the ALU execution state machines (AMU2_SEIRCH*n*[S1XS, S23XS]) to idle. The use of this bit is intended for situations where halted or stopped calculations are *not* continued. This bit always reads as a zero. 0 Do not reset the ALU execution state machines to idle 1 Reset both of the ALU execution state machines to idle See *Section 94.3.5.1: ALU execution states* for more information.
23 S23STP	Stage23 STop request The setting of this indicator bit signals the ALU to stop at the completion of the current outer loop execution. If asserted *after* the final stage2 outer loop completes, stage3 also finishes and S23STP remains asserted indicating the stop request was not granted.The ALU's stage23 status is available in AMU2_SEIRCH*n*[S23XS]. Once the ALU is stopped, this indicator bit is automatically cleared. 0 No stage23 stop request; the ALU operates normally 1 Stage23 stop request; stop execution at the completion of the current outer loop iteration
25 WFSS23	Wait For Sdma complete before starting Stage23 This is a conditional start bit for stage23. It signals the stage23 execution start is enabled by the completion of an SDMA data transfer. This indicator is used when the SDMA is used to copy an AMU result into system memory. The setting of this flag means the SDMA result retrieval must be finished before a stage23 calculation is initiated. Once stage23 begins execution, this indicator bit is automatically cleared. When streaming P3[*] or V[*] data, or both, this indicator bit is ignored. 0 No conditional stage23 start condition 1 Initiate stage23's execution when the SDMA completes its current data transfer

Table 2365. AMU2_CCR2CH*n* field descriptions (continued)

Field	Description
26 WFDS23	Wait For aDma complete before starting Stage23 This is a conditional start bit for stage23. It signals the stage23 execution start is enabled by the completion of the ADMA data transfer. The WFDS23 indicator is automatically cleared if the ADMA halts due to a bus or non-correctable ECC error, or if a checksum error is detected. This prevents the ALU stage23 execution from starting after the ADMA halt condition is processed by a service routine. The WFDS23 conditional start indicator can be combined with WFDS1. If both bits are asserted, as the ADMA completes its data transfer, the ALU begins execution of stage1, followed by stage23. Once stage23 begins execution, this indicator bit is automatically cleared. When streaming P3[*] or V[*] data, or both, this indicator bit is ignored. 0 No conditional stage23 start condition 1 Initiate stage23's execution when the ADMA completes its current data transfer
27 WFDS1	Wait For aDma complete before starting Stage1 This is a conditional start bit for stage1. It signals the stage1 execution start is enabled by the completion of the ADMA data transfer. The WFDS1 indicator is automatically cleared if the ADMA halts due to a bus or non-correctable ECC error, or if a checksum error is detected. This prevents the ALU stage1 execution from starting after the ADMA halt condition is processed by a service routine. The WFDS1 conditional start indicator can be combined with WFDS23. If both bits are asserted, as the ADMA completes its data transfer, the ALU begins execution of stage1, followed by stage23. Once stage1 begins execution, this indicator bit is automatically cleared. When streaming P3[*] or V[*] data, or both, this indicator bit is ignored. 0 No conditional stage1 start condition 1 Initiate stage1's execution when the ADMA completes its current data transfer.
28 S23RSM	Stage23 ReSuMe This bit resumes a stopped stage23 calculation. If stage23 is not stopped at the time of an attempted write to assert this bit, it is ignored. Once stage23 resumes execution, this indicator bit is automatically cleared. 0 Do not resume a stopped stage23 calculation 1 Resume a stopped stage23 calculation
30 S23SRT	Stage23 StaRT request. This assertion of a properly-qualified start condition ((S23SRT == 1) && (S23STP == 0)) initiates the ALU's stage23 execution. If S1SRT is also asserted, the ALU hardware sequences through the execution, first of stage1 and then stage23. Once stage23 starts execution, this indicator bit is automatically cleared. 0 Do not unconditionally start ALU's stage23 execution 1 Unconditionally start ALU's stage23 execution, either immediately or after stage1 completes
31 S1SRT	Stage1 StaRT request. This assertion of this start condition initiates the ALU's stage1 execution. Once stage1 starts execution, this indicator bit is automatically cleared. 0 Do not unconditionally start ALU's stage1 execution 1 Unconditionally start ALU's stage1 execution

94.2.3.23 AMU RES64M Channel *n* Register (AMU2_RES64MCH*n*)

This is the read-only final 64-bit double-precision result register. It is the z64 final accumulation from the stage3 output normalization calculation and is loaded into this register as the calculation completes. It is described here as the concatenation of two 32-bit words with RES64{M,L} defining the most significant and least significant words.

Offset: 0x10080 + n * 0x100 (n = 0 to 1) Access: Supervisor Read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	IEEE64															
W																
Reset:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	IEEE64															
W																
Reset:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 2479. AMU RES64M Channel *n* Register (AMU2_RES64MCH*n*)

Table 2366. AMU2_RES64MCH*n* field descriptions

Field	Description
0:31 IEEE64	Most significant word of the final result value in IEEE 64-bit floating-point format. This is the RES64MCH*n* register.

94.2.3.24 AMU RES64L Channel *n* Register (AMU2_RES64LCH*n*)

This is the read-only final 64-bit double-precision result register. It is the z64 final accumulation from the stage3 output normalization calculation and is loaded into this register as the calculation completes. It is described here as the concatenation of two 32-bit words with RES64{M,L} defining the most significant and least significant words.

Offset: 0x10084 + n * 0x100 (n = 0 to 1) Access: Supervisor Read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	IEEE64															
W																
Reset:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	IEEE64															
W																
Reset:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 2480. AMU RES64L Channel *n* Register (AMU2_RES64LCH*n*)

Table 2367. AMU2_RES64LCH*n* field descriptions

Field	Description
0:31 IEEE64	Least significant word of the final result value in IEEE 64-bit floating-point format. This is the RES64LCH*n* register.

94.2.3.25 AMU Status, Error & Interrupt Channel *n* Register (AMU2_SEIRCH*n*)

This is the AMU's status register. The most significant three bytes contain read-only status and error flags, while the low-order byte contains the interrupt flag bits. Specifically, there are 15 status bits signaling the state of the ADMA, stage1 and stage23 ALU calculations, along with the current speed settings, 9 error flags signaling arithmetic faults in the ALU or other error conditions, and 6 interrupt request flags.

There are aliased copies of this register available at multiple addresses to allow the SDMA to retrieve the AMU2 result (either as a 64-bit double-precision or a 32-bit single-precision format based on the access address) and this status register in a simple, sequential block transfer. For example, a 16-byte block transfer beginning at offset address 0x1_0080 retrieves the AMU2_RES64{M,L}CH0 result, the AMU2_SEIRCH0 status and the AMU2_ACKSCH0 checksum result. Conversely, a 12-byte block transfer beginning at offset address 0x1_0090 retrieves the AMU2_RES32CH0 result, the aliased copy of the AMU2_SEIRCH0 register and the aliased copy of the AMU2_ACHKSCH0 register. Similar 16- and 12-byte block transfers in the CH1 section of the memory map return the comparable registers.

Certain error flag indicators in this register are cleared by a read of the register; there is an alternate location of this register (see *Section 94.2.3.33: AMU Status, Error & Interrupt No Clear Channel n Register (AMU2_SEINCCHn)*) where a read does not affect these status flags. Accordingly, software polling of the CHBSY flag should use the alternate AMU2_SEINCCH*n* register so that no fault status bits are inadvertently cleared.

It should be noted the interrupt flags are intended to record the completion of an event or the occurrence of an error condition. These flags are set when the event occurs, while the subsequent reporting of the event is defined by the corresponding interrupt configuration field, specifically AMU2_DCCRCH*n*[ADICFG, DEICFG] and AMU2_CCR1CH*n*[AEICFG, OVICFG, NCICFG, DNICFG]. The event's reporting can be configured to *not* perform any type of asynchronous notification (intended for software polling applications), a processor interrupt, or depending on the event, a system DMA or ADMA request. For DMA requests (both system and AMU), the hardware clears the indicator event flag as the request is acknowledged; for the software polling configuration or processor interrupts, a service routine or another software task must explicitly clear the event flag by writing a logical 1 to the corresponding interrupt flag. This is a "write 1" clear protocol.

Offset: 0x10088 + n * 0x100 (n = 0 to 1) Access: Supervisor Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	CHBSY	ADSTM	S23XS		S1XS		ADMAS		QDSL	ACURSP			0	DCURSP		
W																
Reset:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	YCAFLT	ZCAFLT	S23AFLT	S1AFLT	IDVERR	ADMERR	CKSERR	ECCERR	RGCERR	0	DEIRQ	ADIRQ	ERIRQ	OVIRQ	NCIRQ	DNIRQ
W											w1c	w1c	w1c	w1c	w1c	w1c
Reset:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 2481. AMU Status, Error & Interrupt Channel *n* Register (AMU2_SEIRCH*n*)

Table 2368. AMU2_SEIRCH*n* field descriptions

Field	Description
0 CHBSY	Channel Busy This read-only bit is intended to serve as a polling flag as it signals the AMU2 channel is busy. In this context, the “channel” status represents the logical summation of the ADMA data channel and the ALU calculation engine status fields indicating a non-idle “channel”. This flag is asserted when ((AMU2_SEIRCH*n*[ADMAS] == 0b11) || (AMU2_SEIRCH*n*[S1XS] == 0b11) || (AMU2_SEIRCH*n*[S23XS] == 0b11)). 0 Channel *n* is idle 1 Channel *n* is busy performing an ADMA data transfer or an ALU calculation
1 ADSTM	ADMA is Streaming This read-only flag signals whether the ADMA is performing a data streaming transfer of the P3[*] or V[*] input vectors as enabled by AMU2_DCCRCH*n*[STMP3, STMV{32, 16}] control bits. 0 The ADMA is not performing any P3[*] nor V[*] data streaming transfers 1 The ADMA is busy performing a P3[*] or V[*] data streaming transfers
2:3 S23XS	Stage23 Execution Status This read-only 2-bit field defines the execution status of the ALU’s stage23 logic. See *Section 94.3.5.1: ALU execution states* for more details on the execution states. The status encodings are: 00 Idle or pending start condition 01 Halted 10 Stopped 11 Running or stalled

Table 2368. AMU2_SEIRCH*n* field descriptions (continued)

Field	Description
4:5 S1XS	Stage1 eXecution Status This read-only 2-bit field defines the execution status of the ALU's stage1 logic. See *Section 94.3.5.1: ALU execution states* for more details on the execution states. The status encodings are: 00 Idle or pending start condition 01 Halted 10 Unused, reserved as stage1 execution cannot be stopped via a request 11 Running or stalled
6:7 ADMAS	ADMA Status This read-only 2-bit field defines the status of the ADMA. See *Section 94.3.6.2: ADMA execution states* for more details on the execution states. The status encodings are: 00 Idle or pending start condition 01 Halted 10 Stopped 11 Running or stalled
8 QDSL	Qualified Debug Stali This read-only bit signals a qualified debug stall condition as defined by (debug_event == 1) && (AMU2_CCR1CH*n*[DDSL] == 0). 0 No qualified debug stall 1 The ALU calculation engine and ADMA are stalled due to a qualified debug event.
9:11 ACURSP	Alu Current Speed This read-only 3-bit encoded value defines the current speed for the ALU engine. The ALU speed control is defined by AMU2_CCR1CH*n*[SRTSP, MAXSP] and AMU2_P8CH*n*[THRHC]. The ALU speed control provides a mechanism to control the operation efficiency of the submodule for coarse power control of the AMU. See *Section 94.3.5.2: ALU speed control* for more details on the ALU speed control mechanism. There is a single ALU speed control function shared by both channels. 000 Maximum speed; speed control disabled 001 1/8 speed 010 2/8 speed 011 3/8 speed 100 4/8 speed 101 5/8 speed 110 6/8 speed 111 7/8 speed
13:15 DCURSP	ADMA Current Speed This read-only 3-bit encoded value defines the current speed for the ADMA channel. The ADMA speed control is defined by AMU2_DCCRCH*n*[SRTSP, MAXSP] and AMU2_DCNTCH*n*[THRHC]. The ADMA speed control provides a mechanism to control the operation efficiency of the submodule for coarse power control of the AMU. See *Section 94.3.6.5: ADMA speed control* for more details on the ADMA speed control mechanism. There is a single ADMA speed control function shared by both channels. The DCURSP uses the same encodings as the ACURSP field above.

Table 2368. AMU2_SEIRCH*n* field descriptions (continued)

Field	Description
16 YCAFLT	Y Conversion Arithmetic Fault The assertion of this read-only flag indicates the conversion from double-precision ieee64 format of the y64 result to single-precision ieee32 format generated an arithmetic fault (overflow, underflow). This flag is cleared by a read of this register (r1c); reads of AMU2_SEINCCH*n* do not clear this flag. 0 No arithmetic fault associated with the conversion of y64 to y32 1 Arithmetic fault associated with the conversion of y64 to y32 was detected. For the overflow case, the y32 value is set to (+/-) INF, while an underflow produces a (+/-) 0 value
17 ZCAFLT	Z Conversion Arithmetic Fault The assertion of this read-only flag indicates the conversion from double-precision ieee64 format of the z64 result to single-precision ieee32 format generated an arithmetic fault (overflow, underflow). This flag is cleared by a read of this register (r1c); reads of AMU2_SEINCCH*n* do not clear this flag. 0 No arithmetic fault associated with the conversion of z64 to z32. 1 Arithmetic fault associated with the conversion of z64 to z32 was detected. For the overflow case, the z32 value is set to (+/-) INF, while an underflow produces a (+/-) 0 value.
18 S23AFLT	Stage2, 3 Arithmetic Fault The assertion of this read-only flag indicates an arithmetic fault (overflow, underflow) was detected in the ALU's stage2 or stage3 calculation. The ALU's calculation is halted. This flag is cleared by a read of this register (r1c); reads of AMU2_SEINCCH*n* do not clear this flag. 0 No arithmetic fault associated with the ALU's stage3 calculation. 1 Arithmetic fault associated with the ALU's stage3 calculation was detected.
19 S1AFLT	Stage1 Arithmetic Fault The assertion of this read-only flag indicates an arithmetic fault (overflow, underflow) was detected in the ALU's stage1 calculation. The ALU's calculation is halted. This flag is cleared by a read of this register (r1c); reads of AMU2_SEINCCH*n* do not clear this flag. 0 No arithmetic fault associated with the ALU's stage1 calculation. 1 Arithmetic fault associated with the ALU's stage1 calculation was detected.
20 IDVERR	Input Data Validation Error The assertion of this read-only flag indicates an error (NaN, +/- INF) was detected in the input data before the ALU's calculation. The ALU's calculation is halted. This flag is cleared by a read of this register (r1c); reads of AMU2_SEINCCH*n* do not clear this flag. 0 No error associated with the input data 1 Arithmetic fault associated with the input data was detected

Table 2368. AMU2_SEIRCH*n* field descriptions (continued)

Field	Description
21 ADMERR	ADMA Error The assertion of this read-only flag indicates an error (such as bus error, illegal address, multi-bit ECC event) was detected by the ADMA. This flag does not include CRC checksum errors as these are directly signaled by SEIRCH*n*[CKSERR]. If the ADMA is streaming P3 or V data, or both, at the time of the error, this flag is set and the ALU calculation is stalled with AMU2_SEIRCH*n*[S{1,23}XS] indicating the running or stalled state. This flag is cleared by a read of this register (r1c); reads of AMU2_SEINCCH*n* do not clear this flag. 0 No bus error, illegal address, multi-bit ECC, such as errors associated with the ADMA 1 An ADMA bus error, illegal address, multi-bit ECC event, other errors, was detected
22 CKSERR	ADMA Checksum Error The assertion of this read-only flag indicates an ADMA CRC checksum error was detected. This flag is cleared by a read of this register (r1c); reads of AMU2_SEINCCH*n* do not clear this flag. 0 No CRC checksum error associated with the ADMA 1 An ADMA CRC checksum error was detected
23 ECCERR	Error Correcting Code Error The assertion of this read-only flag indicates a non-correctable ECC error was detected on an ARAM read by the ALU. Non-correctable ECC errors detected by the ADMA are reported with AMU2_SEIRCH*n*[ADMERR] = 1, while similar errors detected during a slave peripheral bus access from the CPU or SDMA are reported via an error-terminated bus cycle. This flag is cleared by a read of this register (r1c); reads of AMU2_SEINCCH*n* do not clear this flag. 0 No ARAM ECC error 1 An ARAM non-correctable ECC error was detected
24 RGCERR	Register Conflict Error The assertion of this read-only flag indicates an unexpected write of the AMU2's programming model while the ALU is executing was attempted. The protected registers include: AMU2_{P1* - P9*, VPTR, LPTR, UPTR}CH*n*. Attempted writes to the AMU2_{P1PTR, P2PTR, P7, UPTR}CH*n* registers while the ALU is executing stage 1 set this indicator bit and halt the ALU. Attempted writes to the AMU2_{P3PTR, P4, P5, P6, P7, P8, P9{M,L}, VPTR, LPTR}CH*n* registers while the ALU is executing stage 23 set this indicator bit and halt the ALU. The AMU2_{N, DSA, DDA, DCNT, DCKS, DCCR, CCR1, CCR2}CH*n* registers can safely be written while the ALU is executing. This flag is cleared by a read of this register (r1c); reads of AMU2_SEINCCH*n* do not clear this flag. 0 No register conflict error 1 An AMU2 register conflict error was detected

Table 2368. AMU2_SEIRCH*n* field descriptions (continued)

Field	Description
26 DEIRQ	ADMA Error Interrupt Request This flag is set to signal the ADMA has detected an error. The error conditions which set this indicator flag include: – Multi-bit ECC event during a data transfer (ADMERR = 1) – Bus error termination during a data transfer (ADMERR = 1) – Detection of a checksum error (CKSERR = 1) The subsequent system behavior is defined by the AMU2_DCCRCH*n*[DEICFG] field. This flag must be cleared by an explicit write of a 1 (“write 1 clear”) to the bit position. A read of this register does not affect the state of this indicator bit. 0 ADMA has not detected an error 1 ADMA has detected an error
27 ADIRQ	ADMA Interrupt ReQuest. This flag is set to signal the ADMA has completed its data transfer. This flag is not set when streaming P3[*] or V[*] data, or both, completes. The subsequent system behavior is defined by the AMU2_DCCRCH*n*[ADICFG] field. This flag must be cleared by an explicit write of a 1 (“write 1 clear”) to the bit position if AMU2_DCCRCH*n*[ADICFG] = 0b0-. For other configuration values, the SDMA hardware provides an acknowledgement signal which clears this indicator bit. A read of this register does not affect the state of this indicator bit. 0 ADMA has not completed its data transfer 1 ADMA has completed its data transfer
28 ERIRQ	ALU Error Interrupt Request This flag is set to signal the ALU has detected certain errors. The error conditions which set this indicator flag include: – Multi-bit ECC event on an ARAM read by the ALU (ECCERR = 1) – Detection of a register conflict error (RGCERR = 1) – Detection of an input data validation error (IDVERR = 1) The subsequent system behavior is defined by the AMU2_CCR1CH*n*[AEICFG] field. This flag must be cleared by an explicit write of a 1 (“write 1 clear”) to the bit position. A read of this register does not affect the state of this indicator bit. 0 ALU has not detected an error 1 ALU has detected an error
29 OVIRQ	ALU Overflow Interrupt Request This flag is set to signal the ALU has detected an overflow condition. The error conditions which set this indicator flag include: – Arithmetic overflow during stage1 (S1AFLT = 1), stage2 (S23AFLT = 1), or stage3 (S23AFLT = 1) – Overflow when performing 64-bit to 32-bit conversions for ut[*] (S1AFLT = 1), t (S23AFLT = 1), y (YCAFLT = 1), z (ZCAFLT =1) The ALU halts execution at specific places within the algorithm if an overflow is detected: stage1 overflows halt at the end of its calculation, stage2 overflows halt at the completion of the outer loop iteration and stage3 overflows complete at the end of its calculation. See *Table 2384: ALU Completion State Details in AMU2_SEIRCHn* for additional details. The subsequent system behavior is defined by the AMU2_CCR1CH*n*[OVICFG] field. This flag must be cleared by an explicit write of a 1 (“write 1 clear”) to the bit position. A read of this register does not affect the state of this indicator bit. 0 ALU has not detected an overflow condition 1 ALU has detected an overflow condition

Table 2368. AMU2_SEIRCH*n* field descriptions (continued)

Field	Description
30 NCIRQ	ALU N Crossing Interrupt Request This flag is set to signal the ALU has completed “N” iterations of the stage2 calculation in response to a properly configured N crossing event. The subsequent system behavior is defined by the AMU2_CCR1CH*n*[NCICFG] field. This flag must be cleared by an explicit write of a 1 (“write 1 clear”) to the bit position if AMU2_CCR1CH*n*[NCICFG] = 0b0-. For other configuration values, the SDMA or ADMA hardware provides an acknowledgement signal which clears this indicator bit. A read of this register does not affect the state of this indicator bit. 0 ALU has not completed N iterations of a properly-configured N crossing event 1 ALU has completed N iterations of a properly-configured N crossing event
31 DNIRQ	ALU Done Interrupt Request This flag is set to signal the ALU has successfully completed an error-free calculation. The subsequent system behavior is defined by the AMU2_CCR1CH*n*[DNICFG] field. This flag must be cleared by an explicit write of a 1 (“write 1 clear”) to the bit position if AMU2_CCR1CH*n*[DNICFG] = 0b0-. For other configuration values, the SDMA or ADMA hardware provides an acknowledgement signal which clears this indicator bit. A read of this register does not affect the state of this indicator bit. 0 ALU has not completed its calculation 1 ALU has completed its calculation

94.2.3.26 AMU Actual Checksum Channel *n* Register (AMU2_ACKSCH*n*)

This is the actual read-only ADMA checksum data register. The ADMA supports an optional calculation of a 32-bit checksum on the data transferred into the ARAM. This register defines the actual checksum result. If enabled, this register is cleared at the beginning of the ADMA transfer, the actual checksum accumulated by the ADMA in this register and then compared to expected value contained in the AMU2_DCKSCH*n* register at the completion of the data transfer. If the actual and expected checksums are equivalent, the AMU continues, else a data transmission error has been detected (AMU2_SEIRCH*n*[CKSERR] = 1) and the AMU responds according to the programmed configuration (AMU2_DCCRCH*n*[ADICFG]).

There are aliased copies of this register available at multiple addresses to allow the SDMA to retrieve the AMU2 result (either as a 64-bit double-precision or a 32-bit single-precision format based on the access address), the status register and this register in a simple, sequential block transfer. For example, a 16-byte block transfer beginning at offset address 0x1_0080 retrieves the AMU2_RES64{M,L}CH0 result, the AMU2_SEIRCH0 status and the AMU2_ACKSCH0 checksum result. Conversely, a 12-byte block transfer beginning at offset address 0x1_0090 retrieves the AMU2_RES32CH0 result, the aliased copy of the AMU2_SEIRCH0 register and the aliased copy of the AMU2_ACHKSCH0 register. Similar 16- and 12-byte block transfers in the CH1 section of the memory map return the comparable registers.

Offset: 0x1008C + n * 0x100 (n = 0 to 1) Access: Supervisor Read

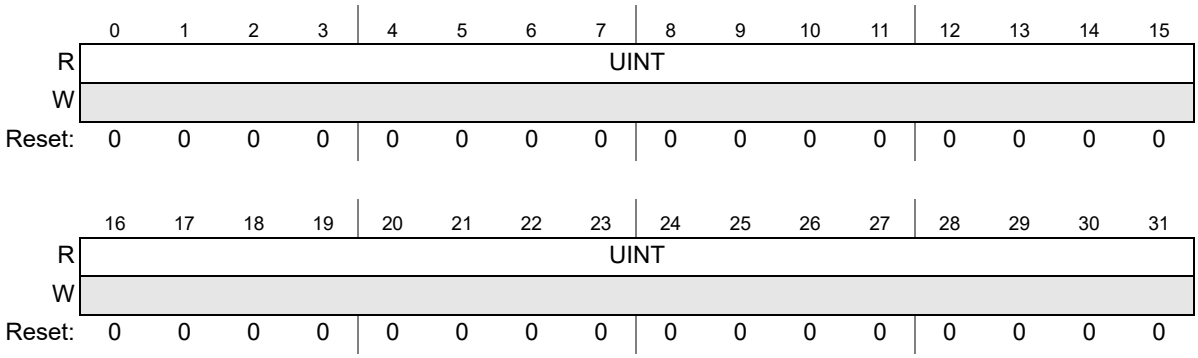


Figure 2482. AMU Actual Checksum Channel *n* Register (AMU2_ACKSCH*n*)

Table 2369. AMU2_ACKSCH*n* field descriptions

Field	Description
0:31 UINT	Actual ADMA checksum in a 32-bit unsigned integer format.

94.2.3.27 AMU RES32 Channel *n* Register (AMU2_RES32CH*n*)

This is the read-only 32-bit single-precision version of the final 64-bit double-precision result register, that is, this is “z32”, derived from the z64 final accumulation from the stage3 output normalization calculation.

Offset: 0x10090 + n * 0x100 (n = 0 to 1) Access: Supervisor Read

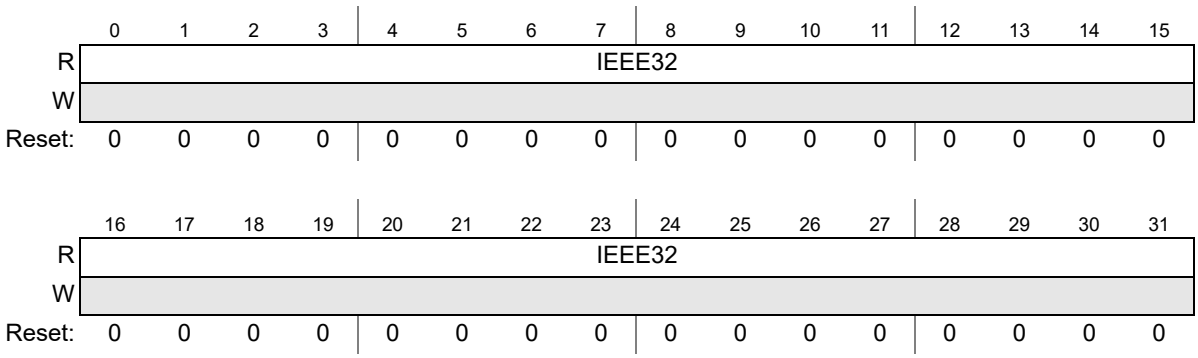


Figure 2483. AMU RES32 Channel *n* Register (AMU2_RES32CH*n*)

Table 2370. AMU2_RES32CH*n* field descriptions

Field	Description
0:31 IEEE32	Single-precision version (IEEE 32-bit floating-point format) of the final 64-bit result (z64) value.

94.2.3.28 Aliased copy of AMU Status, Error & Interrupt Channel *n* Register (AMU2_ASEIRCH*n*)

This is the AMU's status register. The most significant three bytes contain read-only status and error flags, while the low-order byte contains the interrupt flag bits. Specifically, there are 15 status bits signaling the state of the ADMA, stage1 and stage23 ALU calculations, along with the current speed settings, 9 error flags signaling arithmetic faults in the ALU or other error conditions, and 6 interrupt request flags.

There are aliased copies of this register available at multiple addresses to allow the SDMA to retrieve the AMU2 result (either as a 64-bit double-precision or a 32-bit single-precision format based on the access address) and this status register in a simple, sequential block transfer. For example, a 16-byte block transfer beginning at offset address 0x1_0080 retrieves the AMU2_RES64{M,L}CH0 result, the AMU2_SEIRCH0 status and the AMU2_ACKSCH0 checksum result. Conversely, a 12-byte block transfer beginning at offset address 0x1_0090 retrieves the AMU2_RES32CH0 result, the aliased copy of the AMU2_SEIRCH0 register and the aliased copy of the AMU2_ACHKSCH0 register. Similar 16- and 12-byte block transfers in the CH1 section of the memory map return the comparable registers.

Certain error flag indicators in this register are cleared by a read of the register; there is an alternate location of this register (see *Section 94.2.3.33: AMU Status, Error & Interrupt No Clear Channel n Register (AMU2_SEINCCHn)*) where a read does not affect these status flags. Accordingly, software polling of the CHBSY flag should use the alternate AMU2_SEINCCH*n* register so that no fault status bits are inadvertently cleared.

It should be noted the interrupt flags are intended to record the completion of an event or the occurrence of an error condition. These flags are set when the event occurs, while the subsequent reporting of the event is defined by the corresponding interrupt configuration field, specifically AMU2_DCCRCH*n*[ADICFG, DEICFG] and AMU2_CCR1CH*n*[AEICFG, OVICFG, NCICFG, DNICFG]. The event's reporting can be configured to *not* perform any type of asynchronous notification (intended for software polling applications), a processor interrupt, or depending on the event, a system DMA or ADMA request. For DMA requests (both system and AMU), the hardware clears the indicator event flag as the request is acknowledged; for the software polling configuration or processor interrupts, a service routine or another software task must explicitly clear the event flag by writing a logical 1 to the corresponding interrupt flag. This is a "write 1" clear protocol.

Offset: 0x10094 + n * 0x100 (n = 0 to 1) Access: Supervisor Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	CHBSY	ADSTM	S23XS		S1XS		ADMAS		QDSL	ACURSP			0	DCURSP		
W																
Reset:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	YCAFLT	ZCAFLT	S23AFLT	S1AFLT	IDVERR	ADMERR	CKSERR	ECCERR	RGCERR	0	DEIRQ	ADIRQ	ERIRQ	OVIRQ	NCIRQ	DNIRQ
W											w1c	w1c	w1c	w1c	w1c	w1c
Reset:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 2484. Aliased copy of AMU Status, Error & Interrupt Channel *n* Register (AMU2_ASEIRCH*n*)

Table 2371. AMU2_ASEIRCH*n* field descriptions

Field	Description
0 CHBSY	Channel Busy This read-only bit is intended to serve as a polling flag as it signals the AMU2 channel is busy. In this context, the “channel” status represents the logical summation of the ADMA data channel and the ALU calculation engine status fields indicating a non-idle “channel”. 0 Channel *n* is idle 1 Channel *n* is busy performing an ADMA data transfer or an ALU calculation
1 ADSTM	ADMA is Streaming This read-only flag signals whether the ADMA is performing a data streaming transfer of the P3[*] or V[*] input vectors as enabled by AMU2_DCCRCH*n*[STMP3, STMV{32, 16}] control bits. 0 The ADMA is not performing any P3[*] nor V[*] data streaming transfers 1 The ADMA is busy performing a P3[*] or V[*] data streaming transfers
2:3 S23XS	Stage23 Execution Status This read-only 2-bit field defines the execution status of the ALU’s stage23 logic. See *Section 94.3.5.1: ALU execution states* for more details on the execution states. The status encodings are: 00 Idle or pending start condition 01 Halted 10 Stopped 11 Running or stalled

Table 2371. AMU2_ASEIRCH*n* field descriptions (continued)

Field	Description
4:5 S1XS	Stage1 Execution Status This read-only 2-bit field defines the execution status of the ALU's stage1 logic. See *Section 94.3.5.1: ALU execution states* for more details on the execution states. The status encodings are: 00 Idle or pending start condition 01 Halted 10 Unused, reserved as stage1 execution cannot be stopped via a request 11 Running or stalled
6:7 ADMAS	ADMA Status This read-only 2-bit field defines the status of the ADMA. See *Section 94.3.6.2: ADMA execution states* for more details on the execution states. The status encodings are: 00 Idle or pending start condition 01 Halted 10 Stopped 11 Running or stalled
8 QDSL	Qualified Debug Stali This read-only bit signals a qualified debug stall condition as defined by (debug_event == 1) && (AMU2_CCR1CH*n*[DDSL] == 0). 0 No qualified debug stall 1 The ALU calculation engine and ADMA are stalled due to a qualified debug event.
9:11 ACURSP	ALU Current Speed This read-only 3-bit encoded value defines the current speed for the ALU engine. The ALU speed control is defined by AMU2_CCR1CH*n*[SRTSP, MAXSP] and AMU2_P8CH*n*[THRHC]. The ALU speed control provides a mechanism to control the operation efficiency of the submodule for coarse power control of the AMU. See *Section 94.3.5.2: ALU speed control* for more details on the ALU speed control mechanism. There is a single ALU speed control function shared by both channels. 000 Maximum speed; speed control disabled 001 1/8 speed 010 2/8 speed 011 3/8 speed 100 4/8 speed 101 5/8 speed 110 6/8 speed 111 7/8 speed
13:15 DCURSP	ADMA Current Speed This read-only 3-bit encoded value defines the current speed for the ADMA channel. The ADMA speed control is defined by AMU2_DCCRCH*n*[SRTSP, MAXSP] and AMU2_DCNTCH*n*[THRHC]. The ADMA speed control provides a mechanism to control the operation efficiency of the submodule for coarse power control of the AMU. See *Section 94.3.6.5: ADMA speed control* for more details on the ADMA speed control mechanism. There is a single ADMA speed control function shared by both channels. The DCURSP uses the same encodings as the ACURSP field above.

Table 2371. AMU2_ASEIRCH*n* field descriptions (continued)

Field	Description
16 YCAFLT	Y Conversion Arithmetic Fault The assertion of this read-only flag indicates the conversion from double-precision ieee64 format of the y64 result to single-precision ieee32 format generated an arithmetic fault (overflow, underflow). This flag is cleared by a read of this register (r1c); reads of AMU2_SEINCCH*n* do not clear this flag. 0 No arithmetic fault associated with the conversion of y64 to y32 1 Arithmetic fault associated with the conversion of y64 to y32 was detected. For the overflow case, the y32 value is set to (+/-) INF, while an underflow produces a (+/-) 0 value
17 ZCAFLT	Z Conversion Arithmetic Fault The assertion of this read-only flag indicates the conversion from double-precision ieee64 format of the z64 result to single-precision ieee32 format generated an arithmetic fault (overflow, underflow). This flag is cleared by a read of this register (r1c); reads of AMU2_SEINCCH*n* do not clear this flag. 0 No arithmetic fault associated with the conversion of z64 to z32 1 Arithmetic fault associated with the conversion of z64 to z32 was detected. For the overflow case, the z32 value is set to (+/-) INF, while an underflow produces a (+/-) 0 value
18 S23AFLT	Stage2, 3 Arithmetic Fault The assertion of this read-only flag indicates an arithmetic fault (overflow, underflow) was detected in the ALU's stage2 or stage3 calculation. The ALU's calculation is halted. This flag is cleared by a read of this register (r1c); reads of AMU2_SEINCCH*n* do not clear this flag. 0 No arithmetic fault associated with the ALU's stage3 calculation 1 Arithmetic fault associated with the ALU's stage3 calculation was detected
19 S1AFLT	Stage1 Arithmetic Fault The assertion of this read-only flag indicates an arithmetic fault (overflow, underflow) was detected in the ALU's stage1 calculation. The ALU's calculation is halted. This flag is cleared by a read of this register (r1c); reads of AMU2_SEINCCH*n* do not clear this flag. 0 No arithmetic fault associated with the ALU's stage1 calculation 1 Arithmetic fault associated with the ALU's stage1 calculation was detected
20 IDVERR	Input Data Validation Error The assertion of this read-only flag indicates an error (NaN, +/- INF) was detected in the input data before the ALU's calculation. The ALU's calculation is halted. This flag is cleared by a read of this register (r1c); reads of AMU2_SEINCCH*n* do not clear this flag. 0 No error associated with the input data. 1 Arithmetic fault associated with the input data was detected.

Table 2371. AMU2_ASEIRCH*n* field descriptions (continued)

Field	Description
21 ADMERR	ADMA Error The assertion of this read-only flag indicates an error (such as bus error, illegal address, multi-bit ECC event) was detected by the ADMA. This flag does *not* include CRC checksum errors as these are directly signaled by SEIRCH*n*[CKSERR]. If the ADMA is streaming P3 or V data, or both, at the time of the error, this flag is set and the ALU calculation is stalled with AMU2_SEIRCH*n*[S{1,23}XS] indicating the running or stalled state. This flag is cleared by a read of this register (r1c); reads of AMU2_SEINCCH*n* do not clear this flag. 0 No bus error, illegal address, multi-bit ECC, such as errors associated with the ADMA 1 An ADMA bus error, illegal address, multi-bit ECC event, other rerrors, was detected
22 CKSERR	ADMA Checksum Error The assertion of this read-only flag indicates an ADMA CRC checksum error was detected. This flag is cleared by a read of this register (r1c); reads of AMU2_SEINCCH*n* do not clear this flag. 0 No CRC checksum error associated with the ADMA 1 An ADMA CRC checksum error was detected
23 ECCERR	Error Correcting Code Error The assertion of this read-only flag indicates a non-correctable ECC error was detected on an ARAM read by the ALU. Non-correctable ECC errors detected by the ADMA are reported with AMU2_SEIRCH*n*[ADMERR] = 1, while similar errors detected during a slave peripheral bus access from the CPU or SDMA are reported via an error-terminated bus cycle. This flag is cleared by a read of this register (r1c); reads of AMU2_SEINCCH*n* do not clear this flag. 0 No ARAM ECC error 1 An ARAM non-correctable ECC error was detected
24 RGCERR	Register Conflict Error The assertion of this read-only flag indicates an unexpected write of the AMU2's programming model while the ALU is executing was attempted. The protected registers include: AMU2_{P1* - P9*, VPTR, LPTR, UPTR}CH*n*. Attempted writes to the AMU2_{P1PTR, P2PTR, P7, UPTR}CH*n* registers while the ALU is executing stage 1 set this indicator bit and halt the ALU. Attempted writes to the AMU2_{P3PTR, P4, P5, P6, P7, P8, P9{M,L}, VPTR, LPTR}CH*n* registers while the ALU is executing stage 23 set this indicator bit and halt the ALU. The AMU2_{N, DSA, DDA, DCNT, DCKS, DCCR, CCR1, CCR2}CH*n* registers can safely be written while the ALU is executing. This flag is cleared by a read of this register (r1c); reads of AMU2_SEINCCH*n* do not clear this flag. 0 No register conflict error 1 An AMU2 register conflict error was detected

Table 2371. AMU2_ASEIRCH*n* field descriptions (continued)

Field	Description
26 DEIRQ	ADMA Error Interrupt Request This flag is set to signal the ADMA has detected an error. The error conditions which set this indicator flag include: – Multi-bit ECC event during a data transfer (ADMERR = 1) – Bus error termination during a data transfer (ADMERR = 1) – Detection of a checksum error (CKSERR = 1) The subsequent system behavior is defined by the AMU2_DCCRCH*n*[DEICFG] field. This flag must be cleared by an explicit write of a 1 (“write 1 clear”) to the bit position. A read of this register does not affect the state of this indicator bit. 0 ADMA has not detected an error 1 ADMA has detected an error
27 ADIRQ	ADMA Interrupt Request This flag is set to signal the ADMA has completed its data transfer. This flag is not set when streaming P3[*] or V[*] data, or both, completes. The subsequent system behavior is defined by the AMU2_DCCRCH*n*[ADICFG] field. This flag must be cleared by an explicit write of a 1 (“write 1 clear”) to the bit position if AMU2_DCCRCH*n*[ADICFG] = 0b0-. For other configuration values, the SDMA hardware provides an acknowledgment signal which clears this indicator bit. A read of this register does not affect the state of this indicator bit. 0 ADMA has not completed its data transfer 1 ADMA has completed its data transfer
28 ERIRQ	ALU Error Interrupt Request This flag is set to signal the ALU has detected certain errors. The error conditions which set this indicator flag include: – Multi-bit ECC event on an ARAM read by the ALU (ECCERR = 1) – Detection of a register conflict error (RGCERR = 1) – Detection of an input data validation error (IDVERR = 1) The subsequent system behavior is defined by the AMU2_CCR1CH*n*[AEICFG] field. This flag must be cleared by an explicit write of a 1 (“write 1 clear”) to the bit position. A read of this register does not affect the state of this indicator bit. 0 ALU has not detected an error 1 ALU has detected an error

Table 2371. AMU2_ASEIRCH*n* field descriptions (continued)

Field	Description
29 OVIRQ	ALU Overflow Interrupt Request This flag is set to signal the ALU has detected an overflow condition. The error conditions which set this indicator flag include: – Arithmetic overflow during stage1 (S1AFLT = 1), stage2 (S23AFLT = 1), or stage3 (S23AFLT = 1) – Overflow when performing 64-bit to 32-bit conversions for ut[*] (S1AFLT = 1), t (S23AFLT = 1), y (YCAFLT = 1), z (ZCAFLT =1) The ALU halts execution at specific places within the algorithm if an overflow is detected: stage1 overflows halt at the end of its calculation, stage2 overflows halt at the completion of the outer loop iteration and stage3 overflows complete at the end of its calculation. See *Table 2384: ALU Completion State Details in AMU2_SEIRCHn* for additional details. The subsequent system behavior is defined by the AMU2_CCR1CH*n*[OVICFG] field. This flag must be cleared by an explicit write of a 1 (“write 1 clear”) to the bit position. A read of this register does not affect the state of this indicator bit. 0 ALU has not detected an overflow condition 1 ALU has detected an overflow condition
30 NCIRQ	ALU N Crossing Interrupt Request This flag is set to signal the ALU has completed “N” iterations of the stage2 calculation in response to a properly configured N crossing event. The subsequent system behavior is defined by the AMU2_CCR1CH*n*[NCICFG] field. This flag must be cleared by an explicit write of a 1 (“write 1 clear”) to the bit position if AMU2_CCR1CH*n*[NCICFG] = 0b0-. For other configuration values, the SDMA or ADMA hardware provides an acknowledgment signal which clears this indicator bit. A read of this register does not affect the state of this indicator bit. 0 ALU has not completed N iterations of a properly-configured N crossing event 1 ALU has completed N iterations of a properly-configured N crossing event
31 DNIRQ	ALU Done Interrupt Request This flag is set to signal the ALU has successfully completed an error-free calculation. The subsequent system behavior is defined by the AMU2_CCR1CH*n*[DNICFG] field. This flag must be cleared by an explicit write of a 1 (“write 1 clear”) to the bit position if AMU2_CCR1CH*n*[DNICFG] = 0b0-. For other configuration values, the SDMA or ADMA hardware provides an acknowledgment signal which clears this indicator bit. A read of this register does not affect the state of this indicator bit. 0 ALU has not completed its calculation 1 ALU has completed its calculation

94.2.3.29 Aliased copy of AMU Actual Checksum Channel *n* Register (AMU2_AACKSCH*n*)

This is the actual read-only ADMA checksum data register. The ADMA supports an optional calculation of a 32-bit checksum on the data transferred into the ARAM. This register defines the actual checksum result. If enabled, this register is cleared at the beginning of the ADMA transfer, the actual checksum accumulated by the ADMA in this register and then compared to expected value contained in the AMU2_DCKSCH*n* register at the completion of the data transfer. If the actual and expected checksums are equivalent, the AMU continues, else a data transmission error has been detected (AMU2_SEIRCH*n*[CKSERR] = 1) and the AMU responds according to the programmed configuration (AMU2_DCCRCH*n*[ADICFG]).

There are aliased copies of this register available at multiple addresses to allow the SDMA to retrieve the AMU2 result (either as a 64-bit double-precision or a 32-bit single-precision format based on the access address), the status register and this register in a simple, sequential block transfer. For example, a 16-byte block transfer beginning at offset address 0x1_0080 retrieves the AMU2_RES64{M,L}CH0 result, the AMU2_SEIRCH0 status and the AMU2_ACKSCH0 checksum result. Conversely, a 12-byte block transfer beginning at offset address 0x1_0090 retrieves the AMU2_RES32CH0 result, the aliased copy of the AMU2_SEIRCH0 register and the aliased copy of the AMU2_ACHKSCH0 register. Similar 16- and 12-byte block transfers in the CH1 section of the memory map return the comparable registers.

Offset: 0x10098 + n * 0x100 (n = 0 to 1) Access: Supervisor Read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	UINT															
W																
Reset:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	UINT															
W																
Reset:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 2485. Aliased copy of AMU Actual Checksum Channel *n* Register (AMU2_AACKSCH*n*)

Table 2372. AMU2_AACKSCH*n* field descriptions

Field	Description
0:31 UINT	Actual ADMA checksum in a 32-bit unsigned integer format.

94.2.3.30 AMU Y64M Channel *n* Register (AMU2_Y64MCH*n*)

This is the read-only 64-bit double-precision register containing the y64 value at the completion of each iteration of the outer (stage 2) loop. It is described here as the concatenation of two 32-bit words with Y64{M,L} defining the most significant and least significant words.

Offset: 0x100A0 + n * 0x100 (n = 0 to 1) Access: Supervisor Read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	IEEE64															
W																
Reset:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	IEEE64															
W																
Reset:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 2486. AMU Y64M Channel *n* Register (AMU2_Y64MCH*n*)

Table 2373. AMU2_Y64MCH*n* field description

Field	Description
0:31 IEEE64	Most significant word of the y64 value in IEEE 64-bit floating-point format at the completion of the stage2 outer loop. This is the Y64MCH*n* register.

94.2.3.31 AMU Y64L Channel *n* Register (AMU2_Y64LCH*n*)

This is the read-only 64-bit double-precision register containing the y64 value at the completion of each iteration of the outer (stage 2) loop. It is described here as the concatenation of two 32-bit words with Y64{M,L} defining the most significant and least significant words.

Offset: 0x100A4 + n * 0x100 (n = 0 to 1) Access: Supervisor Read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	IEEE64															
W																
Reset:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	IEEE64															
W																
Reset:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 2487. AMU Y64L Channel *n* Register (AMU2_Y64LCH*n*)

Table 2374. AMU2_Y64LCH*n* field description

Field	Description
0:31 IEEE64	Least significant word of the y64 value in IEEE 64-bit floating-point format at the completion of the stage2 outer loop. This is the Y64LCH*n* register.

94.2.3.32 Aliased copy of AMU P8 Channel *n* Register (AMU2_AP8CH*n*)

The P8 register contains two 16-bit unsigned integers related to the outer loop at the stage2 calculation. The upper halfword contains the threshold count for the ALU's speed control (see *Section 94.3.5.2: ALU speed control*), while the lower halfword is the starting value for the stage2 outer loop calculation.

As the stage2 outer loop iterates, the lower halfword of the AMU2_P8CH*n* register is incremented. If the calculation is stopped or halted, the register reflects the incremented loop count value. When the outer loop completes execution, it is restored to its initial value. This provides hardware support for the loop control such that this register does *not* have to be explicitly re-initialized for every calculation.

There is also an aliased read-only copy of this register available so that intermediate results associated with Y64 and P8 can be retrieved by the system DMA (SDMA) in a simple, sequential block transfer.

Offset: 0x100A8 + n * 0x100 (n = 0 to 1) Access: Supervisor Read

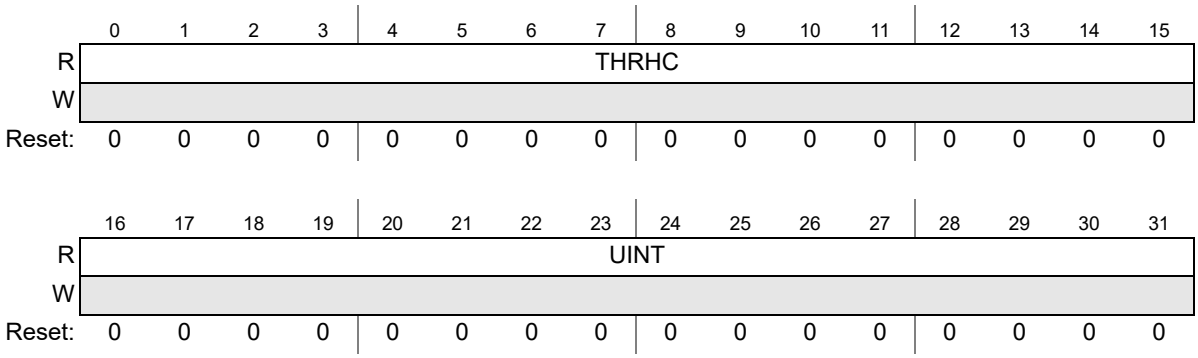


Figure 2488. AMU P8 Channel *n* Register (AMU2_AP8CH*n*)

Table 2375. AMU2_AP8CH*n* field descriptions

Field	Description
0:15 THRHC	Threshold Count This 16-bit unsigned integer defines the threshold count used in the ALU speed control function. There are additional control fields associated with this function in the AMU2_CCR1CH*n* register. There is a single ALU speed control function shared by the calculation engine and accessible from either the CH0 or CH1 programming models.
16:31 UINT	Parameter p8 value in a 16-bit unsigned integer format.

94.2.3.33 AMU Status, Error & Interrupt No Clear Channel *n* Register (AMU2_SEINCCH*n*)

This is an alternate view of the AMU's status register. It contains read-only status (SEIRCH*n*[31:16]) and error flags (SEIRCH*n*[15:7]), plus the module's interrupt bits (SEIRCH*n*[4:0]).

For this version of the register, specific error status indicators are not cleared by a read of the register. This specifically affects the operation of AMU2_SEINCCH*n*[YCAFLT, ZCAFLT, S23AFLT, S1AFLT, IDVERR, ADMERR, CKSERR, ECCERR, RGCERR].

This register is read-only. When clearing the state of the interrupt request flags, use the AMU2_SEIRCH*n* registers.

Offset: 0x100AC + n * 0x100 (n = 0 to 1) Access: Supervisor Read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	CHBSY	ADSTM	S23XS		S1XS		ADMAS		QDSL	ACURSP			0	DCURSP		
W																
Reset:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	YCAFLT	ZCAFLT	S23AFLT	S1AFLT	IDVERR	ADMERR	CKSERR	ECCERR	RGCERR	0	DEIRQ	ADIRQ	ERIRQ	OVIRQ	NCIRQ	DNIRQ
W																
Reset:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 2489. AMU Status, Error & Interrupt No Clear Channel n Register (AMU2_SEINCCHn)

Table 2376. AMU2_SEINCCHn field descriptions

Field	Description
0 CHBSY	Channel Busy This read-only bit is intended to serve as a polling flag as it signals the AMU2 channel is busy. In this context, the “channel” status represents the logical summation of the ADMA data channel and the ALU calculation engine status fields indicating a non-idle “channel”. 0 Channel *n* is idle 1 Channel *n* is busy performing an ADMA data transfer or an ALU calculation
1 ADSTM	ADMA is Streaming This read-only flag signals whether the ADMA is performing a data streaming transfer of the P3[*] or V[*] input vectors as enabled by AMU2_DCCRCH*n*[STMP3, STMV{32, 16}] control bits. 0 The ADMA is not performing any P3[*] nor V[*] data streaming transfers 1 The ADMA is busy performing a P3[*] or V[*] data streaming transfers
2:3 S23XS	Stage23 Execution Status This read-only 2-bit field defines the execution status of the ALU’s stage23 logic. See *Section 94.3.5.1: ALU execution states* for more details on the execution states. The status encodings are: 00 Idle or pending start condition 01 Halted 10 Stopped 11 Running or stalled
4:5 S1XS	Stage1 Execution Status This read-only 2-bit field defines the execution status of the ALU’s stage1 logic. See *Section 94.3.5.1: ALU execution states* for more details on the execution states. The status encodings are: 00 Idle or pending start condition 01 Halted 10 Unused, reserved as stage1 execu tion cannot be stopped via a request 11 Running or stalled

Table 2376. AMU2_SEINCCHn field descriptions (continued)

Field	Description
6:7 ADMAS	ADMA Status This read-only 2-bit field defines the status of the ADMA. See *Section 94.3.6.2: ADMA execution states* for more details on the execution states. The status encodings are: 00 Idle or pending start condition 01 Halted 10 Stopped 11 Running or stalled
8 QDSL	Qualified Debug Stali This read-only bit signals a qualified debug stall condition as defined by (debug_event == 1) && (AMU2_CCR1CH*n*[DDSL] == 0). 0 No qualified debug stall 1 The ALU calculation engine and ADMA are stalled due to a qualified debug event
9:11 ACURSP	Alu CURrent SPeed. This read-only 3-bit encoded value defines the current speed for the ALU engine. The ALU speed control is defined by AMU2_CCR1CH*n*[SRTSP, MAXSP] and AMU2_P8CH*n*[THRHC]. The ALU speed control provides a mechanism to control the operation efficiency of the submodule for coarse power control of the AMU. See *Section 94.3.5.2: ALU speed control* for more details on the ALU speed control mechanism. There is a single ALU speed control function shared by both channels. 000 Maximum speed; speed control disabled 001 1/8 speed 010 2/8 speed 011 3/8 speed 100 4/8 speed 101 5/8 speed 110 6/8 speed 111 7/8 speed
13:15 DCURSP	ADMA Current Speed This read-only 3-bit encoded value defines the current speed for the ADMA channel. The ADMA speed control is defined by AMU2_DCCRCH*n*[SRTSP, MAXSP] and AMU2_DCNTCH*n*[THRHC]. The ADMA speed control provides a mechanism to control the operation efficiency of the submodule for coarse power control of the AMU. See *Section 94.3.6.5: ADMA speed control* for more details on the ADMA speed control mechanism. There is a single ADMA speed control function shared by both channels. The DCURSP uses the same encodings as the ACURSP field above.
16 YCAFLT	Y Conversion Arithmetic Fault The assertion of this read-only flag indicates the conversion from double-precision ieee64 format of the y64 result to single-precision ieee32 format generated an arithmetic fault (overflow, underflow). This flag is cleared by a read of this register (r1c); reads of AMU2_SEINCCH*n* do not clear this flag. 0 No arithmetic fault associated with the conversion of y64 to y32 1 Arithmetic fault associated with the conversion of y64 to y32 was detected. For the overflow case, the y32 value is set to (+/-) INF, while an underflow produces a (+/-) 0 value

Table 2376. AMU2_SEINCCHn field descriptions (continued)

Field	Description
17 ZCAFLT	Z Conversion Arithmetic Fault The assertion of this read-only flag indicates the conversion from double-precision ieee64 format of the z64 result to single-precision ieee32 format generated an arithmetic fault (overflow, underflow). This flag is cleared by a read of this register (r1c); reads of AMU2_SEINCCH*n* do not clear this flag. 0 No arithmetic fault associated with the conversion of z64 to z32. 1 Arithmetic fault associated with the conversion of z64 to z32 was detected. For the overflow case, the z32 value is set to (+/-) INF, while an underflow produces a (+/-) 0 value.
18 S23AFLT	Stage2, 3 Arithmetic Fault The assertion of this read-only flag indicates an arithmetic fault (overflow, underflow) was detected in the ALU's stage2 or stage3 calculation. The ALU's calculation is halted. This flag is cleared by a read of this register (r1c); reads of AMU2_SEINCCH*n* do not clear this flag. 0 No arithmetic fault associated with the ALU's stage3 calculation 1 Arithmetic fault associated with the ALU's stage3 calculation was detected
19 S1AFLT	Stage1 Arithmetic Fault The assertion of this read-only flag indicates an arithmetic fault (overflow, underflow) was detected in the ALU's stage1 calculation. The ALU's calculation is halted. This flag is cleared by a read of this register (r1c); reads of AMU2_SEINCCH*n* do not clear this flag. 0 No arithmetic fault associated with the ALU's stage1 calculation 1 Arithmetic fault associated with the ALU's stage1 calculation was detected
20 IDVERR	Input Data Validation Error The assertion of this read-only flag indicates an error (NaN, +/- INF) was detected in the input data before the ALU's calculation. The ALU's calculation is halted. This flag is cleared by a read of this register (r1c); reads of AMU2_SEINCCH*n* do not clear this flag. 0 No error associated with the input data 1 Arithmetic fault associated with the input data was detected
21 ADMERR	ADMA Error The assertion of this read-only flag indicates an error (such bus error, illegal address, multi-bit ECC event) was detected by the ADMA. This flag does *not* include CRC checksum errors as these are directly signaled by SEIRCH*n*[CKSERR]. If the ADMA is streaming P3 or V data, or both, at the time of the error, this flag is set and the ALU calculation is stalled with AMU2_SEIRCH*n*[S{1,23}XS] indicating the running or stalled state. This flag is cleared by a read of this register (r1c); reads of AMU2_SEINCCH*n* do not clear this flag. 0 No bus error, illegal address, multi-bit ECC, such as errors associated with the ADMA 1 An ADMA bus error, illegal address, multi-bit ECC event, other errors, was detected

Table 2376. AMU2_SEINCCHn field descriptions (continued)

Field	Description
22 CKSERR	ADMA Checksum Error The assertion of this read-only flag indicates an ADMA CRC checksum error was detected. This flag is cleared by a read of this register (r1c); reads of AMU2_SEINCCH*n* do not clear this flag. 0 No CRC checksum error associated with the ADMA 1 An ADMA CRC checksum error was detected
23 ECCERR	Error Correcting Code Error The assertion of this read-only flag indicates a non-correctable ECC error was detected on an ARAM read by the ALU. Non-correctable ECC errors detected by the ADMA are reported with AMU2_SEIRCH*n*[ADMERR] = 1, while similar errors detected during a slave peripheral bus access from the CPU or SDMA are reported via an error-terminated bus cycle. This flag is cleared by a read of this register (r1c); reads of AMU2_SEINCCH*n* do not clear this flag. 0 No ARAM ECC error 1 An ARAM non-correctable ECC error was detected
24 RGCERR	Register Conflict Error The assertion of this read-only flag indicates an unexpected write of the AMU2's programming model while the ALU is executing was attempted. The protected registers include: AMU2_{P1* - P9*, VPTR, LPTR, UPTR}CH*n*. Attempted writes to the AMU2_{P1PTR, P2PTR, P7, UPTR}CH*n* registers while the ALU is executing stage 1 set this indicator bit and halt the ALU. Attempted writes to the AMU2_{P3PTR, P4, P5, P6, P7, P8, P9{M,L}, VPTR, LPTR}CH*n* registers while the ALU is executing stage 23 set this indicator bit and halt the ALU. The AMU2_{N, DSA, DDA, DCNT, DCKS, DCCR, CCR1, CCR2}CH*n* registers can safely be written while the ALU is executing. This flag is cleared by a read of this register (r1c); reads of AMU2_SEINCCH*n* do not clear this flag. 0 No register conflict error 1 An AMU2 register conflict error was detected
26 DEIRQ	ADMA Error Interrupt Request This flag is set to signal the ADMA has detected an error. The error conditions which set this indicator flag include: – Multi-bit ECC event during a data transfer (ADMERR = 1) – Bus error termination during a data transfer (ADMERR = 1) – Detection of a checksum error (CKSERR = 1) The subsequent system behavior is defined by the AMU2_DCCRCH*n*[DEICFG] field. This flag must be cleared by an explicit write of a 1 ("write 1 clear") to the bit position. A read of this register does not affect the state of this indicator bit. 0 ADMA has not detected an error 1 ADMA has detected an error.

Table 2376. AMU2_SEINCCHn field descriptions (continued)

Field	Description
27 ADIRQ	ADMA Interrupt Request This flag is set to signal the ADMA has completed its data transfer. This flag is not set when streaming P3[*] or V[*] data, or both, completes. The subsequent system behavior is defined by the AMU2_DCCRCH*n*[ADICFG] field. This flag must be cleared by an explicit write of a 1 (“write 1 clear”) to the bit position if AMU2_DCCRCH*n*[ADICFG] = 0b0-. For other configuration values, the SDMA hardware provides an acknowledgment signal which clears this indicator bit. A read of this register does not affect the state of this indicator bit. 0 ADMA has not completed its data transfer 1 ADMA has completed its data transfer
28 ERIRQ	ALU Error Interrupt Request This flag is set to signal the ALU has detected certain errors. The error conditions which set this indicator flag include: – Multi-bit ECC event on an ARAM read by the ALU (ECCERR = 1) – -Detection of a register conflict error (RGCERR = 1) – Detection of an input data validation error (IDVERR = 1) The subsequent system behavior is defined by the AMU2_CCR1CH*n*[AEICFG] field. This flag must be cleared by an explicit write of a 1 (“write 1 clear”) to the bit position. A read of this register does not affect the state of this indicator bit. 0 ALU has not detected an error 1 ALU has detected an error
29 OVIRQ	ALU Overflow Interrupt Request This flag is set to signal the ALU has detected an overflow condition. The error conditions which set this indicator flag include: – Arithmetic overflow during stage1 (S1AFLT = 1), stage2 (S23AFLT = 1), or stage3 (S23AFLT = 1) – Overflow when performing 64-bit to 32-bit conversions for ut[*] (S1AFLT = 1), t (S23AFLT = 1), y (YCAFLT = 1), z (ZCAFLT =1) The ALU halts execution at specific places within the algorithm if an overflow is detected: stage1 overflows halt at the end of its calculation, stage2 overflows halt at the completion of the outer loop iteration and stage3 overflows complete at the end of its calculation. See *Table 2384: ALU Completion State Details in AMU2_SEIRCHn* for additional details. The subsequent system behavior is defined by the AMU2_CCR1CH*n*[OVICFG] field. This flag must be cleared by an explicit write of a 1 (“write 1 clear”) to the bit position. A read of this register does not affect the state of this indicator bit. 0 ALU has not detected an overflow condition 1 ALU has detected an overflow condition

Table 2376. AMU2_SEINCCHn field descriptions (continued)

Field	Description
30 NCIRQ	ALU N Crossing Interrupt Request This flag is set to signal the ALU has completed “N” iterations of the stage2 calculation in response to a properly configured N crossing event. The subsequent system behavior is defined by the AMU2_CCR1CH*n*[NCICFG] field. This flag must be cleared by an explicit write of a 1 (“write 1 clear”) to the bit position if AMU2_CCR1CH*n*[NCICFG] = 0b0-. For other configuration values, the SDMA or ADMA hardware provides an acknowledgment signal which clears this indicator bit. A read of this register does not affect the state of this indicator bit. 0 ALU has not completed N iterations of a properly-configured N crossing event 1 ALU has completed N iterations of a properly-configured N crossing event
31 DNIRQ	ALU Done Interrupt Request This flag is set to signal the ALU has successfully completed an error-free calculation. The subsequent system behavior is defined by the AMU2_CCR1CH*n*[DNICFG] field. This flag must be cleared by an explicit write of a 1 (“write 1 clear”) to the bit position if AMU2_CCR1CH*n*[DNICFG] = 0b0-. For other configuration values, the SDMA or ADMA hardware provides an acknowledgment signal which clears this indicator bit. A read of this register does not affect the state of this indicator bit. 0 ALU has not completed its calculation 1 ALU has completed its calculation

94.2.3.34 AMU Y32 Channel *n* Register (AMU2_Y32CH*n*)

This is a read-only 32-bit single-precision (“y32”) version of the 64-bit double-precision y64 value at the completion of each iteration of the outer (stage 2) loop.

Offset: 0x100B0 + n * 0x100 (n = 0 to 1) Access: Supervisor Read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	IEEE32															
W																
Reset:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	IEEE32															
W																
Reset:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 2490. AMU Y32 Channel *n* Register (AMU2_Y32CH*n*)

Table 2377. AMU2_Y32CH*n* field descriptions

Field	Description
0:31 IEEE32	Single-precision version (IEEE 32-bit floating-point format) of the 64-bit result (y64) value.

94.3 Functional description

The AMU2 includes two independent channels, named channel 0 (ch0) and channel 1 (ch1), and featuring independent registers. Each channel logically includes a calculation engine and a DMA channel. The calculation engines share the same arithmetic logic using a time multiplexed scheme. Likewise, the DMA channels use a time multiplexed scheme to share common interfaces for fetching from the system Flash memory and writing into the ARAM. Once a calculation is initiated, the ALU engine executes the defined algorithm using data from the local ARAM (which is visible in the system memory map via the peripheral bus) or data streamed into the module from the private Flash connection by the ADMA. The ADMA executes data transfers from the system Flash (or the calibration remapped destination memories, or both) to the ARAM using a private interconnect.

94.3.1 AMU2 algorithm

As mentioned previously, the defined floating-point algorithm includes several loops with vector multiply-accumulate operations plus an evaluation of an exponential function while calculating a single double-precision output value. The algorithm is partitioned into 3 stages:

- Stage1: Input normalization within a single for() loop
- Stage2: Within two nested for() loops, accumulation of t in the inner loop, followed by calculation of exp(-t) and accumulation of y in the outer loop
- Stage3: Output normalization using y

The AMU2's hardware structure segments the algorithm into two parts for implementation: stage1 and stage23, where the latter includes both stage2 + stage3. The terms stage1, stage2 and stage23 are used throughout the document.

The stage1 and stage23 algorithms are defined in the following C pseudo-code. This description uses explicit references to `ieeexy` tags where *xy* = {16, 32, 64} defines the cast (and associated rounding) to the indicated precision in the subsequent floating-point calculations and `uintxy` defines an unsigned *xy*-bit integer.

94.3.1.1 Stage1 of the algorithm

```
void amuStage1(
    ieee32*  u,             // pointer to  u[p7] 32-bit single data array
    ieee32*  p1,            // pointer to p1[p7] 32-bit single data array
    ieee32*  p2,            // pointer to p2[p7] 32-bit single data array
    ieee32*  ut,            // pointer to ut[p7] 32-bit single data array
    uint05   p7             // dimension of the ASC function
) {
    uint16   k;             // local loop counter

    /* stage 1 */
    for (k = 0; k < p7; k++) {
        ut[k] = (ieee32) (u[k] * p1[k] + (ieee64) p2[k]); // 32x32 -> 64 FMA
    }
}
```

94.3.1.2 Stage23 (2 & 3) of the algorithm

```
void amuStage23(
    ieee32*  ut,            // pointer to ut[p7] 32-bit single data array
    uint01   v16,           // v[p6*p7] is 16-bit data array
    uint01   vfh,           // v[p6*p7] is 16-bit half precision float
    ieee32*  v32f,          // pointer to v[p6*p7] as ieee32
    ieee16*  v16h,          // pointer to v[p6*p7] as ieee16
    uint16*  v16x,          // pointer to v[p6*p7] as 16-bit fixed-point
    uint07   cmExp,         // v[p6*p7] common exponent for 16-bit fixed
    ieee32*  L,             // pointer to  L[p7] 32-bit single data array
    ieee32*  p3,            // pointer to p3[p7] 32-bit single data array
    ieee32   p4,            // scalar factor in the output normalization
    ieee32   p5,            // optional start value in the output
normalization
    uint16   p6,            // size of the ASC function
    uint05   p7,            // dimension of the ASC function
    uint16   &p8,           // outer stage2 "for" loop count
    ieee64   p9,            // starting value of double-precision y
accumulator
    uint01   setz,          // configuration to define initial z64 value
    ieee64   y64,           // 64-bit double-precision outer loop output
value
    ieee32   y32,           // 32-bit single-precision version of y64
    ieee64   z64,           // 64-bit double-precision final output value
    ieee32   z32            // 32-bit single-precision version of final
output
) {
    ieee32   d, e, v_ieee32;// internal stage23 local single-precision
variables
    ieee64   t;             // internal stage23 local double-precision
accumulator
    uint16   p8_init;       // internal unsigned integer variable
    uint16   i,j,k;         // internal unsigned integer array index (i),
loop count variables (j,k)

    /* data initialization */
    y64 = p9;
    p8_init = p8;

    /* stage 2: t accumulation, exp(-t), y accumulation */
    for (j = p8; j < p6; j++) {
        i = j * p7;

        /* stage 2a: t accumulation */
        t = (ieee64) 0.0;
```

```
            for (k = 0; k < p7; k++) {
                switch ((v16 << 1) + vfh) {
                    case 0:
                    case 1:                 // V array = ieee32 format
                        v_ieee32 = v32f[i+k];
                        break;
                    case 2:                 // V array = ieee16 format
                        v_ieee32 = (ieee32) v16h[i+k];
                        break;
                    case 3:
                        v_ieee32 = convertFixedPoint (cmExp-63, v16x[i+k]);
                        break;
                }
                d = ut[k] - v_ieee32;
                d = (ieee32)(d * d);
                t += (L[k] * d);            // 32 x 32 -> 64 FMA
            }

            /* stage 2b: exp() */
            e = amuExp(-(ieee32)t);         // amu exp()

            /* stage 2c: y64 accumulation */
            y64 += p3[j] * e;               // 32 x 32 -> 64 FMA
            p8++;                           // update global loop counter

        // user stop requests pause the algorithm in-place at this point
        }

        y32 = (ieee32) y64;                 // ieee32 version of y

        /* stage 3: output normalization */
        if (setz) {
            z64 = (ieee64) p5;
        }

        z64 = z64 + y64 * p4;               // 32 x 32 -> 64 FMA
        z32 = (ieee32) z64;

        p8 = p8_init;                       // return p8 to original value

    }
```

94.3.2 The exp(x) function and implementation

As previously described, a single-precision evaluation of the exp(x) is a key function in the defined algorithm (part of the stage2 outer loop calculations). The AMU2's implementation consists of four steps:

1. Special case analysis, for example, value out of range produces +Inf
2. Argument reduction
3. Function evaluation
4. Range reconstruction

94.3.2.1 Special case analysis

The first step of the exp(x) function consists of a series of conditional tests to filter out these special cases and perform the proper operation, for example returning an infinity when the input argument is too large.

94.3.2.2 Argument reduction

Standard algorithms for exp(x) typically approximate e^x in a bounded domain, typically defined as [-ln(2)/2, ln(2)/2] or [0, ln(2)]. The AMU2 implementation uses the [0, ln(2)) range. Further, the following mathematical identity is applied to the function definition:

$e^x = e^{E*\ln(2)+z} = (e^{\ln(2)})^E * e^z = 2^E * e^z$

where E is a signed *integer* and z is such that $|z| < \ln(2)$. After a series of arithmetic manipulations, z and E can be computed as:

$E = \text{int}(x / \ln(2))$

$z = x - E * \ln(2)$

where E is the integer part of the (x/ln(2)) computation with the same sign.

Further, the e^z computation can be rewritten as:

$e^z = e^{(A+B)} = e^A * e^B$

The range of the input argument can be further reduced by rewriting z as the sum of two fixed-point numbers such that z = A + B. This can be accomplished trivially by assigning the first *n* bits of z to A and the remaining bits of z to B. For the AMU2, the first 6 bits of the input argument are assigned to A, and the remaining 17 bits to B, thus allowing the use of a simple 64-element lookup table to compute e^A and a polynomial approximation for e^B.

94.3.2.3 Function evaluation

The next step is the selection of a polynomial approximation of e^B that is sufficiently accurate in the range [0, ln(2)/64]. In the AMU2, a Minimax approximation is used to model the e^x function with a low-degree polynomial. Minimax approximations are polynomials of degree N that calculate a given function within a given interval such that the absolute maximum error is minimized. This allows an error bound on the approximation *without* incurring unnecessary terms in the polynomial.

The difficult part of finding a Minimax polynomial is the iterative nature of the algorithm to compute the coefficients. For this, the Remez Algorithm as implemented in an indefinite precision solver is applied. The form of the polynomial is:

$f(x) = a_0 + a_1 * x + a_2 * x^2 + a_3 * x^3 + ...$

Using the Remez Algorithm, the function implements a three-term expression of the form:

$exp(x) = 1 + x * (C_0 + x * (C_1 + x* C_2))$

where C_n represent the required coefficients and x is the range-limited input argument. This produces the following expression in a more typical polynomial form:

$exp(x) = 1 + C_0 * x + C_1 * x^2 + C_2 * x^3$

In solving for the C_n coefficients to provide the minimum error, it becomes obvious that C_0 is very close to 1.0 and C_1 is reasonably close to 0.5. Since the assignment of these two coefficients to these constants allows a simplified hardware implementation, the next step is calculation of a revised C_2 coefficient to account for the simplifications involving C_0 and C_1. This calculation produces the following coefficient values:

$C_0 = 1.0$

$C_1 = 0.5$

$C_2 = 0.167876224340382$

Using this form of the exp(x) function and comparing its precision and accuracy versus Excel's computation of e^x over the range [0, ln(2)/64] shows the following results:

- Error terms approximately bounded as [-7.0E-10, +7.0E-10]
- > 30 bits of precision across the entire input range

The AMU2 limits the datapath width to 25 bits to minimize the exp() implementation cost.

94.3.2.4 Range reconstruction

The final step in the exp(x) function is the use of multiplicative range reconstruction to return the answer back into the proper range. Recall the equations of interest are:

$e^x = 2^E * e^z$

$e^z = e^{(A + B)} = e^A * e^B$

$exp(x) = 2^E * e^A * e^B$

where E was calculated during argument reduction, e^A from a table lookup and e^B using the polynomial evaluation. Finally, the result must be converted into a single-precision floating point number. Due to the nature of floating point numbers, this is a simple step; because the range of z is [0, ln(2)), the range of e^z is [1, 2). Since floating point numbers must be normalized to the range [1, 2), the result is guaranteed to be normalized, and in the proper range. The calculated e^z is stored as the single-precision mantissa and E as the exponent.

Note: If the exp(x) evaluation generates an overflow, the single-precision function result is set to positive infinity (+Inf = 0x7F800000). Likewise, if an underflow is generated, the result is set to a positive zero (+0 = 0x00000000).

The AMU2 includes a single instance of the exp(x) hardware function which is shared via time-multiplexing across the two calculation channels.

94.3.3 AMU data types and formats

The algorithm calculations performed by the AMU2 use floating-point values for the parameters and data, and unsigned integers for all memory pointers and loop counts. This section provides details on the supported floating-point data types as well as one additional format available for the V[*] array.

As an example of the data set size, consider the following two variables as a typical use case:

- Dimensionality of the ASC function, p7, with an example value of 10
- Size of the ASC function, p6, with an example value of 500

Based on these two variables, the input vector data can be categorized into three sizes:

- The V[*] array has a size of p6 * p7 = 500 * 10 = 5000 elements
- The P3[*] vector has a size of p6 = 500 elements
- All remaining vectors (L[*], P1[*], P2[*], u[*], ut[*]) have a size of p7 = 10 elements

94.3.3.1 Floating-point data formats

The AMU2 uses 16-bit half-, 32-bit single- and 64-bit double-precision floating-point data formats as defined in the IEEE 754-2008 “Standard for Floating-Point Arithmetic” specification. These formats are described in the following subsections. The multiple floating-point formats are used to minimize the data memory storage requirements for input operands and increase computational accuracy while allowing a reasonable module implementation size.

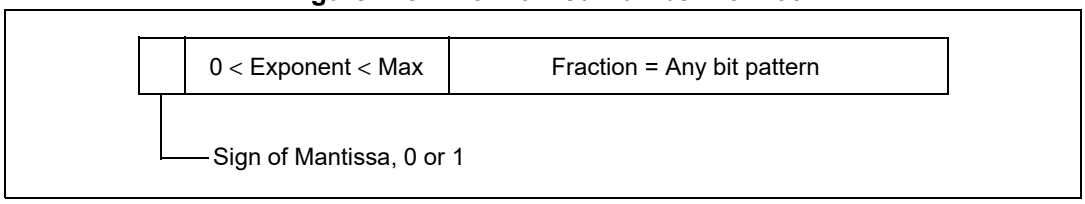
Each floating-point format supports five unique data types: normalized numbers, zeros, infinities, NaNs, and denormalized numbers. The normalized data type, as depicted in *Figure 2491*, never uses the maximum or minimum exponent value for a given format.

94.3.3.1.1 Normalized numbers

Normalized numbers include all positive or negative numbers with exponents between the maximum and minimum values. For half-, single- and double-precision normalized numbers, the implied integer bit (the so-called “hidden bit”, see more in the next paragraph) is one and the biased exponent cannot be zero.

In IEEE 754-2008, all normalized numbers are represented in sign-magnitude format as +/- 1.fff...fff * 2^e, where fff...fff is called the significant (or mantissa; it represents the fractional portion of the number), and e is called the exponent. Since all normalized numbers have a “1” as the first bit, it need not be stored in memory, and thus is often called the “hidden bit.” Implementations detect whether a number is normal or denormal by inspecting the exponent value. Denormalized numbers always have a zero exponent, and normalized numbers always have a non-zero exponent.

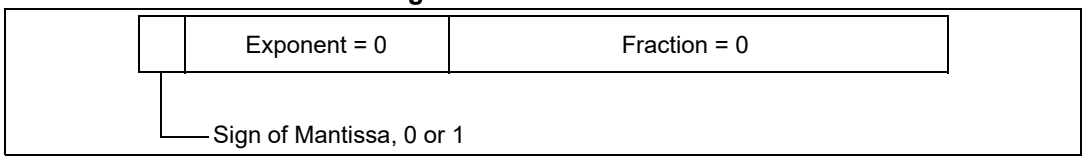
Figure 2491. Normalized Number Format



94.3.3.1.2 Zeros

Zeros can be positive or negative and represent real values, +0.0 and –0.0. See *Figure 2492*. The difference between +0.0 and -0.0 is significant: +0.0 is used to represent values that approach zero from the positive direction, and -0.0 is used to represent values that approach zero from the negative direction.

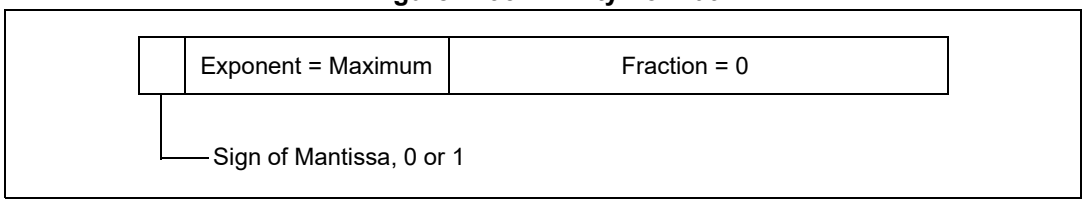
Figure 2492. Zero Format



94.3.3.1.3 Infinities

Infinities can be positive or negative and represent real values that exceed the overflow threshold. A result's exponent greater than or equal to the maximum exponent value indicates an overflow for a given data format and operation. This overflow description ignores the effects of rounding. For half-, single- and double-precision infinities, the fraction is a zero. See *Figure 2493*.

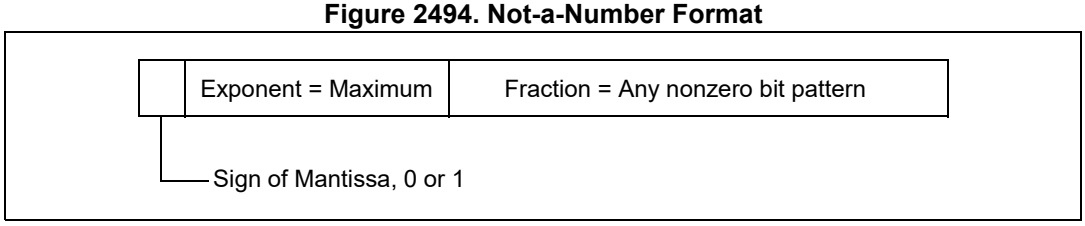
Figure 2493. Infinity Format



94.3.3.1.4 Not-A-Number

NaNs represent the results of operations having no mathematical interpretation, such as infinity divided by infinity. Operations using a NaN operand as an input return a NaN result. User-created NaNs can protect against uninitialized variables and arrays. See *Figure 2494*.

Figure 2494. Not-a-Number Format



The IEEE 754 specification defines quiet and signaling NaNs, based on the state of the most significant bit of the fraction. If set, the NaN is quiet, else it is a signaling NaN.

As the AMU2 performs each algorithm calculation, if one (or more) input operand is a NaN, the logic selects one NaN input and passes it as the result of that computation. Accordingly, if any input operand is a NaN, the result of the ALU calculation is also a NAN.

The IEEE 754 specification also defines a calculation producing an invalid operation should generate a quiet NaN that provides some diagnostic information. Accordingly, the AMU2 produces the following quiet NaN results from these invalid operations:

```
Inf - Inf// ieee32 = 0x7fc0_0001, ieee64 = 0x7ff8_0000_0000_0001
Inf * 0// ieee32 = 0x7fc0_0002, ieee64 = 0x7ff8_0000_0000_0002
0   * Inf// ieee32 = 0x7fc0_0003, ieee64 = 0x7ff8_0000_0000_0003
FMA(0,  Inf,  x)// ieee32 = 0x7fc0_0004, ieee64 = 0x7ff8_0000_0000_0004
FMA(Inf,  0,  x)// ieee32 = 0x7fc0_0005, ieee64 = 0x7ff8_0000_0000_0005
FMA(Inf,  0, -Inf)// ieee32 = 0x7fc0_0006, ieee64 = 0x7ff8_0000_0000_0006
FMA(x,  Inf, -Inf)// ieee32 = 0x7fc0_0006, ieee64 = 0x7ff8_0000_0000_0006
```

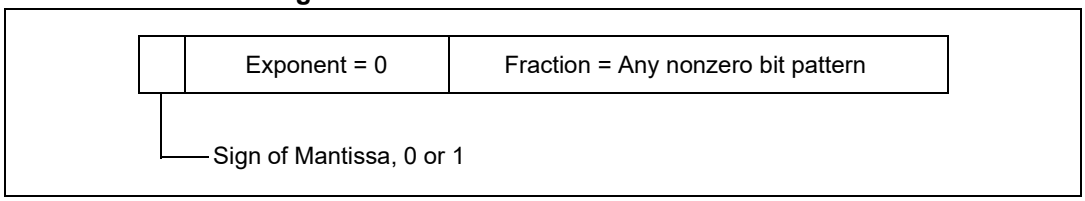
The following definition is used when converting an ieee64 NaN into an ieee32 NaN:

```
ieee32[31] = ieee64[63]// copy the sign bit
ieee32[30:23] = 0xff    // exponent = maximum
ieee32[22] = ieee64[51]// copy (signaling, quiet) bit from the ieee64 NaN
ieee32[21:0]  = if (ieee64[50:22] == 0) then ieee64[21:0], else 0x3f_ffff
```

94.3.3.1.5 Denormalized numbers

Denormalized numbers represent real values near the underflow threshold. Denormalized numbers can be positive or negative. For denormalized numbers in half-, single- and double-precision, the implied (hidden) integer bit is a zero and the biased exponent is zero. See *Figure 2495*.

Figure 2495. Denormalized Number Format



Traditionally, the detection of underflow causes floating-point number systems to perform a flush-to-zero. The IEEE 754 standard implements gradual underflow: the result mantissa is shifted right (denormalized) while the result exponent is incremented until reaching the minimum value. If all the mantissa bits of the result are shifted off to the right during this denormalization, the result becomes zero.

In the AMU2, all denormalized numbers are treated as zeros.

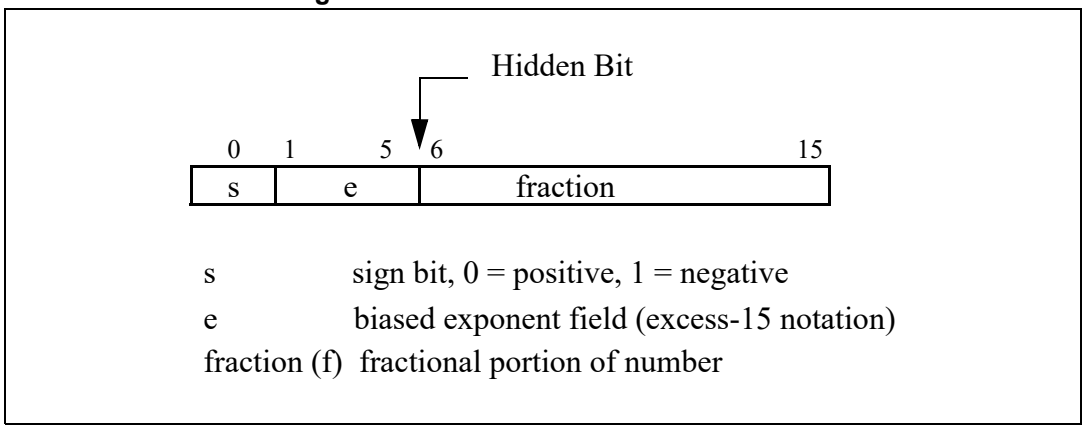
94.3.3.1.6 Half-precision floating-point format

Half-precision (16-bit) floating-point storage format is supported by the AMU2 only for the V[*] array with automatic conversion to the 32-bit single-precision floating-point format. No computational operations are defined for half-precision numbers as this format is supported *to minimize the memory storage associated with the V[*] array*. Recall the V array is two dimensional of size [p6 * p7].

Each half-precision floating-point data element is 16 bits wide with one sign bit (s), 5 bits of biased exponent (*e*) and 10 bits of fraction (*f*).

In the IEEE 754-2008 specification, half-precision floating-point values are represented in a format consisting of three explicit fields (sign field, biased exponent field, and fraction field) and an implicit hidden bit.

Figure 2496. Half-Precision Data Format



For normalized numbers, the biased exponent value '*e*' lies in the [1 - 30] range corresponding to an actual exponent value E in the [-14 - +15] range, the hidden bit is a '1', and the value of the number is interpreted as:

$$(-1)^{s} \times 2^{E} \times (1.\text{fraction})$$

where E is the unbiased exponent and 1.fraction is the significand consisting of a leading '1' (the hidden bit) and a fractional part (fraction field). With this format, the maximum positive normalized number ($pmax_{hp}$) is represented by the encoding `0x7bff` which is 65504, and the minimum positive normalized number ($pmin_{hp}$) is represented by the encoding `0x0400` which is approximately 6.1E-5.

Two specific values of the biased exponent are reserved: 0 and 31, for encoding special values of ±0, ±∞, *NaN and Denorm*.

Zeros of both positive and negative sign are represented by a biased exponent value *e* of zero and a fraction which is zero.

Infinities of both positive and negative sign are represented by a biased exponent value of 31 and a fraction which is zero.

Denormalized numbers of both positive and negative sign are represented by a biased exponent value *e* of 0 and a fraction *f* which is non-zero. For these numbers, the hidden bit is defined to be '0'. This number type is not directly supported in hardware and is converted to zero by the AMU2.

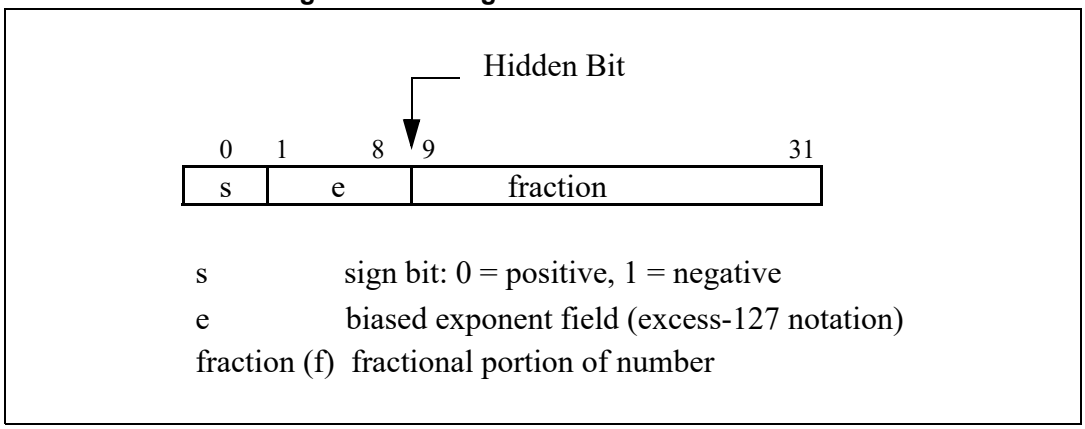
Not a Numbers (*NaNs*) are represented by a biased exponent value *e* of 31 and a fraction *f* which is non-zero.

94.3.3.1.7 Single-precision floating-point format

Each single-precision floating-point data element is 32 bits wide with one sign bit (s), 8 bits of biased exponent (*e*) and 23 bits of fraction (*f*).

In the IEEE-754 specification, floating-point values are represented in a format consisting of three explicit fields (sign field, biased exponent field, and fraction field) and an implicit hidden bit.

Figure 2497. Single-Precision Data Format



For normalized numbers, the biased exponent value '*e*' lies in the [1 - 254] range corresponding to an actual exponent value E in the [-126 - +127] range, the hidden bit is a '1', and the value of the number is interpreted as:

$$(-1)^{s} \times 2^{E} \times (1.\text{fraction})$$

where E is the unbiased exponent and 1.fraction is the significand consisting of a leading '1' (the hidden bit) and a fractional part (fraction field). With this format, the maximum positive normalized number (*pmax*) is represented by the encoding `0x7f7fffff` which is approximately 3.4E+38, and the minimum positive normalized number (*pmin*) is represented by the encoding `0x00800000` which is approximately 1.2E-38.

Two specific values of the biased exponent are reserved: 0 and 255, for encoding special values of ±0, ±∞, *NaN and Denorm*.

Zeros of both positive and negative sign are represented by a biased exponent value *e* of zero and a fraction *f* which is zero.

Infinities of both positive and negative sign are represented by a biased exponent value of 255 and a fraction which is zero.

Denormalized numbers of both positive and negative sign are represented by a biased exponent value *e* of 0 and a fraction *f* which is non-zero. For these numbers, the hidden bit is defined by the IEEE-754 standard to be '0'. This number type is not directly supported in hardware and is converted to zero by the AMU2.

Not a Numbers (*NaNs*) are represented by a biased exponent value *e* of 255 and a fraction *f* which is non-zero.

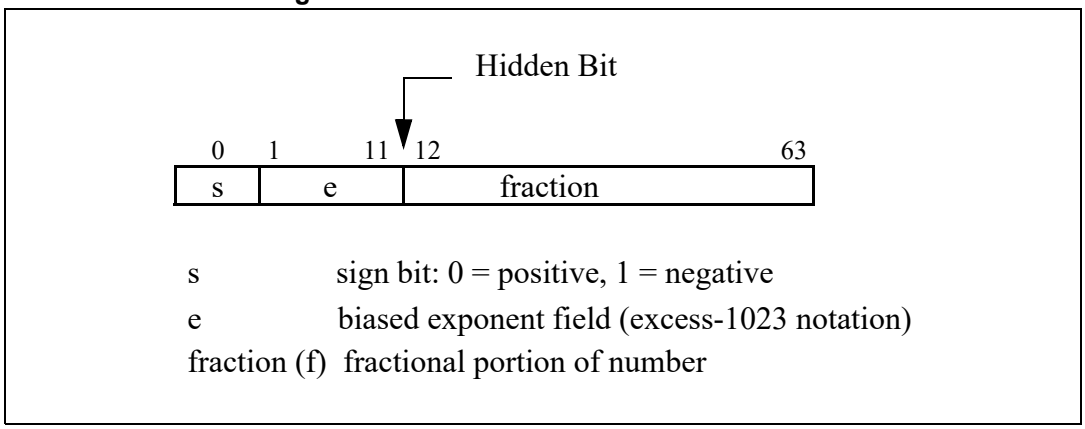
Defining *pmax* to be the most positive normalized number (farthest from zero), *pmin* the smallest positive normalized number (closest to zero), *nmax* the most negative normalized number (farthest from zero) and *nmin* the smallest normalized negative number (closest to zero), an overflow is said to have occurred if the numerically correct result (r) of an operation is such that r > *pmax* or r < *nmax*. An underflow is said to have occurred if the numerically correct result of an operation is such that 0 < r < *pmin* or *nmin* < r < 0. In this case, r may be denormalized, or may be smaller than the smallest denormalized number.

94.3.3.1.8 Double-precision floating-point format

Each double-precision floating-point data element is 64 bits wide with one sign bit (s), 11 bits of biased exponent (*e*) and 52 bits of fraction (*f*).

In the IEEE-754 specification, floating-point values are represented in a format consisting of three explicit fields (sign field, biased exponent field, and fraction field) and an implicit hidden bit.

Figure 2498. Double-Precision Data Format



For normalized numbers, the biased exponent value ‘*e*’ lies in the [1 - 2046] range corresponding to an actual exponent value E in the [-1022 - +1023] range, the hidden bit is a ‘1’, and the value of the number is interpreted as:

$$(-1)^{s} \times 2^{E} \times (1.\text{fraction})$$

where E is the unbiased exponent and 1.fraction is the significand consisting of a leading ‘1’ (the hidden bit) and a fractional part (fraction field). With this format, the maximum positive normalized number (*pmax*) is represented by the encoding `0x7feeffff_ffffffff` which is approximately 1.8E+308, and the minimum positive normalized number (*pmin*) is represented by the encoding `0x00100000_00000000` which is approximately 2.2E-308.

Two specific values of the biased exponent are reserved: 0 and 2047, for encoding special values of ±0, ±∞, *NaN and Denorm*.

Zeros of both positive and negative sign are represented by a biased exponent value *e* of zero and a fraction *f* which is zero.

Infinities of both positive and negative sign are represented by a biased exponent value of 2047 and a fraction which is zero.

Denormalized numbers of both positive and negative sign are represented by a biased exponent value *e* of 0 and a fraction *f* which is non-zero. For these numbers, the hidden bit is defined by the IEEE-754 standard to be ‘0’. This number type is not directly supported in hardware and is converted to zero by the AMU2.

Not a Numbers (*NaNs*) are represented by a biased exponent value *e* of 2047 and a fraction *f* which is non-zero.

Defining *pmax* to be the most positive normalized value (farthest from zero), *pmin* the smallest positive normalized value (closest to zero), *nmax* the most negative normalized value (farthest from zero) and *nmin* the smallest normalized negative value (closest to zero), an overflow is said to have occurred if the numerically correct result (r) of an operation is such that r > *pmax* or r < *nmax*. An underflow is said to have occurred if the numerically correct result of an operation is such that 0 < r < *pmin* or *nmin* < r < 0. In this case, r may be denormalized, or may be smaller than the smallest denormalized number.

94.3.3.1.9 Floating-point data format summary

Table 2378 summarizes the data type specifications for the three supported floating-point formats.

Table 2378. Floating-point Format Summary

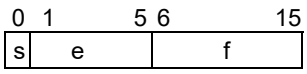
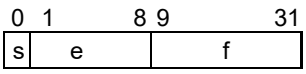
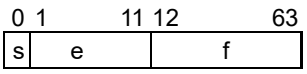
Parameter	Half-Precision	Single-Precision	Double-Precision
Data Format			
Field Size in Bits			
Sign (s)	1	1	1
Biased exponent (e)	5	8	11
Fraction (f)	10	23	52
Total	16	32	64
Interpretation of Sign			
Positive fraction	s = 0	s = 0	s = 0
Negative fraction	s = 1	s = 1	s = 1
Normalized Numbers			
Bias of biased exponent	+15 (0xf)	+127 (0x7F)	+1023 (0x3FF)
Range of biased exponent	0 < e < 31 (0x1F)	0 < e < 255 (0xFF)	0 < e < 2047 (0x7FF)
Range of fraction	Zero or Nonzero	Zero or Nonzero	Zero or Nonzero
Mantissa	1.f	1.f	1.f
Relation to representation of real numbers	$(-1)^s \times 2^{e-15} \times 1.f$	$(-1)^s \times 2^{e-127} \times 1.f$	$(-1)^s \times 2^{e-1023} \times 1.f$
Denormalized Numbers			
Biased exponent format minimum	0 (0x0)	0 (0x00)	0 (0x000)
Bias of biased exponent	+14 (0xE)	+126 (0x7E)	+1022 (0x3FE)

Table 2378. Floating-point Format Summary (continued)

Parameter	Half-Precision	Single-Precision	Double-Precision
Range of fraction	Nonzero	Nonzero	Nonzero
Mantissa	0.f	0.f	0.f
Relation to representation of real numbers	$(-1)^s \times 2^{-14} \times 0.f$	$(-1)^s \times 2^{-126} \times 0.f$	$(-1)^s \times 2^{-1022} \times 0.f$
Signed Zeros			
Biased exponent format minimum	0 (0x0)	0 (0x00)	0 (0x00)
Mantissa	0.f = 0.0	0.f = 0.0	0.f = 0.0
Signed Infinities			
Biased exponent format maximum	31 (0x1F)	255 (0xFF)	2047 (0x7FF)
Mantissa	0.f = 0.0	0.f = 0.0	0.f = 0.0
NaNs			
Sign	0 or 1	0 or 1	0 or 1
Biased exponent format maximum	31 (0x1F)	255 (0xFF)	2047 (0x7FF)
Fraction	Nonzero	Nonzero	Nonzero
Representation of fraction Nonzero bit pattern created by user Fraction when created by AMU2	xxxxx…xxxx N/A	xxxxx…xxxx 10000…0[001-110]	xxxxx…xxxx 1000…0[001-110]

Table 2378. Floating-point Format Summary (continued)

Parameter	Half-Precision	Single-Precision	Double-Precision
Approximate Ranges			
Maximum positive normalized	6.5×10^{4}	3.4×10^{38}	1.8×10^{308}
Minimum positive normalized	6.1×10^{-5}	1.2×10^{-38}	2.2×10^{-308}

94.3.3.1.10 Calculation Precision

The AMU2 implementation uses the IEEE 754 floating-point precisions explicitly defined in *Section 94.3.1: AMU2 algorithm*. Other details include:

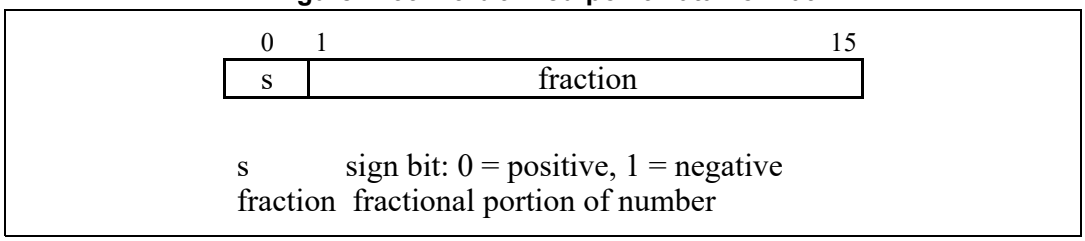
- The AMU2 supports the standard IEEE 754 “round-to-nearest, ties-to-even” mode for all rounding operations, including all conversions from double-precision to single-precision. Denormalized numbers are not supported. Denormalized numbers are interpreted as zero. Input NaNs are passed as the result of a computation and certain operations producing an invalid result generate a quiet NaN with a diagnostic value embedded in the fraction. More details on the handling of NaNs is presented in *Section 94.3.3.1.4: Not-A-Number*.
- If an overflow condition is detected, the result is forced to (+/-) Inf value before the ALU is halted.
- The exponential function inputs a 32-bit converted *t* operand and produces an IEEE 32-bit floating-point result. The externally-observable exp(x) precision is 24 bits, ± 1 ULP.

94.3.3.2 16-bit fixed-point with common exponent

The AMU2 also supports a fixed-point format for the V[*] array. This format specifies a signed 16-bit two’s complement fixed-point fraction in the [-1, 1.0) range with a common 7-bit exponent using an excess-63 format. This format provides more bits of fraction precision versus the IEEE 754 half-precision floating-point format at the cost of a reduced dynamic range within the same memory footprint.

In this format, values are represented with two explicit fields (sign field and fraction field) along with a 7-bit common biased exponent field (AMU2_CCR1CH*n*[CMEXP]) applied to all elements in the V[*] array.

Figure 2499. 16-bit fixed-point Data Format



Note that the 7-bit common biased exponent is not shown in *Figure 2499*.

Using the same variables as the floating-point data types, the biased exponent value '*e*' lies in the [0 - 127] range corresponding to an actual exponent value E in the [-63 to +64] range, the hidden bit is a '0', and the value of this 2's complement number is interpreted as:

((-1 x *s*) + 0.*fraction*) x 2^E

where E = CMEXP - 63.

When the V[*] array is specified in this format, the AMU2 automatically converts the number into a single-precision floating-point value before it is used in the stage2 calculation. This transformation is described by the following C language *convert fixed halfword-to-single precision floating-point* function:

```
ieee32 convertFixedPoint(uint07 cmExp, uint16 v16x)
{
    int i, s;

    if (v16x & 0x8000) {
        s = 1;
        v16x = (~v16x + 1) & 0xffff;              // same as v16x = -v16x
    } else
        s = 0;

    if (v16x & 0x8000)
        ieee32Result = 1<<31 | (64+cmExp)<<23;    // minimum value
    else if (v16x == 0)
        ieee32Result = 0x00000000;
    else {
        for (i = 0; i < 16; i++) {
            v16x <<= 1;
            if (v16x & 0x8000)
                break;
        }
        ieee32Result = s<<31 | (63+cmExp-i)<<23 | (v16x & 0x7fff)<<8;
    }

    return ieee32Result;
}
```

94.3.4 Register format and data types

As described in the pseudo-code of *Section 94.3.1: AMU2 algorithm*, the programming model provides address pointers and data values. The register types are defined in *Table 2379* and presented in their order of appearance in the pseudo-code.

Table 2379. Data types in registers

Operand	AMU2 Register	Data Type
u_pointer(1)	UPTRCH*n*	ARAM pointer to ieee32 floating-point data
p1_pointer(1)	P1PTRCH*n*	ARAM pointer to ieee32 floating-point data

Table 2379. Data types in registers (continued)

Operand	AMU2 Register	Data Type
p2_pointer(1)	P2PTRCH*n*	ARAM pointer to ieee32 floating-point data
ut_pointer(1)	Implicit	Special ARAM pointer to ieee32 floating-point data
p9	P9{M,L}CH*n*	ieee64 floating-point data
p8	P8CH*n*	Unsigned int16 data
p7	P7CH*n*	Unsigned non-zero int6 data (if value ==0, then p7 = 32)
p6	P6CH*n*	Unsigned int16 data
v_pointer(2)	VPTRCH*n*	ARAM or system pointer to ieee32 floating-point data, or ieee16 floating-point data, or 16-bit fixed-point data with 7-bit common exponent
L_pointer(1)	LPTRCH*n*	ARAM pointer to ieee32 floating-point data
p3_pointer(2)	P3PTRCH*n*	ARAM or system pointer to ieee32 floating-point data
y64	Y64{M,L}CH*n*	ieee64 floating-point data
y32	Y32CH*n*	ieee32 floating-point data
p5	P5CH*n*	ieee32 floating-point data
p4	P4CH*n*	ieee32 floating-point data
result, z64	RES64{M,L}CH*n*	ieee64 floating-point data
res32, z32	RES32CH*n*	ieee32 floating-point data

1. All pointers reference the AMU local RAM (ARAM), and are limited to 15 bits in size as the RAM is a maximum of 32 KB.
2. For “normal” operations, the v_pointer and p3_pointer are like the other pointers and reference the ARAM and are limited to 15 bits in size. When ADMA data streaming is enabled, these same registers are configured as a 8-byte aligned 32-bit system Flash address.

94.3.5 ALU calculation engine

The ALU calculation engine supports a time-multiplexed hardware computation pipeline supporting 2 execution channels. This structure implements the algorithm detailed in *Section 94.3.1: AMU2 algorithm* including the transcendental function described in *Section 94.3.2: The exp(x) function and implementation*.

94.3.5.1 ALU execution states

The ALU maintains two finite state machines that record the execution state of the stage1 and stage23 calculations and is visible in the AMU2_SEIRCH*n*[S23XS, S1XS] fields. The execution states and their descriptions are detailed in *Table 2380*.

Table 2380. ALU S1, S23 Execution States

State	AMU2_SEIRCH*n*[S{1,23}XS]	Description
Idle or Pending Start	0b00	Idle or pending start. A pending start is created when any of the conditional start bits, AMU2_CCR2CH*n*[WFS1D, WFS23D, WFSS23], are set.
Halted	0b01	Halted after detection of an error condition or arithmetic fault.
Stopped	0b10	Stopped due to an explicit request (AMU2_CCR2CH*n*[S23STP]) or a properly-enabled N crossing event. Note the stage1 execution cannot be stopped via either of these mechanisms and stage23 is stopped after the current iteration of the outer loop. N crossing events also stop at the same point in the algorithm execution.
Running or Stalled	0b11	Actively executing or stalled due to a properly-enabled debug event (AMU2_CCR1CH*n*[DDSL] = 0). The presence of a properly-qualified debug event is reflected in the AMU2_SEIRCH*n*[QDSL] indicator.

Once a state machine enters the halted state, the state needs to be cleared by starting a new calculation, or by explicitly resetting it by a register write with AMU2_CCR2CH*n*[CSXXS] = 1. Likewise, once the status signals a stopped state, it can be resumed by a register write with AMU2_CCR2CH*n*[S23RSM] = 1, cleared by starting a new calculation or explicitly reset by a register write with AMU2_CCR2CH*n*[CSXXS] = 1.

94.3.5.2 ALU speed control

The ALU includes a “speed control” function that can be used to throttle the execution rate of the channel. There are four registers associated with this function: three 3-bit registers defining the starting, maximum and current speed settings plus a threshold count. The 3-bit settings define a fractional operation “speed” as {1/8, 2/8, 3/8,..., 7/8, 8/8} where 8/8 represents the maximum, that is full speed. The registers for this function include:

AMU2_CCR1CH*n*[SRTSP] // start speed

AMU2_CCR1CH*n*[MAXSP] // maximum speed

AMU2_SEIRCH*n*[ACURSP] // ALU current speed

AMU2_P8CH*n*[THRHC] // threshold P8 count

The ALU controls its execution speed by stalling the calculation engine pipeline as defined by the current speed. As the ALU executes, the operation speed is increased by 1/8 from the starting value to the maximum value each time a multiple of the P8 threshold count is reached. As an example, if SRTSP = 1/8 and MAXSP = 8/8 (full speed), then the current speed reflected in the ACURSP field follows the following sequence:

For the 1st THRHC loops of stage 23 calculation, ACURSP = 1/8

For the 2nd THRHC loops of stage 23 calculation, ACURSP = 2/8

For the 3rd THRHC loops of stage 23 calculation, ACURSP = 3/8

For the 4th THRHC loops of stage 23 calculation, ACURSP = 4/8

For the 5th THRHC loops of stage 23 calculation, ACURSP = 5/8

For the 6th THRHC loops of stage 23 calculation, ACURSP = 6/8

For the 7th THRHC loops of stage 23 calculation, ACURSP = 7/8

For the 8th THRHC loops of stage 23 calculation until complete, ACURSP = 8/8

It is software's responsibility to initialize the three controlling registers (SRTSP, MAXSP, THRHC) with appropriate values (SRTSP < MAXSP) else the speed control behavior may be indeterminate.

Note: Recall the pipelined calculation engine hardware is shared by the two channels via a time multiplexing scheme. To make the speed control of the calculation engine manageable, there is a single ALU speed control mechanism implemented, affecting both channels. Note the four registers associated with this function are multiply-mapped into the programming models of both channels, but only a single function is implemented. Accordingly, the global values of the speed control registers are used if one (or both) channels are executing. If both channels are idle and one begins execution, the speed ramps up from SRTSP to MAXSP as described. If the other channel begins execution while the first channel is already executing, the second channel begins its execution at ACURSP. The speed is only returned to the SRTSP value when both channels are idle and one (or both) becomes active.

94.3.6 AMU Direct Memory Access (ADMA)

94.3.6.1 Basic ADMA data transfer capabilities

Each AMU2 channel is equipped with an ADMA channel. This allows optimized block copying of data from Flash and system calibration memories (if applicable) to the local RAM (ARAM).

The programming model associated with each ADMA channel contains five registers: source and destination addresses, transfer count, expected checksum and control & configuration. Since the AMU2 supports the ADMA transfers using a private 64-bit interface to the platform Flash memory controller, all addresses and transfer counts must be 8-byte aligned values.

The control & configuration register (AMU2_DCCRCH*n*) supports the ability to separately update the source or destination address registers, or both, so repeated ADMA transfers can sequence through a Flash data structure while keeping the local ARAM structure constant. See the AMU2_DCCRCH*n*[SARU, DARU] fields. The DARU register field supports the ability to transfer the last 64 bits fetched from the Flash source memory location to the two AMU Control & Configuration Registers (AMU2_CCR{1,2}CH*n*). This provides the ADMA with the ability to load a new vector data set from Flash, followed by writes to initiate a calculation. This special "last write to CCR{1,2}" mode is enabled when AMU2_DCCRCH*n*[DARU] = 0b01.

94.3.6.2 ADMA execution states

The ADMA maintains a finite state machine that records its current operating condition and is visible in the AMU2_SEIRCH*n*[ADMAS] field. These execution states and their descriptions are detailed in *Table 2381*.

Table 2381. ADMA Execution States

State	AMU2_SEIRCH*n* [ADMAS]	Description
Idle or Pending Start	0b00	Idle or pending start. A pending start is created when any of the conditional start bits, AMU2_DCCRCH*n*[WFS1D, WFS23D, WFDNI], are set.
Halted	0b01	Halted after detection of an error condition, such as a bus error, illegal address, multi-bit ECC event.

Table 2381. ADMA Execution States (continued)

State	AMU2_SEIRCH*n* [ADMAS]	Description
Stopped	0b10	Stopped due to an explicit request (AMU2_DCCRCH*n*[STP]) or an implicit request caused by an ALU stop condition enabled by AMU2_DCCRCH*n*[DSAS] = 1.
Running or Stalled	0b11	Actively executing the data transfer or stalled due to a properly-enabled debug event (AMU2_CCR1CH*n*[DDSL] = 0). The presence of a properly-qualified debug event is reflected in the AMU2_SEIRCH*n*[QDSL] indicator.

Once a state machine enters the halted state, the state needs to be cleared by starting a new data transfer or by explicitly resetting it by a register write with AMU2_DCCRCH*n*[CADS] = 1. Likewise, once the status signals a stopped state, it can be resumed by a register write with AMU2_DCCRCH*n*[RSM] = 1, cleared by starting a new data transfer or explicitly reset by a register write with AMU2_DCCRCH*n*[CADS] = 1.

94.3.6.3 Cyclic redundancy checking

The ADMA optionally supports a standard CRC checksum on transfers into the ARAM. If enabled (AMU2_DCCRCH*n*[ECRC] = 1), the checksum is calculated by the ADMA on each data transfer and accumulated. When the data transfers are complete, the final actual CRC value is compared against the expected checksum value contained in the AMU2_DCKSCH*n* register. If the checksums are identical, the AMU operation continues. However, if the checksums are not equal, a data transmission error has occurred and is signaled by AMU2_SEIRCH*n*[CKSERR] = 1. In the case of an enabled CRC checksum error, any ALU calculation which depends on the completion of the ADMA transfer (as defined by AMU2_CCR2CH*n*[WFDS1] == 1 or AMU2_CCR2CH*n*[WFDS23] == 1) is aborted and the corresponding AMU2_CCR2CH*n*[WFDS1, WFDS23] indicators cleared.

The ADMA supports the standard “Ethernet” CRC32 polynomial, which is defined as:

$$CRC32 = x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^{8} + x^{7} + x^{5} + x^{4} + x^{2} + x^{1} + x^{0}$$

The checksum calculation initializes the CRC to 0xFFFF_FFFF, and processes the fetched data beginning with the most significant bit of the most significant byte. The C code in *Figure 2500* mirrors the ADMA’s hardware implementation. It uses a 256-entry table lookup function (containing unique 32-bit words based on the combined CRC state and a data byte) and is based on the algorithm detailed in “Byte-wise CRC Calculations”, Aram Perez, IEEE Micro June 1983. The `gen_crc_table()` function is executed once to populate the `crc_table[]` array; the `update_crc()` function is called for each CRC to be calculated.

Figure 2500. ADMA CRC32 C Language Function

```
#define POLYNOMIAL 0x04c11db7L
static unsigned long crc_table[256];
void gen_crc_table()
{
    /* generate the table of CRC remainders for all possible bytes */
    register int i, j;
    register unsigned long crc_accum;

    for (i = 0; i < 256; i++) {
        crc_accum = ((unsigned long) i << 24);
        for (j = 0; j < 8; j++) {
            if (crc_accum & 0x80000000L)
                crc_accum = (crc_accum << 1) ^ POLYNOMIAL;
            else
                crc_accum = (crc_accum << 1);
        }
        crc_table[i] = crc_accum;
    }
}

unsigned long update_crc(unsigned long crc_accum,
                         char *data_blk_ptr,
                         int data_blk_size)
{
    /* update the CRC on the data block one byte at a time */
    register int i, j;

    for (j = 0; j < data_blk_size; j++) {
        i = ((int)(crc_accum >> 24) ^ *data_blk_ptr++) & 0xff;
        crc_accum = (crc_accum << 8) ^ crc_table[i];
    }
    return crc_accum;
}
```

The actual checksum value calculated by the ADMA is available in the AMU2_ACKSCH*n* register or the aliased AMU2_AACKSCH*n*.

94.3.6.4 ADMA V and P3 Data Streaming Modes

Recall the V[*] and P3[*] input data vectors have larger array dimensions compared to all the other input data vectors. Accordingly, the ADMA supports a special type of data streaming for these 2 input vectors. At any point in time, the ADMA supports *either* normal data transfers *or* the special streaming modes.

Specifically, *it is not necessary to pre-load the contents* of V[*] or P3[*], or both, into the ARAM before initiating the ALU's stage23 calculation if one of the special data streaming modes is enabled. When enabled by AMU2_DCCRCH*n*[STMP3, STMV{32,16}], the ADMA

manages the streamed input vector(s) using the circular data FIFO(s) implemented in the ARAM. The ADMA prefetches the streamed data and writes it into the FIFOs and the FIFO contents are read based of the ALU's stage23 calculation speed.

The V[*] *and* P3[*] input data vectors can be streamed at the same time, if both AMU2_DCCRCH*n*[STMP3, STMV{32,16}] are [1, != 00]. Typically, the execution time of the ALU calculation engine increases when streaming P3 or V, or both, due to Flash arbitration delays as the AMU streaming traffic may be stalled due to requests from other bus masters.

When operating in the data streaming mode, certain register fields associated with a normal ADMA transfer are ignored, or are interpreted in a special manner. See *Table 2382*.

Table 2382. ADMA Register Definitions when V or P3 Data Streaming is Enabled

Register/Field	Used when Data Streaming	Description when Data Streaming
AMU2_P3PTRCH*n*	Yes	if AMU2_DCCRCH*n*[STMP3] = 1, this register defines the 32-bit system address for the P3[*] input data vector.
AMU2_VPTRCH*n*	Yes	if AMU2_DCCRCH*n*[STMV{32,16}] != 00, this register defines the 32-bit system address for the V[*] input data vector.
AMU2_DSACH*n*	No	The normal ADMA source address register is *not* used when data streaming is enabled; the AMU2_{P3,V}PTRCH*n* registers are used instead.
AMU2_DDACH*n*	No	The normal ADMA destination address register is *not* used when data streaming is enabled. The ADMA uses a reserved space within the ARAM to implement circular data FIFOs (256 bytes for V[*], 128 bytes for P3[*]). The offset ARAM addresses of interest are: 0x0_7C00 - 0x0_7CFF Channel 0 V[*] FIFO 0x0_7D00 - 0x0_7DFF Channel 1 V[*] FIFO 0x0_7E00 - 0x0_7E7F Channel 0 P3[*] FIFO 0x0_7E80 - 0x0_7EFF Channel 1 P3[*] FIFO
AMU2_DCNTCH*n*	No	The ADMA's fetching of V[*] or P3[*] data, or both, is paced by the ALU's stage23 calculation speed. Due to the execution pipeline structure, there may be speculative data fetches that are not actually used in the calculation.
AMU2_DCKSCH*n*	No	Due to the speculative, ALU-paced data transfers, CRC checking is not supported while data streaming. Note the data fetches from system Flash are still protected by the standard single-error correction, double-error detection ECC. A multi-bit non-correctable ECC event during a streaming operation causes the AMU2_SEIRCH*n*[ADMERR] bit to be set and the ADMA to be halted. The ALU calculation then stalls due to the lack of input data.
AMU2_DCCRCH*n*[SRTSP], AMU2_DCCRCH*n*[MAXSP]	No	Since the ADMA transfers are ALU-paced when data streaming, the normal "speed control" function is ignored.
AMU2_DCCRCH*n*[ADICFG]	No	Since the ADMA transfers are ALU-paced when data streaming, the normal interrupt configurations are ignored.
AMU2_DCCRCH*n*[SARU], AMU2_DCCRCH*n*[DARU]	No	Since the ADMA transfers use different source and destination addresses when data streaming, these fields are ignored.

Table 2382. ADMA Register Definitions when V or P3 Data Streaming is Enabled (continued)

Register/Field	Used when Data Streaming	Description when Data Streaming
AMU2_DCCRCH*n*[HCRCE], AMU2_DCCRCH*n*[ECRC]	No	Due to the speculative, ALU-paced data transfers, CRC checking is not supported while data streaming.
AMU2_DCCRCH*n*[STMP3]	Yes	When asserted in conjunction with AMU2_DCCRCH*n*[SRT] = 1, this bit enables the P3[*] data streaming mode.
AMU2_DCCRCH*n*[STMV{32,16}]	Yes	When asserted in conjunction with AMU2_DCCRCH*n*[SRT] = 1, this bit enables the V[*] data streaming mode.
AMU2_DCCRCH*n*[DSAS]	No	This bit is ignored when data streaming is enabled. Since the ADMA data transfers are ALU-paced, stopping the ALU also stops the ADMA.
AMU2_DCCRCH*n*[STP]	No	This bit is ignored when data streaming is enabled. Since the ADMA data transfers are ALU-paced, stopping the ALU also stops the ADMA, given AMU2_DDCRCH*n*[DSAS] = 1.
AMU2_DCCRCH*n*[WFS23], AMU2_DCCRCH*n*[WFS1]	No	Since the ADMA data transfers are paced by the ALU stage23 execution speed, these bits are ignored when data streaming is enabled.
AMU2_DCCRCH*n*[RSM]	No	This bit is ignored when data streaming is enabled. Since the ADMA data transfers are ALU-paced, resuming the ALU also resumes the ADMA.
AMU2_DCCRCH*n*[SRT]	Yes	The simultaneous assertion of SRT = 1 and (STMV{32,16} != 00 or STMP3 = 1) initiates an ADMA transfer with data streaming enabled.

94.3.6.5 ADMA speed control

The ADMA includes a “speed control” function that can be used to throttle its access rate. There are four register fields associated with this function: three 3-bit registers defining the starting, maximum and current speed settings plus a threshold count. The 3-bit settings define a fractional operation “speed” as {1/8, 2/8, 3/8,..., 7/8, 8/8} where 8/8 represents the maximum, that is full speed. The registers for this function include:

AMU2_DCCRCH*n*[SRTSP] // start speed

AMU2_DCCRCH*n*[MAXSP] // maximum speed

AMU2_SEIRCH*n*[DCURSP] // ADMA current speed

AMU2_DCNTCH*n*[THRHC] // threshold byte count (32-byte aligned)

The ADMA controls the transfer speed by inserting idle states between the receipt of data and the initiation of the next Flash read. As the ADMA fetches data, the operation speed is increased by 1/8 from the starting value to the maximum value each time a multiple of the byte threshold count is reached. As an example, if SRTSP = 1/8 and MAXSP = 8/8 (full

speed), then the current speed reflected in the DCURSP field follows the following sequence:

For the 1st THRHC bytes, DCURSP = 1/8

For the 2nd THRHC bytes, DCURSP = 2/8

For the 3rd THRHC bytes, DCURSP = 3/8

For the 4th THRHC bytes, DCURSP = 4/8

For the 5th THRHC bytes, DCURSP = 5/8

For the 6th THRHC bytes, DCURSP = 6/8

For the 7th THRHC bytes, DCURSP = 7/8

For the 8th THRHC bytes until the transfer is complete, DCURSP = 8/8

It is software's responsibility to initialize the three controlling registers (SRTSP, MAXSP, THRHC) with appropriate values (SRTSP < MAXSP) else the speed control behavior may be indeterminate.

Note: Recall the ADMA hardware is shared by the two channels via a time multiplexing scheme. To make the speed control of the ADMA manageable, there is a single ADMA speed control mechanism implemented, affecting both channels. Note the four registers associated with this function are multiply-mapped into the programming models of both channels, but only a single function is implemented. Accordingly, the global values of the speed control registers are used if one (or both) channels are executing. The speed is only returned to the starting value when both channels are idle and one (or both) becomes active.

94.4 Use case examples

This section presents overviews of multiple use cases describing different scenarios for data loading, ALU calculation and post-processing of the result. This is *not* intended to provide a complete list, but rather examples of the types of configurations supported by the AMU2.

In general, there are three distinct phases associated with an AMU2 calculation: data loading, the ALU calculation and result retrieval & post-processing. The data loading can be performed by any processing element, such as a processor core, the system DMA or the ADMA, or both. The actual algorithm execution is always performed by the AMU2's ALU calculation engine. The retrieval of results and status can be performed by a processor core or the system DMA, while the post-processing is typically handled by a processor core. Recall, the AMU2's programming model is specifically organized in a manner that promotes the use of a DMA, either the system DMA or the ADMA, to handle the data loading operations.

In general, the relative data load speeds among the possible masters are:

CPU < SDMA < ADMA

that is, the CPU is the slowest, the system DMA is faster and the ADMA is the fastest. Additionally, for certain scenarios, the ADMA's data transfers can be overlapped with the ALU calculation to further reduce the execution times for multiple algorithm evaluations.

Consider the following use case examples presented in order of increasing complexity.

94.4.1 Single calculation using a CPU as the master device

For this scenario, a processor core performs *all the required data loading*, moving the coefficients and parameters from system Flash into the ARAM and then initializing all the

corresponding pointer and parameter registers. Once the data load and register initialization are completed, the calculation is started by writing the AMU2_CCR{1,2}CH*n* registers. In this case, the AMU completion and error indicators are configured to generate interrupts to the processor; the calculation actually begins when the AMU2_CCR2CH*n*[S1SRT, S23SRT] bits are set.

The AMU2 performs the calculation and then typically generates an interrupt, signaling successful completion or some type of error termination or an N crossing debug event.

As the processor handles the interrupt, the 64-bit double precision result (AMU2_RES64{M,L}CH*n*) is retrieved along with the status flags in the AMU2_SEIRCH*n* register. After processing the interrupt, the service routine clears the interrupt source by writing a “1” to clear the appropriate flag in the low-order byte of the AMU2_SEIRCH*n* register. The processor then performs any required analysis of the status indicators before saving the result.

The preceding description assumes the processor is interrupted when the AMU2 completes as this would be the typical scenario. The AMU2 also provides a channel busy flag (AMU2_SEIRCH*n*[CHBSY]) that the processor can poll to determine whether the calculation is finished.

94.4.2 Single calculation using SDMA for data load, CPU for post-processing

In this example, it is assumed the processor has created the appropriate data structure in system Flash memory so that the System DMA can perform the data load into the ARAM plus the initialization of the pointer and parameter registers.

The SDMA would typically require two separate TCDs (Transfer Control Descriptors): one to perform the data move from system Flash into the ARAM, and a second to load the programming model registers. The SDMA can use the scatter/gather feature to process the two TCDs in a single channel, or 2 channels can be used where the first channel performing the data move “links” to the second channel performing the register loads. In any case, the second data transfer loads the programming model using an 80-byte block transfer. Using CH0 as an example, this block data transfer writes from the AMU2_P1PTRCH*n* register (offset address 0x1_0010) to the AMU2_CCR2CH*n* registers (offset address 0x1_005c) before beginning the actual calculation.

It should be noted the programming model of the AMU2 is organized such that an optimized 80-byte burst transfer (5 x 8-byte 8-beat bursts) can be used to load the registers. The optimized block transfer does not include the AMU2_NCH*n* register for two reasons:

1. The use of the N crossing functionality is a debug feature and not intended for normal system operation;
2. The inclusion of the AMU2_NCH*n* functionality must be properly enabled by setting AMU2_CCR2CH*n*[ENCRS] = 1. Stated differently, the AMU2_NCH*n* functionality is completely disabled if AMU2_CCR2CH*n*[ENCRS] = 0. In the optimized block transfer, the AMU2_NCH*n* register does not need to be initialized, and AMU2_CCR2CH*n* register (with ENCRS = 0) is always the last register written.

Note, the AMU2_NCH*n* register can be included in the sequential data transfer for the register writes by increasing the total number of bytes to be transferred to 84 and using 32-bit writes.

As before, the AMU completion and error indicators are configured to generate interrupts to the processor; the calculation actually begins when the AMU2_CCR2CH*n*[S1SRT, S23SRT] bits are set by the SDMA's last transfer.

The channel performs the calculation and generates an interrupt, signaling a successful completion or some type of error termination or an N crossing debug event.

As the processor handles the interrupt, the 64-bit double precision result (AMU2_RES64{M,L}CH*n*) is retrieved along with the status flags in the AMU2_SEIRCH*n* register. After processing the interrupt, the service routine clears the interrupt source by writing a "1" to clear the appropriate flag in the low-order byte of the AMU2_SEIRCH*n* register. The processor then performs any required analysis of the status indicators before saving the result.

94.4.3 Single calculation using SDMA + ADMA for data load, CPU for post-processing

For this case, it is assumed the processor has created the appropriate data structure in system Flash memory so that the System DMA and the ADMA collectively can perform the data load into the ARAM plus the initialization of the pointer and parameter registers.

In this example, let the ADMA move the U[*] vector into the ARAM and the SDMA move all the rest of the data.

The SDMA would typically require two separate TCDs (Transfer Control Descriptors): one to perform the data move from system Flash into the ARAM, and a second to load the programming model registers. The SDMA can use the scatter/gather feature to process the two TCDs in a single channel, or 2 channels can be used where the first channel performing the data move "links" to the second channel performing the register load. The second data transfer loads the programming model using an 80-byte (5 x 16 byte 4-beat bursts) block transfer. Using CH0 as an example, this block data transfer writes from the AMU2_P1PTRCH*n* register (offset address 0x1_0010) to the AMU2_CCR2CH*n* registers (offset address 0x1_005c) before beginning the actual calculation. In this example, the programming model initialization configures the ADMA to move the U[*] vector into the ARAM and the AMU2_CCR2CH*n*[WFDS1, WFDS23] conditional start bits are set (and [S1SRT, S23SRT] cleared) to signal the start of the calculation engine is dependent on the completion of the ADMA.

As before, the AMU completion and error indicators are configured to generate interrupts to the processor; the calculation actually begins when the ADMA completes the data transfer of the U[*] vector into the ARAM.

The channel performs the calculation and generates an interrupt, signaling a successful completion or some type of error termination or an N crossing debug event.

As the processor handles the interrupt, the 64-bit double precision result (AMU2_RES64{M,L}CH*n*) is retrieved along with the status flags in the AMU2_SEIRCH*n* register. After processing the interrupt, the service routine clears the interrupt source by writing a "1" to clear the appropriate flag in the low-order byte of the AMU2_SEIRCH*n* register. The processor then performs any required analysis of the status indicators before saving the result.

94.4.4 Single calculation using SDMA for partial data load, streaming ADMA for V[*], CPU for post-processing

For this case, it is assumed the processor has created the appropriate data structure in system Flash memory so that the System DMA can perform the partial data load into the ARAM plus the initialization of the pointer and parameter registers.

In this example, let the ADMA use the special data streaming mode to fetch the V[*] array into the ARAM with the SDMA moving all the rest of the data including the programming model register initialization.

The SDMA would typically require two separate TCDs (Transfer Control Descriptors): one to perform the data move from system Flash into the ARAM, and a second to load the programming model registers. The SDMA can use the scatter/gather feature to process the two TCDs in a single channel, or 2 channels can be used where the first channel performing the data move “links” to the second channel performing the register load. In any case, the second data transfer loads the programming model using an 80-byte (5 x 16 byte 4-beat bursts) block transfer. Using CH0 as an example, this block data transfer writes from the AMU2_P1PTRCH*n* register (offset address 0x1_0010) to the AMU2_CCR2CH*n* registers (offset address 0x1_005c) before beginning the actual calculation. In this scenario, the programming model initialization configures the ADMA to stream the V[*] array into the ARAM by setting both AMU2_DCCRCH*n*[STMV{32,16}, SRT] and the AMU2_CCR2CH*n*[S1SRT, S23SRT] indicators set to signal the start of the calculation engine for this special data streaming mode.

As before, the AMU completion and error indicators are configured to generate interrupts to the processor; the calculation is paced by the availability of V[*] input data being managed by the ADMA.

The channel performs the calculation and generates an interrupt, signaling successful completion or some type of error termination or an N crossing debug event.

As the processor handles the interrupt, the 64-bit double precision result (AMU2_RES64{M,L}CH*n*) is retrieved along with the status flags in the AMU2_SEIRCH*n* register. After processing the interrupt, the service routine clears the interrupt source by writing a “1” to clear the appropriate flag in the low-order byte of the AMU2_SEIRCH*n* register. The processor then performs any required analysis of the status indicators before saving the result.

As detailed in *Section 94.3.6.4: ADMA V and P3 Data Streaming Modes*, the AMU2 supports streaming of V or P3, or both, concurrently. Additionally, neither the checksum nor the ADMA speed control is supported in this mode. Additionally for this case, the AMU2_VPTRCH*n* register must be reloaded before the next calculation.

94.4.5 Multiple calculations using ADMA for partial data load, SDMA for result retrieval

For this example, let a processor or system DMA perform the *initial* data and programming model register load, and configuring the AMU2 for the first calculation. Next, configure the ADMA to perform repeated loads of the second, third, fourth, … last U[*] vectors and the SDMA to retrieve results as each calculation is finished.

For this scenario, the ADMA would load a data structure from Flash (or a calibration remapped destination memories, or both) having the U[*] vectors and the values for the AMU2_CCR{1,2}CH*n* registers. The configuration sequences through the Flash data

structure and loads each U[*] vector into the same ARAM locations for all calculations. Consider the following sequence of events:

1. A processor or system DMA loads the initial data set and initializes the programming model registers. The initial register load sets AMU2_DCCRCH*n*[WFDNI] to delay the fetching of the next (2nd) U[*] until the first calculation is complete. Additionally, the ADMA is configured to load the next U[*] input vector and the AMU2_CCR{1,2}CH*n* registers by setting AMU2_DCCRCH*n*[SARU, DARU] = [0b10, 0b01]. Finally, as part of the initial register writes, AMU2_CCR1CH*n*[DNICFG] = 0b10 to generate a SDMA request after the ALU calculation completes, and AMU2_CCR2CH*n*[S1SRT, S23SRT] = 1,1 to start the first calculation.
2. As the first complete calculation finishes, the assertion of AMU2_SEIRCH*n*[DNIRQ] is configured to assert an SDMA request and initiate the ADMA to fetch the next U[*] vector from Flash and writing it into the same ARAM locations. Since AMU2_DCCRCH*n*[DARU] = 0b01, the last 64 bits of fetch data are written into the AMU2_CCR{1,2}CH*n* registers. The write sets AMU2_CCR2CH*n*[WFDS1, WFSS23] = 1,1 to conditionally start the second calculation when the ADMA completes the new U[*] fetch. The ADMA fetch also needs to set AMU2_CCR2CH*n*[ADCCR25] = AMU2_DCCRCH*n*[WFDNI] = 1 on all data sets *except* the final iteration. The SDMA responds by copying the AMU2_RES64{M,L}CH*n* and AMU2_SEIRCH*n* registers into system RAM; the AMU2_SEIRCH*n*[DNIRQ] indicator is cleared by the SDMA acknowledge.
3. The process continues with the ALU performing the calculation, the ADMA fetching a new U[*] vector, and the SDMA copying the 64-bit result and status into system RAM until the last U[*] vector is fetched from Flash. The final value loaded into AMU2_CCR1CH*n*[DNICFG] = 0b01 to generate a processor interrupt request when the final calculation completes.

94.5 Operation details

94.5.1 Details on operation completion status indicators

Recall the AMU2_SEIRCH*n* register provides a sizable number of status flags. See *Section 94.2.3.25: AMU Status, Error & Interrupt Channel n Register (AMU2_SEIRCHn)* for the details. This section provides information on the detailed bit status based on a number of expected operation completion states. For ADMA completion state information, see *Table 2383*, while *Table 2384* contains the ALU completion state details.

Table 2383. ADMA Completion State Details in AMU2_SEIRCH*n*

ADMA Completion State	ADMAS	ADMERR	CKSERR	DEIRQ	ADIRQ
Completed error-free	00	0	0	0	1
Error (Bus error, non-correctable ECC event)	01	1	0	1	0
Checksum error with AMU2_DCCRCHn[HCRCE] = 0	00	0	1	0	1
Checksum error with AMU2_DCCRCHn[HCRCE] = 1	01	0	1	1	0

Table 2384. ALU Completion State Details in AMU2_SEIRCH*n*

ALU Completion State	S23XS	S1XS	YCAFLT	ZCAFLT	S23AFLT	S1AFLT	IDVERR	ECCERR	RGCERR	ERIRQ	OVIRQ	NCIRQ	DNIRQ
S1 error-free completion	00	00	0	0	0	0	0	0	0	0	0	0	1(1)
S1 input data validation error	00	01	0	0	0	0	1	0	0	1	0	0	0
S1 ECC error	00	01	0	0	0	0	0	1	0	1	0	0	0
S1 register conflict error	00	01	0	0	0	0	0	0	1	1	0	0	0
S1 overflow	00	01	0	0	0	1	0	0	0	0	1	0	0
S1 underflow	00	00	0	0	0	1	0	0	0	0	0	0	1(1)
S23 error-free completion	00	00	0	0	0	0	0	0	0	0	0	0	1
S23 N crossing	10	00	0	0	0	0	0	0	0	0	0	1	0
S23 input data validation error	01	00	0	0	0	0	1	0	0	1	0	0	0
S23 ECC error	01	00	0	0	0	0	0	1	0	1	0	0	0
S23 register conflict error	01	00	0	0	0	0	0	0	1	1	0	0	0
S23 overflow	01	00	0	0	1	0	0	0	0	0	1	0	0
S23 underflow	00	00	0	0	1	0	0	0	0	0	0	0	1
Final Y32 overflow	01	00	1	0	0	0	0	0	0	0	1	0	0
Final Y32 underflow	00	00	1	0	0	0	0	0	0	0	0	0	1
Z32 overflow	01	00	0	1	0	0	0	0	0	0	1	0	0
Z32 underflow	00	00	0	1	0	0	0	0	0	0	0	0	1

1. For these completion states, if there is a pending Stage23 calculation, the AMU2_SEIRCH*n*[DNIRQ] indicator does *not* set until and unless that calculation completes error-free or with any underflow.

94.5.2 Details on operation stop/resume

The AMU2 supports the ability to temporarily stop an operation and later resume execution. Consider the following examples of the required operations to support this stop/resume control.

94.5.2.1 Stop ALU calculation, check progress, then resume

For this basic stop/resume sequence, the following steps are needed:

1. Set AMU2_CCR2CH*n*[S23STP] = 1.
2. Wait for AMU2_CCR2CH*n*[S23STP] to (auto)clear, indicating the stop request has been processed and the ALU has entered the stopped state. The ALU's stopped state is signaled by AMU2_SEIRCH*n*[S23XS] = 2 (stopped).
3. Check calculation progress as needed.
4. Resume the calculation by setting AMU2_CCR2CH*n*[S23RSM] = 1. As the ALU continues the calculation, AMU2_SEIRCH*n*[S23XS] = 11 (running).

This sequence can be used whether P3[*] or V[*], or both, streaming is enabled or not. If data streaming is enabled, the stopping of the ALU automatically causes the ADMA to stop fetching data since the data transfers are paced by the ALU execution.

If a stop request is made during the final iteration of the stage2 outer loop, the AMU honors the request and stops prior to executing stage3.

94.5.2.2 Stop ALU calculation, perform a new calculation, then restore and resume the original calculation

For this stop/resume sequence, the following steps are needed:

1. Set AMU2_CCR2CH*n*[S23STP] = 1.
2. Wait for AMU2_CCR2CH*n*[S23STP] to (auto)clear, indicating the stop request has been processed and the ALU has entered the stopped state. The ALU's stopped state is signaled by AMU2_SEIRCH*n*[S23XS] = 2 (stopped).
3. Temporarily save the contents of the required programming model. This includes the AMU2_{P3PTR, P4, P5, P6, P7, P8, VPTR, LPTR, CCR{1,2}, Y64, RES32} registers as well as the CH*n* UT[*] in the ARAM data FIFO.
4. Load the new data set, execute the calculation, store the results as required.
5. Restore the saved data contents from Step 3, including restoring the saved Y64 to P9 and RES32 to P5 (see Note below).
6. Continue the calculation by setting AMU2_CCR2CH*n*[S23SRT] = 1. As the ALU continues the calculation, AMU2_SEIRCH*n*[S23XS] = 11 (running).

Again, this sequence can be used whether P3[*] or V[*], or both, streaming is enabled or not. This sequence assumes the two calculations *do not corrupt each other's input data in ARAM*, otherwise software must save/restore the ARAM contents as required.

Note: In the case where the interrupted calculation is configured to initialize Z to P5 as indicated by AMU2_CCR1CHn[SETZ] = 1, then P5 is restored to the original P5 value. Otherwise, P5 is restored to the saved RES32 value. Note that in the latter case, the double precision RES64 value is lost when resuming the interrupted calculation as only the single-precision RES32 can be saved and restored using P5.

94.5.3 AMU2 Performance

The execution time for the M=p7=10, N=p6=500 test case is 10,084 cycles with all the operands pre-loaded into the ARAM. At 200 MHz, this corresponds to an execution time of 50.42 microseconds. This time applies for both a single or two simultaneous channel execution.

The AMU2 execution time, when all operands are pre-loaded into the ARAM, can be approximated by the equation:

Estimated AMU2 Execution Time [cycles] ~= 66 + 2*M + 2*M*(N + (N & 1))

where the constant represents the summation of the time associated with the stage1 and stage2 startup including the initial data FIFO loading and filling the execution pipeline plus the stage3 execution, the second term represents the stage1 time, and the third term represents the stage2 time. The stage1 and stage2 times include the "2*" multiplier since these operations are time-multiplexed across the two ALU channels.

See *Table 2385* for a few estimated and measured execution times with varying M and N values.

Table 2385. AMU2 Execution Times, all data pre-loaded into ARAM

M (p7)	N (p6)	Estimated Execution Time [cycles]	Measured Execution Time [cycles]	Measured Execution Time [usec @ 200 MHz]
10	500	10086	10084	50.42
10	250	5086	5084	25.42
5	500	5076	5076	25.38
5	250	2576	2576	12.88
5	249	2576	2576	12.88
5	248	2556	2556	12.78
5	247	2556	2556	12.78
5	246	2536	2536	12.68

For a 200 MHz operating speed, the Flash access time set to the normal value (RWSC=6) and an otherwise idle Flash, if both P3[*] *and V[*] are streamed by the ADMA* with all the *other* input data already loaded in the ARAM, the incremental execution time is estimated at 1000 cycles producing a total execution time of 11,084 cycles.

Table 2386. Single Channel AMU2 Performance for M=p7=10, N=p6=500

Input Operand Mode	Execution Cycles	Execution Time @ 200 MHz
All pre-loaded into ARAM	10084	50.42 usec
ADMA load of new P3[*] + V32[*][*], all else pre-loaded into ARAM	16959	84.80 usec
ADMA load of new P3[*] + V16[*][*], all else pre-loaded into ARAM	13834	69.17 usec
Stream V32[*][*], all else pre-loaded into ARAM	10086	50.43 usec
Stream V16[*][*], all else pre-loaded into ARAM	10086	50.43 usec
Stream P3[*] + V32[*][*], all else pre-loaded into ARAM	11524	57.62 usec
Stream P3[*] + V16[*][*], all else pre-loaded into ARAM	10090	50.45 usec

95 Real Time Clock / Autonomous Periodic Interrupt (RTC/API)

95.1 Overview

The RTC is a free running counter used for time keeping applications. The RTC may be configured to generate an interrupt at a predefined interval independent of the mode of operation (run mode or low power mode). When the RTC interval is reached, the RTC will generate a wakeup request to the wakeup unit. The RTC can be configured to generate an interrupt on the RTC interval timeout.

The RTC also supports an autonomous periodic interrupt (API) function which can be used to generate a periodic event to the wakeup unit or an interrupt request.

95.2 Features

Features of the RTC/API include:

- 3 selectable counter clock sources:
  - LPRC prescaled by 8 (128 kHz)
  - SXOSC (32 kHz)
  - IRCOSC (16 MHz)
- Clock sources can optionally be prescaled by 512 or 32
- 32-bit counter
  - Supports times up to 1.5 months with 1 ms resolution
  - Runs in all modes of operation, including normal RESET
  - Reset when disabled by software and by POR
- 12-bit compare value to support an extended range of interrupt intervals
- RTC compare value changeable while counter is running
- RTC status and control register are reset only by POR
- Autonomous periodic interrupt (API) to support an extended range of inerrupt intervals
- Configurable interrupt for RTC match, API match, and RTC rollover
- Configurable wakeup event for RTC match, API match, and RTC rollover

Figure 2501. RTC/API block diagram

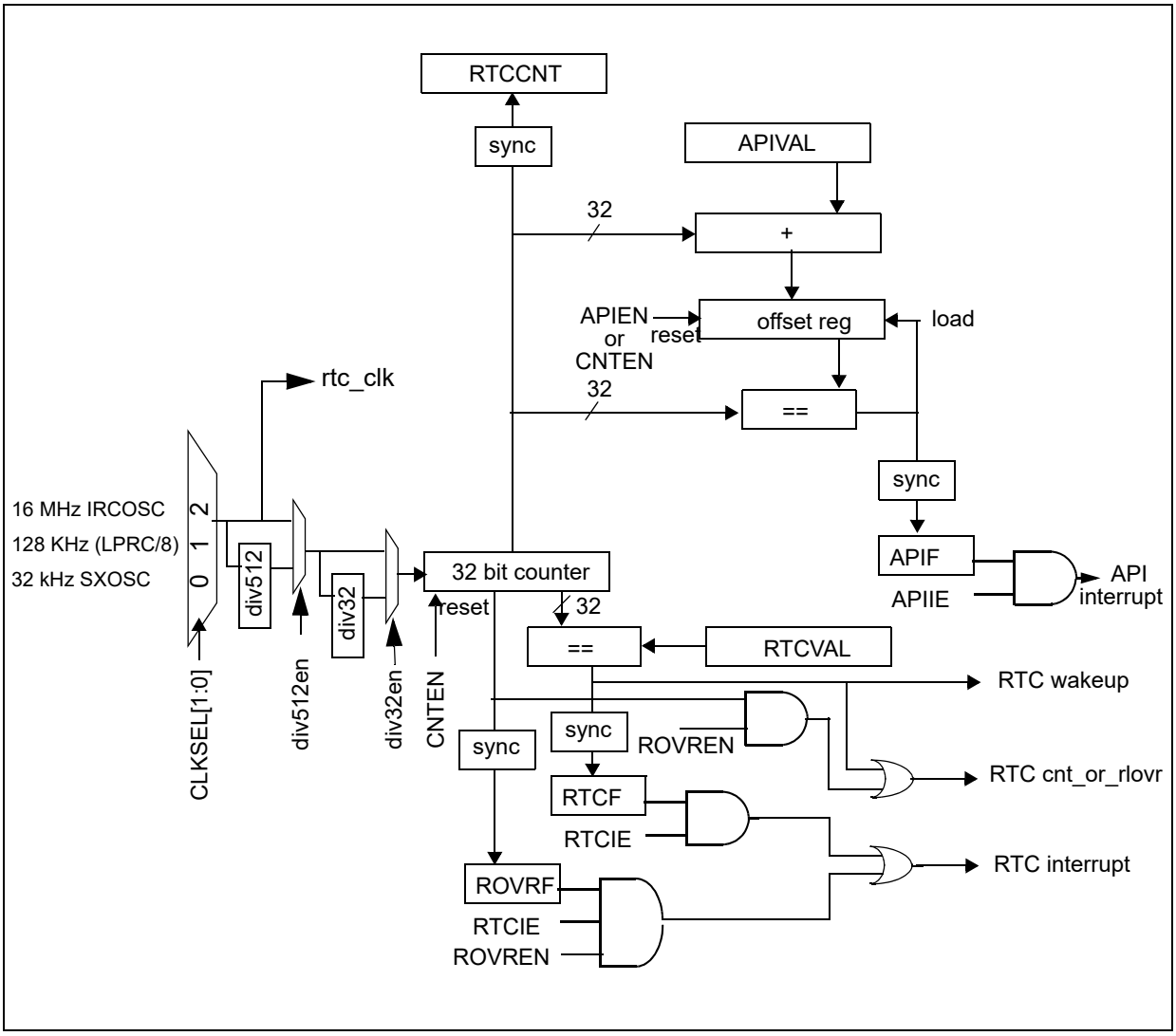
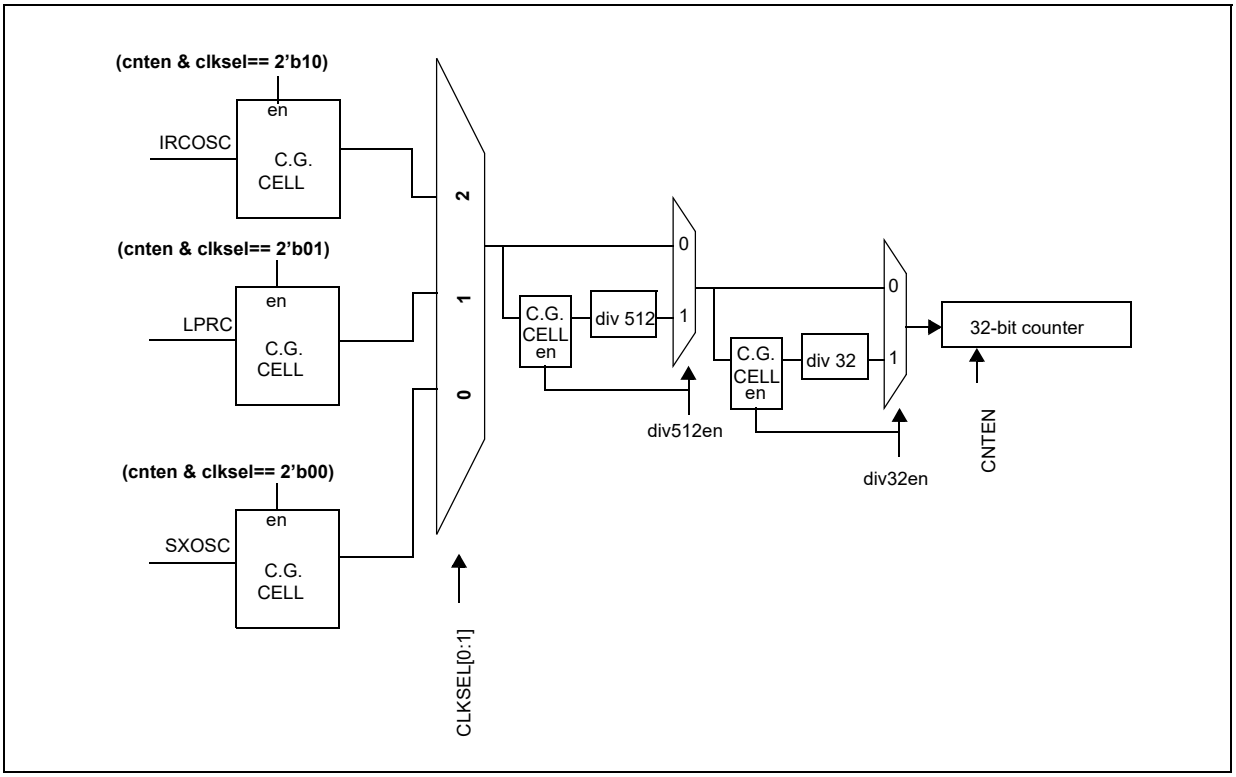


Figure 2502. Clock gating for RTC clocks



95.3 Modes of operation

95.3.1 Functional mode

There are two functional modes of operation for the RTC: normal operation and low power mode. In normal operation, all RTC registers can be read or written. The RTC/API and associated interrupts are optionally enabled. In low power mode, the bus interface is disabled and no configuration changes are permitted. The RTC/API is enabled if enabled prior to entry into low power mode.

95.3.2 Debug mode

If RTCC[FRZEN] is set, the counter will stop on the last valid count when the device enters debug mode. On debug mode exit, the counter will resume counting from the frozen value. If RTCC[FRZEN] is clear, the counter will continue counting (and set flags accordingly) when the device is in debug mode.

95.4 Register descriptions

The registers listed in *Table 2387* are described in the following sections.

Table 2387. RTC/API register map

Address offset	Register	Location
0x0000	RTC Supervisor Control Register (RTCSUPV)	*Section 95.4.1*
0x0004	RTC Control Register (RTCC)	*Section 95.4.2*
0x0008	RTC Status Register (RTCS)	*Section 95.4.3*
0x000C	RTC Counter Register (RTCCNT)	*Section 95.4.4*
0x0010	API Compare Value Register (RTC_APIVAL)	*Section 95.4.5*
0x0014	RTC Compare Value Register (RTC_RTCVAL)	*Section 95.4.6*

95.4.1 RTC Supervisor Control Register (RTCSUPV)

The RTCSUPV register contains the SUPV bit which determines whether other registers are accessible in supervisor mode or user mode.

Note: RTCSUPV register is accessible only in supervisor mode.

Offset: 0x0000 Access: Supervisor R/W

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	SUPV	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

Figure 2503. RTC Supervisor Control Register (RTCSUPV)

Table 2388. RTCSUPV register field descriptions

Field	Description
0 SUPV	RTC Supervisor Bit 0 All registers are accessible in both user as well as supervisor mode. 1 All other registers are accessible in supervisor mode only.

95.4.2 RTC Control Register (RTCC)

The RTCC register contains:

- RTC counter enable
- RTC interrupt enable
- RTC clock source select
- RTC compare value
- API enable
- API interrupt enable
- API compare value

Offset: 0x0004 Access: User R/W

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	CNTEN	RTCIE	FRZEN	ROVREN	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	APIEN	APIIE	CLKSEL		DIV512EN	DIV32EN	0	0	0	0	0	0	0	0	0	TRIG_EN
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 2504. RTC Control Register (RTCC)

Table 2389. RTCC register field descriptions

Field	Description
0 CNTEN	Counter Enable The CNTEN bit enables the RTC counter. Clearing CNTEN has the effect of asynchronously resetting (synchronous reset negation) all the RTC and API logic as well as resetting the 32-bit counter. This allows for the RTC configuration and clock source selection to be updated without causing synchronization issues. 0 Counter disabled and reset 1 Counter enabled
1 RTCIE	RTC Interrupt Enable The RTCIE bit enables interrupts requests to the system if RTCF is asserted. 0 RTC interrupts disabled 1 RTC interrupts enabled
2 FRZEN	Freeze Enable Bit If RTCC[FRZEN] is set, the counter will stop on the last valid count when the device enters debug mode. On debug mode exit, the counter will resume counting from the frozen value. 0 Counter running in debug mode 1 Counter stops (freezes) in debug mode
3 ROVREN	Counter Roll Over Interrupt Enable The ROVREN enables interrupt requests when the RTC has rolled over from 0xFFFF_FFFF to 0x0000_0000. The RTCIE bit must also be set in order to generate an interrupt from a counter rollover. 0 RTC rollover interrupt disabled 1 RTC rollover interrupt enabled
16 APIEN	Autonomous Periodic Interrupt Enable The APIEN bit enables the autonomous periodic interrupt function. 0 API disabled 1 API enabled

Table 2389. RTCC register field descriptions (continued)

Field	Description
17 APIIE	API Interrupt Enable The APIIE bit enables interrupts requests to the system if APIF is asserted. 0 API interrupts disabled 1 API interrupts enabled
18:19 CLKSEL	Clock Select The CLKSEL[0:1] bits select the clock source for the RTC. CLKSEL may only be updated when CNTEN is 0. The user should ensure that oscillator is enabled before selecting it as a clock source for RTC. 00 SXOSC 01 LPRC prescaled by 8 10 IRCOSC 11 Reserved
20 DIV512EN	Divide by 512 enable The DIV512EN bit enables the 512 clock divider. DIV512EN may only be updated when CNTEN is 0. 0 Divide by 512 is disabled. 1 Divide by 512 is enabled.
21 DIV32EN	Divide by 32 enable The DIV32EN bit enables the 32 clock divider. DIV32EN may only be updated when CNTEN is 0. 0 Divide by 32 is disabled. 1 Divide by 32 is enabled.
31 TRIG_EN	Trigger enable Used for self clearing of WKPU API event. This bit is required in case API event is being used to trigger SSWU eCTU commands. 0 Trigger enable is disabled. 1 Trigger enable is enabled.

95.4.3 RTC Status Register (RTCS)

The RTCS register contains:

- RTC interrupt flag
- API interrupt flag
- ROLLOVR Flag

Offset: 0x0008 Access: User R/W

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	RTCF	0	0	0	0	0	0	0	0	0	0	INV_RTC	INV_API	INV_ANL
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	APIF	0	0	ROVRF	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 2505. RTC Status Register (RTCS)

Table 2390. RTCS register field descriptions

Field	Description
2 RTCF	RTC Interrupt Flag The RTCF bit indicates that the RTC counter has reached the counter value matching RTCVAL. RTCF is cleared by writing a 1 to RTCF. Writing a 0 to RTCF has no effect. 0 RTC counter is not equal to RTCVAL. 1 RTC counter matches RTCVAL.
13 INV_RTC	Invalid RTC write This bit becomes value 1 after a value has been written to the RTCVAL register and the synchronization process is continuing. During this synchronization period, any attempt to write to the RTCVAL register again will be ignored.
14 INV_API	Invalid APIVAL write This bit becomes value 1 after a value has been written to the APIVAL register and the synchronization process is continuing. During this synchronization period, any attempt to write to the APIVAL register again will be ignored.
15 INV_ANL	Invalid ANLCMP write This bit becomes value 1 after a value has been written to the ANLCMP_CNT register and the synchronization process is continuing. During this synchronization period, any attempt to write to the ANLCMP_CNT register again will be ignored.
18 APIF	API Interrupt Flag The APIF bit indicates that the RTC counter has reached the counter value matching API offset value. APIF is cleared by writing a 1 to APIF. Writing a 0 to APIF has no effect. 0 No API interrupt 1 API interrupt
21 ROVRF	Counter Roll Over Interrupt Flag The ROVRF bit indicates that the RTC has rolled over from 0xffff_ffff to 0x0000_0000. ROVRF is cleared by writing a 1 to ROVRF. 0 RTC has not rolled over. 1 RTC has rolled over.

95.4.4 RTC Counter Register (RTCCNT)

The RTCCNT register contains the current value of the RTC counter.

Offset: 0x000C Access: User read-only

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	RTCCNT															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	RTCCNT															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 2506. RTC Counter Register (RTCCNT)

Table 2391. RTCCNT register field descriptions

Field	Description
0:31 RTCCNT	RTC Counter Value Due to the clock synchronization, the RTCCNT value may actually represent a previous counter value.

95.4.5 API Compare Value Register (RTC_APIVAL)

The APIVAL bits are compared to the RTC counter bits and if a match occurs, an interrupt/wakeup request is asserted.

Offset: 0x0010 Access: R/W

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	APIVAL															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	APIVAL															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 2507. API Compare Value Register (RTC_APIVAL)

Table 2392. RTC_APIVAL register field descriptions

Field	Description
0:31 APIVAL	API Compare Value The APIVAL bits are compared to the RTC counter bits and if a match occurs, an interrupt/wakeup request is asserted. **Note:** API functionality is active only when APIVAL is non zero. The first API interrupt takes two more cycles because of synchronization of APIVAL to rtc clock. After that, interrupts are periodic in nature and it takes APIVAL+1 cycles. Minimum supported value of APIVAL is 4. This is due to synchronization issues.

95.4.6 RTC Compare Value Register (RTC_RTCVAL)

The RTCVAL bits are compared to the RTC counter bits and if a match occurs, RTCF is set.

Note: If RTC counter is reset (RTCC[CTNEN] = 0), it is required to reload the RTCVAL via software to re-enable the RTCVAL match interrupt before starting the RTC counter.

With RTCC[CTNEN] = 1 (RTC running) the user needs to wait the RTCS[RTC_INV] goes to zero (completed the synchronization process) before updating the RTCVAL counter. Otherwise the latter update will have no effect.

Offset: 0x0014 Access: R/W

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	RTCVAL															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	RTCVAL															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 2508. RTC Compare Value Register (RTC_RTCVAL)

Table 2393. RTC_RTCVAL register field descriptions

Field	Description
0:31 RTCVAL	RTC Compare Value The RTCVAL bits are compared to the RTC counter bits and if a match occurs, RTCF is set. **Note:** Minimum supported value of RTCVAL is 4. This is due to synchronization issues.

95.5 RTC functional description

The RTC consists in a 32-bit free running counter enabled with the RTCC[CNTEN] bit. (When CNTEN is cleared, the counter is asynchronously reset.) The value of the counter may be read via the RTCCNT register. Note that due to the clock synchronization, the RTCCNT value may actually represent a previous counter value. The difference between

the counter and the read value depends on ratio of counter clock and main clock. Maximum possible difference between the two is 6 count values.

The clock source to the counter is selected with the RTCC[CLKSEL] field, which gives four options for clocking the RTC/API. The three clock sources are:

- LPRC prescaled by 8 (128 kHz)
- SXOSC (32 kHz)
- IRCOSC (16 MHz)

The output of the clock mux can be optionally divided by combination of 512 and 32 to give a 1 ms RTC/API count period for different clock sources. Note that the RTCC[CNTEN] bit must be disabled when the RTC/API clock source is switched.

When the counter value matches the 32-bit value in the RTCC[RTCVAL] field, then the RTCS[RTCF] interrupt flag bit is set (after proper clock synchronization). If the RTCC[RTCIE] interrupt enable bit is set, then the RTC interrupt request is generated. If there is a match while in low power mode then the RTC will first generate a wakeup request to force a wakeup to run mode, then the RTCF flag will be set.

A rollover interrupt can be generated when the RTC transitions from a count of 0xFFFF_FFFF to 0x0000_0000. Rollover events are enabled by setting the RTCC[ROVEN] bit. If this bit is set, an RTC rollover will cause a request. To enable interrupts on a rollover request, the RTCC[RTCIE] bit also needs to be set.

All the flags and counter values are synchronized with ipg_clk the main clock. It is assumed that ipg_clk the main clock frequency is always more than or equal to the rtc_clk RTC clock used to run the counter.

95.6 API functional description

Setting the RTCC[APIEN] bit enables the autonomous interrupt function. The 32-bit RTCC[APIVAL] field selects the time interval for triggering an interrupt and/or wakeup event. Since the RTC is a free running counter, the APIVAL is added to the current count to calculate an offset. When the counter reaches the offset count, a interrupt and/or wakeup request is generated. Then the offset value is recalculated and re-triggers a new request when the new value is reached. APIVAL may only be updated when APIEN is disabled. When a compare is reached, the RTCS[APIF] interrupt flag bit is set (after proper clock synchronization). If the RTCC[APIIE] interrupt enable bit is set, then the API interrupt request is generated. If there is a match while in low power mode, then the API will first generate a wakeup request to force a wakeup into normal operation, then the APIF flag will be set.

96 Wakeup unit (WKPU)

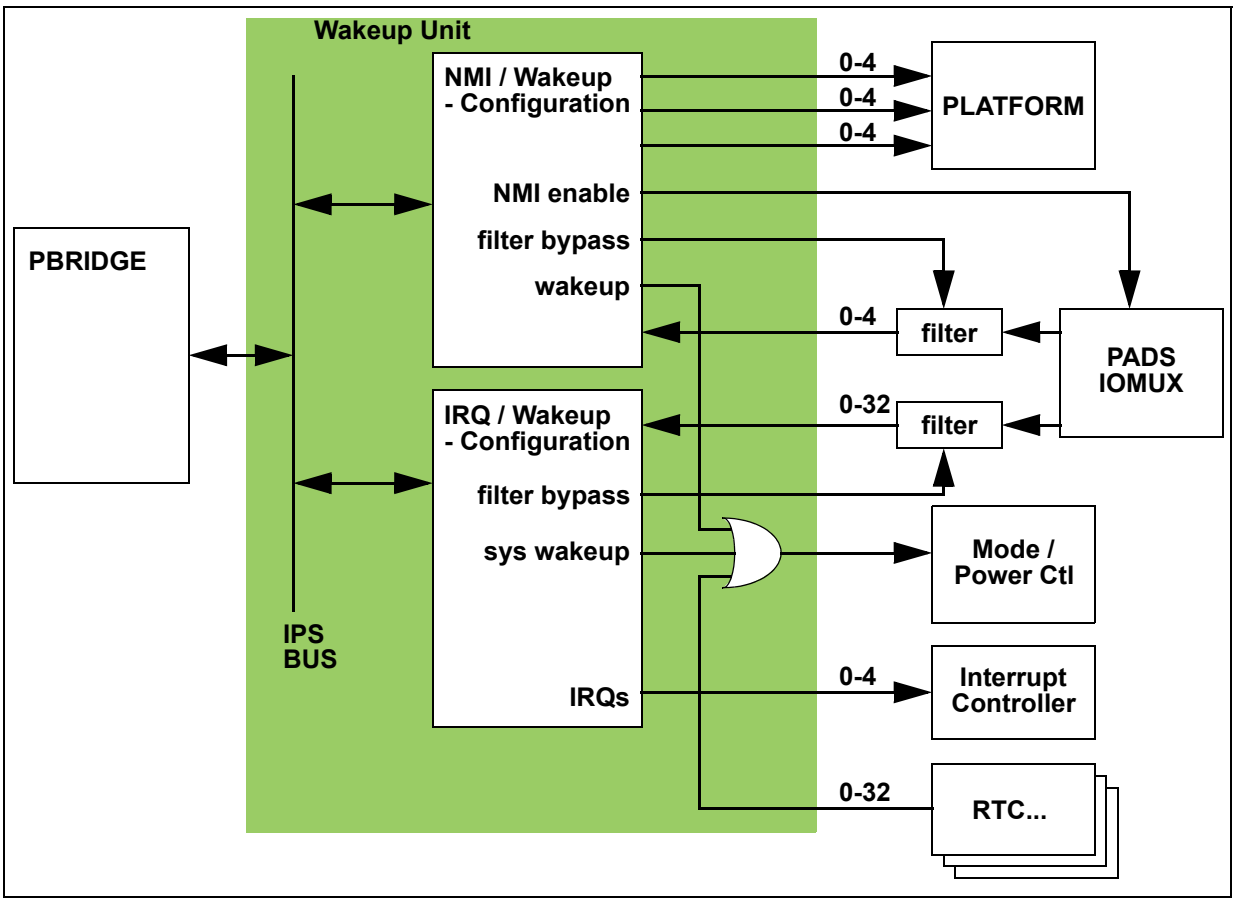
96.1 Introduction

96.1.1 Overview

The Wakeup unit supports up to 32 external sources that can generate CPU exceptions or wakeup events and up to 3 external sources that can cause non-maskable interrupt requests or wakeup events. In addition, it combines its wakeup events with those generated from up to 32 other wakeup sources to supply a single wakeup to the system. This NMI external source can also generate a functional reset event to the MC_RGM. *Figure 2509* is a block diagram of the Wakeup unit and its interfaces to other system components.

Note: For the chip-specific implementation details of this module refer to the Device Configuration chapter.

Figure 2509. Wakeup unit block diagram



96.1.2 Features

The Wakeup unit supports these distinctive features:

- Non-maskable interrupt support with:
  - Up to 4 NMI sources
  - Up to 4 analog glitch filters
  - Independent CPU exceptions destination: non-maskable interrupt, critical interrupt, or machine check request
  - Edge detection
  - Configurable system wakeup triggering from all NMI sources
- External wakeup/interrupt support with:
  - Up to 4 system interrupt vectors for up to 32 interrupt sources
  - Up to 32 analog glitch filters
  - Independent interrupt mask
  - Edge detection
  - Configurable system wakeup triggering from all interrupt sources
  - Configurable pull functionality
- On-chip wakeup support with:
  - Up to 32 wakeup sources
  - Wakeup status mapped to same register as external wakeup/interrupt status

Note: For the chip-specific feature details of this module refer to the Device Configuration chapter.

96.2 External signal description

The Wakeup unit has 0 to 32 signal inputs that can be used as external interrupt sources in normal run mode or as system wakeup sources in certain power down modes.

The module has 0 to 4 signal inputs that can be used as non-maskable interrupt sources in normal run mode or as system wakeup sources in certain power down modes.

96.3 Memory map and register description

This section provides a detailed description of all registers accessible in the WKPU module.

96.3.1 Memory map

Table 2394 gives an overview on the WKPU registers implemented.

Table 2394. WKPU memory map

Address offset	Registers	Supported access sizes	Location
0x0000	NMI Status Flag Register (NSR)	32/16/8	*Section 96.3.2.1*
0x0004–0x0007	Reserved		
0x0008	NMI Configuration Register (NCR)	32/16/8	*Section 96.3.2.2*

Table 2394. WKPU memory map (continued)

Address offset	Registers	Supported access sizes	Location
0x000C–0x0013	Reserved		
0x0014	Wakeup/Interrupt Status Flag Register (WISR)	32	*Section 96.3.2.3*
0x0018	Interrupt Request Enable Register (IRER)	32	*Section 96.3.2.4*
0x001C	Wakeup Request Enable Register (WRER)	32	*Section 96.3.2.5*
0x0020–0x0027	Reserved		
0x0028	Wakeup/Interrupt Rising-Edge Event Enable Register (WIREER)	32	*Section 96.3.2.6*
0x002C	Wakeup/Interrupt Falling-Edge Event Enable Register (WIFEER)	32	*Section 96.3.2.7*
0x0030	Wakeup/Interrupt Filter Enable Register (WIFER)	32	*Section 96.3.2.8*
0x0034	Wakeup/Interrupt Pullup Enable Register (WIPUER)	32	*Section 96.3.2.9*
0x0038–0x3FFF	Reserved		

Note: Reserved registers will read as 0, writes will have no effect. If supported and enabled by the SoC, a transfer error will be issued when trying to access completely reserved register space.

96.3.2 Register descriptions

This section describes in address order all the Wakeup unit registers.

96.3.2.1 NMI Status Flag Register (NSR)

This register holds the non-maskable interrupt status flags.

Offset: 0x0000 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	NIF0	NOVF0	0	0	0	0	0	0	NIF1	NOVF1	0	0	0	0	0	0	NIF2	NOVF2	0	0	0	0	0	0	RIF	ROVF	0	0	0	0	0	0
W	w1c	w1c							w1c	w1c							w1c	w1c							w1c	w1c						
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 2510. NMI Status Flag Register (NSR)

Table 2395. NSR field descriptions

Field	Description
0 NIF0	NMI Status Flag 0 This flag can be cleared only by writing a 1. Writing a 0 has no effect. If enabled (NREE0 or NFEE0 set), NIF0 causes an exception to the associated CPU, as configured within the NDSS bit of the NCR register. In addition this event can also be configured to trigger a system wakeup. 1 An event as defined by NREE0 and NFEE0 has occurred. 0 No event has occurred on the pad.
1 NOVF0	NMI Overrun Status Flag 0 This flag can be cleared only by writing a 1. Writing a 0 has no effect. It will be a copy of the current NIF0 value whenever a NMI event occurs, thereby indicating to the software that a NMI occurred while the last one was not yet serviced. If enabled (NREE0 or NFEE0 set), NOVF0 causes an interrupt request. 1 An overrun has occurred on NMI input 0. 0 No overrun has occurred on NMI input 0.
8 NIF1	NMI Status Flag 1 This flag can be cleared only by writing a 1. Writing a 0 has no effect. If enabled (NREE1 or NFEE1 set), NIF1 causes an exception to the associated CPU, as configured within the NDSS bit of the NCR register. In addition this event can also be configured to trigger a system wakeup. 1 An event as defined by NREE1 and NFEE1 has occurred. 0 No event has occurred on the pad.
9 NOVF1	NMI Overrun Status Flag 1 This flag can be cleared only by writing a 1. Writing a 0 has no effect. It will be a copy of the current NIF1 value whenever a NMI event occurs, thereby indicating to the software that a NMI occurred while the last one was not yet serviced. If enabled (NREE1 or NFEE1 set), NOVF1 causes an interrupt request. 1 An overrun has occurred on NMI input 1. 0 No overrun has occurred on NMI input 1.
16 NIF2	NMI Status Flag 2 This flag can be cleared only by writing a 1. Writing a 0 has no effect. If enabled (NREE2 or NFEE2 set), NIF2 causes an exception to the associated CPU, as configured within the NDSS bit of the NCR register. In addition this event can also be configured to trigger a system wakeup. 1 An event as defined by NREE2 and NFEE2 has occurred. 0 No event has occurred on the pad.
17 NOVF2	NMI Overrun Status Flag 2 This flag can be cleared only by writing a 1. Writing a 0 has no effect. It will be a copy of the current NIF2 value whenever a NMI event occurs, thereby indicating to the software that a NMI occurred while the last one was not yet serviced. If enabled (NREE2 or NFEE2 set), NOVF2 causes an interrupt request. 1 An overrun has occurred on NMI input 2. 0 No overrun has occurred on NMI input 2.

Table 2395. NSR field descriptions (continued)

Field	Description
24 RIF	Reset status Flag This flag can be cleared only by writing a 1. Writing a 0 has no effect. If enabled (that is if RREE or RFEE is set), RIF status bit is set and will cause a functional reset request to RGM if RDSS[1:0] of NCR is '00'. This bit is automatically cleared if the F_WAKEUP reset request is configured in the MC_RGM to generate a long or short "functional" reset. 1 An event as defined by RREE and RFEE has occurred. 0 No event has occurred on the pad.
25 ROVF	Reset overrun status flag This flag can be cleared only by writing a 1. Writing a 0 has no effect. It will be a copy of the current RIF value whenever a NMI event occurs, thereby indicating to the software that an NMI occurred while the last one was not yet serviced. If enabled (RREE or RFEE set), ROVF can cause a reset request to MC_RGM if RDSS[1:0] of NCR is '00'. This bit is automatically cleared if the F_WAKEUP reset request is configured in the MC_RGM to generate a long or short "functional" reset. 1 An overrun has occurred on NMI input. 0 No overrun has occurred on NMI input.

Note: Refer to the Device Configuration chapter to see how the NMI Status Flag Register (NSR) is used in the device.

96.3.2.2 NMI Configuration Register (NCR)

This register holds the configuration bits for the non-maskable interrupt settings.

Offset: 0x0008 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	NLOCK0	NDSS0		NWRE0	0	NREE0	NFEE0	NFE0	NLOCK1	NDSS1		NWRE1	0	NREE1	NFEE1	NFE1	NLOCK2	NDSS2		NWRE2	0	NREE2	NFEE2	NFE2	RLOCK	RDSS		RWRE	0	RREE	RFEE	RFE(1)
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0

Figure 2511. NMI Configuration Register (NCR)

1. Writing this bit has no functional impact.

Table 2396. NCR field descriptions

Field	Description
0 NLOCK0	NMI Configuration Lock Register 0 Writing a 1 to this bit locks the configuration for the NMI until it is unlocked by a system reset. Writing a 0 has no effect.
1:2 NDSS0	NMI Destination Source Select 0 This value is reset on system reset. 00 non-maskable interrupt 01 critical interrupt 10 machine check request 11 reset request

Table 2396. NCR field descriptions (continued)

Field	Description
3 NWRE0	NMI Wakeup Request Enable 0 1 NMI0 wakeup is enabled. A set NIF0 bit or set NOVF0 bit causes a system wakeup request. 0 NMI0 wakeup is disabled. System wakeup requests from the corresponding NIF0 bit are disabled.
5 NREE0	NMI Rising-edge Events Enable 0 1 Rising-edge event is enabled. 0 Rising-edge event is disabled.
6 NFEE0	NMI Falling-edge Events Enable 0 1 Falling-edge event is enabled. 0 Falling-edge event is disabled.
7 NFE0	NMI Filter Enable 0 Enable analog glitch filter on the NMI pad input. 1 Filter is enabled. 0 Filter is disabled.
8 NLOCK1	NMI Configuration Lock Register 1 Writing a 1 to this bit locks the configuration for the NMI until it is unlocked by a system reset. Writing a 0 has no effect.
9:10 NDSS1	NMI Destination Source Select 1 This value is reset on system reset. 00 non-maskable interrupt 01 critical interrupt 10 machine check request 11 reset request
11 NWRE1	NMI Wakeup Request Enable 1 1 NMI1 wakeup is enabled. A set NIF1 bit or set NOVF1 bit causes a system wakeup request 0 NMI1 wakeup is disabled. System wakeup requests from the corresponding NIF1 bit are disabled
13 NREE1	NMI Rising-edge Events Enable 1 1 Rising-edge event is enabled. 0 Rising-edge event is disabled.
14 NFEE1	NMI Falling-edge Events Enable 1 1 Falling-edge event is enabled. 0 Falling-edge event is disabled.
15 NFE1	NMI Filter Enable 1 Enable analog glitch filter on the NMI pad input. 1 Filter is enabled. 0 Filter is disabled.
16 NLOCK2	NMI Configuration Lock Register 2 Writing a 1 to this bit locks the configuration for the NMI until it is unlocked by a system reset. Writing a 0 has no effect.

Table 2396. NCR field descriptions (continued)

Field	Description
17:18 NDSS2	NMI Destination Source Select 2 This value is reset on system reset. 00 non-maskable interrupt 01 critical interrupt 10 machine check request 11 reset request
19 NWRE2	NMI Wakeup Request Enable 2 1 NMI2 wakeup is enabled. A set NIF2 bit or set NOVF2 bit causes a system wakeup request. 0 NMI2 wakeup is disabled. System wakeup requests from the corresponding NIF2 bit are disabled.
21 NREE2	NMI Rising-edge Events Enable 2 1 Rising-edge event is enabled. 0 Rising-edge event is disabled.
22 NFEE2	NMI Falling-edge Events Enable 2 1 Falling-edge event is enabled. 0 Falling-edge event is disabled.
23 NFE2	NMI Filter Enable 2 Enable analog glitch filter on the NMI pad input. 1 Filter is enabled. 0 Filter is disabled.
24 RLOCK	Reset Configuration Lock Register Writing a 1 to this bit locks the configuration for the reset until it is unlocked by a system reset. Writing a 0 has no effect.
25:26 RDSS	Reset Destination Source Select This value is reset on system reset. 00 Reset request to RGM 01 No reaction 10 No reaction 11 No reaction
27 RWRE	Reset Wakeup Request Enable 1 A set RIF bit or set ROVF bit causes a system wakeup request. 0 System wakeup requests from the corresponding RIF bit are disabled. **Note:** To wake up the system from standby by this way, ESR1 option has to be disable in MC_RGM. To disable the MC_RGM ESR1 function, the bit 30 of RGM_FERD and RGM_FEAR registers have to be set.
29 RREE	Reset Rising-edge Events Enable 1 Rising-edge event is enabled. 0 Rising-edge event is disabled.

Table 2396. NCR field descriptions (continued)

Field	Description
30 RFEE	Reset Falling-edge Events Enable 1 Falling-edge event is enabled. 0 Falling-edge event is disabled.
31 RFE	Reset Filter Enable Enable analog glitch filter on the NMI pad input. 1 Filter is enabled. 0 Filter is disabled.

Note: Refer to the Device configuration chapter to see how the NMI Configuration Register (NCR) is used in the device.

Writing a '0' to both NREEx and NFEEx disables the NMI functionality completely (no system wakeup or interrupt will be generated on any pad activity).

96.3.2.3 Wakeup/Interrupt Status Flag Register (WISR)

This register holds the wakeup/interrupt flags.

Offset: 0x0014 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	EIF31	EIF30	EIF29	EIF28	EIF27	EIF26	EIF25	EIF24	EIF23	EIF22	EIF21	EIF20	EIF19	EIF18	EIF17	EIF16	EIF15	EIF14	EIF13	EIF12	EIF11	EIF10	EIF9	EIF8	EIF7	EIF6	EIF5	EIF4	EIF3	EIF2	EIF1	EIF0
W	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 2512. Wakeup/Interrupt Status Flag Register (WISR)

Table 2397. WISR field descriptions

Field	Description
0 EIF31	External Wakeup/Interrupt Status Flag 31 This flag can be cleared only by writing a 1. Writing a 0 has no effect. If enabled (IRER31), EIF31 causes an interrupt request. 1 An event as defined by WIREER and WIFEER has occurred. 0 No event has occurred on the pad.
1 EIF30	External Wakeup/Interrupt Status Flag 30 This flag can be cleared only by writing a 1. Writing a 0 has no effect. If enabled (IRER30), EIF30 causes an interrupt request. 1 An event as defined by WIREER and WIFEER has occurred. 0 No event has occurred on the pad.
2 EIF29	External Wakeup/Interrupt Status Flag 29 This flag can be cleared only by writing a 1. Writing a 0 has no effect. If enabled (IRER29), EIF29 causes an interrupt request. 1 An event as defined by WIREER and WIFEER has occurred. 0 No event has occurred on the pad.

Table 2397. WISR field descriptions (continued)

Field	Description
3 EIF28	External Wakeup/Interrupt Status Flag 28 This flag can be cleared only by writing a 1. Writing a 0 has no effect. If enabled (IRER28), EIF28 causes an interrupt request. 1 An event as defined by WIREER and WIFEER has occurred. 0 No event has occurred on the pad.
4 EIF27	External Wakeup/Interrupt Status Flag 27 This flag can be cleared only by writing a 1. Writing a 0 has no effect. If enabled (IRER27), EIF27 causes an interrupt request. 1 An event as defined by WIREER and WIFEER has occurred. 0 No event has occurred on the pad.
5 EIF26	External Wakeup/Interrupt Status Flag 26 This flag can be cleared only by writing a 1. Writing a 0 has no effect. If enabled (IRER26), EIF26 causes an interrupt request. 1 An event as defined by WIREER and WIFEER has occurred. 0 No event has occurred on the pad.
6 EIF25	External Wakeup/Interrupt Status Flag 25 This flag can be cleared only by writing a 1. Writing a 0 has no effect. If enabled (IRER25), EIF25 causes an interrupt request. 1 An event as defined by WIREER and WIFEER has occurred. 0 No event has occurred on the pad.
7 EIF24	External Wakeup/Interrupt Status Flag 24 This flag can be cleared only by writing a 1. Writing a 0 has no effect. If enabled (IRER24), EIF24 causes an interrupt request. 1 An event as defined by WIREER and WIFEER has occurred. 0 No event has occurred on the pad.
8 EIF23	External Wakeup/Interrupt Status Flag 23 This flag can be cleared only by writing a 1. Writing a 0 has no effect. If enabled (IRER23), EIF23 causes an interrupt request. 1 An event as defined by WIREER and WIFEER has occurred. 0 No event has occurred on the pad.
9 EIF22	External Wakeup/Interrupt Status Flag 22 This flag can be cleared only by writing a 1. Writing a 0 has no effect. If enabled (IRER22), EIF22 causes an interrupt request. 1 An event as defined by WIREER and WIFEER has occurred. 0 No event has occurred on the pad.
10 EIF21	External Wakeup/Interrupt Status Flag 21 This flag can be cleared only by writing a 1. Writing a 0 has no effect. If enabled (IRER21), EIF21 causes an interrupt request. 1 An event as defined by WIREER and WIFEER has occurred. 0 No event has occurred on the pad.
11 EIF20	External Wakeup/Interrupt Status Flag 20 This flag can be cleared only by writing a 1. Writing a 0 has no effect. If enabled (IRER20), EIF20 causes an interrupt request. 1 An event as defined by WIREER and WIFEER has occurred. 0 No event has occurred on the pad.

Table 2397. WISR field descriptions (continued)

Field	Description
12 EIF19	External Wakeup/Interrupt Status Flag 19 This flag can be cleared only by writing a 1. Writing a 0 has no effect. If enabled (IRER19), EIF19 causes an interrupt request. 1 An event as defined by WIREER and WIFEER has occurred. 0 No event has occurred on the pad.
13 EIF18	External Wakeup/Interrupt Status Flag 18 This flag can be cleared only by writing a 1. Writing a 0 has no effect. If enabled (IRER18), EIF18 causes an interrupt request. 1 An event as defined by WIREER and WIFEER has occurred. 0 No event has occurred on the pad.
14 EIF17	External Wakeup/Interrupt Status Flag 17 This flag can be cleared only by writing a 1. Writing a 0 has no effect. If enabled (IRER17), EIF17 causes an interrupt request. 1 An event as defined by WIREER and WIFEER has occurred. 0 No event has occurred on the pad.
15 EIF16	External Wakeup/Interrupt Status Flag 16 This flag can be cleared only by writing a 1. Writing a 0 has no effect. If enabled (IRER16), EIF16 causes an interrupt request. 1 An event as defined by WIREER and WIFEER has occurred. 0 No event has occurred on the pad.
16 EIF15	External Wakeup/Interrupt Status Flag 15 This flag can be cleared only by writing a 1. Writing a 0 has no effect. If enabled (IRER15), EIF15 causes an interrupt request. 1 An event as defined by WIREER and WIFEER has occurred. 0 No event has occurred on the pad.
17 EIF14	External Wakeup/Interrupt Status Flag 14 This flag can be cleared only by writing a 1. Writing a 0 has no effect. If enabled (IRER14), EIF14 causes an interrupt request. 1 An event as defined by WIREER and WIFEER has occurred. 0 No event has occurred on the pad.
18 EIF13	External Wakeup/Interrupt Status Flag 13 This flag can be cleared only by writing a 1. Writing a 0 has no effect. If enabled (IRER13), EIF13 causes an interrupt request. 1 An event as defined by WIREER and WIFEER has occurred. 0 No event has occurred on the pad.
19 EIF12	External Wakeup/Interrupt Status Flag 12 This flag can be cleared only by writing a 1. Writing a 0 has no effect. If enabled (IRER12), EIF12 causes an interrupt request. 1 An event as defined by WIREER and WIFEER has occurred. 0 No event has occurred on the pad.
20 EIF11	External Wakeup/Interrupt Status Flag 11 This flag can be cleared only by writing a 1. Writing a 0 has no effect. If enabled (IRER11), EIF11 causes an interrupt request. 1 An event as defined by WIREER and WIFEER has occurred. 0 No event has occurred on the pad.

Table 2397. WISR field descriptions (continued)

Field	Description
21 EIF10	External Wakeup/Interrupt Status Flag 10 This flag can be cleared only by writing a 1. Writing a 0 has no effect. If enabled (IRER10), EIF10 causes an interrupt request. 1 An event as defined by WIREER and WIFEER has occurred. 0 No event has occurred on the pad.
22 EIF9	External Wakeup/Interrupt Status Flag 9 This flag can be cleared only by writing a 1. Writing a 0 has no effect. If enabled (IRER9), EIF9 causes an interrupt request. 1 An event as defined by WIREER and WIFEER has occurred. 0 No event has occurred on the pad.
23 EIF8	External Wakeup/Interrupt Status Flag 8 This flag can be cleared only by writing a 1. Writing a 0 has no effect. If enabled (IRER8), EIF8 causes an interrupt request. 1 An event as defined by WIREER and WIFEER has occurred. 0 No event has occurred on the pad.
24 EIF7	External Wakeup/Interrupt Status Flag 7 This flag can be cleared only by writing a 1. Writing a 0 has no effect. If enabled (IRER7), EIF7 causes an interrupt request. 1 An event as defined by WIREER and WIFEER has occurred. 0 No event has occurred on the pad.
25 EIF6	External Wakeup/Interrupt Status Flag 6 This flag can be cleared only by writing a 1. Writing a 0 has no effect. If enabled (IRER6), EIF6 causes an interrupt request. 1 An event as defined by WIREER and WIFEER has occurred. 0 No event has occurred on the pad.
26 EIF5	External Wakeup/Interrupt Status Flag 5 This flag can be cleared only by writing a 1. Writing a 0 has no effect. If enabled (IRER5), EIF5 causes an interrupt request. 1 An event as defined by WIREER and WIFEER has occurred. 0 No event has occurred on the pad.
27 EIF4	External Wakeup/Interrupt Status Flag 4 This flag can be cleared only by writing a 1. Writing a 0 has no effect. If enabled (IRER4), EIF4 causes an interrupt request. 1 An event as defined by WIREER and WIFEER has occurred. 0 No event has occurred on the pad.
28 EIF3	External Wakeup/Interrupt Status Flag 3 This flag can be cleared only by writing a 1. Writing a 0 has no effect. If enabled (IRER3), EIF3 causes an interrupt request. 1 An event as defined by WIREER and WIFEER has occurred. 0 No event has occurred on the pad.
29 EIF2	External Wakeup/Interrupt Status Flag 2 This flag can be cleared only by writing a 1. Writing a 0 has no effect. If enabled (IRER2), EIF2 causes an interrupt request. 1 An event as defined by WIREER and WIFEER has occurred. 0 No event has occurred on the pad.

Table 2397. WISR field descriptions (continued)

Field	Description
30 EIF1	External Wakeup/Interrupt Status Flag 1 This flag can be cleared only by writing a 1. Writing a 0 has no effect. If enabled (IRER1), EIF1 causes an interrupt request. 1 An event as defined by WIREER and WIFEER has occurred. 0 No event has occurred on the pad.
31 EIF0	External Wakeup/Interrupt Status Flag 0 This flag can be cleared only by writing a 1. Writing a 0 has no effect. If enabled (IRER0), EIF0 causes an interrupt request. 1 An event as defined by WIREER and WIFEER has occurred. 0 No event has occurred on the pad.

Note: Status bits associated with on-chip wakeup sources are located to the left of the external wakeup/interrupt status bits and are read only. The wakeup for these sources must be configured and cleared at the on-chip wakeup source. Also, the configuration registers for the external interrupts/wakeups do not have corresponding bits.

96.3.2.4 Interrupt Request Enable Register (IRER)

This register is used to enable the interrupt messaging from the wakeup/interrupt pads to the interrupt controller.

Offset: 0x0018 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	EIRE31	EIRE30	EIRE29	EIRE28	EIRE27	EIRE26	EIRE25	EIRE24	EIRE23	EIRE22	EIRE21	EIRE20	EIRE19	EIRE18	EIRE17	EIRE16	EIRE15	EIRE14	EIRE13	EIRE12	EIRE11	EIRE10	EIRE9	EIRE8	EIRE7	EIRE6	EIRE5	EIRE4	EIRE3	EIRE2	EIRE1	EIRE0
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 2513. Interrupt Request Enable Register (IRER)

Table 2398. IRER field descriptions

Field	Description
0 EIRE31	External Interrupt Request Enable 31 1 A set EIF31 bit causes an interrupt request. 0 Interrupt requests from the corresponding EIF31 bit are disabled.
1 EIRE30	External Interrupt Request Enable 30 1 A set EIF30 bit causes an interrupt request. 0 Interrupt requests from the corresponding EIF30 bit are disabled.
2 EIRE29	External Interrupt Request Enable 29 1 A set EIF29 bit causes an interrupt request. 0 Interrupt requests from the corresponding EIF29 bit are disabled.
3 EIRE28	External Interrupt Request Enable 28 1 A set EIF28bit causes an interrupt request. 0 Interrupt requests from the corresponding EIF28 bit are disabled.

Table 2398. IRER field descriptions (continued)

Field	Description
4 EIRE27	External Interrupt Request Enable 27 1 A set EIF27 bit causes an interrupt request. 0 Interrupt requests from the corresponding EIF27 bit are disabled.
5 EIRE26	External Interrupt Request Enable 26 1 A set EIF26 bit causes an interrupt request. 0 Interrupt requests from the corresponding EIF26 bit are disabled.
6 EIRE25	External Interrupt Request Enable 25 1 A set EIF25 bit causes an interrupt request. 0 Interrupt requests from the corresponding EIF25 bit are disabled.
7 EIRE24	External Interrupt Request Enable 24 1 A set EIF24 bit causes an interrupt request. 0 Interrupt requests from the corresponding EIF24 bit are disabled.
8 EIRE23	External Interrupt Request Enable 23 1 A set EIF23 bit causes an interrupt request. 0 Interrupt requests from the corresponding EIF23 bit are disabled.
9 EIRE22	External Interrupt Request Enable 22 1 A set EIF22bit causes an interrupt request. 0 Interrupt requests from the corresponding EIF22 bit are disabled.
10 EIRE21	External Interrupt Request Enable 21 1 A set EIF21 bit causes an interrupt request. 0 Interrupt requests from the corresponding EIF21 bit are disabled.
11 EIRE20	External Interrupt Request Enable 20 1 A set EIF20 bit causes an interrupt request. 0 Interrupt requests from the corresponding EIF20 bit are disabled.
12 EIRE19	External Interrupt Request Enable 19 1 A set EIF19 bit causes an interrupt request. 0 Interrupt requests from the corresponding EIF19 bit are disabled.
13 EIRE18	External Interrupt Request Enable 18 1 A set EIF18 bit causes an interrupt request. 0 Interrupt requests from the corresponding EIF18 bit are disabled.
14 EIRE17	External Interrupt Request Enable 17 1 A set EIF17bit causes an interrupt request. 0 Interrupt requests from the corresponding EIF17 bit are disabled.
15 EIRE16	External Interrupt Request Enable 16 1 A set EIF16 bit causes an interrupt request. 0 Interrupt requests from the corresponding EIF16 bit are disabled.
16 EIRE15	External Interrupt Request Enable 15 1 A set EIF15 bit causes an interrupt request. 0 Interrupt requests from the corresponding EIF15 bit are disabled.
17 EIRE14	External Interrupt Request Enable 14 1 A set EIF14 bit causes an interrupt request. 0 Interrupt requests from the corresponding EIF14 bit are disabled.

Table 2398. IRER field descriptions (continued)

Field	Description
18 EIRE13	External Interrupt Request Enable 13 1 A set EIF13 bit causes an interrupt request. 0 Interrupt requests from the corresponding EIF13 bit are disabled.
19 EIRE12	External Interrupt Request Enable 12 1 A set EIF12 bit causes an interrupt request. 0 Interrupt requests from the corresponding EIF12 bit are disabled.
20 EIRE11	External Interrupt Request Enable 11 1 A set EIF11 bit causes an interrupt request. 0 Interrupt requests from the corresponding EIF11 bit are disabled.
21 EIRE10	External Interrupt Request Enable 10 1 A set EIF10 bit causes an interrupt request. 0 Interrupt requests from the corresponding EIF10 bit are disabled.
22 EIRE9	External Interrupt Request Enable 9 1 A set EIF9 bit causes an interrupt request. 0 Interrupt requests from the corresponding EIF9 bit are disabled.
23 EIRE8	External Interrupt Request Enable 8 1 A set EIF8 bit causes an interrupt request. 0 Interrupt requests from the corresponding EIF8 bit are disabled.
24 EIRE7	External Interrupt Request Enable 7 1 A set EIF7 bit causes an interrupt request. 0 Interrupt requests from the corresponding EIF7 bit are disabled.
25 EIRE6	External Interrupt Request Enable 6 1 A set EIF6 bit causes an interrupt request. 0 Interrupt requests from the corresponding EIF6 bit are disabled.
26 EIRE5	External Interrupt Request Enable 5 1 A set EIF5 bit causes an interrupt request. 0 Interrupt requests from the corresponding EIF5 bit are disabled.
27 EIRE4	External Interrupt Request Enable 4 1 A set EIF4 bit causes an interrupt request. 0 Interrupt requests from the corresponding EIF4 bit are disabled.
28 EIRE3	External Interrupt Request Enable 3 1 A set EIF3 bit causes an interrupt request. 0 Interrupt requests from the corresponding EIF3 bit are disabled.
29 EIRE2	External Interrupt Request Enable 2 1 A set EIF2 bit causes an interrupt request. 0 Interrupt requests from the corresponding EIF2 bit are disabled.
30 EIRE1	External Interrupt Request Enable 1 1 A set EIF1 bit causes an interrupt request. 0 Interrupt requests from the corresponding EIF1 bit are disabled.
31 EIRE0	External Interrupt Request Enable 0 1 A set EIF0 bit causes an interrupt request. 0 Interrupt requests from the corresponding EIF0 bit are disabled.

96.3.2.5 Wakeup Request Enable Register (WRER)

This register is used to enable the system wakeup messaging from the wakeup/interrupt pads to the mode entry and power control modules.

Note: Any wakeup pad must be properly driven either externally or internally (enabling weak pull-up/pull-down) before enabling its corresponding WREx bit.

Offset: 0x001C Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	WRE31	WRE30	WRE29	WRE28	WRE27	WRE26	WRE25	WRE24	WRE23	WRE22	WRE21	WRE20	WRE19	WRE18	WRE17	WRE16	WRE15	WRE14	WRE13	WRE12	WRE11	WRE10	WRE9	WRE8	WRE7	WRE6	WRE5	WRE4	WRE3	WRE2	WRE1	WRE0
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 2514. Wakeup Request Enable Register (WRER)

Table 2399. WRER field descriptions

Field	Description
0 WRE31	External Wakeup Request Enable 31 1 A set EIF31 bit causes a system wakeup request. 0 System wakeup requests from the corresponding EIF31 bit are disabled.
1 WRE30	External Wakeup Request Enable 30 1 A set EIF30 bit causes a system wakeup request. 0 System wakeup requests from the corresponding EIF30 bit are disabled.
2 WRE29	External Wakeup Request Enable 29 1 A set EIF29 bit causes a system wakeup request. 0 System wakeup requests from the corresponding EIF29 bit are disabled.
3 WRE28	External Wakeup Request Enable 28 1 A set EIF28 bit causes a system wakeup request. 0 System wakeup requests from the corresponding EIF28 bit are disabled.
4 WRE27	External Wakeup Request Enable 27 1 A set EIF27 bit causes a system wakeup request. 0 System wakeup requests from the corresponding EIF27 bit are disabled.
5 WRE26	External Wakeup request.Enable 26 1 A set EIF26 bit causes a system wakeup request 0 System wakeup requests from the corresponding EIF26 bit are disabled.
6 WRE25	External Wakeup request.Enable 25 1 A set EIF25 bit causes a system wakeup request 0 System wakeup requests from the corresponding EIF25 bit are disabled.
7 WRE24	External Wakeup request.Enable 24 1 A set EIF24 bit causes a system wakeup request 0 System wakeup requests from the corresponding EIF24 bit are disabled.
8 WRE23	External Wakeup request.Enable 23 1 A set EIF23 bit causes a system wakeup request 0 System wakeup requests from the corresponding EIF23 bit are disabled

Table 2399. WRER field descriptions (continued)

Field	Description
9 WRE22	External Wakeup Request Enable 22 1 A set EIF22 bit causes a system wakeup request. 0 System wakeup requests from the corresponding EIF22 bit are disabled.
10 WRE21	External Wakeup Request Enable 21 1 A set EIF21 bit causes a system wakeup request. 0 System wakeup requests from the corresponding EIF21 bit are disabled.
11 WRE20	External Wakeup Request Enable 20 1 A set EIF20 bit causes a system wakeup request. 0 System wakeup requests from the corresponding EIF20 bit are disabled.
12 WRE19	External Wakeup Request Enable 19 1 A set EIF19 bit causes a system wakeup request. 0 System wakeup requests from the corresponding EIF19 bit are disabled.
13 WRE18	External Wakeup Request Enable 18 1 A set EIF18 bit causes a system wakeup request. 0 System wakeup requests from the corresponding EIF18 bit are disabled.
14 WRE17	External Wakeup request.Enable 17 1 A set EIF17 bit causes a system wakeup request 0 System wakeup requests from the corresponding EIF17 bit are disabled.
15 WRE16	External Wakeup request.Enable 16 1 A set EIF16 bit causes a system wakeup request 0 System wakeup requests from the corresponding EIF16 bit are disabled.
16 WRE15	External Wakeup request.Enable 15 1 A set EIF15 bit causes a system wakeup request 0 System wakeup requests from the corresponding EIF15 bit are disabled.
17 WRE14	External Wakeup request.Enable 14 1 A set EIF14 bit causes a system wakeup request 0 System wakeup requests from the corresponding EIF14 bit are disabled.
18 WRE13	External Wakeup request.Enable 13 1 A set EIF13 bit causes a system wakeup request 0 System wakeup requests from the corresponding EIF13 bit are disabled.
19 WRE12	External Wakeup request.Enable 12 1 A set EIF12 bit causes a system wakeup request 0 System wakeup requests from the corresponding EIF12 bit are disabled.
20 WRE11	External Wakeup request.Enable 11 1 A set EIF11 bit causes a system wakeup request 0 System wakeup requests from the corresponding EIF11 bit are disabled.
21 WRE10	External Wakeup request.Enable 10 1 A set EIF10 bit causes a system wakeup request 0 System wakeup requests from the corresponding EIF10 bit are disabled.
22 WRE9	External Wakeup request.Enable 9 1 A set EIF9 bit causes a system wakeup request 0 System wakeup requests from the corresponding EIF9 bit are disabled.

Table 2399. WRER field descriptions (continued)

Field	Description
23 WRE8	External Wakeup request.Enable 8. 1 A set EIF8 bit causes a system wakeup request 0 System wakeup requests from the corresponding EIF8 bit are disabled
24 WRE7	External Wakeup Request Enable 7 1 A set EIF7 bit causes a system wakeup request. 0 System wakeup requests from the corresponding EIF7 bit are disabled.
25 WRE6	External Wakeup Request Enable 6 1 A set EIF6 bit causes a system wakeup request. 0 System wakeup requests from the corresponding EIF6 bit are disabled.
26 WRE5	External Wakeup Request Enable 5 1 A set EIF5 bit causes a system wakeup request. 0 System wakeup requests from the corresponding EIF5 bit are disabled.
27 WRE4	External Wakeup Request Enable 4 1 A set EIF4 bit causes a system wakeup request. 0 System wakeup requests from the corresponding EIF4 bit are disabled.
28 WRE3	External Wakeup Request Enable 3 1 A set EIF3 bit causes a system wakeup request. 0 System wakeup requests from the corresponding EIF3 bit are disabled.
29 WRE2	External Wakeup Request Enable 2 1 A set EIF2 bit causes a system wakeup request. 0 System wakeup requests from the corresponding EIF2 bit are disabled.
30 WRE1	External Wakeup Request Enable 1 1 A set EIF1 bit causes a system wakeup request. 0 System wakeup requests from the corresponding EIF1 bit are disabled.
31 WRE0	External Wakeup Request Enable 0 1 A set EIF0 bit causes a system wakeup request 0 System wakeup requests from the corresponding EIF0 bit are disabled.

96.3.2.6 Wakeup/Interrupt Rising-Edge Event Enable Register (WIREER)

This register is used to enable rising-edge triggered events on the corresponding wakeup/interrupt pads.

Offset: 0x0028 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	IREE31	IREE30	IREE29	IREE28	IREE27	IREE26	IREE25	IREE24	IREE23	IREE22	IREE21	IREE20	IREE19	IREE18	IREE17	IREE16	IREE15	IREE14	IREE13	IREE12	IREE11	IREE10	IREE9	IREE8	IREE7	IREE6	IREE5	IREE4	IREE3	IREE2	IREE1	IREE0
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 2515. Wakeup/Interrupt Rising-Edge Event Enable Register (WIREER)

Table 2400. WIREER field descriptions

Field	Description
0 IREE31	External Interrupt Rising-edge Events Enable 31 1 Rising-edge event is enabled. 0 Rising-edge event is disabled.
1 IREE30	External Interrupt Rising-edge Events Enable 30 1 Rising-edge event is enabled. 0 Rising-edge event is disabled.
2 IREE29	External Interrupt Rising-edge Events Enable 29 1 Rising-edge event is enabled. 0 Rising-edge event is disabled.
3 IREE28	External Interrupt Rising-edge Events Enable 28 1 Rising-edge event is enabled. 0 Rising-edge event is disabled.
4 IREE27	External Interrupt Rising-edge Events Enable 27 1 Rising-edge event is enabled. 0 Rising-edge event is disabled.
5 IREE26	External Interrupt Rising-edge Events Enable 26 1 Rising-edge event is enabled. 0 Rising-edge event is disabled.
6 IREE25	External Interrupt Rising-edge Events Enable 25 1 Rising-edge event is enabled. 0 Rising-edge event is disabled.
7 IREE24	External Interrupt Rising-edge Events Enable 24 1 Rising-edge event is enabled. 0 Rising-edge event is disabled.
8 IREE23	External Interrupt Rising-edge Events Enable 23 1 Rising-edge event is enabled. 0 Rising-edge event is disabled.
9 IREE22	External Interrupt Rising-edge Events Enable 22 1 Rising-edge event is enabled. 0 Rising-edge event is disabled.
10 IREE21	External Interrupt Rising-edge Events Enable 21 1 Rising-edge event is enabled. 0 Rising-edge event is disabled.
11 IREE20	External Interrupt Rising-edge Events Enable 20 1 Rising-edge event is enabled. 0 Rising-edge event is disabled.
12 IREE19	External Interrupt Rising-edge Events Enable 19 1 Rising-edge event is enabled. 0 Rising-edge event is disabled.
13 IREE18	External Interrupt Rising-edge Events Enable 18 1 Rising-edge event is enabled. 0 Rising-edge event is disabled.

Table 2400. WIREER field descriptions (continued)

Field	Description
14 IREE17	External Interrupt Rising-edge Events Enable 17 1 Rising-edge event is enabled. 0 Rising-edge event is disabled.
15 IREE16	External Interrupt Rising-edge Events Enable 16 1 Rising-edge event is enabled. 0 Rising-edge event is disabled.
16 IREE15	External Interrupt Rising-edge Events Enable 15 1 Rising-edge event is enabled. 0 Rising-edge event is disabled.
17 IREE14	External Interrupt Rising-edge Events Enable 14 1 Rising-edge event is enabled. 0 Rising-edge event is disabled.
18 IREE13	External Interrupt Rising-edge Events Enable 13 1 Rising-edge event is enabled. 0 Rising-edge event is disabled.
19 IREE12	External Interrupt Rising-edge Events Enable 12 1 Rising-edge event is enabled. 0 Rising-edge event is disabled.
20 IREE11	External Interrupt Rising-edge Events Enable 11 1 Rising-edge event is enabled. 0 Rising-edge event is disabled.
21 IREE10	External Interrupt Rising-edge Events Enable 10 1 Rising-edge event is enabled. 0 Rising-edge event is disabled.
22 IREE9	External Interrupt Rising-edge Events Enable 9 1 Rising-edge event is enabled. 0 Rising-edge event is disabled.
23 IREE8	External Interrupt Rising-edge Events Enable 8 1 Rising-edge event is enabled. 0 Rising-edge event is disabled.
24 IREE7	External Interrupt Rising-edge Events Enable 7 1 Rising-edge event is enabled. 0 Rising-edge event is disabled.
25 IREE6	External Interrupt Rising-edge Events Enable 6 1 Rising-edge event is enabled. 0 Rising-edge event is disabled.
26 IREE5	External Interrupt Rising-edge Events Enable 5 1 Rising-edge event is enabled. 0 Rising-edge event is disabled.
27 IREE4	External Interrupt Rising-edge Events Enable 4 1 Rising-edge event is enabled. 0 Rising-edge event is disabled.

Table 2400. WIREER field descriptions (continued)

Field	Description
28 IREE3	External Interrupt Rising-edge Events Enable 3 1 Rising-edge event is enabled. 0 Rising-edge event is disabled.
29 IREE2	External Interrupt Rising-edge Events Enable 2 1 Rising-edge event is enabled. 0 Rising-edge event is disabled.
30 IREE1	External Interrupt Rising-edge Events Enable 1 1 Rising-edge event is enabled. 0 Rising-edge event is disabled.
31 IREE0	External Interrupt Rising-edge Events Enable 0 1 Rising-edge event is enabled. 0 Rising-edge event is disabled.

96.3.2.7 Wakeup/Interrupt Falling-Edge Event Enable Register (WIFEER)

This register is used to enable falling-edge triggered events on the corresponding wakeup/interrupt pads.

Offset: 0x002C Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	IFEE31	IFEE30	IFEE29	IFEE28	IFEE27	IFEE26	IFEE25	IFEE24	IFEE23	IFEE22	IFEE21	IFEE20	IFEE19	IFEE18	IFEE17	IFEE16	IFEE15	IFEE14	IFEE13	IFEE12	IFEE11	IFEE10	IFEE9	IFEE8	IFEE7	IFEE6	IFEE5	IFEE4	IFEE3	IFEE2	IFEE1	IFEE0
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 2516. Wakeup/Interrupt Falling-Edge Event Enable Register (WIFEER)

Table 2401. WIFEER field descriptions

Field	Description
0 IFEE31	External Interrupt Falling-edge Events Enable 31 1 Falling-edge event is enabled. 0 Falling-edge event is disabled.
1 IFEE30	External Interrupt Falling-edge Events Enable 30 1 Falling-edge event is enabled. 0 Falling-edge event is disabled.
2 IFEE29	External Interrupt Falling-edge Events Enable 29 1 Falling-edge event is enabled. 0 Falling-edge event is disabled.
3 IFEE28	External Interrupt Falling-edge Events Enable 28 1 Falling-edge event is enabled. 0 Falling-edge event is disabled.
4 IFEE27	External Interrupt Falling-edge Events Enable 27 1 Falling-edge event is enabled. 0 Falling-edge event is disabled.

Table 2401. WIFEER field descriptions (continued)

Field	Description
5 IFEE26	External Interrupt Falling-edge Events Enable 26 1 Falling-edge event is enabled. 0 Falling-edge event is disabled.
6 IFEE25	External Interrupt Falling-edge Events Enable 25 1 Falling-edge event is enabled. 0 Falling-edge event is disabled.
7 IFEE24	External Interrupt Falling-edge Events Enable 24 1 Falling-edge event is enabled. 0 Falling-edge event is disabled.
8 IFEE23	External Interrupt Falling-edge Events Enable 23 1 Falling-edge event is enabled. 0 Falling-edge event is disabled.
9 IFEE22	External Interrupt Falling-edge Events Enable 22 1 Falling-edge event is enabled. 0 Falling-edge event is disabled.
10 IFEE21	External Interrupt Falling-edge Events Enable 21 1 Falling-edge event is enabled. 0 Falling-edge event is disabled.
11 IFEE20	External Interrupt Falling-edge Events Enable 20 1 Falling-edge event is enabled. 0 Falling-edge event is disabled.
12 IFEE19	External Interrupt Falling-edge Events Enable 19 1 Falling-edge event is enabled. 0 Falling-edge event is disabled.
13 IFEE18	External Interrupt Falling-edge Events Enable 18 1 Falling-edge event is enabled. 0 Falling-edge event is disabled.
14 IFEE17	External Interrupt Falling-edge Events Enable 17 1 Falling-edge event is enabled. 0 Falling-edge event is disabled.
15 IFEE16	External Interrupt Falling-edge Events Enable 16 1 Falling-edge event is enabled. 0 Falling-edge event is disabled.
16 IFEE15	External Interrupt Falling-edge Events Enable 15 1 Falling-edge event is enabled. 0 Falling-edge event is disabled.
17 IFEE14	External Interrupt Falling-edge Events Enable 14 1 Falling-edge event is enabled. 0 Falling-edge event is disabled.
18 IFEE13	External Interrupt Falling-edge Events Enable 13 1 Falling-edge event is enabled. 0 Falling-edge event is disabled.

Table 2401. WIFEER field descriptions (continued)

Field	Description
19 IFEE12	External Interrupt Falling-edge Events Enable 12 1 Falling-edge event is enabled. 0 Falling-edge event is disabled.
20 IFEE11	External Interrupt Falling-edge Events Enable 11 1 Falling-edge event is enabled. 0 Falling-edge event is disabled.
21 IFEE10	External Interrupt Falling-edge Events Enable 10 1 Falling-edge event is enabled. 0 Falling-edge event is disabled.
22 IFEE9	External Interrupt Falling-edge Events Enable 9 1 Falling-edge event is enabled. 0 Falling-edge event is disabled.
23 IFEE8	External Interrupt Falling-edge Events Enable 8 1 Falling-edge event is enabled. 0 Falling-edge event is disabled.
24 IFEE7	External Interrupt Falling-edge Events Enable 7 1 Falling-edge event is enabled. 0 Falling-edge event is disabled.
25 IFEE6	External Interrupt Falling-edge Events Enable 6 1 Falling-edge event is enabled. 0 Falling-edge event is disabled.
26 IFEE5	External Interrupt Falling-edge Events Enable 5 1 Falling-edge event is enabled. 0 Falling-edge event is disabled.
27 IFEE4	External Interrupt Falling-edge Events Enable 4 1 Falling-edge event is enabled. 0 Falling-edge event is disabled.
28 IFEE3	External Interrupt Falling-edge Events Enable 3 1 Falling-edge event is enabled. 0 Falling-edge event is disabled.
29 IFEE2	External Interrupt Falling-edge Events Enable 2 1 Falling-edge event is enabled. 0 Falling-edge event is disabled.
30 IFEE1	External Interrupt Falling-edge Events Enable 1 1 Falling-edge event is enabled. 0 Falling-edge event is disabled.
31 IFEE0	External Interrupt Falling-edge Events Enable 0 1 Falling-edge event is enabled. 0 Falling-edge event is disabled.

96.3.2.8 Wakeup/Interrupt Filter Enable Register (WIFER)

This register is used to enable an analog filter on the corresponding interrupt pads to filter out glitches on the inputs. The number of wakeups/interrupts supporting this feature is SoC dependent and can be configured to be between 1 and 32.

Note: Glitch filter must be enabled in case the wake-up sources are used to exit from Standby.

Offset: 0x0030 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	IFE31	IFE30	IFE29	IFE28	IFE27	IFE26	IFE25	IFE24	IFE23	IFE22	IFE21	IFE20	IFE19	IFE18	IFE17	IFE16	IFE15	IFE14	IFE13	IFE12	IFE11	IFE10	IFE9	IFE8	IFE7	IFE6	IFE5	IFE4	IFE3	IFE2	IFE1	IFE0
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 2517. Wakeup/Interrupt Filter Enable Register (WIFER)

Table 2402. WIFER field descriptions

Field	Description
0 IFE31	External Interrupt Filter Enable 31 Enable analog glitch filter on the external interrupt pad input. 1 Filter is enabled. 0 Filter is disabled.
1 IFE30	External Interrupt Filter Enable 30 Enable analog glitch filter on the external interrupt pad input. 1 Filter is enabled. 0 Filter is disabled.
2 IFE29	External Interrupt Filter Enable 29 Enable analog glitch filter on the external interrupt pad input. 1 Filter is enabled. 0 Filter is disabled.
3 IFE28	External Interrupt Filter Enable 28 Enable analog glitch filter on the external interrupt pad input. 1 Filter is enabled. 0 Filter is disabled.
4 IFE27	External Interrupt Filter Enable 27 Enable analog glitch filter on the external interrupt pad input. 1 Filter is enabled. 0 Filter is disabled.
5 IFE26	External Interrupt Filter Enable 26 Enable analog glitch filter on the external interrupt pad input. 1 Filter is enabled. 0 Filter is disabled.
6 IFE25	External Interrupt Filter Enable 25 Enable analog glitch filter on the external interrupt pad input. 1 Filter is enabled. 0 Filter is disabled.

Table 2402. WIFER field descriptions (continued)

Field	Description
7 IFE24	External Interrupt Filter Enable 24 Enable analog glitch filter on the external interrupt pad input. 1 Filter is enabled. 0 Filter is disabled.
8 IFE23	External Interrupt Filter Enable 23 Enable analog glitch filter on the external interrupt pad input. 1 Filter is enabled. 0 Filter is disabled.
9 IFE22	External Interrupt Filter Enable 22 Enable analog glitch filter on the external interrupt pad input. 1 Filter is enabled. 0 Filter is disabled.
10 IFE21	External Interrupt Filter Enable 21 Enable analog glitch filter on the external interrupt pad input. 1 Filter is enabled. 0 Filter is disabled.
11 IFE20	External Interrupt Filter Enable 20 Enable analog glitch filter on the external interrupt pad input. 1 Filter is enabled. 0 Filter is disabled.
12 IFE19	External Interrupt Filter Enable 19 Enable analog glitch filter on the external interrupt pad input. 1 Filter is enabled. 0 Filter is disabled.
13 IFE18	External Interrupt Filter Enable 18 Enable analog glitch filter on the external interrupt pad input. 1 Filter is enabled. 0 Filter is disabled.
14 IFE17	External Interrupt Filter Enable 17 Enable analog glitch filter on the external interrupt pad input. 1 Filter is enabled. 0 Filter is disabled.
15 IFE16	External Interrupt Filter Enable 16 Enable analog glitch filter on the external interrupt pad input. 1 Filter is enabled. 0 Filter is disabled.
16 IFE15	External Interrupt Filter Enable 15 Enable analog glitch filter on the external interrupt pad input. 1 Filter is enabled. 0 Filter is disabled.
17 IFE14	External Interrupt Filter Enable 14 Enable analog glitch filter on the external interrupt pad input. 1 Filter is enabled. 0 Filter is disabled.

Table 2402. WIFER field descriptions (continued)

Field	Description
18 IFE13	External Interrupt Filter Enable 13 Enable analog glitch filter on the external interrupt pad input. 1 Filter is enabled. 0 Filter is disabled.
19 IFE12	External Interrupt Filter Enable 12 Enable analog glitch filter on the external interrupt pad input. 1 Filter is enabled. 0 Filter is disabled.
20 IFE11	External Interrupt Filter Enable 11 Enable analog glitch filter on the external interrupt pad input. 1 Filter is enabled. 0 Filter is disabled.
21 IFE10	External Interrupt Filter Enable 10 Enable analog glitch filter on the external interrupt pad input. 1 Filter is enabled. 0 Filter is disabled.
22 IFE9	External Interrupt Filter Enable 9 Enable analog glitch filter on the external interrupt pad input. 1 Filter is enabled. 0 Filter is disabled.
23 IFE8	External Interrupt Filter Enable 8 Enable analog glitch filter on the external interrupt pad input. 1 Filter is enabled. 0 Filter is disabled.
24 IFE7	External Interrupt Filter Enable 7 Enable analog glitch filter on the external interrupt pad input. 1 Filter is enabled. 0 Filter is disabled.
25 IFE6	External Interrupt Filter Enable 6 Enable analog glitch filter on the external interrupt pad input. 1 Filter is enabled. 0 Filter is disabled.
26 IFE5	External Interrupt Filter Enable 5 Enable analog glitch filter on the external interrupt pad input. 1 Filter is enabled. 0 Filter is disabled.
27 IFE4	External Interrupt Filter Enable 4 Enable analog glitch filter on the external interrupt pad input. 1 Filter is enabled. 0 Filter is disabled.
28 IFE3	External Interrupt Filter Enable 3 Enable analog glitch filter on the external interrupt pad input. 1 Filter is enabled. 0 Filter is disabled.

Table 2402. WIFER field descriptions (continued)

Field	Description
29 IFE2	External Interrupt Filter Enable 2 Enable analog glitch filter on the external interrupt pad input. 1 Filter is enabled. 0 Filter is disabled.
30 IFE1	External Interrupt Filter Enable 1 Enable analog glitch filter on the external interrupt pad input. 1 Filter is enabled. 0 Filter is disabled.
31 IFE0	External Interrupt Filter Enable 0 Enable analog glitch filter on the external interrupt pad input. 1 Filter is enabled. 0 Filter is disabled.

96.3.2.9 Wakeup/Interrupt Pull Enable Register (WIPUER)

This register is used to enable a pull on the corresponding interrupt pads. The pull direction (up or down) is selected by SIUL2_MSCR_IO_n.WPDE-WPUE bits.

Note: WIPUER bits related to those bonded LP pads that are used to wake-up the device from STANDBY Mode, should be left at 0 default value.

- *Bits setting has undesired effect in STANDBY mode (extra consumption or anticipated exiting from STANDBY mode)*
- *In running mode, pull-up/down IO configuration shall be done only through SIUL*

Offset: 0x0034 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	IPUE31	IPUE30	IPUE29	IPUE28	IPUE27	IPUE26	IPUE25	IPUE24	IPUE23	IPUE22	IPUE21	IPUE20	IPUE19	IPUE18	IPUE17	IPUE16	IPUE15	IPUE14	IPUE13	IPUE12	IPUE11	IPUE10	IPUE9	IPUE8	IPUE7	IPUE6	IPUE5	IPUE4	IPUE3	IPUE2	IPUE1	IPUE0
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 2518. Wakeup/Interrupt Pullup Enable Register (WIPUER)

Table 2403. WIPUER field descriptions

Field	Description[1]
0 IPUE31	External Interrupt Pull Enable 31 1 Pull is enabled. 0 Pull is disabled.
1 IPUE30	External Interrupt Pull Enable 30 1 Pull is enabled. 0 Pull is disabled.
2 IPUE29	External Interrupt Pull Enable 29 1 Pull is enabled. 0 Pull is disabled.

Table 2403. WIPUER field descriptions (continued)

Field	Description[1]
3 IPUE28	External Interrupt Pull Enable 28 1 Pull is enabled. 0 Pull is disabled.
4 IPUE27	External Interrupt Pull Enable 27 1 Pull is enabled. 0 Pull is disabled.
5 IPUE26	External Interrupt Pull Enable 26 1 Pull is enabled. 0 Pull is disabled.
6 IPUE25	External Interrupt Pull Enable 25 1 Pull is enabled. 0 Pull is disabled.
7 IPUE24	External Interrupt Pull Enable 24 1 Pull is enabled. 0 Pull is disabled.
8 IPUE23	External Interrupt Pull Enable 23 1 Pull is enabled. 0 Pull is disabled.
9 IPUE22	External Interrupt Pull Enable 22 1 Pull is enabled. 0 Pull is disabled.
10 IPUE21	External Interrupt Pull Enable 21 1 Pull is enabled. 0 Pull is disabled.
11 IPUE20	External Interrupt Pull Enable 20 1 Pull is enabled. 0 Pull is disabled.
12 IPUE19	External Interrupt Pull Enable 19 1 Pull is enabled. 0 Pull is disabled.
13 IPUE18	External Interrupt Pull Enable 18 1 Pull is enabled. 0 Pull is disabled.
14 IPUE17	External Interrupt Pull Enable 17 1 Pull is enabled. 0 Pull is disabled.
15 IPUE16	External Interrupt Pull Enable 16 1 Pull is enabled. 0 Pull is disabled.
16 IPUE15	External Interrupt Pull Enable 15 1 Pull is enabled. 0 Pull is disabled.

Table 2403. WIPUER field descriptions (continued)

Field	Description[1]
17 IPUE14	External Interrupt Pull Enable 14 1 Pull is enabled. 0 Pull is disabled.
18 IPUE13	External Interrupt Pull Enable 13 1 Pull is enabled. 0 Pull is disabled.
19 IPUE12	External Interrupt Pull Enable 12 1 Pull is enabled. 0 Pull is disabled.disabled.
20 IPUE11	External Interrupt Pull Enable 11 1 Pull is enabled. 0 Pull is disabled.
21 IPUE10	External Interrupt Pull Enable 10 1 Pull is enabled. 0 Pull is disabled.
22 IPUE9	External Interrupt Pull Enable 9 1 Pull is enabled. 0 Pull is disabled.
23 IPUE8	External Interrupt Pull Enable 8 1 Pull is enabled. 0 Pull is disabled.
24 IPUE7	External Interrupt Pull Enable 7 1 Pull is enabled. 0 Pull is disabled.
25 IPUE6	External Interrupt Pull Enable 6 1 Pull is enabled. 0 Pull is disabled.
26 IPUE5	External Interrupt Pull Enable 5 1 Pull is enabled. 0 Pull is disabled.
27 IPUE4	External Interrupt Pull Enable 4 1 Pull is enabled. 0 Pull is disabled.
28 IPUE3	External Interrupt Pull Enable 3 1 Pull is enabled. 0 Pull is disabled.
29 IPUE2	External Interrupt Pull Enable 2 1 Pull is enabled. 0 Pull is disabled.

Table 2403. WIPUER field descriptions (continued)

Field	Description[1]
30 IPUE1	External Interrupt Pull Enable 1 1 Pull is enabled. 0 Pull is disabled.
31 IPUE0	External Interrupt Pull Enable 0 1 Pull is enabled. 0 Pull is disabled.

1. For the actual chip-specific bit description details refer to the Device Configuration chapter.

96.4 Functional description

96.4.1 General

This section provides a complete functional description of the Wakeup unit.

96.4.2 Non-maskable interrupts

The Wakeup unit supports up to 4 non-maskable interrupts which can be allocated to any pad necessary at the SoC level. This allocation is fixed per SoC.

The Wakeup unit supports the generation of 3 types of interrupts per NMI input to the SoC. The Wakeup unit supports the capturing of a second event per NMI input before the interrupt is cleared, thus reducing the chance of losing an NMI event.

Each NMI passes through a by-passable analog glitch filter.

Note: Glitch filter control and pad configuration should be done while the NMI is disabled in order to avoid erroneous triggering by glitches caused by the configuration process itself.

96.4.2.1 NMI management

Each NMI can be enabled or disabled independently. This can be performed using the single NCR register laid out to contain all configuration bits for a given NMI in a single byte (see *Figure 2511*). A pad defined as an NMI can be configured by the user to recognize interrupts with an active rising edge, an active falling edge or both edges being active. A setting of having both edge events disabled results in no interrupt being detected and should not be configured.

The active NMI edge is controlled by the user through the configuration of the NREE and NFEE bits.

Note: After reset, NREE and NFEE are set to '0', therefore the NMI functionality is disabled after reset and must be enabled explicitly by software.

Once a pad's NMI functionality has been enabled, the pad cannot be reconfigured in the IOMUX to override or disable the NMI.

The NMI destination interrupt is controlled by the user through the configuration of the NDSS bits. See *Table 2396* for details.

Each NMI supports a status flag and an overrun flag which are located in the NSR register (see *Figure 2510*). This register is a clear-by-write-1 register type, preventing inadvertent overwriting of other flags in the same register. The status flag is set whenever an NMI event

is detected. The overrun flag is set whenever an NMI event is detected and the status flag is set (it has not yet been cleared).

Note: The overrun flag is cleared by writing a '1' to the appropriate overrun bit in the NSR register. If the status bit is cleared and the overrun bit is still set, the pending interrupt will not be cleared.

96.4.3 External wakeups/interrupts

The Wakeup unit supports up to 32 external wakeup/interrupts which can be allocated to any pad necessary at the SoC level. This allocation is fixed per SoC.

The Wakeup unit supports up to 4 interrupt vectors to the interrupt controller of the SoC. Each interrupt vector can support up to the number of external interrupt sources from the device pads with the total across all vectors being equal to the number of external interrupt sources. Each external interrupt sources is assigned to exactly one interrupt vector. The interrupt vector assignment is sequential so that one interrupt vector is for external interrupt sources 0 through N - 1, the next is for N through N + M - 1, and so forth.

The parameter IRQ_VEC defines the number of interrupt vectors (the ipi_int_pins output size).

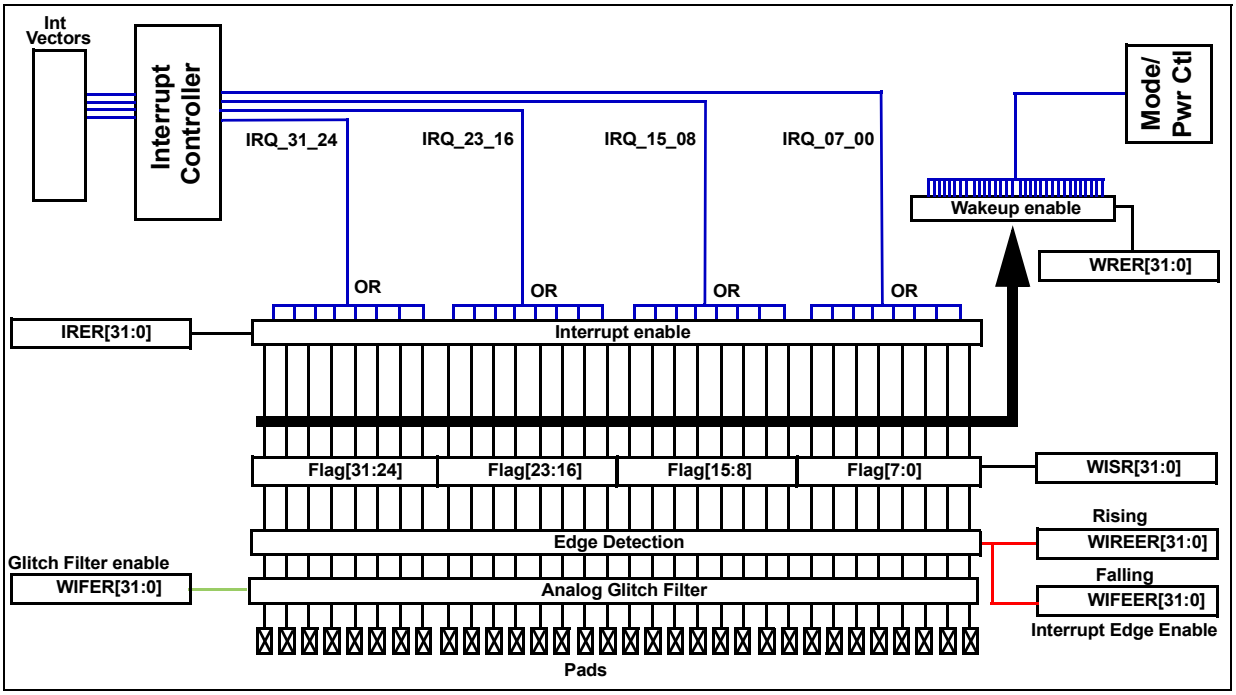
The parameter EIRQ_PER_VEC defines the number of external interrupt sources assigned to each interrupt vector. It is split up into four 6-bit fields, one for each possible interrupt vector, with the lowest field corresponding to ipi_int_pins[0], the second field to ipi_int_pins[1], and so forth.

For example, for EIRQS = 17, IRQ_VEC = 3 and EIRQ_PER_VEC = {6'd0, 6'd5, 6'd3, 6'd9}, ipp_ind_int[8:0] are assigned to ipi_int_pins[0], ipp_ind_int[11:9] are assigned to ipi_int_pins[1], and ipp_ind_int[16:12] are assigned to ipi_int_pins[2]. The signals ipp_ind_int[31:17] and ipi_int_pins[3] do not exist.

Note: If EIRQS = 0, the unused ipp_ind_int[0] input has no effect on the sys_wakeup output.

Refer to *Figure 2519* for an overview of the external interrupt implementation for the example of four interrupt vectors with eight external interrupt sources each.

Figure 2519. External interrupt pad diagram



All of the external interrupt pads within a single group have equal priority. It is the responsibility of the user software to search through the group of sources in the most appropriate way for their application.

The priority of the vectors used by the external interrupt pads is fixed based on the platform and the interrupt controller and its priority levels, but the allocation of pads to each group of interrupts can be independently configured by the SoC.

An SoC specific number of external interrupt lines can have a digital glitch filters applied to them. The supported range is 0 to 32.

Note: Glitch filter control and pad configuration should be done while the external interrupt line is disabled in order to avoid erroneous triggering by glitches caused by the configuration process itself.

96.4.3.1 External interrupt management

Each interrupt can be enabled or disabled independently. This can be performed using a single rolled up register (*Figure 2513*). A pad defined as an external interrupt can be configured by the user to recognize interrupts with an active rising edge, an active falling edge or both edges being active.

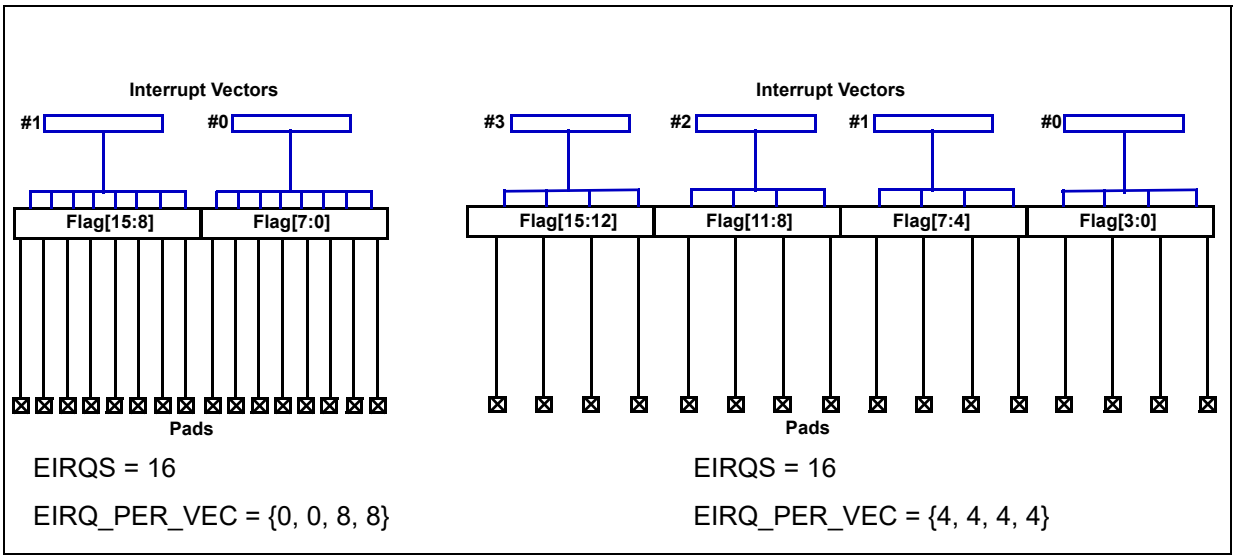
Note: Writing a '0' to both IREE[x] and IFEE[x] disables the external interrupt functionality for that pad completely (no system wakeup or interrupt will be generated on any activity on that pad)!

The active IRQ edge is controlled by the users through the configuration of the registers WIREER and WIFEER.

Each external interrupt supports an individual flag which is held in the flag register (WISR). This register is a clear-by-write-1 register type, preventing inadvertent overwriting of other flags in the same register.

For devices with less than the maximum number of interrupt sources, it is possible for the SoC to implement either all four interrupt vectors with fewer sources or to reduce the number of vectors as the number of sources are reduced. *Figure 2520* shows two examples of how this could be supported.

Figure 2520. Example of different interrupt to vector mapping at SOC level



96.4.4 On-Chip wakeups

The Wakeup unit supports up to 32 on-chip wakeup sources. It combines the on-chip wakeups with the external ones to generate a single wakeup to the system.

Note: If WUPS = 0, the unused wakeup_in input has no effect on the sys_wakeup output.

96.4.4.1 On-Chip wakeup management

In order to allow software to determine the wakeup source at one location, on-chip wakeups are reported along with external wakeups in the WISR register (see *Figure 2512* for details). Enabling and clearing of these wakeups are done via the on-chip wakeup source's own registers.

97 Emulation and Debug Device Introduction

The emulation and debug device is a multi-die MCU that adds additional functionality specifically for ease in calibration and debug. The Emulation and debug device (ED) consists of the production die (PD) and a buddy die (BD).

97.1 Functionality

The ED is a PD packaged with a BD to provide additional capability during development. The ED is used for development, debug, and calibration of automotive powertrain controller applications for 4- to 8-cylinder engines. To enable these operations, the buddy device supports the following functions:

- Debug and calibration tool interface: it allows tool read/write access to buddy device resources even if main MCU supplies are removed as long as special ED debug supplies are maintained. The BD duplicates the DCI, JTAGM and LFAST functionality that is also available in the PD.
- High speed tool trace port (based on the IEEE-ISTO 5001-2012 Nexus Aurora standard)
- 2 MB on-chip SRAM usable as calibration overlay or trace RAM. Entire SRAM contents are maintained as long as ED debug supply is maintained.

97.2 Feature summary

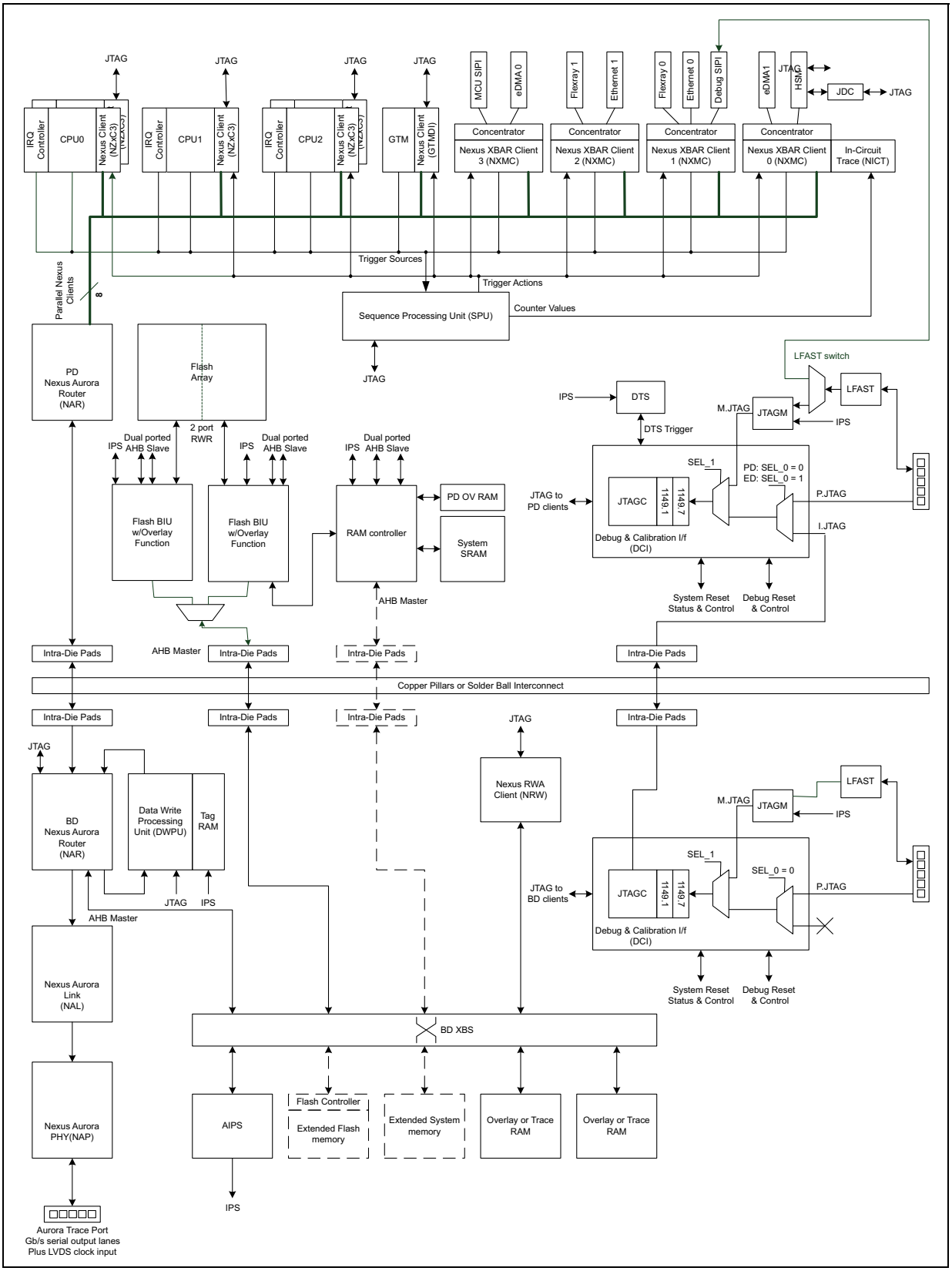
On-chip modules available on the BD include the following features:

- 2 MB on-chip SRAM
  - Access timing matches internal PD flash accesses when used in conjunction with PD calibration remap logic
  - Includes ECC
  - Implemented as 4 × 512 KB partitions where each partition can be accessed by different masters without causing blockages to access of any other partition
- XBAR switch architecture
- XBAR slave interface
  - 64-bit data width with ECC checkbits
  - Supports “Early Write” AHB protocol (AHB-EW) for one clock trace data writes
  - Allows a PD with AHB master to support read and write to buddy device memory-mapped resources. Memory-mapped resources include buddy device SRAM and PBRIDGE resources
- Nexus Aurora auxiliary trace port
  - 1.25 Gbit/s maximum data rate per lane
  - Four lanes supported in ED package
  - Port configurable to operate in 2- or 4-lane width
  - One lane LVDS reference clock in
  - Supports Nexus development interface (NDI) per IEEE-ISTO 5001-2003 standard, with some support for 2012 standard
- Data Write Processing Unit (DWPU)
  - Filters data trace stream to retain only writes to user selected locations
  - Detects user configured out of range condition on writes to selected locations
- Debug and Calibration Interface: supports both JTAG (IEEE 1149.1 and IEEE 1149.7) mode and LFAST mode
- Independent high and low voltage supplies allow tool control of BD debug and calibration resources even when the main MCU supplies are off
- Device and board test support per Joint Test Action Group (JTAG) (IEEE 1149.1)

97.3 Block diagram

This figure shows the top-level block diagram of the emulation and debug device.

Figure 2521. Emulation and debug device architecture

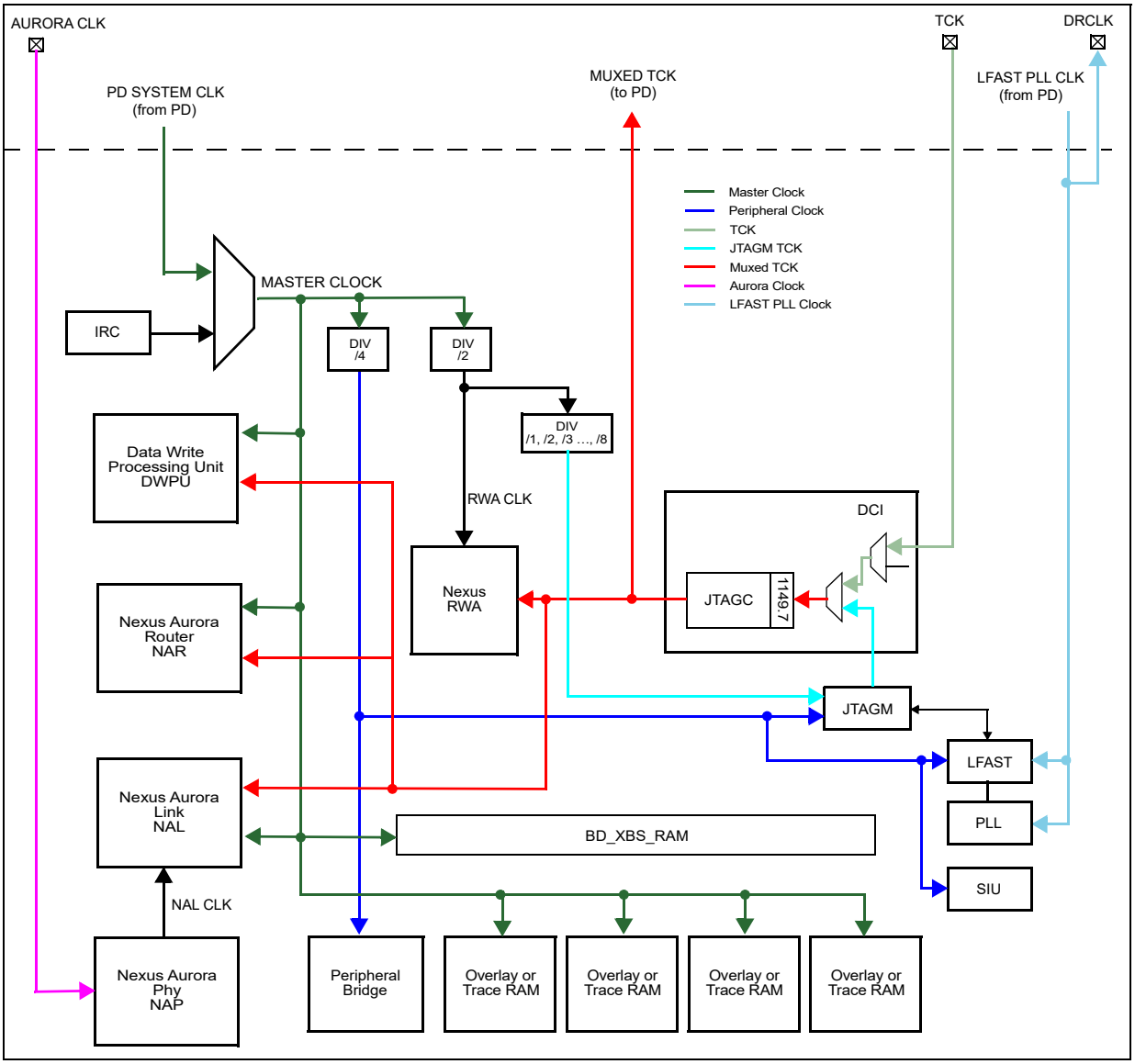


97.4 Clocking

Figure 2522 shows the clocking structure of the BD. There is one internal clock source and four external clock sources:

- Internal RC oscillator (IRC) clock
- System clock coming from the PD (identified as BD_CLK clock in the clock generation figure of the Clocking chapter)
- LFAST PLL reference clock coming from the PD (BD LFAST PLL clock in the clock generation figure of the Clocking chapter)
- JTAG clock coming from the TCK pin in the ED
- Nexus Aurora PHY clock coming from differential LVDS pins in the ED

Figure 2522. BD clock structure



The IRC is the source of the BD master clock when the PD is unpowered, or powered but still in reset state. It has a nominal frequency of 64 MHz, does not have any user accessible register and is always enabled. It does not have any trimming capability, therefore the expected precision is on the order of ±30%.

The PD system clock is the source of the BD master clock when the PD is powered and out of reset state. When the PD PLL is engaged and locked this clock has a nominal frequency of 200 MHz, but while the PLL is not engaged the frequency is that of the PD internal RC oscillator.

The master clock directly feeds the XBAR, SRAMs and the Aurora sub-system. It is also divided down to generate the following internal clocks:

- Peripheral clock (divided by 4): Provides the clock to memory mapped registers and other low-speed logic.
- Nexus RWA clock (divided by 2): Provides the clock to the Nexus Read Write Access module.
- JTAGM TCK (divided by 1, 2, …, 8): Provides a high speed TCK clock to the JTAGM module.

The LFAST PLL clock from the PD (also referred to as lfast_sysclk) feeds the LFAST PLL, but it can also directly clock the LFAST in the low data rate mode. This clock is also made available as an output clock to the DRCLK package pin.

The internal TCK clock (referred to as muxed clock in *Figure 2522*) is used in the Nexus RWA, NAR, NAL, and it also goes to the PD to be used as the JTAG clock. (In the ED, the JTAG package pins are bonded to the BD JTAG interface; therefore, the PD receives the JTAG signals from the BD.) This clock can come from two sources (see the Buddy Device–Debug and Calibration Interface (DCI) chapter):

- TCK package pin, when using IEEE 1149.1 or IEEE 1149.7 access
- Divided down version of the master clock to be used in LFAST/JTAGM debug accesses

Note: The division factor for the LFAST/JTAGM clock is defined by the TCKSEL field of the JTAGM_MCR (refer to the JTAGM chapter for more details). However the division factor described there applies to an already divided-by-2 clock, as shown in Figure 2522.

When the PD is unpowered, access through LFAST/JTAGM is not possible. In this case, JTAG accesses can be done only by IEEE 1149.1 or IEEE 1149.7 modes accessing through the Nexus RWA module. In this mode, the TCK clock comes from the package pin and the RWA clock is a divided-by-2 version of the IRC; therefore, these clocks are asynchronous to each other. When clocks at the input of the RWA are asynchronous, there is a restriction that the RWA clock must be at least double the frequency of the JTAG clock. This means that the master clock frequency must be at least four times the JTAG clock frequency.

In the LFAST/JTAGM mode of operation, both RWA clock and internal TCK are synchronous to each other, therefore they can both be the same frequency. This means that TCK frequency can be half of the master clock frequency. *Table 2404* summarizes the frequency restrictions on the Nexus RWA accesses.

Table 2404. JTAG operating frequencies

PD configuration	BD configuration	PD system clock frequency [MHz]	BD master clock frequency [MHz]	Max RWA JTAG frequency [MHz]	Notes
Unpowered or in reset state	1149.1 or 1149.7	—	44.8	11.2	Assuming 64 MHz BD IRC with 30% tolerance
Powered, out of reset, PLL not engaged, system clock at IRC frequency	1149.1 or 1149.7	15.52	15.52	3.88	Assuming 16 MHz PD IRC with 3% tolerance
Powered, out of reset, PLL engaged	1149.1 or 1149.7	200	200	27.7	JTAG speed is limited by TCK negedge to TDO delay (14 ns) plus external setup to TCK posedge (4 ns)
Powered, out of reset, PLL engaged	LFAST/JTAGM	200	200	100	—

97.5 Reset logic

There are five sources of reset that are combined to generate the reset of system portions as described by *Table 2405*:

1. BD power-on reset: asserts immediately upon assertion of BD POR, and remains asserted for 10 clock cycles after negation of POR.
2. PD reset: asserts two clock rising edges after assertion of reset signal coming from PD and remains asserted while the PD reset signal is asserted plus two clock rising edges after the PD reset signal is de-asserted (this reset source is reflected in the PD_RST bit of the BD_SIU_RSR register).
3. DCI FBD reset: asserts immediately when the FBD_RESET bit of the DCI Control Register is set, and remains asserted for one clock cycle after FBD_RESET is negated.
4. JTAG controller reset: asserts immediately when a reset request signal is received from the JTAG Controller by the execution of EXTEST, CLAMP, HIZ boundary instructions, and remains asserted for one clock cycle after the reset request is negated.
5. Nexus reset: asserts by DCI logic based on JCOMP state and JTAG source selected. DCI FBD also generates a Nexus reset.

In *Table 2405*, an ‘X’ indicates that the reset source generates a reset signal to the specified block. JTAG mode only indicates that those blocks are reset by the corresponding reset sources only in the case where the debug interface is running through the JTAG external pins. When the debug interface is running through the LFAST interface or by software through JTAGM, those blocks are not reset by the mentioned reset sources.

Table 2405. Reset sources and functional blocks

Blocks	BD POR	PD reset	DCI FBD reset	JTAGC reset	Nexus reset
DCI	X	—	—	—	—
Z0 Nexus RWA	X	—	X	—	X
BD XBAR	X	—	X	X	—
Overlay/Trace RAM	X	—	X	X	—
JTAGM	X	—	JTAG mode only	JTAG mode only	—
LFAST	X	—	JTAG mode only	JTAG mode only	—
NAR/DWPU	X	—	X	X	X
NAL	X	—	X	X	X
NAP	X	—	X	X	X
BD SIU	X	—	X	X	—
BD SIU GPIO registers	X	X	—	X	—
BD INIT flag	X	—	X	X	—
PD/BD interface logic	X	—	X	X	—

Note: The PD reset is not affected by the BD reset. The BD does not interfere with startup or shut down timing of the PD.

97.6 BD memory map

The BD memory map is contained within a 16 MB range. This entire range is accessed through the PD Flash BIU. The BD memory map is divided into two regions: overlay RAM and device registers.

The device register region allows the CPUs on the PD to access configuration registers of the peripherals in the BD, such as LFAST, JTAGM, XBAR configuration. The device register region are divided into 16 KB regions for assignment to each peripheral block.

The memory map for the BD is shown below as an absolute address relative to the PD address map. Within the BD, only a 24-bit address bus is required. The upper 8 bits of the complete 32-bit address are decoded and removed by the PD PFLASH controller.

Table 2406. BD memory map

Start address	End address	Allocated size (KB)	Description
BD overlay RAM			
0x0C00_0000	0x0C1F_FFFF	2048	Extended overlay RAM
0x0C20_0000	0x0C3F_FFFF	2048	Reserved BD overlay RAM
0x0C40_0000	0x0C5F_FFFF	2048	Reserved BD overlay RAM
0x0C60_0000	0x0C7F_FFFF	2048	Reserved BD overlay RAM

Table 2406. BD memory map (continued)

Start address	End address	Allocated size (KB)	Description
BD registers			
0x0C80_0000	0x0C80_3FFF	16	BD SIUL2
0x0C80_4000	0x0C80_7FFF	16	LFAST
0x0C80_8000	0x0C80_BFFF	16	JTAGM
0x0C80_C000	0x0C80_FFFF	16	Reserved
0x0C81_0000	0x0C81_3FFF	16	BD XBAR
0x0C81_4000	0x0C81_7FFF	16	Reserved
0x0C81_8000	0x0C81_BFFF	16	Reserved
0x0C81_C000	0x0C81_FFFF	16	Reserved
0x0C82_0000	0x0CFF_FFFF	8064	Reserved

98 Buddy Device—Data Write Processing Unit (DWPU)

98.1 Introduction

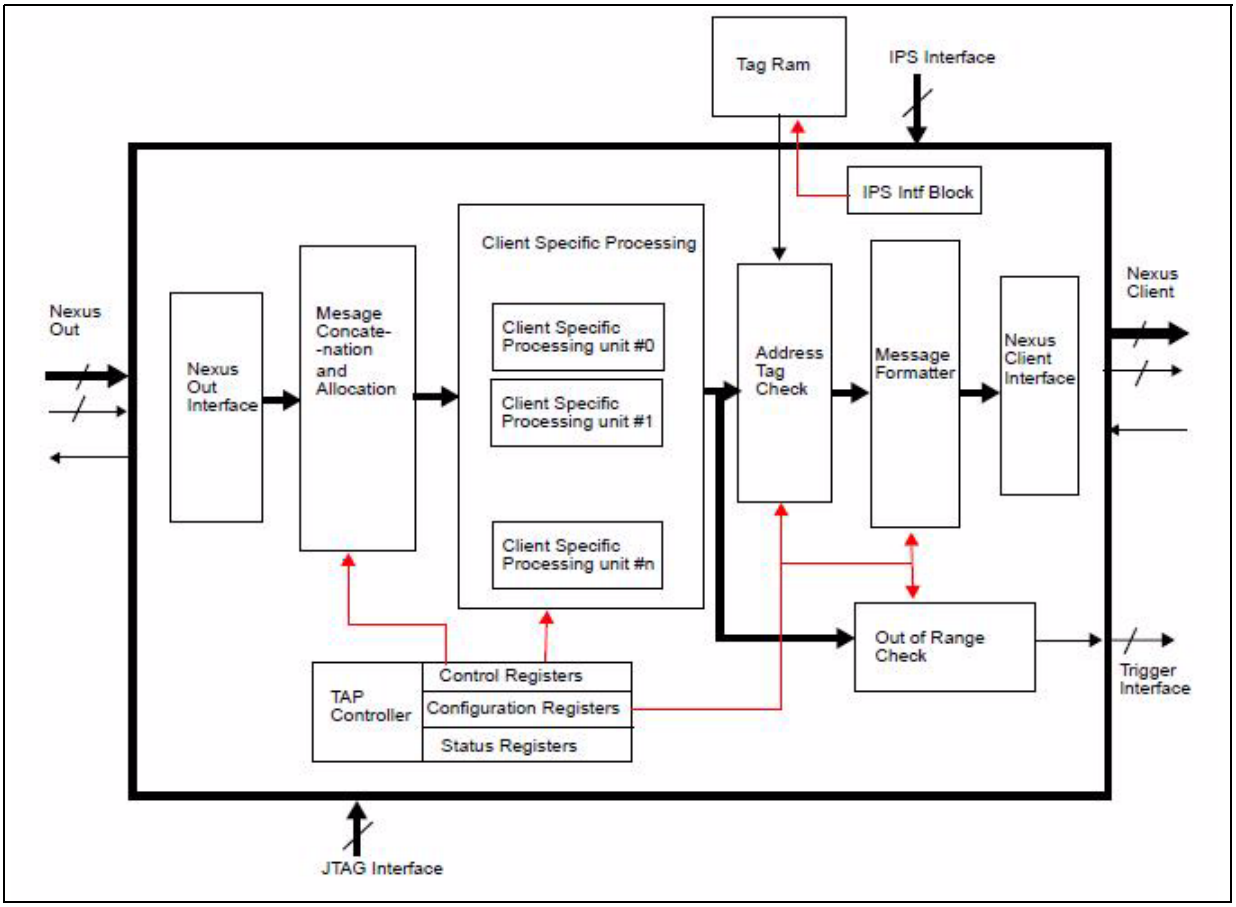
The Data Write Processing Unit (DWPU) supports two specific debug and calibration functions:

- Data trace filtering
- Out of range parameter identification

DWPU takes the Nexus standard messages and processes the data write trace messages to filter out messages based on the tagged addressing schema contained in the Tag RAM. DWPU also generates triggers indicating out of range data values for a set of identified addresses and data ranges.

This figure shows the block diagram of the DWPU identifying various functional blocks and the data flows.

Figure 2523. DWPU block diagram



98.2 Features

DWPU supports the following features:

- Out of range data value detection and data write trace filtering simultaneously from up to five Nexus clients
- Filtering operation that can be individually enabled or disabled for each 64-bit word in system RAM or data memory
- High bandwidth throughput
- Filtered trace stream is returned to the BD NAR to allow routing to the trace port or overlay RAM.
- Detection of out of range data values writes within data write trace stream

98.3 Functional description

DWPU includes the following functional blocks:

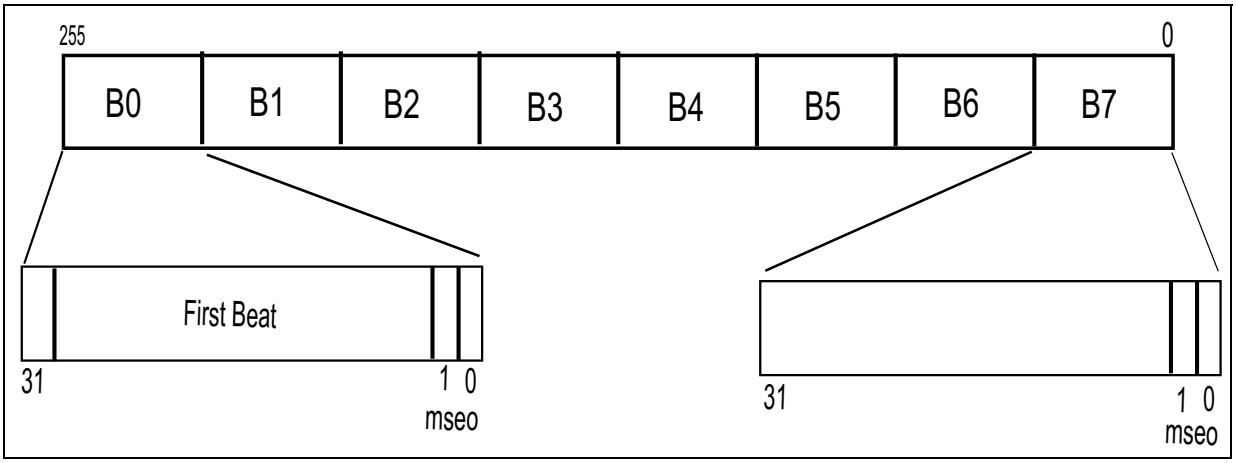
- Nexus Out Interface (NOI)
- Message Concatenation and Allocation (MCA)
- Client Specific Processing (CSP)
- Address Tag Check (ATC)
- Out of Range Check (ORC)
- Message Formatter (MF)
- Nexus Client Interface (NCI)

98.3.1 Nexus Out Interface (NOI)

The Nexus Out Interface (NOI) enables DWPU to receive Nexus message for data write filtering from NAR. NOI maintains a minimum queue of two buffers to accommodate incoming messages. The NAR only initiates a transfer to the DWPU when it has all the beats of a message. The incoming messages are buffered and then forwarded to the Message Concatenation and Allocation (MCA) for message concatenation and allocation. NOI ensures the temporal order of the incoming messages while forwarding to MCA. NOI also acknowledges the processing of data write messages by MCA.

The format for incoming message with NEXUS_WORD = 32 and NUM_BEAT_IN = 8 is shown below.

Figure 2524. Incoming message format



98.3.2 Message Concatenation and Allocation (MCA)

Message Concatenation and Allocation (MCA) starts concatenating message beats once the NOI has a valid input message to form a complete message. Concatenated messages with selected source IDs contained in Nexus messages are passed over to appropriate Client Specific Processing Unit (CSPU) for further processing. The following messages are dropped by MCA:

- Messages that do not have a selected source ID
- IDLE messages
- Messages with number of beats larger than defined beats for data-write messages

MCA has the capability of providing one fully concatenated message to a CSPU each clock.

98.3.3 Client Specific Processing (CSP)

The Client Specific Processing (CSP) block receives the messages from MCA. The received message are processed after extracting all the relevant fields. Different types of message from various clients are received and processed differently based on their type by a specialized unit called the Client Specific Processing Unit (CSPU).

The CSPU receives concatenated data write messages (including synchronization messages) from MCA and provides the following for the Out of Range Check (ORC) and Address Tag Check (ATC) logic to perform filter checks:

- Reconstructed address
- Data value
- Source ID
- Master ID
- Timestamp

CSPU also confirms that each message is a data write message. All other message types are dropped.

There are two CSPU types, one associated with Core Nexus clients and another associated with NXMC clients. Each CSPU simultaneously supports data write messages with up to four different source IDs.Data write with synchronization message is always required to start an address thread for a particular source ID, then each subsequent data write message

associated with that source ID contains an encoded address. *Table 2407* shows the CSPU type encoding to be used while mapping the source IDs to a CSPU type using SRC_MAP_CTLR.

Table 2407. CSPU encoding

CSTU type	Encoding
Standard	01
NXMC	10

98.3.3.1 Input message formats

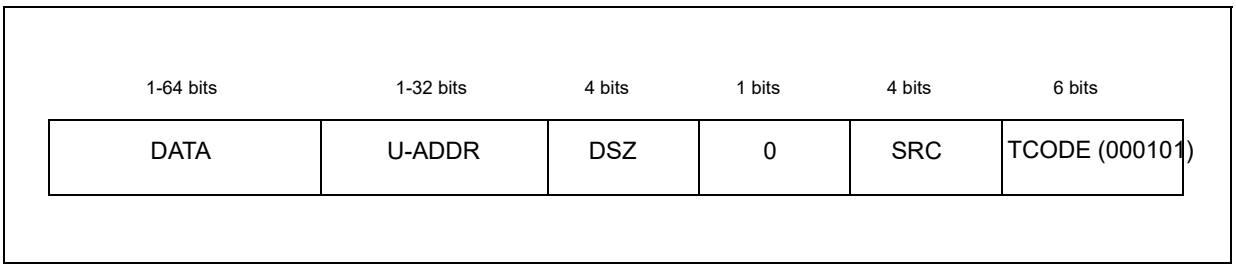
DWPU supports various message formats from different source clients. These formats are further processed by the CSPU. The following message formats are supported by the CSPUs, as also described in details the sub-sections:

- Standard data write message format (SDWM)—Includes sync and non-sync type. This is used by most of the clients, such as e200 peripheral and e200 main cores.
- AHBMM (modified) data write message format (MDWM)—Includes both sync and non-sync type of messages. Used by NXMC clients only.

98.3.3.1.1 Standard data trace write message format

The standard data trace write messages are of two types, sync and non-sync. The non-sync type contains the data write value and the address of the write access, relative to the previous data trace message. The address and DATA field is considered variable. Data write message information is messaged out in the format shown below. The SRC field values are described in the device-specific chapter that describes how the clients are configured.

Figure 2525. Non-sync data trace write message format



The TSTAMP field not shown is an optional fixed field of 26 bits, is inserted optionally by the core clients based on controls received from the BD NAR.

The second type is the DWM synchronization message, whose format is exactly same as above but with full-address (without leading zeros) instead of the relative address. This is to ensure that development tools fully synchronize with data trace regularly. Synchronization messages provide a reference address for subsequent DWMs, in which only the unique portion of the data trace address is transmitted. The format for data trace write with synchronization messages is shown below.

Figure 2526. Sync data trace write message format

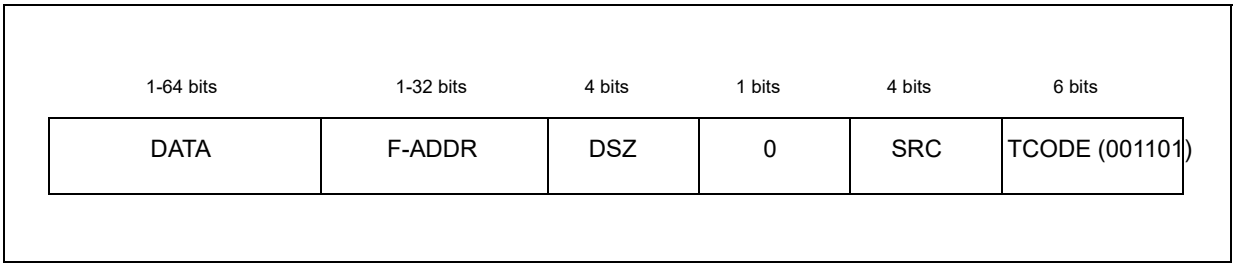


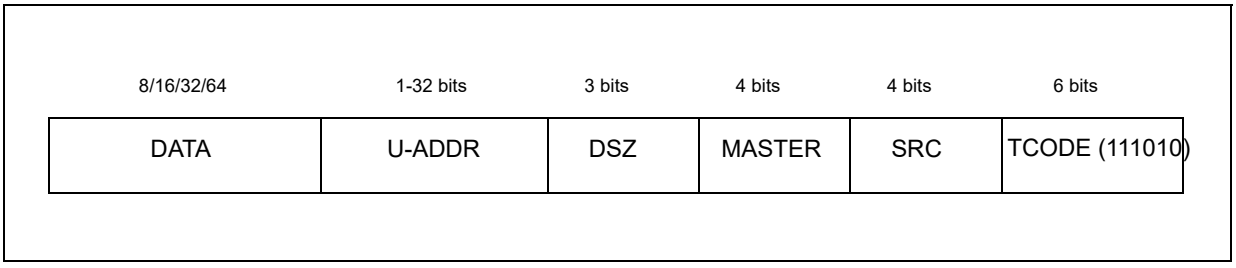
Table 2408 shows the data size (DSZ) encoding for standard data trace message.

Table 2408. Standard DSZ encoding

DSZ	Data size
0000	0-byte
0001	1-byte
0010	2-byte
0011	3-byte
0100	4-byte
0101	5-byte
0110	6-byte
0111	7-byte
1000	8-byte
1001–1111	Reserved

The NXMC (modified) data trace write sync messages contain the data write value and the address of the write access, relative to the previous data trace message. The DATA field is considered fixed based on the DSZ encoding to either 8, 16, 32 or 64. The TSTAMP field is 26 bits. Data write message information is messaged out in the format shown below. The SRC field values are described in the device-specific chapter that describes how the clients are configured.

Figure 2527. Non-sync M-Data data trace write message format



The TSTAMP field not shown over here is an optional fixed field of 26 bits, is inserted optionally by the AHBMM clients based on controls received from NAR.

The second type is the MDWM synchronization message, whose format is exactly same as above but with full-address (without leading zeros) instead of the relative address. This is to

ensure that development tools fully synchronize with data trace regularly. Synchronization messages provide a reference address for subsequent MDWMs, in which only the unique portion of the data trace address is transmitted. The format for data trace write with synchronization messages is shown below.

Figure 2528. Sync M-Data data trace write message format

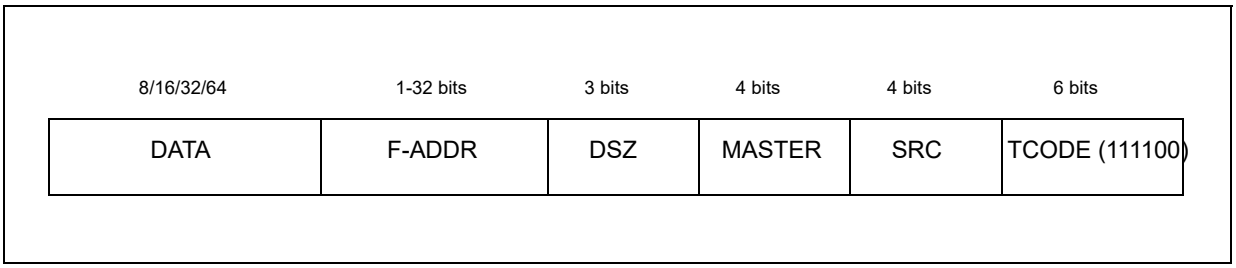


Table 2409 shows the data size (DSZ) encoding for NXMC data trace message.

Table 2409. NXMC DSZ encoding

DSZ	Data size
001	1-byte
010	2-byte
011	3-byte
100	4-byte
000,101–111	Reserved

98.3.4 Address Tag Check (ATC)

DWPU allows filtering operation on data writes to any address within system RAM and tightly coupled data memory. Each 64-bit word in the supported system RAM or data memory address space corresponds to a single bit in Tag RAM; therefore, the required Tag RAM size is $1/64^{th}$ of the address space to be filtered. Data write messages within the supported address range are dropped by ATC if the corresponding bit in Tag RAM is not set. The data write messages with the corresponding Tag bit set are passed over to Message Formatter for further processing.

Writes to addresses outside of the supported address range can be configured to be either removed from the trace stream or retained. This option is controlled by a single user accessible control bit. Data reads and other Nexus message formats submitted in the DWPU trace stream are always dropped by DWPU. Tag RAM is accessible through software (IPS) as well as JTAG interface via PD addressing. All the IPS access is possible in supervisor mode only. The access to Tag RAM is carried out indirectly through three registers: TRC_CTRL, TRC_ADR and TRC_DATA registers. Users should write the TRC_CTRL register followed by the TRC_ADR register before accessing the TRC_DATA register. Any write access to the TRC_ADR register results in a Tag RAM access request. Users should ensure that the Tag RAM request is completed by providing the required TRC_DATA access based on the length of the transfer. In case of block access, the TRC_DATA register is accessed subsequently and users should ensure that the block access length does not cross the region boundary. Deviation in the sequence of register access for Tag RAM access results in erroneous and incomplete transfers.

In PD addressing, the configured PD address in the TRC_ADR register is translated to a Tag RAM address using mapping derived from the FAR_SA_CFGn and FAR_RS_CFGn register. The PD address configured in the FAR_SA_CFG0 register is always translated to the zeroth address of Tag RAM.

The equation to convert system/PD address to Tag RAM address where the System/PD address falls within region n is given below:

Tag RAM address = System/PD address[31:8] - Region n base address[31:8] + Total sum of previous region (n-1) size[31:8]

The Tag RAM address for System/PD address that falls on region 2 is given by:

System/PD address[31:8] - Region 2 base address[31:8] + (region 0 size[31:8] + region 1 size[31:8])

Figure 2529 shows an example of the Tag RAM organization for 8 Kbyte Tag RAM with PD base address configured.

Figure 2529. Tag RAM organization example

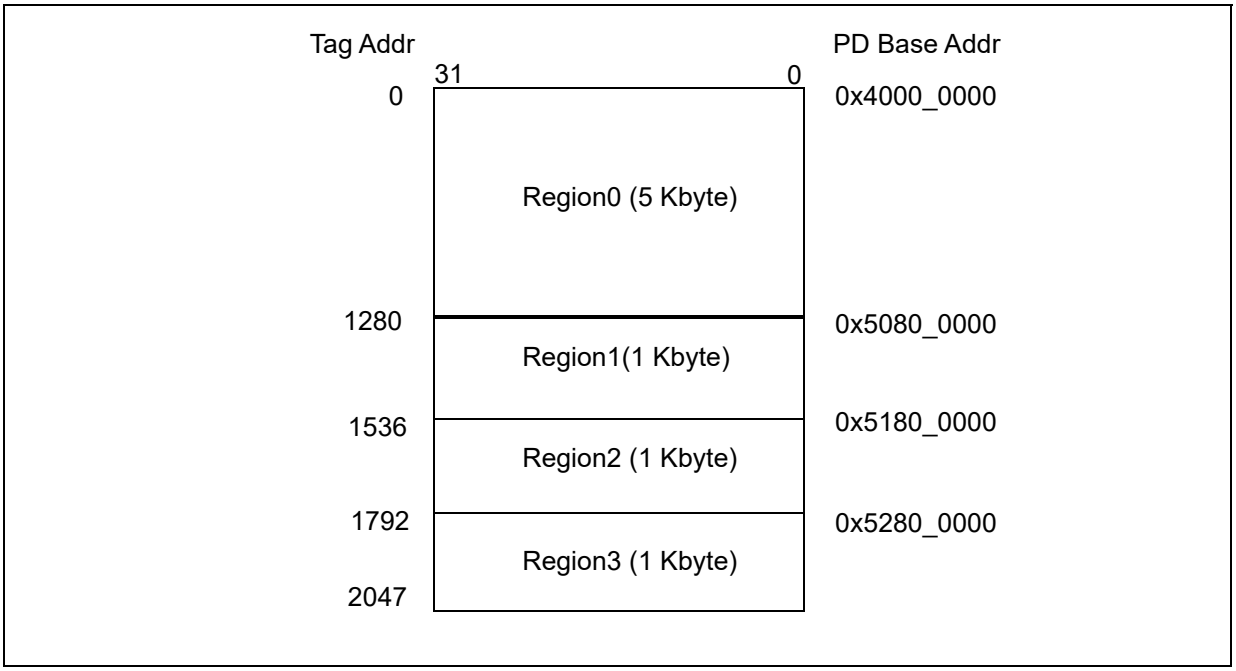


Table 2410 shows an example of address translation from PD addresses to Tag RAM address.

Table 2410. Address translation example

PD address		Tag RAM address
Start	End	
0x4000_0000	0x4000_00FF	0x0
0x4000_0100	0x4000_01FF	0x1
0x4004_FF00	0x4004_FFFF	0x4FF
0x5080_0000	0x5080_00FF	0x500

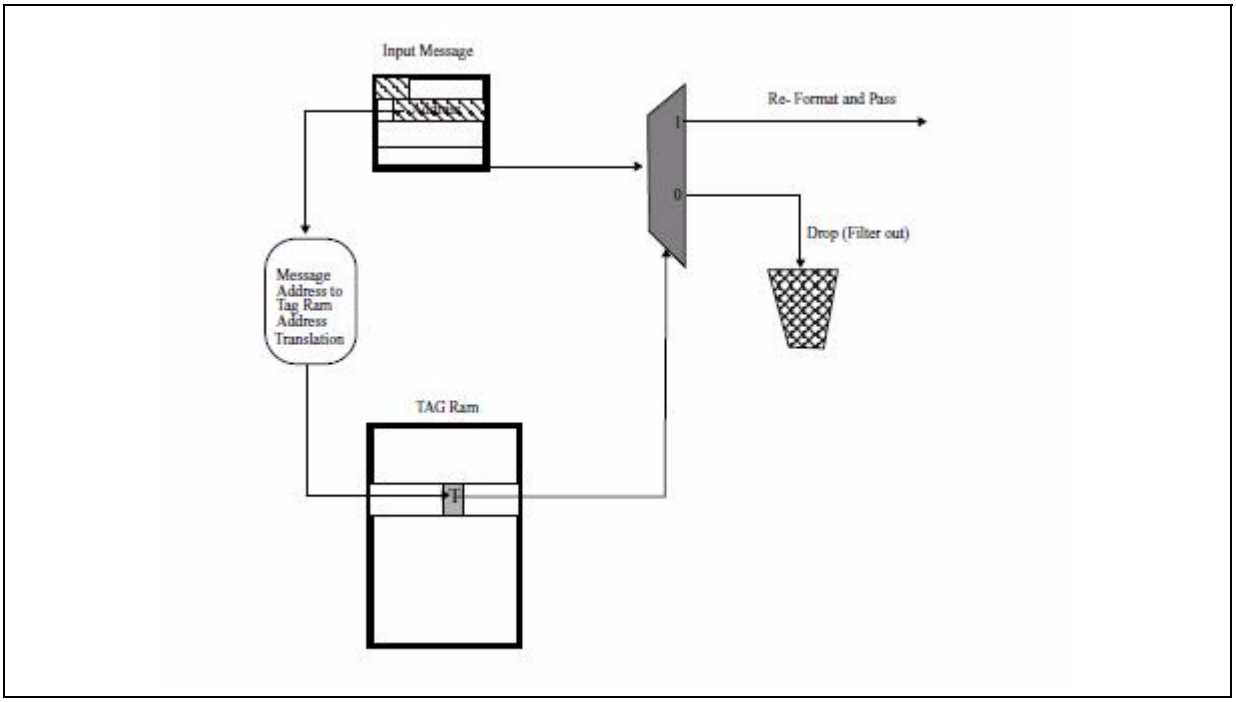
Table 2410. Address translation example (continued)

PD address		Tag RAM address
Start	End	
0x5180_0000	0x5180_00FF	0x600
0x5280_0000	0x5280_00FF	0x700
0x5280_FF00	0x5280_FFFF	0x7FF

During the data write filtering operation, the system RAM or data memory address is extracted from the input message. It is then compared to ascertain the address space (whether it is within system RAM space or data memory or outside the supported address range). Finally it is translated to Tag RAM address and initiate a read to Tag RAM. From the Tag RAM 32-bit read accessed, the system/PD address bit [7:3] determine the Tag bit position that is associated with the decoded absolute system/PD address. If the corresponding bit is set, then the input message is passed for message formatting else it is dropped by ATC.

This figure shows the data flow for address Tag filtering.

Figure 2530. Address tag filtering

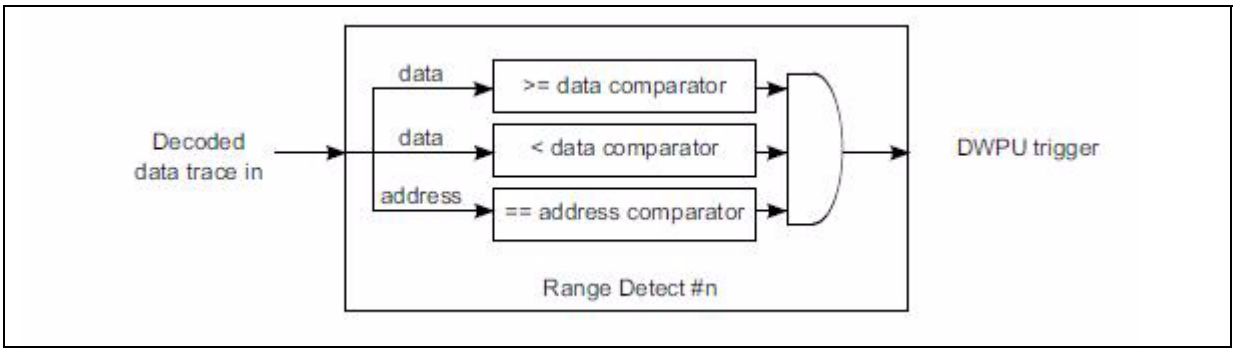


98.3.5 Out of Range Check (ORC)

In addition to the trace filtering function, the DWPU also supports detection of outside/inside range writes to RAM locations. Both the data range values and address of RAM locations to be checked are user specified. It also has configurable support to detect target inside or outside data range values specified.

A total of four sets of address and data range comparators are supported, where each set of comparators include an address match comparator, a greater than or equal data value comparator and a less than or equal data value comparator. This figure shows the block diagram for range detect comparators.

Figure 2531. Range detect comparators



ORC supports unsigned, signed and floating point data ranges. Signed numbers are represented in 2's complement. 32-bit IEEE-754 single-precision floating-point format is used for floating point representation.

ORC has a configurable option to support for 8/16/32 bit unsigned/signed range detection and 32-bit floating point range comparison. Address comparison is carried out with full the device address. An output signal from DWPU is triggered in response to a detected out of range condition along with a status bit being set in an ORC status register indicating which range comparator has caused the trigger.

98.3.5.1 Comparator capabilities

The primary usage for out of range detect function is to detect out of range conditions on unsigned integer and pointer variables, although it also supports signed and floating point data types comparison. It also has a configurable option to support for 8/16/32 bit unsigned range detection. The unsigned data is directly compared with the stored data ranges.

To support use with signed variables, each data comparator supports 8/16/32 bit signed operation. All the signed numbers are represented in two’s complement. Data comparison happens only when the sign bit matches.

To detect range of floating point variables, each data comparator supports only 32-bit comparison mode. All the floating point numbers are represented in 32-bit IEEE-754 single precision with biased exponent. The floating point data is directly compared with the stored floating point ranges which are also represented in 32-bit IEEE-754 single precision with biased exponent.

98.3.6 Message Formatter (MF)

Message Formatter (MF) module receives message information from ATC and reformats the data write messages to comply with output Nexus message format. A user defined message format is used for sending out the filtered messages as shown in *Figure 2532* below.

Figure 2532. DWPU message format

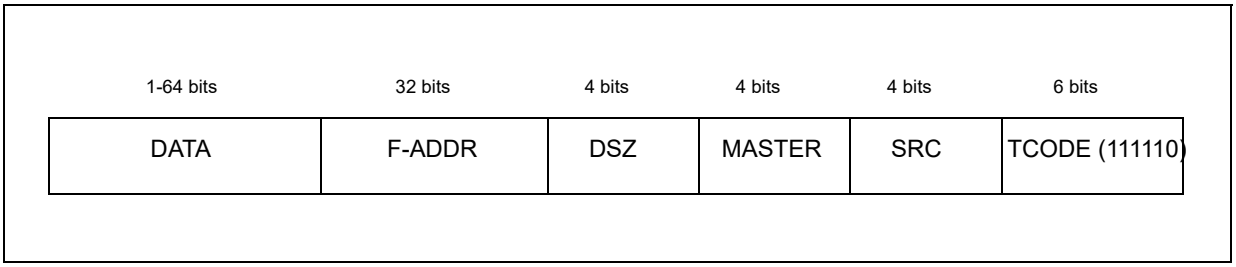


Table 2411. Message field description

Message name	Min packet size (bits)	Max packet size (bits)	Packet name	Packet type	Packet description
DWPU message	6	6	TCODE	fixed	TCODE number = 62
	4	4	SRC	fixed	Source client identifier
	4	4	MASTER	fixed	Indicates which master (ID) initiated the data access. For core clients 1111 is inserted. For the NXMC clients, same value is inserted as received.
	4	4	DSZ	fixed	Data size
	32	32	F-ADDR	fixed	Full data write address
	1	64	DATA	fixed	Data write value (size fixed based on DSZ field)
	26	26	TSTAMP	fixed	Timestamp (Optional and omitted if there is no valid timestamp)

98.3.7 Nexus Client Interface (NCI)

The Nexus Client Interface (NCI) sends out the formatted filtered data write messages to BD NAR. It also supports the buffer reallocation to increase the Q size.

98.4 Non-memory mapped registers

This section describes the DWPU Nexus register definition. Nexus registers are accessed using the JTAG port in compliance with IEEE 1149.1. In addition, some register are accessible through the IPS interface.

The table below shows the JTAG mapped DWPU registers.

Table 2412. JTAG DWPU registers

Nexus register index	Register	Access	Read address	Write address	Reset value
0x02	Global Control (GLB_CTRL)	R/W	0x04	0x05	0x0000_0002
0x03	Source Mapping Control 0 (SRC_MAP_CTLR0)	R/W	0x06	0x07	0x0000_0000
0x04	Source Mapping Control 1 (SRC_MAP_CTLR1)	R/W	0x08	0x09	0x0000_0000
0x05	Source Mapping Control 2 (SRC_MAP_CTLR2)	R/W	0x0A	0x0B	0x0000_0000

Table 2412. JTAG DWPU registers (continued)

Nexus register index	Register	Access	Read address	Write address	Reset value
0x0A	Filter Address Range Start Address 0 (FAR_SA_CFG0)	R/W	0x14	0x15	0x0000_0000
0x0B	Filter Address Range Region Size 0 (FAR_RS_CFG0)	R/W	Ox16	0x17	0x0000_0000
0x0C	Filter Address Range Start Address 1 (FAR_SA_CFG1)	R/W	0x18	0x19	0x0000_0000
0x0D	Filter Address Range Region Size 1 (FAR_RS_CFG1)	R/W	0x1A	0x1B	0x0000_0000
0x0E	Filter Address Range Start Address 2 (FAR_SA_CFG2)	R/W	0x1C	0x1D	0x0000_0000
0x0F	Filter Address Range Region Size 2 (FAR_RS_CFG2)	R/W	0x1E	0x1F	0x0000_0000
0x10	Filter Address Range Start Address 3 (FAR_SA_CFG3)	R/W	0x20	0x21	0x0000_0000
0x11	Filter Address Range Region Size 3 (FAR_RS_CFG3)	R/W	0x22	0x23	0x0000_0000
0x1A	Tag RAM Control (TRC_CTRL)	R/W	0x34	0x35	0x0000_0000
0x1B	Tag RAM Configuration Address (TRC_ADR)	R/W	0x36	0x37	0x0000_0000
0x1C	Tag RAM Configuration Data (TRC_DATA)	R/W	0x38	0x39	0x0000_0000
0x20	Out of Range Check Control 1 (ORC_CTRL1)	R/W	0x40	0x41	0x0000_0000
0x21	Out of Range Check Control 2 (ORC_CTRL2)	R/W	0x42	0x43	0x0000_0000
0x22	Global Status (GLB_STAT)	R	0x44	0x45	0x0004_0000
0x24	Out of Range Configuration Address 0 (ORC_CFG0_ADR)	R/W	0x48	0x49	0x0000_0000
0x25	Out of Range Configuration Data High 0 (ORC_CFG0_DH)	R/W	0x4A	0x4B	0x0000_0000
0x26	Out of Range Configuration Data Low 0 (ORC_CFG0_DL)	R/W	0x4C	0x4D	0x0000_0000
0x27	Out of Range Configuration Data Mask 0 (ORC_CFG0_DM)	R/W	0x4E	0x4F	0x0000_0000
0x28	Out of Range Configuration Address 1 (ORC_CFG1_ADR)	R/W	0x50	0x51	0x0000_0000
0x29	Out of Range Configuration Data High 1 (ORC_CFG1_DH)	R/W	0x52	0x53	0x0000_0000
0x2A	Out of Range Configuration Data Low 1 (ORC_CFG1_DL)	R/W	0x54	0x55	0x0000_0000
0x2B	Out of Range Configuration Data Mask 1 (ORC_CFG1_DM)	R/W	0x56	0x57	0x0000_0000
0x2C	Out of Range Configuration Address 2 (ORC_CFG2_ADR)	R/W	0x58	0x59	0x0000_0000
0x2D	Out of Range Configuration Data High 2 (ORC_CFG2_DH)	R/W	0x5A	0x5B	0x0000_0000
0x2E	Out of Range Configuration Data Low 2 (ORC_CFG2_DL)	R/W	0x5C	0x5D	0x0000_0000
0x2F	Out of Range Configuration Data Mask 2 (ORC_CFG2_DM)	R/W	0x5E	0x5F	0x0000_0000

Table 2412. JTAG DWPU registers (continued)

Nexus register index	Register	Access	Read address	Write address	Reset value
0x30	Out of Range Configuration Address 3 (ORC_CFG3_ADR)	R/W	0x60	0x61	0x0000_0000
0x31	Out of Range Configuration Data High 3 (ORC_CFG3_DH)	R/W	0x62	0x63	0x0000_0000
0x32	Out of Range Configuration Data Low 3 (ORC_CFG3_DL)	R/W	0x64	0x65	0x0000_0000
0x33	Out of Range Configuration Data Mask 3 (ORC_CFG3_DM)	R/W	0x66	0x67	0x0000_0000

98.4.1 Register descriptions

98.4.1.1 Global Control (GLB_CTRL)

Index 0x02 Access: Read

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	FARE3	FARE2	FARE1	FARE0	TSC	0	0	0	0	0	ORC	SFT_RST	MEN
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0

Figure 2533. Global Control register (GLB_CTRL)

Table 2413. GLB_CTRL field descriptions

Field	Description
19 FARE3	0 Filter Address Range 3 Configuration (FAR_SA_CFG3, FAR_RS_CFG3) disabled. 1 Filter Address Range 3 Configuration (FAR_SA_CFG3, FAR_RS_CFG3) enabled.
20 FARE2	0 Filter Address Range 2 Configuration (FAR_SA_CFG2, FAR_RS_CFG2) disabled. 1 Filter Address Range 2 Configuration (FAR_SA_CFG2, FAR_RS_CFG2) enabled.
21 FARE1	0 Filter Address Range 1 Configuration (FAR_SA_CFG1, FAR_RS_CFG1) disabled. 1 Filter Address Range 1 Configuration (FAR_SA_CFG1, FAR_RS_CFG1) enabled.
22 FARE0	0 Filter Address Range 0 Configuration (FAR_SA_CFG0, FAR_RS_CFG0) disabled. 1 Filter Address Range 0 Configuration (FAR_SA_CFG0, FAR_RS_CFG0) enabled.

Table 2413. GLB_CTRL field descriptions (continued)

Field	Description
23 TSC	0 Pass incoming timestamp and include in output message format (requires more trace bandwidth or overlay RAM locations) 1 Drop timestamp from incoming messages
29 ORC	0 Data writes outside the identified RAM areas are flushed 1 Data writes outside the identified RAM areas are passed
30 SFT_RST	Active low soft reset. 0 Enabled soft reset 1 Disable soft reset
31 MEN	Module Enable. 0 DWPU disabled 1 DWPU enabled

98.4.1.2 Source Mapping Control *n* (SRC_MAP_CTLR*n*)

Index 0x03 + n (n = 0 to 2) Access: Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	EN1	0	0	0	0	CSP_TYPE1			SRC_ID1							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	EN0	0	0	0	0	CSP_TYPE0			SRC_ID0							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0

Figure 2534. Source Mapping Control *n* register (SRC_MAP_CTLR*n*)

Table 2414. SRC_MAP_CTLR*n* field descriptions

Field	Description
0 EN1	0 SRC_ID0 field is not used or mapped to a CSPU. Messages with unmapped source IDs are flushed 1 SRC_ID and CSP_TYPE0 are used for selecting and mapping data write messages
5:7 CSP_TYPE1	CSPU Type to which the SRC_ID1 is mapped 000 Unmapped (to be dropped by CSP) 001 Standard message 010 NXMC message 011 Reserved 100 Reserved 101 Reserved 110 Reserved 111 Reserved
8:15 SRC_ID1	Source ID selected

Table 2414. SRC_MAP_CTLR*n* field descriptions (continued)

Field	Description
16 EN0	0 SRC_ID0 field is not used or mapped to a CSPU. Messages with unmapped source IDs are flushed 1 SRC_ID and CSP_TYPE0 are used for selecting and mapping data write messages
21:23 CSP_TYPE0	CSPU Type to which the SRC_ID0 is mapped 000 Unmapped (to be dropped by CSP) 001 Standard message 010 NXMC message 011 Reserved 100 Reserved 101 Reserved 110 Reserved 111 Reserved
24:31 SRC_ID0	Source ID selected

98.4.1.3 Filter Address Range Start Address *n* (FAR_SA_CFG*n*)

Index 0x0A + n * 2 (n = 0 to 3) Access: Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	SA[0:15]															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	SA[16:23]								0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0

Figure 2535. Filter Address Range Start Address *n* (FAR_SA_CFG*n*)

Table 2415. FAR_SA_CFG*n* field descriptions

Field	Description
0:23 SA	Start PD address for the region, should be at 265-byte boundaries

98.4.1.4 Filter Address Range Region Size *n* (FAR_RS_CFG*n*)

Index 0x0B + n * 2 (n = 0 to 3) Access: Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	RS[12:15]			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	RS[16:23]								0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0

Figure 2536. Filter Address Range Start Address *n* (FAR_SA_CFG*n*)

Table 2416. FAR_RS_CFG*n* field descriptions

Field	Description
12:23 RS	Region size in 256-bytes

98.4.1.5 Tag RAM Control (TRC_CTRL)

Index 0x1A Access: Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	BAL											
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	ERR	EXC	PROG	INV	0	DSM			0	BAE	CMD	
W					w1c	w1c		w1c								
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0

Figure 2537. Tag RAM Control (TRC_CTRL)

Table 2417. TRC_CTRL field descriptions

Field	Description
4:15 BAL	Block Access Length.Should not cross the region boundary. Valid only when BAE = 1
20 ERR	Error JTAG access. Generated when read/write access to Tag RAM before getting the grant. Also generated on new JTAG Tag RAM access request before completing the previous access. **Note:** This field is not present in the IPS mapped version of this register. 0 No Error 1 Error

Table 2417. TRC_CTRL field descriptions (continued)

Field	Description
21 EXC	Excess JTAG access. Generated when JTAG access to Tag RAM exceeds the configured access length 0 No excess access to Tag RAM 1 Exceeds the configure access length
22 PROG	JTAG access to Tag RAM in progress 0 No Tag RAM access by JTAG 1 Tag RAM access by JTAG is in progress
23 INV	Invalid JTAG access. Generated when the targeted system/PD address in the TRC_ADR register falls outside the configured Tag RAM region 0 No error 1 Invalid address
25:27 DSM	Data Size Mode. Valid only for writes, all reads are fixed 32 bits only. 000 Word (32 bit) Access to the Tag RAM. 001 Half-Word (16-bit) access to Tag RAM. 16-bit LSBs of the TRC_DATA is used. 010 Byte (8-bit) access to the Tag RAM. 8-bit LSBs of the TRC_DATA is used. 100 Bit Access to the Tag RAM. LS bit of the TRC_DATA is used.
29 BAE	Block Access Enable. Block access is carried out with 32-bit Data Size Mode only. 0 Block Access disable 1 Block Access enable
30:31 CMD	00 No read or write access to Tag RAM 01 Read access to Tag RAM. Any write to TRC_ADR register results in read access to the Tag RAM. Read data is stored in the TRC_DATA register. 10 Write access to Tag RAM. Any write to TRC_ADR register results in write access request to the Tag RAM and subsequent TRC_DATA writes result in actual data to be written into tag RAM addressed by the TRC_DATA register. 11 Reserved

98.4.1.6 Tag RAM Configuration Address (TRC_ADR)

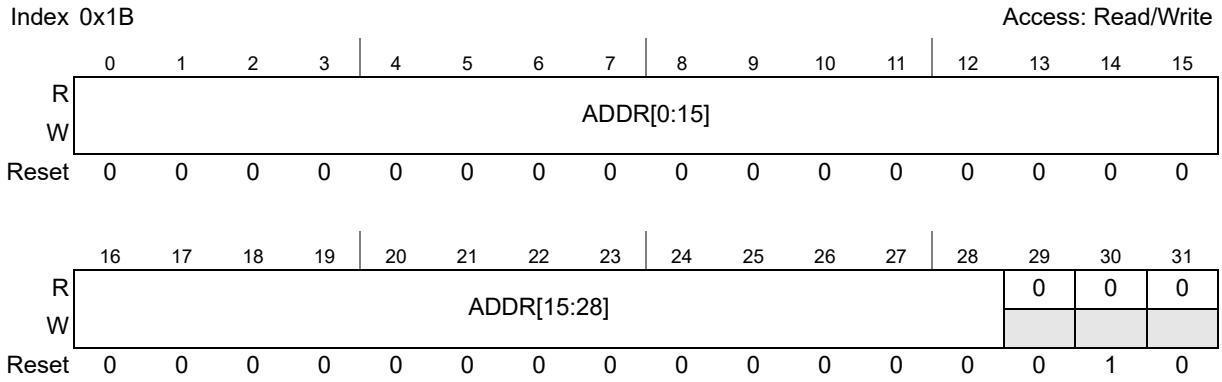


Figure 2538. Tag RAM Configuration Address (TRC_ADR)

Table 2418. TRC_ADR field descriptions

Field	Description
0:28 ADDR	PD Address. It should be aligned depending on Data Size mode field. DSM = 000, 256 byte aligned DSM = 001, 128 byte aligned DSM = 010, 64 byte aligned DSM = 100, 8 byte aligned (default) **Note:** If the PD address is not aligned to the DSM, this results in a erroneous data read/write.

98.4.1.7 Tag RAM Configuration Data (TRC_DATA)

Index 0x1C Access: Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	DATA[0:15]															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	DATA[16:31]															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0

Figure 2539. Tag RAM Configuration Data (TRC_DATA)

Table 2419. TRC_DATA field descriptions

Field	Description
0:31 DATA	Reflects the content of Tag RAM on Tag RAM read access. Also provides data for Tag RAM write access.

98.4.1.8 Out of Range Check Control 1 (ORC_CTRL1)

Index 0x20 Access: Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	CMP_MD3		CMP_MD2		CMP_MD1		CMP_MD0	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	TOR3	TOR2	TOR1	TOR0	0	0	0	0	RDCE3	RDCE2	RDCE1	RDCE0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0

Figure 2540. Out of Range Check Control 1 (ORC_CTRL1)

Table 2420. ORC_CTRL1 field descriptions

Field	Description
8:9 CMP_MD3	00 8 bit data comparison mode for comparator 3. 01 16 bit data comparison mode for comparator 3. 10 32 bit data comparison mode for comparator 3. 11 Reserved **Note:** All size of data writes are used to compare against the comparator size specified if data write size is greater than or equal to comparator size.
10:11 CMP_MD2	00 8 bit data comparison mode for comparator 2. 01 16 bit data comparison mode for comparator 2. 10 32 bit data comparison mode for comparator 2. 11 Reserved *Note: All size of data writes are used to compare against the comparator size specified if data write size is greater than or equal to comparator size.*
12:13 CMP_MD1	00 8 bit data comparison mode for comparator 1. 01 16 bit data comparison mode for comparator 1. 10 32 bit data comparison mode for comparator 1. 11 Reserved **Note:** All size of data writes are used to compare against the comparator size specified if data write size is greater than or equal to comparator size.
14:15 CMP_MD0	00 8 bit data comparison for comparator 0. 01 16 bit data comparison for comparator 0. 10 32 bit data comparison for comparator 0. 11 Reserved **Note:** All size of data writes are used to compare against the comparator size specified if data write size is greater than or equal to comparator size.
20 TOR3	Comparison Mode for Comparator 3 0 Trigger is generated if data write value is within the range ORC_CFG3_DL ≤ data_wr_value ≤ ORC_CFG3_DH 1 Trigger is generated if data write value is outside the range data_wr_value **<** ORC_CFG3_DL or data_wr_value > ORC_CFG3_DH
21 TOR2	Comparison Mode for Comparator 2 0 Trigger is generated if data write value is within the range ORC_CFG2_DL ≤ data_wr_value ≤ ORC_CFG2_DH 1 Trigger is generated if data write value is outside the range data_wr_value **<** ORC_CFG2_DL or data_wr_value **>** ORC_CFG2_DH
22 TOR1	Comparison Mode for Comparator 1 0 Trigger is generated if data write value is within the range ORC_CFG1_DL ≤ data_wr_value ≤ ORC_CFG1_DH 1 Trigger is generated if data write value is outside the range data_wr_value **<** ORC_CFG1_DL or data_wr_value >ORC_CFG1_DH
23 TOR0	Comparison Mode for Comparator 0 0 Trigger is generated if data write value is within the range ORC_CFG0_DL ≤ data_wr_value ≤ ORC_CFG0_DH 1 Trigger is generated if data write value is outside the range data_wr_value **<** ORC_CFG0_DL or data_wr_value > ORC_CFG0_DH
28 RDCE3	0 Disable Range Detect comparator 3 1 Enable Range Detect comparator 3

Table 2420. ORC_CTRL1 field descriptions (continued)

Field	Description
29 RDCE2	0 Disable Range Detect comparator 2 1 Enable Range Detect comparator 2
30 RDCE1	0 Disable Range Detect comparator 1 1 Enable Range Detect comparator 1
31 RDCE0	0 Disable Range Detect comparator 0 1 Enable Range Detect comparator 0

98.4.1.9 Out of Range Check Control 2 (ORC_CTRL2)

The ORC_CTRL2 register is shown in this figure.

Index 0x21 Access: Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	DTYP3		DTYP2		DTYP1		DTYP0	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	GTM	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0

Figure 2541. Out of Range Check Control 2 (ORC_CTRL2)

Table 2421. ORC_CTRL2 field descriptions

Field	Description
8:9 DTYP3	00 Unsigned data type comparison 01 Signed data type comparison 10 Floating data type comparison 11 Reserved
10:11 DTYP2	00 Unsigned data type comparison 01 Signed data type comparison 10 Floating data type comparison 11 Reserved
12:13 DTYP1	00 Unsigned data type comparison 01 Signed data type comparison 10 Floating data type comparison 11 Reserved

Table 2421. ORC_CTRL2 field descriptions (continued)

Field	Description
14:15 DTYP0	00 Unsigned data type comparison 01 Signed data type comparison 10 Floating data type comparison 11 Reserved
23 GTM	Generate Trigger on Mismatch 0 Disable trigger generation on data-size mismatch 1 Enable trigger generation on data-size mismatch **Note:** Mismatch happens if the data write size is less than comparator size (CMP_MD)

98.4.1.10 Global Status (GLB_STAT)

Index 0x22 Access: Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	DIDLE	0	TRSE
W																w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	DSM3	DSM2	DSM1	DSM0	0	0	0	0	TRDC3	TRDC2	TRDC1	TRDC0
W					w1c	w1c	w1c	w1c					w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0

Figure 2542. Global Status (GLB_STAT)

Table 2422. GLB_STAT field descriptions

Field	Description
13 DIDLE	DWPU module idle indication 1 Indicates that there is no more processing going on inside the module, or that all the messages have been completely processed and sent out.
15 TRSE	Tag RAM Size Error 1 Configured PD region for ATC exceeds the maximum Tag RAM size.
20 DSM3	1 Received data write trace size is smaller than comparator size (CMP_MP3)
21 DSM2	1 Received data write trace size is smaller than comparator size (CMP_MP2)
22 DSM1	1 Received data write trace size is smaller than comparator size (CMP_MP1)
23 DSM0	1 Received data write trace size is smaller than comparator size (CMP_MP0)

Table 2422. GLB_STAT field descriptions (continued)

Field	Description
28 TRDC3	1 Data range comparison for ORC_CFG3_ADR was successful
29 TRDC2	1 Data range comparison for ORC_CFG2_ADR was successful
30 TRDC1	1 Data range comparison for ORC_CFG1_ADR was successful
31 TRDC0	1 Data range comparison for ORC_CFG0_ADR was successful

98.4.1.11 Out of Range Configuration Address *n* (ORC_CFGn_ADR)

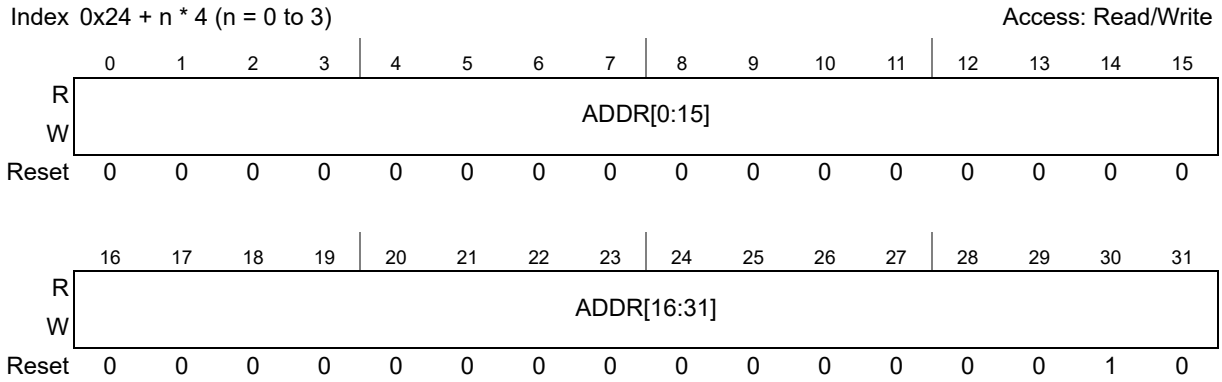


Figure 2543. Out of Range Configuration Address *n* (ORC_CFG*n*_ADR)

Table 2423. ORC_CFGn_ADR field descriptions

Field	Description
0:31 ADDR	Address value for the Data range comparison.

98.4.1.12 Out of Range Configuration Data High n (ORC_CFGn_DH)

Index 0x25 + n * 4 (n = 0 to 3) Access: Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	DATA_H[0:15]															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	DATA_H[16:31]															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0

Figure 2544. Out of Range Configuration Data High *n* (ORC_CFG*n*_DH)

Table 2424. ORC_CFGn_DH field descriptions

Field	Description
0:31 DATA_H	Data High value for the comparator.

98.4.1.13 Out of Range Configuration Data Low n (ORC_CFGn_DL)

Index 0x26 + n * 4 (n = 0 to 3) Access: Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	DATA_L[0:15]															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	DATA_L[16:31]															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0

Figure 2545. Out of Range Configuration Data Low *n* (ORC_CFG*n*_DL)

Table 2425. ORC_CFGn_DL field descriptions

Field	Description
0:31 DATA_L	Data low value for the comparator.

98.4.1.14 Out of Range Configuration Data Mask n (ORC_CFGn_DM)

Index 0x27 + n * 4 (n = 0 to 3) Access: Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	D_MASK[0:15]															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	D_MASK[16:31]															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0

Figure 2546. Out of Range Configuration Data Mask *n* (ORC_CFG*n*_DM)

Table 2426. ORC_CFGn_DM field descriptions

Field	Description
D_MASK	Data Mask value for the comparator (bit wise). 0 Data comparison not masked 1 Data comparison masked

98.5 Memory mapped registers

Table 2427. DWPU memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/page
0xC80_C000	Tag RAM Control (DWPU_TRC_CTRL)	32	R/W	0x0000_0000	*Section 98.5.1 on page 4088*
0xC80_C004	Tag RAM Configuration Address (DWPU_TRC_ADR)	32	R/W	0x0000_0000	*Section 98.5.2 on page 4090*
0xC80_C008	Tag RAM Configuration Data (DWPU_TRC_DATA)	32	R/W	0x0000_0000	*Section 98.5.3 on page 4090*

98.5.1 Tag Ram Control (DPWU_TRC_CTRL)

The Tag RAM Control register provides the configuration for Tag RAM access. It supports byte write IPS access. It also has a status bit to indicate the status of the Tag RAM access.

Offset: 0x0000 Access: Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	BAL											
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	EXC	PROG	INV	0	DSM			0	BAE	CMD	
W						w1c		w1c								
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0

Figure 2547. Tag RAM Control (DPWU_TRC_CTRL)

Table 2428. DPWU_TRC_CTRL field descriptions

Field	Description
4:15 BAL	Block Access Length.Should not cross the region boundary. Valid only when BAE = 1
21 EXC	Excess Tag RAM access by IPS. Generated when the IPS access to Tag RAM exceeds the configured access length 0 No excess access to Tag RAM 1 Exceeds the configure access length
22 PROG	IPS access to Tag RAM in progress 0 No Tag RAM access by IPS 1 Tag RAM access by IPS is in progress
23 INV	Invalid IPS access. Generated when the targeted PD/system address in the TRC_ADR register falls outside the configured Tag RAM region 0 No error 1 Invalid address
25:27 DSM	Data Size Mode. Valid only for writes, all reads are fixed 32 bits only. 000 Word (32 bit) Access to the Tag RAM. 001 Half-Word (16-bit) access to Tag RAM. 16-bit LSBs of the TRC_DATA is used. 010 Byte (8-bit) access to the Tag RAM. 8-bit LSBs of the TRC_DATA is used. 100 Bit Access to the Tag RAM. LS bit of the TRC_DATA is used.
29 BAE	Block Access Enable. Block access is carried out with 32-bit Data Size Mode only. 0 Block Access disable 1 Block Access enable
30:31 CMD	Read/Write access 00 No read or write access to Tag RAM 01 Read access to Tag RAM. Any write to TRC_ADR register results in read access to the Tag RAM. Read data is stored in the TRC_DATA register. 10 Write access to Tag RAM. Any write to TRC_ADR register results in write access request to the Tag RAM and subsequent TRC_DATA register writes result in actual data to be written into Tag RAM addressed by the TRC_DATA register. 11 Reserved

98.5.2 Tag RAM Configuration Address (DWPU_TRC_ADR)

Tag RAM Configuration Address provides the system address for Tag RAM access. Any write to this register initiates a Tag RAM access request at the address which is being translated from system address configured. It is always accessed in 32-bit only.

Offset: 0x0004 Access: Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	ADDR[0:15]															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	ADDR[15:28]													0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0

Figure 2548. Tag RAM Configuration Address (DWPU_TRC_ADR)

Table 2429. DWPU_TRC_ADR field descriptions

Field	Description
0:28 ADDR	PD Address. It should be aligned depending on Data Size mode field. DSM = 000, 256 byte aligned DSM = 001, 128 byte aligned DSM = 010, 64 byte aligned DSM = 100, 8 byte aligned (default) **Note:** If the PD address is not aligned to the DSM, this results in a erroneous data read/write.

98.5.3 Tag RAM Configuration Data (DWPU_TRC_DATA)

Tag RAM Configuration read/write data register. Generate ips_xfer_err on accessing this register before configuring the TRC_ADR register. Also on read access to this register while Tag RAM write is in progress and vice versa. Access is always 32-bit. During write, TRC_ADR.DSM determines which part of TRC_DATA should be written irrespective of byte enable.

Offset: 0x0008 Access: Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	DATA[0:15]															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	DATA[16:31]															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0

Figure 2549. Tag RAM Configuration Data (DWPU_TRC_DATA)

Table 2430. DWPU_TRC_DATA field descriptions

Field	Description
0:31 DATA	Reflects the content of Tag RAM on Tag RAM read access. Also provides data for Tag RAM write access.

99 Buddy Device—Nexus Read Write Access (RWA)

99.1 Introduction

The Nexus Read Write Access (RWA) module allows an external tool connected to the Emulation and Debug Device (ED) to read or write the Buddy Die calibration and trace memory via the JTAG interface. All accesses are setup and initiated by the Read/Write Access Control/Status Register (RWCS), as well as the Read/Write Access Address (RWA) and Read/Write Access Data Registers (RWD). The RWA module supports single as well as block reads and writes.

The RWA module is enabled by loading a single instruction (ACCESS_AUX_RWA_BD = 0x23) into the JTAG Instruction Register (IR). After the RWA module has been enabled, access to the RWA registers is enabled by loading an instruction (CORE_ENABLE) into the JTAG IR, as described in *Section 99.3: Nexus register access via JTAG*. The RWA module is disabled when the JTAG state machine initially reaches the Test-Logic-Reset state from any other state. The RWA module is also disabled if a Power-on-Reset (POR) event occurs. When disabled, the RWA module registers are not available for (32-bit or less) reads or writes.

99.2 Register definition

Table 2431 details the register map for the RWA module.

Table 2431. RWA register map

Register	Nexus access opcode	Read/ write	Read address	Write address
Read/Write Access Control/Status (RWCS)	0x7	R/W	0x0E	0x0F
Read/Write Access Address (RWA)	0x9	R/W	0x12	0x13
Read/Write Access Data (RWD)	0xA	R/W	0x14	0x15

99.2.1 Register descriptions

99.2.1.1 Read/write access control/status (RWCS)

The Read/write access control/status register provides control and status for the RWA module as shown in *Table 2433*.

Access: Read/Write

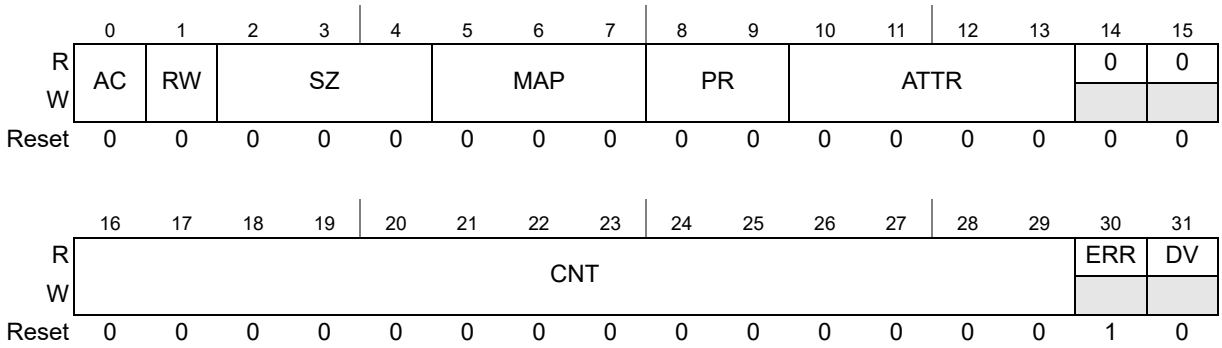


Figure 2550. Read/Write access control/status (RWCS) register

Table 2432. RWCS fields

Field	Description
0 AC	Access Control. 0 End access/ Access has completed 1 Start access
1 RW	Read/Write Select. 0 Read access 1 Write access
2:4 SZ	Word Size. 000 8-bit (byte) 001 16-bit (half-word) 010 32-bit (word) 011 Reserved (default to word) 100 Reserved (default to word) 101 Reserved (default to word) 110 Reserved (default to word) 111 Reserved (default to word)
5:7 MAP	MAP Select. 000 Primary memory map 001 Reserved 010 Reserved 011 Reserved 100 Reserved 101 Reserved 110 Reserved 111 Reserved
8:9 PR	Read/Write Access Priority. 00 Reserved (default to highest priority) 01 Reserved (default to highest priority) 10 Reserved (default to highest priority) 11 Highest access priority

Table 2432. RWCS fields (continued)

Field	Description
10:13 ATTR	Access Attributes. RWA accesses are run as privileged data non-cacheable accesses by default, and drive the p_d_hprot[5:0] bus access attributes to 6'b000011 accordingly. This field is provided to allow a portion of these default values to be modified when performing read or write accesses using the RWA as shown in *Table 2434*. hprot[1] must be set to one. 0xxx Reserved 1xxx Reserved x0xx p_d_hprot[4] driven to 0 for accesses x1xx p_d_hprot[4] driven to 1 for accesses xx0x p_d_hprot[3] driven to 0 for accesses xx1x p_d_hprot[3] driven to 1 for accesses xxx0 p_d_hprot[2] driven to 0 for accesses xxx1 p_d_hprot[2] driven to 1 for accesses
14:15	Reserved for future functionality.
16:29 CNT	Access Control Count. Number of accesses of word size SZ.
30 ERR	Read/Write Access Error (see *Table 2433*).
31 DV	Read/Write Access Data Valid (see *Table 2433*).

Table 2433. Read/Write access status bit encoding

Read action	Write action	ERR	DV
Read access has not completed	Write access completed without error	0	0
Read access error has occurred	Write access error has occurred	1	0
Read access completed without error	Write access has not completed	0	1
Not allowed	Not allowed	1	1

Table 2434. Protection encoding

p_hprot[4]	p_hprot[3]	p_hprot[2]	Transfer type
0	0	0	Cache-inhibited
0	0	1	Guarded, not cache-inhibited
0	1	0	Reserved
0	1	1	Reserved
1	0	0	Reserved
1	0	1	Reserved
1	1	0	Cachable, writethrough
1	1	1	Cachable, writeback

99.2.2 Read/write access data (RWD)

The Read/write access data register (RWD) provides the data to/from memory when initiating a read or a write access.

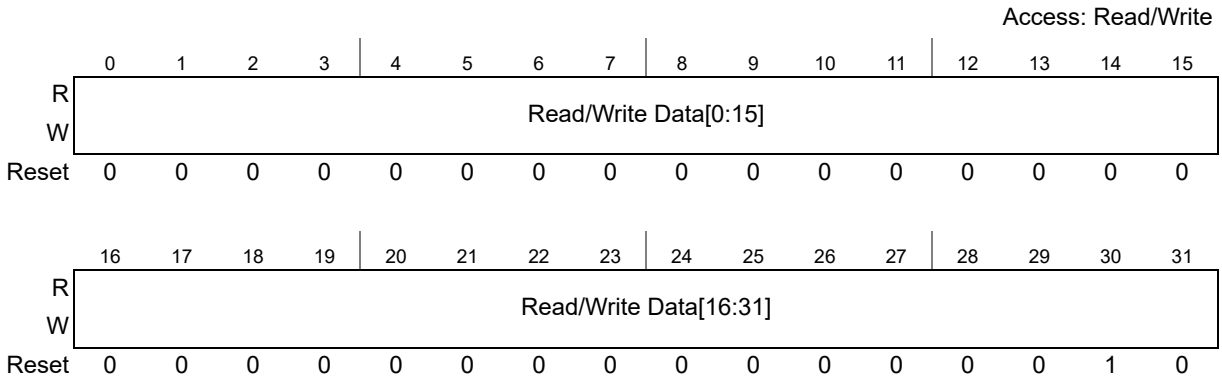


Figure 2551. Read/write access data (RWD) register

Read/write accesses to the memory require that the debug firmware properly retrieve/place the data in the RWD. *Table 2435* shows the proper placement of data into the RWD.

Table 2435. RWD data placement for transfers

Transfer size and byte offset	RWA(2:0)	RWCS[SZ]	RWD(1)			
			31:24	23:16	15:8	7:0
Byte	x x x	0 0 0	—	—	—	X
Half	x x 0	0 0 1	—	—	X	X
Word	x 0 0	0 1 0	X	X	X	X

1. X indicates byte lanes with valid data; — indicated bytes lanes with unused data.

Table 2436 shows the mapping of RWD bytes to byte lanes of the read and write data buses.

Table 2436. RWD byte lane mapping

Transfer size and byte offset	RWA(2:0)	RWD(1)			
		31:24	23:16	15:8	7:0
Byte @000	0 0 0	—	—	—	XBAR[7:0]
Byte @001	0 0 1	—	—	—	XBAR[15:8]
Byte @010	0 1 0	—	—	—	XBAR[23:16]
Byte @011	0 1 1	—	—	—	XBAR[31:24]
Byte @100	1 0 0	—	—	—	XBAR[39:32]
Byte @101	1 0 1	—	—	—	XBAR[47:40]
Byte @110	1 1 0	—	—	—	XBAR[55:48]
Byte @111	1 1 1	—	—	-	XBAR[63:56]

Table 2436. RWD byte lane mapping (continued)

Transfer size and byte offset	RWA(2:0)	RWD(1)			
		31:24	23:16	15:8	7:0
Half @000	0 0 0	—	—	XBAR[15:8]	XBAR[7:0]
Half @010	0 1 0	—-	—	XBAR[31:24]	XBAR[23:16]
Half @100	1 0 0	—-	—	XBAR[47:40]	XBAR[39:32]
Half @110	1 1 0	—	—	XBAR[63:56]	XBAR[55:48]
Word @000	0 0 0	XBAR[31:24]	XBAR[23:16]	XBAR[15:8]	XBAR[7:0]
Word @100	1 0 0	XBAR[63:56]	XBAR[55:48]	XBAR[47:40]	XBAR[39:32]

1. — indicates byte lanes that contain unused data

99.2.3 Read/write access address (RWA)

The Read/write access address register provides the address to be accessed when initiating a read or a write access.

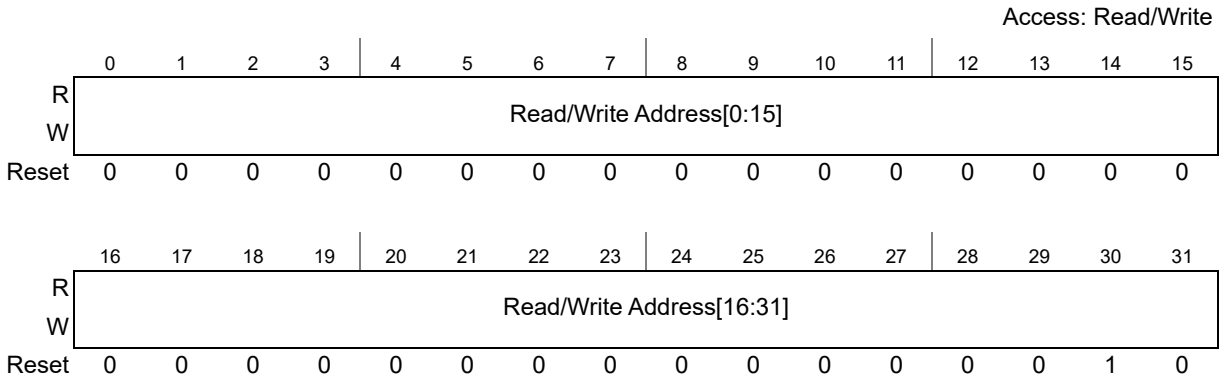


Figure 2552. Read/write access data (RWA) register

99.3 Nexus register access via JTAG

After the RWA module has been enabled, access to the RWA registers is enabled by loading an instruction (CORE_ENABLE = 0x7C) into the JTAG Instruction Register (IR). Once the CORE_ENABLE instruction has been loaded, the JTAG port allows tool/target communications with all Nexus registers according to the map in *Table 2431*.

Reading or writing of a RWA module register then requires two passes through the Data-Scan (DR) path of the JTAG state machine.

1. The first pass through the DR selects the RWA module register to be accessed by providing an index and the direction (read/write) as shown in *Table 2431*. This is achieved by loading an 8-bit value into the JTAG data register (DR).

   This register has the following format:

Figure 2553. JTAG DR for Nexus register access

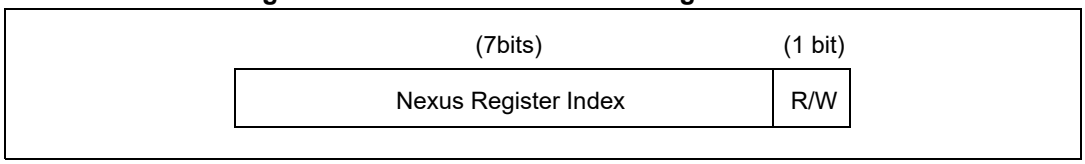


Table 2437. JTAG DR field description for Nexus register access

Field	Description
Nexus Register Index	Selected from values in *Table 2431*
Read/Write (R/W)	0 Read 1 Write

2. The second pass through the DR then shifts the data in or out of the JTAG port, LSB first.
   a) During a read access, data is latched from the selected Nexus register when the JTAG state machine passes through the Capture-DR state.
   b) During a write access, data is latched into the selected Nexus register when the JTAG state machine passes through the Update-DR state.

99.4 Memory access examples

The following subsections describe the steps which are required to access memory.

99.4.1 Single write access

1. Initialize the RWA register through the JTAG. Configure as follows:
   – Write Address = 32h’xxxxxxxx (write address)
2. Initialize the RWCS register through the JTAG. Configure the bits as follows:
   – Access Control (AC) = 1b’1 (to indicate start access)
   – Map Select (MAP) = 3b’000 (primary memory map)
   – Access Priority (PR) = 2b’00 (lowest priority)
   – Read/Write (RW) = 1b’1 (write access)
   – Word Size (SZ) = 3b’0xx (32-bit, 16-bit, 8-bit)
   – Access Count (CNT) = 14h’0000 or 14h’0001(single access)

Note: Access Count (CNT) of 14’h0000 or 14’h0001 performs a single access.

3. Initialize the RWD register through the JTAG. Configure as follows:
   – Write Data = 32h’xxxxxxxx (write data)
4. The RWA module then transfers the data value from the data buffer RWD register to the memory mapped address in the RWA register. When the access has completed without error (ERR = 1’b0), the RWA clears the DV bit in the RWCS register. This indicates that the device is ready for the next access.

99.4.2 Block write access

1. For a block write access, follow Steps 1, 2, and 3 outlined in *Section 99.4.1: Single write access* to initialize the registers, but using a value greater than one (14'h0001) for the CNT field in the RWCS register.
2. The RWA module then transfers the first data value from the RWD register to the memory mapped address in the RWA register. When the transfer has completed without error (ERR=1'b0), the address from the RWA register is incremented to the next word size (specified in the SZ field) and the number from the CNT field is decremented. The RWA module then indicates it is ready for the next access.
3. Repeat Step 3 in *Section 99.4.1: Single write access* until the internal CNT value is zero (0). When this occurs, the DV bit within the RWCS is cleared to indicate the end of the block write access.

Note: The actual RWA register value as well as the CNT field within the RWCS are not changed when executing a block write access. The original values can be read by the external development tool at any time.

99.4.3 Single read access

1. Initialize the RWA register through the JTAG. Configure as follows:
   – Read Address = 32h'xxxxxxxx (read address)
2. Initialize the RWCS register through the JTAG. Configure the bits as follows:
   – Access Control (AC) = 1b'1 (to indicate start access)
   – Map Select (MAP) = 3b'000 (primary memory map)
   – Access Priority (PR) = 2b'00 (lowest priority)
   – Read/Write (RW) = 1b'0 (read access)
   – Word Size (SZ) = 3b'0xx (32-bit, 16-bit, 8-bit)
   – Access Count (CNT) = 14h'0000 or 14h'0001(single access)

Note: Access Count (CNT) of 14'h0000 or 14'h0001 performs a single access.

3. The RWA module then transfers the data to the RWD register. When the transfer is completed without error (ERR = 1'b0), the RWA module sets the DV bit in the RWCS register. This indicates that the RWA module is ready for the next access.
4. The data can then be read from the RWD register through the JTAG.

99.4.4 Block read access

1. For a block read access, follow Steps 1 and 2 outlined in *Section 99.4.3: Single read access* to initialize the registers, but using a value greater than one (14'h0001) for the CNT field in the RWCS register.
2. The RWA module then transfers the data to the RWD register. When the transfer has completed without error (ERR = 1'b0), the address from the RWA register is incremented to the next word size (specified in the SZ field) and the number from the CNT field is decremented. The RWA module then indicates that the device is ready for the next access.
3. The data can then be read from the RWD register through JTAG.
4. Repeat Steps 3 and 4 in *Section 99.4.3: Single read access* until the CNT value is zero (0). When this occurs, the DV bit within the RWCS register is set to indicate the end of the block read access.

Note: The data values must be shifted out 32-bits at a time LSB first. The actual RWA value as well as the CNT field within the RWCS register are not changed when executing a block read access. The original values can be read by the external development tool at any time.

100 Buddy Device—Crossbar and RAM (BD_XBS_RAM)

Within the buddy device (BD), there is a crossbar switch that provides the required connections between multiple bus masters and the overlay RAM. For a diagram of the Emulation device (ED) debug architecture that shows the BD blocks and crossbar, see the Emulation device introduction chapter. This section provides details on the structure of this function.

Note: Throughout this description, the term "overlay RAM" is applied to the BD memory, even though this memory can be used for both overlay and debug trace information.

The 64-bit BD crossbar and RAM structure consists of:

- Input registers to capture transfers (address, attributes and write data) targeted at the overlay RAM or memory-mapped peripheral space
- A 3 x 5 crossbar switch to route requests from masters to either the overlay RAM or the system bus
  - Bus masters include the BD's Nexus Aurora Router (BD-NAR), the production device's flash controller (PFLASH) and the BD's Nexus Read/Write Access Client (RWA)
  - Bus slaves include 4 equal-sized regions of the overlay RAM and a peripheral bridge bus controller
    - 160-bit RAM arrays (128 bits of data, 32 bits of end-to-end ECC)
    - End-to-End ECC32 address and data protection
    - ECC scheme based on 29-bit address and 32-bit data
    - Overlay RAM supports a 64-bit "early write" datapath and a 128-bit read datapath
- Output alignment logic and registers to capture read data being driven to the requesting master

The result is an efficient structure supporting concurrent references of the overlay RAM for calibration accesses from the PFLASH controller on the Production Device (PD) and debug references from the other masters. The crossbar and RAM support 200 MHz operation.

Note: Calibration reads from the PFLASH have the same timing characteristics as normal data reads from the flash array.

100.1 3 x 5 Crossbar switch and attached controllers

In the same manner as the comparable crossbar switches in the PD, the BD crossbar concurrently supports up to 3 simultaneous connections between the master and the slave ports. It implements a 64-bit "early write" datapath and a 128-bit read datapath and fully supports concurrent accesses between calibration reads and debug trace writes to different overlay RAM regions.

This crossbar implements fixed priority arbitration on a per slave port basis where the master number defines the relative priority in a reverse numeric order, that is, m0 > m1 > m2 (PFLASH > BD-NAR > Nexus RWA). Each slave port is "parked" on the last master making a request.

The design supports {8, 16, 32, 64}-bit SINGLE reads, {8, 16, 32, 64}-bit SINGLE writes along with 64-bit WRAP4 burst reads and writes. Two consecutive sequential 128-bit accesses of the RAM are used (only) on the WRAP4 burst read.

100.2 BD_XBS_RAM End-to-End ECC32 Checkbit/Syndrome Coding Scheme

The e2eECC scheme used in the BD overlay RAM is a simplified derivative of the algorithm used in the Production Device. The only difference is the code is based on a 32-bit data field, but the same H-matrix is used with the appropriate 32-bit data upper or lower word zeroed.

Consider the following Boolean equation definitions:

e2eECC64[7:0] = ecc_addr[7:0] ^ ecc_data_upper[7:0] ^ ecc_data_lower[7:0]

where `ecc_addr` is H-matrix evaluation of the 29-bit access address (addr[31:3]), `ecc_data_upper` is the H-matrix evaluation of the upper data word (data[63:32]) and `ecc_data_lower` is the comparable evaluation of the lower data word (data[31:0]).

For the BD RAM definition, the following (re)definitions apply:

e2eECC32_upper[7:0] = ecc_addr[7:0] ^ ecc_data_upper[7:0]

e2eECC32_lower[7:0] = ecc_addr[7:0] ^ ecc_data_lower[7:0]

As a result, the PD's H-matrix is used for the BD RAM's e2eECC32 scheme with the appropriate 32 bit data word zeroed out:

e2eECC32_upper[7:0] = ecc_addr[7:0] ^ ecc_data_upper[7:0]; data_lower[31:00] = 0

e2eECC32_lower[7:0] = ecc_addr[7:0] ^ ecc_data_lower[7:0]; data_upper[63:32] = 0

In any case, the resulting ECC32 is calculated based on the appropriate 32 data bits plus 29 address bits (the upper bits of the 32-bit address field).

The complete H-matrix for this (101, 93) code is shown in *Table 2438*. A '*' in *Table 2438* indicates the corresponding data or address bit is XOR'd to form the final checkbit value on the left. For 32-bit data writes, the table sections corresponding to D[63:32] or D[31:0], together with A[31:3] are logically summed (output of each table section is XOR'd) together to the final value driven on the hwchkbit[7:0] outputs. Note that this table uses *the AHB bit numbering convention where bit[0] is the least significant bit*.

Table 2438. e2eECC Basic H-Matrix Definition

Checkbits [7:0]	Data Bit(1)																													
	Byte 7								Byte 6								Byte 5								Byte 4							
	63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
7	*				*					*			*		*	*		*	*	*	*					*	*			*	*	
6			*	*			*	*					*		*		*				*				*			*	*			*
5	*	*	*	*					*				*	*		*		*	*				*	*	*				*			
4	*		*			*		*					*		*		*			*			*	*			*				*	

Table 2438. e2eECC Basic H-Matrix Definition (continued)

Checkbits [7:0]	Data Bit[1]																													
	Byte 7								Byte 6								Byte 5								Byte 4							
	63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
3									*	*	*	*	*		*				*		*				*	*	*	*				
2					*		*					*		*			*	*				*	*						*		*
1		*		*		*		*	*	*	*	*						*		*		*		*					*		*
0		*			*	*	*				*			*	*	*		*				*					*		*	*	
	Byte 3								Byte 2								Byte 1								Byte 0							
Checkbits [7:0]	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
7					*		*				*		*				*	*			*	*			*				*	*		*
6	*	*		*	*				*				*				*	*		*	*					*	*				*	*
5			*	*			*	*	*			*	*	*					*		*	*		*								
4	*		*	*				*		*	*				*	*	*			*			*	*				*		*		
3	*	*	*			*						*	*		*	*			*				*			*	*	*			*		
2		*		*	*	*	*	*	*	*	*			*					*				*			*		*	*		*		
1		*						*		*			*	*	*			*				*				*			*	*	*	
0		*		*		*		*				*				*				*				*				*	*		*	*
Checkbits [7:0]	Address Bit[1]																													
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
7		*		*		*		*	*	*		*	*	*		*	*	*		*		*		*	*	*		*				
6		*		*	*	*	*			*	*			*	*	*	*		*		*	*		*	*	*		*				
5		*	*	*			*	*			*	*		*		*		*	*			*	*	*		*	*	*	*			
4	*	*	*		*	*		*		*	*	*	*		*					*	*		*	*	*		*	*				
3	*		*		*	*	*		*		*	*	*			*		*		*	*		*		*	*	*		*			
2	*	*			*		*	*	*	*				*	*	*	*	*	*	*		*	*		*		*		*			
1	*		*	*		*	*		*	*		*	*		*		*	*	*		*	*	*					*	*			
0	*		*	*	*			*	*		*		*	*	*		*		*	*	*			*		*	*		*			

1. Bit numbering is AHB convention, bit 0 is LSB. D[7:0] corresponds to byte at address 0. D[63:56] corresponds to byte at address 7.

Figure 2554 through *Figure 2557* show an alternative representation of the ECC32 encode process, written as a C language function.

Figure 2554. C language encodeECC32_upper function description (Part 1 of 4)

```
encodeEcc32_upper (addr, data_a2_is_zero)
    unsigned int        addr;                   /* 32-bit byte address */
    unsigned int       data_a2_is_zero;       /* 32-bit data upper, a[2]=0 */

{
    unsigned int       addr_ecc;              /* 8 bits of ecc for address */
    unsigned int        ecc;                    /* 8 bits of ecc codeword */

    /* the following equation calculates the 8-bit wide ecc codeword by
examining
    each addr or data bits and xor'ing the appropriate H-matrix value if the
bit = 1 */

    addr_ecc
        = (((addr           >> 31) & 1) ? 0x1f : 0x0)      /* addr[31] */
        ^ (((addr           >> 30) & 1) ? 0xf4 : 0x0)      /* addr[30] */
        ^ (((addr           >> 29) & 1) ? 0x3b : 0x0)      /* addr[29] */
        ^ (((addr           >> 28) & 1) ? 0xe3 : 0x0)      /* addr[28] */
        ^ (((addr           >> 27) & 1) ? 0x5d : 0x0)      /* addr[27] */
        ^ (((addr           >> 26) & 1) ? 0xda : 0x0)      /* addr[26] */
        ^ (((addr           >> 25) & 1) ? 0x6e : 0x0)      /* addr[25] */
        ^ (((addr           >> 24) & 1) ? 0xb5 : 0x0)      /* addr[24] */

        ^ (((addr           >> 23) & 1) ? 0x8f : 0x0)      /* addr[23] */
        ^ (((addr           >> 22) & 1) ? 0xd6 : 0x0)      /* addr[22] */
        ^ (((addr           >> 21) & 1) ? 0x79 : 0x0)      /* addr[21] */
        ^ (((addr           >> 20) & 1) ? 0xba : 0x0)      /* addr[20] */
        ^ (((addr           >> 19) & 1) ? 0x9b : 0x0)      /* addr[19] */
        ^ (((addr           >> 18) & 1) ? 0xe5 : 0x0)      /* addr[18] */
        ^ (((addr           >> 17) & 1) ? 0x57 : 0x0)      /* addr[17] */
        ^ (((addr           >> 16) & 1) ? 0xec : 0x0)      /* addr[16] */

        ^ (((addr           >> 15) & 1) ? 0xc7 : 0x0)      /* addr[15] */
        ^ (((addr           >> 14) & 1) ? 0xae : 0x0)      /* addr[14] */
        ^ (((addr           >> 13) & 1) ? 0x67 : 0x0)      /* addr[13] */
        ^ (((addr           >> 12) & 1) ? 0x9d : 0x0)      /* addr[12] */
        ^ (((addr           >> 11) & 1) ? 0x5b : 0x0)      /* addr[11] */
        ^ (((addr           >> 10) & 1) ? 0xe6 : 0x0)      /* addr[10] */
        ^ (((addr           >>  9) & 1) ? 0x3e : 0x0)      /* addr[ 9] */
        ^ (((addr           >>  8) & 1) ? 0xf1 : 0x0)      /* addr[ 8] */

        ^ (((addr           >>  7) & 1) ? 0xdc : 0x0)      /* addr[ 7] */
        ^ (((addr           >>  6) & 1) ? 0xe9 : 0x0)      /* addr[ 6] */
        ^ (((addr           >>  5) & 1) ? 0x3d : 0x0)      /* addr[ 5] */
        ^ (((addr           >>  4) & 1) ? 0xf2 : 0x0)      /* addr[ 4] */
        ^ (((addr           >>  3) & 1) ? 0x2f : 0x0);     /* addr[ 3] */
```

Figure 2555. C language encodeECC32_upper function description (Part 2 of 4)

```
      ecc = (((data_a2_is_zero >> 31) & 1) ? 0xb0 : 0x0)      /* data[63] */
             ^ (((data_a2_is_zero >> 30) & 1) ? 0x23 : 0x0)      /* data[62] */
             ^ (((data_a2_is_zero >> 29) & 1) ? 0x70 : 0x0)      /* data[61] */
             ^ (((data_a2_is_zero >> 28) & 1) ? 0x62 : 0x0)      /* data[60] */
             ^ (((data_a2_is_zero >> 27) & 1) ? 0x85 : 0x0)      /* data[59] */
             ^ (((data_a2_is_zero >> 26) & 1) ? 0x13 : 0x0)      /* data[58] */
             ^ (((data_a2_is_zero >> 25) & 1) ? 0x45 : 0x0)      /* data[57] */
             ^ (((data_a2_is_zero >> 24) & 1) ? 0x52 : 0x0)      /* data[56] */

             ^ (((data_a2_is_zero >> 23) & 1) ? 0x2a : 0x0)      /* data[55] */
             ^ (((data_a2_is_zero >> 22) & 1) ? 0x8a : 0x0)      /* data[54] */
             ^ (((data_a2_is_zero >> 21) & 1) ? 0x0b : 0x0)      /* data[53] */
             ^ (((data_a2_is_zero >> 20) & 1) ? 0x0e : 0x0)      /* data[52] */
             ^ (((data_a2_is_zero >> 19) & 1) ? 0xf8 : 0x0)      /* data[51] */
             ^ (((data_a2_is_zero >> 18) & 1) ? 0x25 : 0x0)      /* data[50] */
             ^ (((data_a2_is_zero >> 17) & 1) ? 0xd9 : 0x0)      /* data[49] */
             ^ (((data_a2_is_zero >> 16) & 1) ? 0xa1 : 0x0)      /* data[48] */

             ^ (((data_a2_is_zero >> 15) & 1) ? 0x54 : 0x0)      /* data[47] */
             ^ (((data_a2_is_zero >> 14) & 1) ? 0xa7 : 0x0)      /* data[46] */
             ^ (((data_a2_is_zero >> 13) & 1) ? 0xa8 : 0x0)      /* data[45] */
             ^ (((data_a2_is_zero >> 12) & 1) ? 0x92 : 0x0)      /* data[44] */
             ^ (((data_a2_is_zero >> 11) & 1) ? 0xc8 : 0x0)      /* data[43] */
             ^ (((data_a2_is_zero >> 10) & 1) ? 0x07 : 0x0)      /* data[42] */
             ^ (((data_a2_is_zero >>  9) & 1) ? 0x34 : 0x0)      /* data[41] */
             ^ (((data_a2_is_zero >>  8) & 1) ? 0x32 : 0x0)      /* data[40] */

             ^ (((data_a2_is_zero >>  7) & 1) ? 0x68 : 0x0)      /* data[39] */
             ^ (((data_a2_is_zero >>  6) & 1) ? 0x89 : 0x0)      /* data[38] */
             ^ (((data_a2_is_zero >>  5) & 1) ? 0x98 : 0x0)      /* data[37] */
             ^ (((data_a2_is_zero >>  4) & 1) ? 0x49 : 0x0)      /* data[36] */
             ^ (((data_a2_is_zero >>  3) & 1) ? 0x61 : 0x0)      /* data[35] */
             ^ (((data_a2_is_zero >>  2) & 1) ? 0x86 : 0x0)      /* data[34] */
             ^ (((data_a2_is_zero >>  1) & 1) ? 0x91 : 0x0)      /* data[33] */
             ^  ((data_a2_is_zero        & 1) ? 0x46 : 0x0);     /* data[32] */

      return(ecc);
      }
```

Figure 2556. C language encodeECC32_upper function description (Part 3 of 4)

```
C Language encodeECC32_lower Function Description
encodeEcc32_lower (addr, data_a2_is_one)
    unsigned int        addr;                   /* 32-bit byte address */
    unsigned int       data_a2_is_one;         /* 32-bit data lower, a[2]=1 */

{
    unsigned int       addr_ecc;               /* 8 bits of ecc for address */
    unsigned int        ecc;                    /* 8 bits of ecc codeword */

    /* the following equation calculates the 8-bit wide ecc codeword by
examining
    each addr or data bits and xor'ing the appropriate H-matrix value if the
bit = 1 */

    addr_ecc
        = (((addr           >> 31) & 1) ? 0x1f : 0x0)      /* addr[31] */
        ^ (((addr           >> 30) & 1) ? 0xf4 : 0x0)      /* addr[30] */
        ^ (((addr           >> 29) & 1) ? 0x3b : 0x0)      /* addr[29] */
        ^ (((addr           >> 28) & 1) ? 0xe3 : 0x0)      /* addr[28] */
        ^ (((addr           >> 27) & 1) ? 0x5d : 0x0)      /* addr[27] */
        ^ (((addr           >> 26) & 1) ? 0xda : 0x0)      /* addr[26] */
        ^ (((addr           >> 25) & 1) ? 0x6e : 0x0)      /* addr[25] */
        ^ (((addr           >> 24) & 1) ? 0xb5 : 0x0)      /* addr[24] */

        ^ (((addr           >> 23) & 1) ? 0x8f : 0x0)      /* addr[23] */
        ^ (((addr           >> 22) & 1) ? 0xd6 : 0x0)      /* addr[22] */
        ^ (((addr           >> 21) & 1) ? 0x79 : 0x0)      /* addr[21] */
        ^ (((addr           >> 20) & 1) ? 0xba : 0x0)      /* addr[20] */
        ^ (((addr           >> 19) & 1) ? 0x9b : 0x0)      /* addr[19] */
        ^ (((addr           >> 18) & 1) ? 0xe5 : 0x0)      /* addr[18] */
        ^ (((addr           >> 17) & 1) ? 0x57 : 0x0)      /* addr[17] */
        ^ (((addr           >> 16) & 1) ? 0xec : 0x0)      /* addr[16] */

        ^ (((addr           >> 15) & 1) ? 0xc7 : 0x0)      /* addr[15] */
        ^ (((addr           >> 14) & 1) ? 0xae : 0x0)      /* addr[14] */
        ^ (((addr           >> 13) & 1) ? 0x67 : 0x0)      /* addr[13] */
        ^ (((addr           >> 12) & 1) ? 0x9d : 0x0)      /* addr[12] */
        ^ (((addr           >> 11) & 1) ? 0x5b : 0x0)      /* addr[11] */
        ^ (((addr           >> 10) & 1) ? 0xe6 : 0x0)      /* addr[10] */
        ^ (((addr           >>  9) & 1) ? 0x3e : 0x0)      /* addr[ 9] */
        ^ (((addr           >>  8) & 1) ? 0xf1 : 0x0)      /* addr[ 8] */

        ^ (((addr           >>  7) & 1) ? 0xdc : 0x0)      /* addr[ 7] */
        ^ (((addr           >>  6) & 1) ? 0xe9 : 0x0)      /* addr[ 6] */
        ^ (((addr           >>  5) & 1) ? 0x3d : 0x0)      /* addr[ 5] */
        ^ (((addr           >>  4) & 1) ? 0xf2 : 0x0)      /* addr[ 4] */
        ^ (((addr           >>  3) & 1) ? 0x2f : 0x0);     /* addr[ 3] */
```

Figure 2557. C language encodeECC32_upper function description (Part 4 of 4)

```
          ecc = (((data_a2_is_one  >> 31) & 1) ? 0x58 : 0x0)      /* data[31] */
              ^ (((data_a2_is_one  >> 30) & 1) ? 0x4f : 0x0)      /* data[30] */
              ^ (((data_a2_is_one  >> 29) & 1) ? 0x38 : 0x0)      /* data[29] */
              ^ (((data_a2_is_one  >> 28) & 1) ? 0x75 : 0x0)      /* data[28] */
              ^ (((data_a2_is_one  >> 27) & 1) ? 0xc4 : 0x0)      /* data[27] */
              ^ (((data_a2_is_one  >> 26) & 1) ? 0x0d : 0x0)      /* data[26] */
              ^ (((data_a2_is_one  >> 25) & 1) ? 0xa4 : 0x0)      /* data[25] */
              ^ (((data_a2_is_one  >> 24) & 1) ? 0x37 : 0x0)      /* data[24] */

              ^ (((data_a2_is_one  >> 23) & 1) ? 0x64 : 0x0)      /* data[23] */
              ^ (((data_a2_is_one  >> 22) & 1) ? 0x16 : 0x0)      /* data[22] */
              ^ (((data_a2_is_one  >> 21) & 1) ? 0x94 : 0x0)      /* data[21] */
              ^ (((data_a2_is_one  >> 20) & 1) ? 0x29 : 0x0)      /* data[20] */
              ^ (((data_a2_is_one  >> 19) & 1) ? 0xea : 0x0)      /* data[19] */
              ^ (((data_a2_is_one  >> 18) & 1) ? 0x26 : 0x0)      /* data[18] */
              ^ (((data_a2_is_one  >> 17) & 1) ? 0x1a : 0x0)      /* data[17] */
              ^ (((data_a2_is_one  >> 16) & 1) ? 0x19 : 0x0)      /* data[16] */

              ^ (((data_a2_is_one  >> 15) & 1) ? 0xd0 : 0x0)      /* data[15] */
              ^ (((data_a2_is_one  >> 14) & 1) ? 0xc2 : 0x0)      /* data[14] */
              ^ (((data_a2_is_one  >> 13) & 1) ? 0x2c : 0x0)      /* data[13] */
              ^ (((data_a2_is_one  >> 12) & 1) ? 0x51 : 0x0)      /* data[12] */
              ^ (((data_a2_is_one  >> 11) & 1) ? 0xe0 : 0x0)      /* data[11] */
              ^ (((data_a2_is_one  >> 10) & 1) ? 0xa2 : 0x0)      /* data[10] */
              ^ (((data_a2_is_one  >>  9) & 1) ? 0x1c : 0x0)      /* data[ 9] */
              ^ (((data_a2_is_one  >>  8) & 1) ? 0x31 : 0x0)      /* data[ 8] */

              ^ (((data_a2_is_one  >>  7) & 1) ? 0x8c : 0x0)      /* data[ 7] */
              ^ (((data_a2_is_one  >>  6) & 1) ? 0x4a : 0x0)      /* data[ 6] */
              ^ (((data_a2_is_one  >>  5) & 1) ? 0x4c : 0x0)      /* data[ 5] */
              ^ (((data_a2_is_one  >>  4) & 1) ? 0x15 : 0x0)      /* data[ 4] */
              ^ (((data_a2_is_one  >>  3) & 1) ? 0x83 : 0x0)      /* data[ 3] */
              ^ (((data_a2_is_one  >>  2) & 1) ? 0x9e : 0x0)      /* data[ 2] */
              ^ (((data_a2_is_one  >>  1) & 1) ? 0x43 : 0x0)      /* data[ 1] */
              ^  ((data_a2_is_one         & 1) ? 0xc1 : 0x0);     /* data[ 0] */

      return(ecc);
}
```

As the ECC syndrome is calculated on a read operation by applying the H-matrix to the data plus the checkbits, an all zero syndrome indicates an error free operation. If the generated syndrome value is non-zero and matches one of the H-matrix values associated with the data or checkbits, it represents a single-bit error correction case and the specific bit is complemented to produce the correct data value. If the syndrome value matches one of the H-matrix values associated with the address bits, the opposite data word, or is an even weight value, or represents an unused odd weight value, a non-correctable ECC event has been detected and the appropriate error termination response is initiated.

101 Buddy Device—System Integration Unit Lite (BD_SIUL2)

101.1 Introduction

The Buddy Device—System Integration Unit Lite2 (BD_SIUL2) provides control over the electrical pads of the buddy device (BD), as well as implementing various SoC functions that are not included in other modules.

The main functions supported by BD_SIUL2 are:

- Device identification
- Device status reporting
- Pad control and signal muxing

It should be noted that some other system level features are located in the BD DCI block including:

- Device identification available via the JTAG accessible Nexus Device ID (DID) register
- Reporting of reset status via the JTAG accessible reset status register
- Control of production device (PD) and BD reset via JTAG accessible reset control register

These registers are also indirectly accessible by software via BD_JTAGM in software debug mode.

101.2 Device identification

The BD_SIUL2 includes a device ID register (DIDR) that identifies the BD type and version, allowing a connected tool or development/calibration software running on a connected PD to determine what debug or calibration resources are available. The DIDR uses similar formatting to the MCU ID registers (MIDR1, MIDR2) included in the SIUL2 module.

This information is made available via the BD SIU rather than via the PD because it is either not available via the PD (for example, information on BD-specific resources, such as extended overlay RAM size) or because it is useful for the tool to be able to access the information before the PD is powered up.

Information available includes:

- An ID code that can be used to uniquely identify the BD type.
- A version field that is changed for every major mask revision of the BD.
- A parameterized field that indicates the amount of extended overlay RAM included
  - The field coding allows indication of up to 4 MB of RAM, with resolution of 256 KB steps.

It should be noted that in addition to information available in the DIDR, a connected tool can also retrieve identification information from other sources including:

- The JTAG ID returned by the buddy DCI/JTAGC module, which is different than the JTAG ID returned by a standard PD DCI/JTAGC.
- The MIDR register included in the PD SIUL2 module, which includes information on the PD device type, version and package type (including identification of emulation and debug device package variants).

101.3 Device status information

101.3.1 Initialization status

The BD SIU includes a flag (BD_INIT) that can be used by the connected PD or any connected debug or calibration tools to track the status of the BD.

The main expected usage mode for this bit is to allow the Boot Assist Flash (BAF) or user calibration software on the PD to determine whether the BD has lost power since it was last initialized. A typical usage sequence for BD_INIT is shown below.

- At BD power on reset (POR), BD_INIT is cleared automatically.
  - BD_INIT is not cleared by external BD reset input.
- A calibration tool or calibration software initializes the BD (for example, writing BD overlay RAM with calibration data and remap configuration).
- The calibration tool or calibration software sets BD_INIT.
- After PD POR, the BAF executes and reads BD_INIT.
  - If BD_INIT is clear, the BD overlay is assumed to be uninitialized and is not used for calibration remap.
  - If BD_INIT is set, the BD overlay is assumed to be initialized and can be used for calibration remap. (Other tests to verify data integrity may be used).

BD_INIT is cleared by BD POR, and can be set or cleared via the debug interface or writes from the PD. BD_INIT is only used to detect full power down events resulting in a subsequent POR. It does not detect brown out drops in the supply voltage.

101.3.2 PD reset status

The IPS accessible BD_SIU_RSR register shows what stage of reset processing the connected PD is in, and whether the PD is powered down or not. This is functionally different from the software accessible registers in the PD reset generation module that indicate which reset source has caused the last PD reset.

101.4 Memory map

Table 2439 gives an overview of the BD_SIUL2 registers.

Table 2439. BD_SIUL2 register summary

Offset	Register	Access	Reset Value	Section/Page
0x0000	BD_SIUL2 Device Identification Register 1 (BD_SIUL2_DIDR1)	R	0x57BD_0000	*Section 101.4.1.1*
0x0004	BD_SIUL2 Device Identification Register 2 (BD_SIUL2_DIDR2)	R	0x0003_0100	*Section 101.4.1.2*
0x0100	BD_SIUL2 Initialization Status Register (BD_SIUL2_ISR)	R/W	0x0000_0000	*Section 101.4.1.3*
0x0104	BD_SIUL2 Reset Status Register (BD_SIUL2_RSR)	R	0x0000_0000	*Section 101.4.1.4*

Table 2439. BD_SIUL2 register summary (continued)

Offset	Register	Access	Reset Value	Section/Page
0x0240	BD_SIUL2 Multiplexed Signal Configuration Register 1 (BD_SIUL2_MSCR1)	R/W	0x0001_0000	*Section 101.4.1.5*
0x0244	BD_SIUL2 Multiplexed Signal Configuration Register 2 (BD_SIUL2_MSCR2)	R/W	0x0001_0000	*Section 101.4.1.5*

101.4.1 Register descriptions

This section describes in address order all the BD_SIUL2 registers. Each description includes a standard register diagram with an associated figure number. Details of register bit and field function follow the register diagrams, in bit order.

Note: All registers support 32-bit accesses only.

101.4.1.1 BD_SIUL2 Device Identification Register 1 (BD_SIUL2_DIDR1)

The BD_SIUL2_DIDR1 contains the part number and mask number.

Offset 0x00 Access: User read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	PARTNUM															
W																
Reset	0	1	0	1	0	1	1	1	1	0	1	1	1	1	0	1

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	MASKNUM_MAJOR				MASKNUM_MINOR			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 2558. BD_SIUL2 Device Identification Register 1 (BD_SIUL2_DIDR1)

Table 2440. BD_SIUL2_DIDR1 Bit Field Descriptions

Field	Description
0:15 PARTNUM	Device Part Number Read-only, mask programmed part number of the device. Reads 0x57BD.
16:23	Reserved.
24:27 MASKNUM_MAJOR	Major Mask Revision Number Read-only, mask programmed mask number of the device. Reads 0x0 for the initial mask set of the device, and changes sequentially for each mask set.
28:31 MASKNUM_MINOR	Minor Mask Revision Number Read-only, mask programmed mask number of the MCU. Reads 0x0 for the initial mask set of the device, and changes sequentially for each mask set.

101.4.1.2 BD_SIUL2 Device Identification Register 2 (BD_SIUL2_DIDR2)

The BD_SIUL2_DIDR2 contains a complement to the part number and amount of RAM available.

Offset 0x04 Access: User read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	RAM_SIZE			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	PARTNUM								0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0

Figure 2559. BD_SIUL2 Device Identification Register 2 (BD_SIUL2_DIDR2)

Table 2441. BD_SIUL2_DIDR2 Bit Field Descriptions

Field	Description
0:11	Reserved.
12:15 RAM_SIZE	RAM Size Read-only, indicates the amount of overlay RAM available in the device, with resolution of 256K byte steps 0000 0.25M byte 0001 0.5M byte 0010 0.75M byte 0011 1M byte 0100 1.25M byte 0101 1.5M byte 0110 1.75M byte 0111 2M byte 1000 2.25M byte 1001 2.5M byte 1010 2.75M byte 1011 3M byte 1100 3.25M byte 1101 3.5M byte 1110 3.75M byte 1111 4M byte
16:23 PARTNUM	Device Part Number Read-only, mask programmed complement to the part number of the device. For the BD 1M byte part with part number 57BD1, BD_SIUL2_DIDR1.PARTNUM=57BD, and BD_SIUL2_DIDR2.PARTNUM=1. 01 1 (BD1M) 02 2 (BD2M)
24:31	Reserved.

101.4.1.3 BD_SIUL2 Initialization Status Register (BD_SIUL2_ISR)

The BD_SIUL2_ISR contains a flag that can be used by the PD or any connected debug or calibration tool to track the status of the BD. This register is reset by POR.

The main expected use case for this flag is to allow software on the PD to determine whether the BD has lost power since it was last initialized. A typical sequence is:

- At power on reset (POR), the flag is cleared automatically (not affected by software reset).
- A calibration tool or PD calibration software initializes the BD (for example, writing BD overlay RAM with calibration data and remap configuration).
- The calibration tool or PD calibration software sets the BD_INIT flag
- After PD POR, the PD boot software executes and reads BD_INIT.
  - If BD_INIT is clear, BD overlay is assumed to be uninitialized and is not used for calibration remap.
  - If BD_INIT is set, BD overlay is assumed to be initialized and can be used for calibration remap (other tests to verify data integrity may be used).

BD_INIT is cleared by BD POR, and can be set or cleared via writes from the PD or from the Nexus RWA.

Offset 0x100 Access: User read/write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	BD_INIT
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 2560. BD_SIUL2 Initialization Status Register (BD_SIUL2_ISR)

Table 2442. BD_SIUL2_ISR Bit Field Descriptions

Field	Description
0:30	Reserved.
31 BD_INIT	Initialization Flag This is a read/write bit that can be used to indicate that the BD was initialized. Software reset does not affect the flag. Power-on reset clear the flag. 0 BD not initialized. 1 BD initialized.

101.4.1.4 BD_SIUL2 Reset Status Register (BD_SIUL2_RSR)

The BD_SIUL2_RSR contains reset status information about the PD.

Offset 0x104 Access: User read

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	PD_RST	PD_TRIM	PD_POR
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 2561. BD_SIUL2 Reset Status Register (BD_SIUL2_RSR)

Table 2443. BD_SIUL2_RSR Bit Field Descriptions

Field	Description
0:28	Reserved.
29 PD_RST	PD Reset Flag Read-only, indicates if PD has finished the reset sequence. 0 PD reset sequence not completed. 1 PD reset sequence completed.
30 PD_TRIM	PD Trim Flag Read-only, indicates if PD has finished trimming phase during the reset sequence. 0 PD trimming phase not completed. 1 PD trimming phase completed.
31 PD_POR	PD POR Flag Read-only, indicates whether PD is powered down or not. 0 PD is powered down or power on reset has not completed. 1 PD is powered and power on reset has completed.

101.4.1.5 BD_SIUL2 Multiplexed Signal Configuration Registers (BD_SIUL2_MSCR1, BD_SIUL2_MSCR2)

The BD_SIUL2_MSCR1–2 registers contain pad multiplexing and control bits. It is reset by system reset.

The debug interface on the BD requires five pins when operating in JTAG 1149.1 and LFAST modes, but can be reduced to three pins when operating in JTAG 1149.7 reduced pin mode, allowing the use of two pins for GPIO. The BD_SIUL2_MSCR registers provides bits to control pad electrical characteristics when the pins are working as GPIO. Note that the registers do not control the multiplexing, but the electrical characteristics for GPIO only. Multiplexing is controlled by the function of the pads. If the debug tool is disconnected and JCOMP negated, the pads are GPIO. When the tool is connected and JCOMP asserted, the pads are automatically assigned to JTAG. If the tool then configures JTAG to work in 1149.7 mode, the pads are assigned back to GPIO. And if the tool selects LFAST mode, the pads are assigned to LFAST.

Note: Multiplexing of GPIO into JTAG pins is not supported.

Offset BD_SIUL2_BASE + 0x240
BD_SIUL2_BASE + 0x244

Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	OERC		0	0	ODC		0	0	0	0	IBE	HYS	WPDE	WPUE
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

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

Figure 2562. BD_SIUL2 Multiplexed Signal Configuration Registers (BD_SIUL2_MSCR1, BD_SIUL2_MSCR2)

Table 2444. BD_SIUL2_MSCR1/2 Bit Field Descriptions

Field	Description
0:1	Reserved.
2:3 OERC	Output Edge Rate Control 00 Minimum slew rate 01 Medium slew rate 10 Invalid value 11 Maximum slew rate
4:5	Reserved.
6:7 ODC	Output Drive Control 00 Output buffer disabled 01 Open drain. Invalid value, not supported by this pad (reverts to “output disabled”) 10 Push-pull. Output buffer enabled 11 Open source. Invalid value, not supported by this pad (reverts to “push-pull”)
8:11	Reserved.
12 IBE	Input Buffer Enable 0 Input buffer disabled 1 Input buffer enabled
13 HYS	Input Hysteresis 0 Hysteresis disabled 1 Hysteresis enabled
14 WPDE	Weak Pull Down Enable This pad does not allow pull down to be set at the same time as pull up. If both are set, only pull up is enabled. 0 Pull down disabled 1 Pull down enabled

Table 2444. BD_SIUL2_MSCR1/2 Bit Field Descriptions (continued)

Field	Description
15 WPUE	Weak Pull Up Enable This pad does not allow pull down to be set at the same time as pull up. If both are set, only pull up is enabled. 0 Pull up disabled 1 Pull up enabled
16:31	Reserved.

102 Buddy Device: Debug and Calibration Interface (DCI)

102.1 Introduction

The Debug and Calibration Interface (DCI) module provides debug and calibration features for the MCU. It includes a standard 1149.1/1149.7 compatible JTAG interface which is used to connect with an external JTAG tool using a low frequency clock interface.

To provide high-speed calibration capability, the DCI also provides a mechanism to use 5-pin LFAST debug tool which shares its pins with the standard JTAG interface.

It also features a software based debug mode which can be controlled by the MCU without using an external tool. Additionally, the DCI provides features for debug control using break-points and synchronous restart of the cores. This chapter describes the DCI features in the Emulation and Debug Device (ED) that includes features in the Production device (PD) as well as features in the Buddy device (BD). There are two DCIs on the ED which are designated PD DCI and BD DCI.

102.1.1 Features

The ED DCI features are the following:

- Debug mode enable control for connected tool or software
- Dual DCI system for multi-device clients
- Port sharing logic to allow the 5-pin debug port to be shared between the JTAG and the LFAST
- 1149.1 and 1149.7 controllers
- Debug break and cross-triggering control
- Synchronous restart control for all CPUs when exiting debug mode
- Tool hot plug capability
- Security access control
- Debug reset control

102.1.2 Overview

The DCI can take the following different inputs for its JTAG signals:

- P.JTAG—JTAG signals coming from the Buddy device JTAG pads. This is the standard 5-pin JTAG interface and is the default operating mode.
- M.JTAG—JTAG signals generated by JTAGM module using LFAST or software debug mode.

  - When using LFAST with a connected calibration/debug tool, JTAG packets are received from the tool via LFAST and output JTAG packets are transmitted back to the tool. The LFAST's 5-pin interface consists of an LVDS pair for transmit signals, an LVDS pair of receive signals and a reference clock output signal.
  - In software debug mode, there is no tool, and debug software running on the MCU generates the JTAG packets and writes them to JTAGM using special debug registers in the JTAGM module.
- I.JTAG— JTAG signals interconnecting the two DCI implementations in the ED coming from the intra-die pads.

By default the DCI operates in standard 5-pin JTAG mode (IEEE 1149.1). It can be configured in 3-pin reduced pin JTAG mode (IEEE 1149.7) after starting from standard 5-pin JTAG mode. The LFAST-based debug interface shares its five pins with the standard JTAG pins.

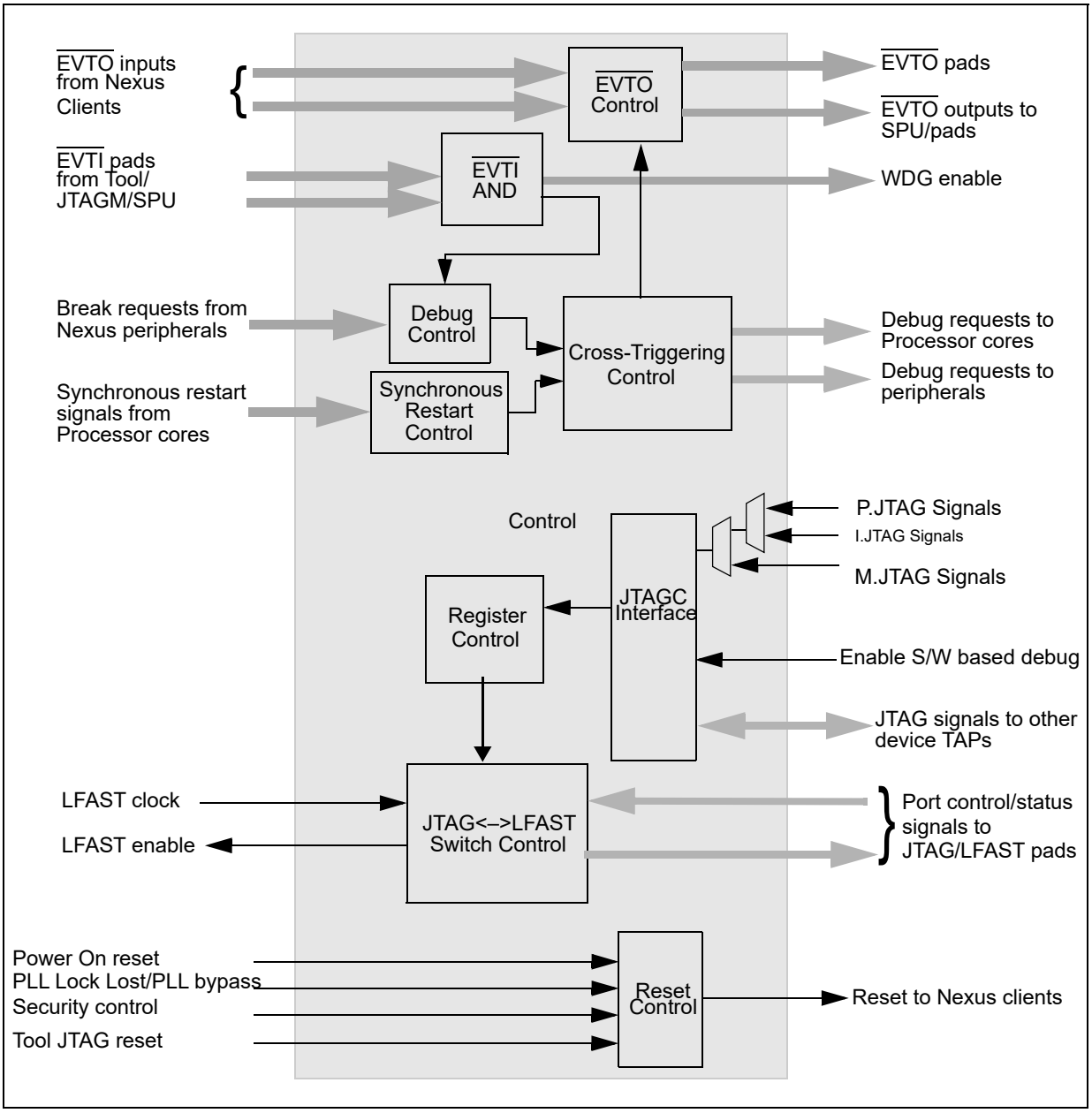
When reduced pin JTAG mode (IEEE 1149.7) is configured TMSC pin is configured in bidirectional mode during operation.

As the LFAST-based debug interface shares its five pins with JTAG pins, the DCI provides a safe switching logic to switch to LFAST-based debug mode.

The ED DCI also provides a centralized break control for system IPs and cores, which can be controlled by an external tool as well as the internal debug peripherals such as SPU. The ED DCI also provides the Event Out ($\overline{\text{EVTO}}$) signals to the debug tool in case of the occurrence of any internal debug event.

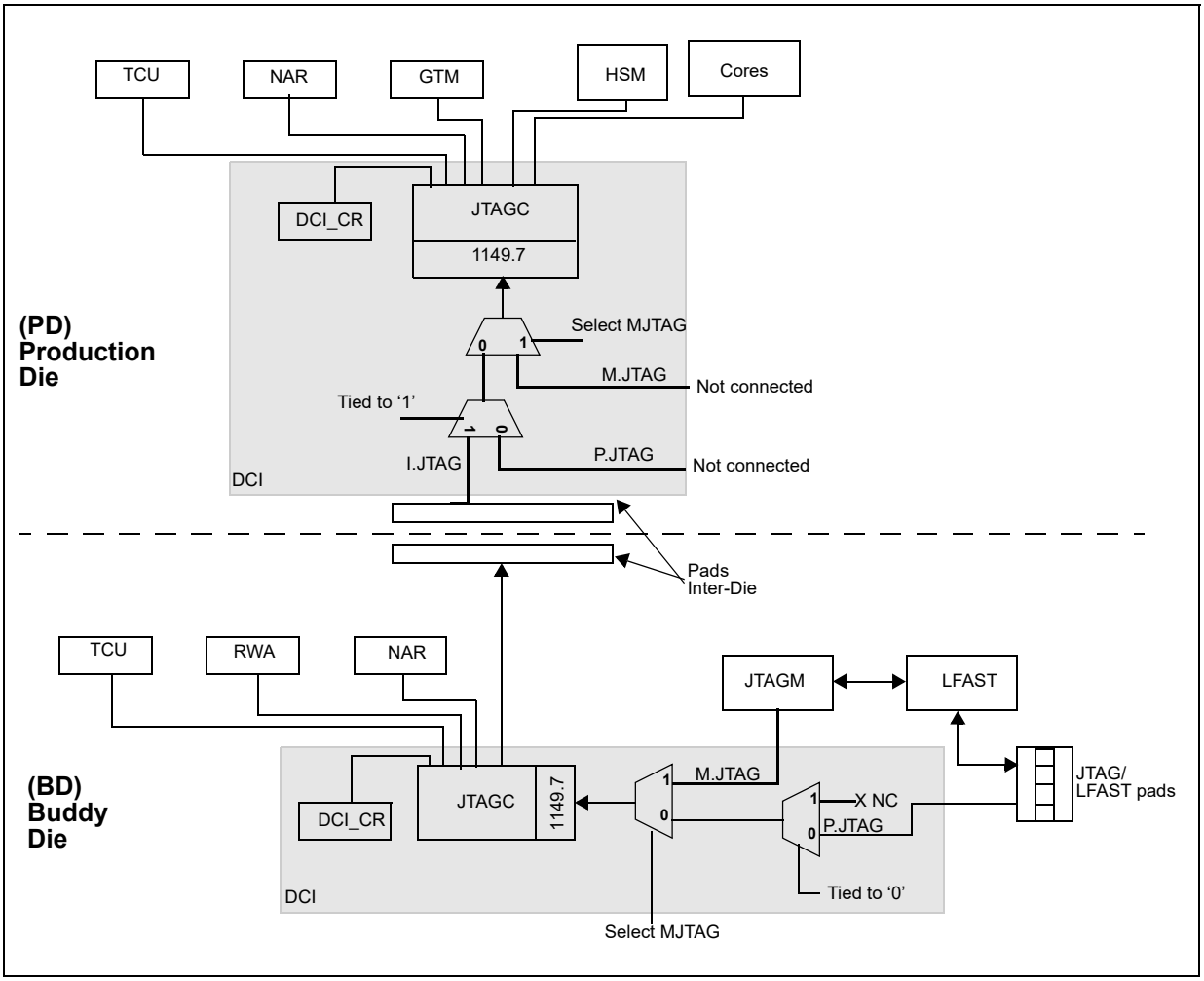
Figure 2563 shows the DCI block diagram.

Figure 2563. DCI block diagram



When the DCI is used in an ED package (PD and the BD together) its configuration is shown as in *Figure 2564*. In the ED package, a debug tool is connected to the BD pins and connects to the PD with the I.JTAG interface which are routed using inter-die pads.

Figure 2564. DCI in an emulation configuration



102.1.3 Operating modes

Selection between various JTAG sources is done using a JTAG register bit, DCI_CR[EN_LFAST], and a JTAGM register bit, JTAGM_MCR[DTM]. The JTAG bit, DCI_CR[EN_LFAST], can only be programmed using JTAG instructions and is used to select standard JTAG or LFAST debug tool. The JTAGM bit, JTAGM_MCR[DTM], is programmable via memory-mapped registers. JTAGM is selected as a JTAG source when either DCI_CR[EN_LFAST] or JTAGM_MCR[DTM] is set.

Table 2445 shows a summary of the configuration options.

Table 2445. DCI JTAG source selection

DCI_CR[EN_LFAST]	JTAGM_MCR[DTM]	DCI Source	JCOMP
0	0	P.JTAG (Standard JTAG)	Asserted
1	x	M.JTAG (LFAST)	Asserted
x	1	M.JTAG (Software)	Negated

The reset signal to the DCI block consists of the following:

- Power-on reset, Low/High voltage detection on IO/Core supplies
- JCOMP reset

The DCI puts the JTAGC TAP in reset when either power-on/low-voltage-detect reset is asserted or JCOMP is negated. In the ED, the BD DCI is reset with BD POR/Low voltage detection on BD_VDD.

The M.JTAG provides an alternate interface to the debug logic.

When enabled, it allows high speed debug through the BD LFAST and BD JTAGM modules. The high speed link uses LFAST commands that are translated to JTAG commands in the JTAGM module. The selection between LFAST based debug or software based debug is managed in the JTAGM. However, it is important to note that the dynamic switching between tool-based debug and software debug is not allowed.Software based debug is done only in those cases when an external tool is not present.

An LFAST based tool starts in JTAG mode and then switches to LFAST mode. This switching is managed through a control set of operations which is described in *Section 102.1.3.1: Port sharing between JTAG and LFAST modes.* During the switch from JTAG to LFAST and back, the DCI uses a latched value of JCOMP to prevent all debug configuration reset.

The LFAST uses LVDS ports for LFAST communication and the JTAG ports uses CMOS level GPIOs. These signals are double-bonded to share the same pins at the tool interface. *Table 2446* shows the port mapping.

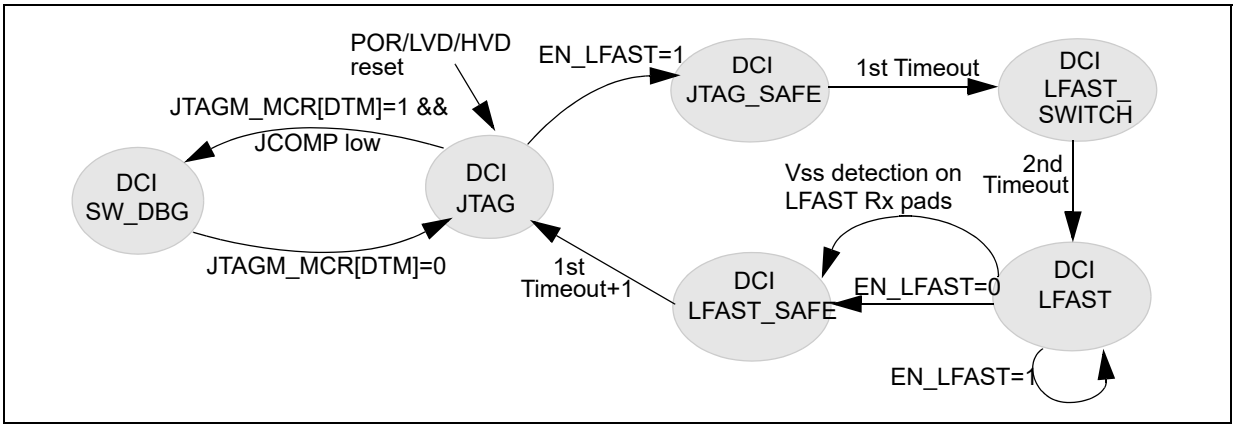
When no debug tool is connected, device power-on reset puts the DCI in standard 5-pin JTAG mode. This configures the TDO pin in output mode with the reset value (1’b0). There is no change in port values if no debug tool is connected.

Table 2446. DCI LFAST/JTAG port sharing

JTAG pad	Type in JTAG mode	LFAST pad	Type in LFAST mode
TDI	Input	LVDS TxN	Output
TMS	Input	LVDS TxP	Output
TDO	Output	LVDS RxP	Input
JCOMP	Input	LVDS RxN	Input
TCK	Input	DRCLK	Output

Figure 2565 shows the DCI mode switching operation.

Figure 2565. DCI operating modes state diagram



Note: JTAGM_MCR[jtagm_JCOMP] should not be asserted in LFAST mode to avoid any undesired behavior.

Table 2447 contains the operating mode descriptions.

Table 2447. DCI operating modes

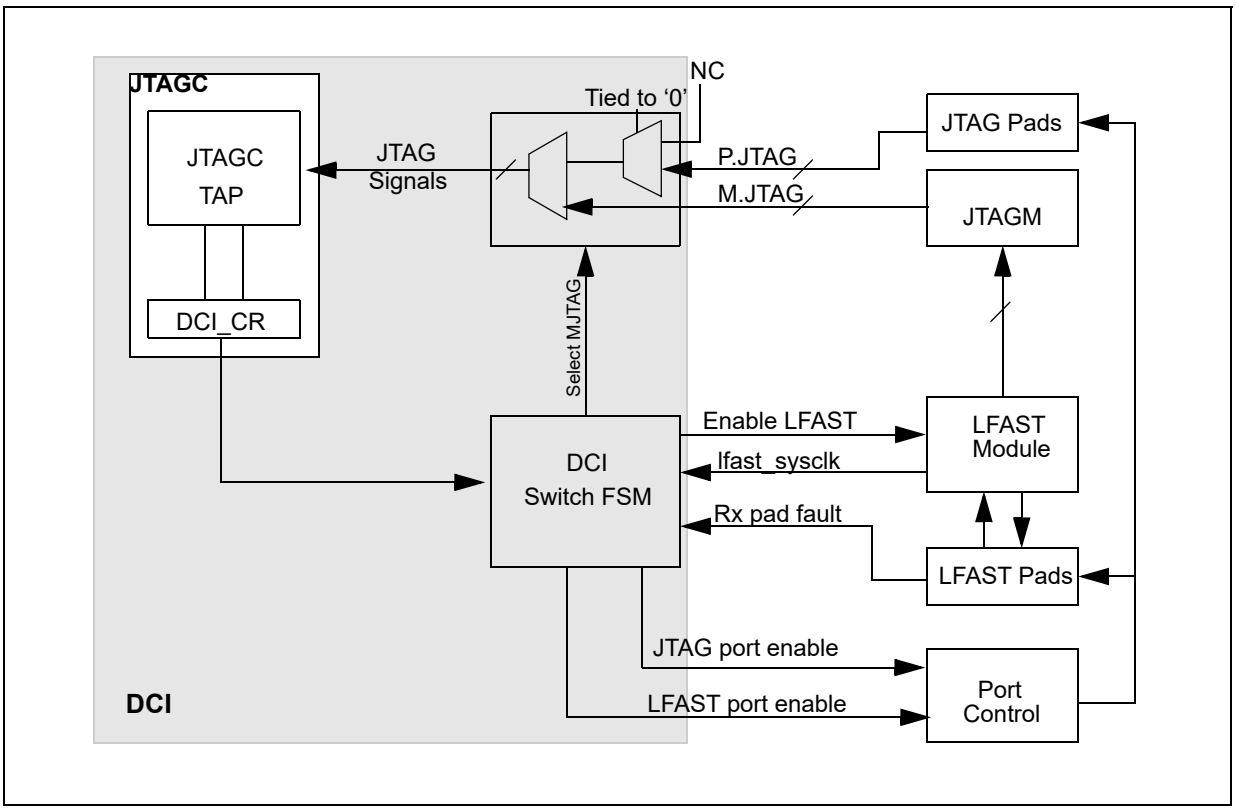
Mode	Description
DCI_JTAG	JTAG mode. This is the default state after power-on reset. In this mode the DCI is in JTAG mode (controlled with P.JTAG or I.JTAG) and is controlled by an external JTAG tool.
DCI_SW_DBG	Software Debug State. This state is achieved when no tool is present and software sets the JTAGM_MCR[DTM] bit. The DCI is then controlled using M.JTAG signals.
DCI_JTAG_SAFE	SAFE_JTAG state. When an external tool sets the DCI_CR[EN_LFAST] bit, this indicates that the external tool is going to switch to the LFAST mode. The external tool puts the DCI in Run-Test-Idle state to start the switching operation. To enable safe switching operation, the DCI enters the JTAG_SAFE state where it performs the following operations: – JCOMP is latched inside to control until switching gets completed – The DCI releases control of the JTAG ports and uses high on TMS inside, thus keeping the DCI in the Run-Test-Idle state – An internal counter starts to count to a tool specified value as programmed in DCI_CR[SWITCH_CNTR1].
DCI_LFAST_SWITCH	LFAST port switching state. In this state, the DCI asserts the enable signal to indicate to the LFAST that it is now ready for LFAST port switching. In this mode, the LFAST Rx pads are enabled. The switch counter starts another count operation to count to the DCI_CR[SWITCH_CNTR2] value.

Table 2447. DCI operating modes (continued)

Mode	Description
DCI_LFAST	LFAST mode with escape mode enabled. This mode indicates that the switching is now completed and the debug tool is operating in the LFAST mode. However, there is another feature called Escape Mode, which enables the LFAST pads to detect any disconnection fault on its Rx pads. When these pads are working in Escape Mode, a sensing logic is enabled in the pad which detects and flags any out of common mode voltage range for the Rx pads. If this flag is asserted, it is used to force the DCI to go to LFAST_SAFE mode to ensure safe switching to the JTAG state.
DCI_LFAST_SAFE	LFAST_SAFE mode. While the tool is switching from LFAST mode to JTAG mode, it resets the DCI_CR[EN_LFAST] bit. The external tool puts the DCI in the Run-Test-Idle state to start the switching operation. To account for the switching time for the pads, the DCI enters in a safe LFAST state. It then restarts a counter to count to a tool specified value in DCI_CR[SWITCH_CNTR1], and disables both the LFAST function on the pads and the LFAST module.

Figure 2566 shows a block diagram of the JTAG/LFAST connections.

Figure 2566. DCI JTAG/LFAST switch integration



102.1.3.1 Port sharing between JTAG and LFAST modes

By default the DCI is configured in JTAG mode using the standard JTAG port (P.JTAG). Therefore, the initial configuration is done using low speed JTAG signals. In order to switch

from JTAG mode to LFAST mode, the tool accesses the following fields in the DCI control register (DCI_CR):

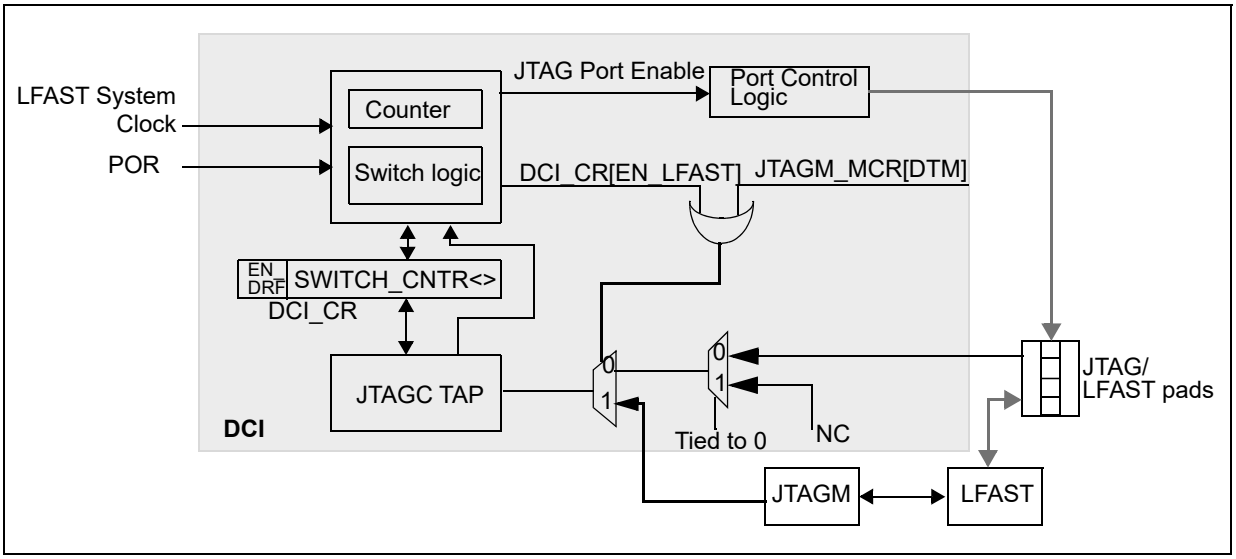
- EN_LFAST—Bit to indicate enable/disable LFAST mode
- SWITCH_CNTR1 and SWITCH_CNTR2—Bit fields to program a count value that corresponds to the delay for switching the pin functions when changing modes.

As the JTAG to LFAST mode switching reflects change in port input/output behavior, it is done through a controlled procedure.The DCI_CR is only accessible through JTAG sequences. While switching from JTAG to LFAST mode, the tool programs this register via P.JTAG to enable switching to LFAST mode. As the pins are shared between JTAG and LFAST functions, a programmable delay based on switch counters is required which allows a safe switching for the MCU as well as for the tool to change its mode. In LFAST mode, the tool can write this register to switch back from high speed debug to low speed debug. Again, the similar switching delay mechanism is used. The switching schematic is shown in *Figure 2567*.

A programmable switching delay and tri-stating port pins during switching allow a connected tool to safely tune the device turn-around of pin functions from JTAG mode to LFAST mode and vice versa. These delays are programmed according to the time required by the tool to change its operation from JTAG mode to LFAST and vice versa. This reduces the possibility of device damage during switching.

The programmable counter uses the LFAST system clock as the clock source.

Figure 2567. DCI switching between JTAG and LFAST modes



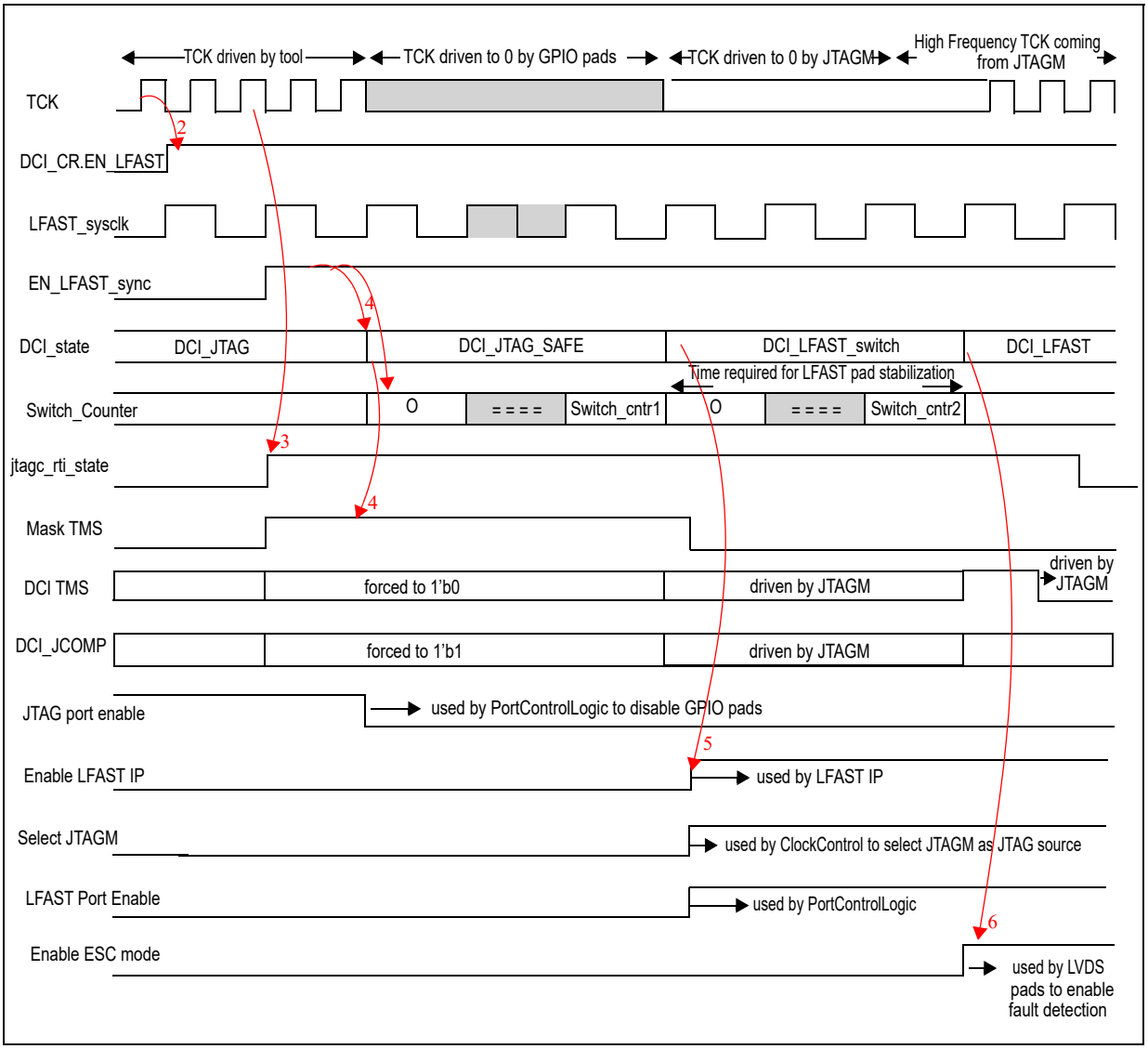
102.1.3.1.1 Switching from JTAG mode to LFAST mode

The steps to switch operation from JTAG to LFAST mode are as follows:

1. System starts in port JTAG mode after power-on-reset.
2. External tool programs the following bits in the DCI_CR:
   – EN_LFAST
   – SWITCH_CNTR1
   – SWITCH_CNTR2
3. Tool puts JTAG TAP in the Run-Test-Idle mode.
4. An internal finite state machine (FSM) starts the switching operation using the LFAST reference clock. The switching process takes four LFAST reference clocks to start, and an external tool should not change the JTAG signals during this period. The DCI forces TMS to 0 and JCOMP to 1 internally so as not to change JTAG TAP status during switching. The DCI also triggers a switching counter to allow for some time for the tool to change its mode from JTAG to LFAST.
5. After the counter counts to the DCI_CR[SWITCH_CNTR1] value, the LFAST IP and LFAST pads are enabled. Then the counter is reset. Also, internally JTAGM is selected as the JTAG source for the DCI.
6. Counter counts to the DCI_CR[SWITCH_CNTR2] value to enable the Escape mode feature for the LFAST Rx pads. This time allows the tool to settle down in LFAST mode.
7. Tool starts LFAST transmission when it detects pulses on the DRCLK pin. Tool now can use the LFAST to program the JTAGM to generate JTAG sequences.

Figure 2568 shows the timing sequence.

Figure 2568. DCI JTAG to LFAST switching diagram



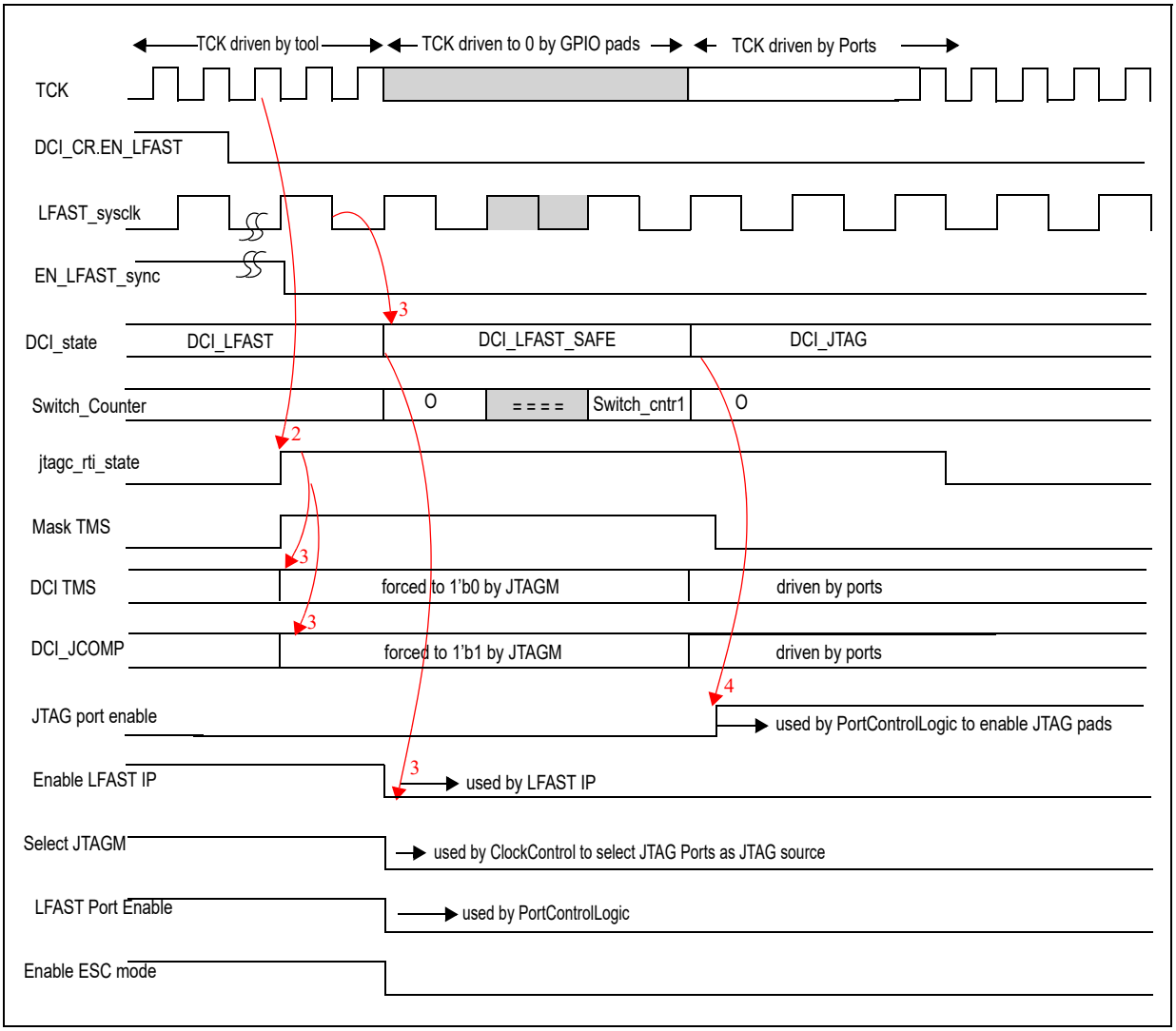
102.1.3.1.2 Switching from LFAST mode to JTAG mode

The steps to switch operation from LFAST to JTAG mode are as follows:

1. External tool programs the DCI_CR to reset the EN_LFAST bit and SWITCH_CNTR1 for the switching delay.
2. Tool puts JTAG TAP in the Run-Test-Idle mode.
3. An internal FSM starts switching operation. It forces TMS to 0 and JCOMP to 1 internally so as not to change JTAG TAP status during switching. It also triggers a switching counter to count to DCI_CR[SWITCH_CNTR1] to allow for the tool to change its mode from LFAST to JTAG.
4. After the counter overflows, the JTAG pads are enabled and JTAGM releases its control for the DCI JTAG source.

Table 2451 and *Table 2453* summarize the MCU and tool behaviors. *Figure 2569* shows the timing sequence.

Figure 2569. DCI LFAST to JTAG switching diagram



102.1.3.2 VSS detection

When the LFAST pads are operating in Escape mode, a sensing logic is enabled in the pad which detects and flags any disconnection of the LFAST Rx pads. This flag, if asserted, is used to force the DCI to go to JTAG mode to ensure safe operation. This flag, when asserted, also resets the JTAG TAP to restart in JTAG mode and the EN_LFAST bit is reset accordingly. This flag resets only JTAG TAPs and does not affect any Nexus blocks.

This feature is enabled only when DCI_CR[EN_ESC_MODE] is set. By default this feature is not enabled. To tune the tool behavior it is possible to emulate the pad fault using the DCI_CR[FLVDS_PAD_FLT] bit.

102.1.3.3 Software enabled debug and calibration mode

Apart from the JTAG sources, such as a JTAG-based tool or LFAST-based tool, it is possible to generate JTAG sequences through software programming using JTAGM IP. The JTAGM, which generates JTAG sequences using LFAST protocol, is also used to generate JTAG sequences using software programming.

To support a software debug session via a CAN or SPI link, the software can enable debug and calibration mode via an memory-mapped register. It is done first by setting a control bit in the JTAGM (JTAGM_MCR[DTM]). For more details, refer to the JTAGM documentation. When enabled, the DCI is configured to receive M.JTAG signals which are driven by JTAGM registers accessible through software. Software can control all DCI features like any other JTAG source.

After a system reset, the JTAGM_MCR[DTM] bit is reset and the JCOMP pad is pulled-low, thus keeping the DCI in the reset state. Therefore, software is required to first change to a non-secured mode. Then it can program the JTAGM_MCR[DTM] bit, to select M.JTAG as the source inside the DCI, and then set the JTAGM_MCR[jtagm_JCOMP] bit present in the JTAGM, which takes all debug modules out of reset.

When MCR[jtagm_JCOMP] is programmed to 0, the JTAGC block reset is asserted, thus putting the debug and calibration logic in reset state. This bit is used during software based debug to initialize the debug and calibration logic.

Software based debug is not possible when a tool is connected. The presence of the tool is determined using the rising edge of the JCOMP pad to set an internal flag which, when set, disables the selection of JTAGM as a JTAG source.

Note: It is recommended to utilize either hardware based debug (JTAG or LFAST) or software based debug. If only one of the debug method is used, all three connection methods (connect and reset target, connect and break application, connect and keep application running) are supported.

If both debug methods are used simultaneously, then an external tool may arbitrarily connect when the SoC has already been halted using the software debug mode. In this case, two consecutive assertions of external JCOMP are necessary to allow the external tool to take control of debug session.

102.1.3.4 Nexus reset control

The JCOMP input that is used as the primary reset signal for the DCI is also used by the DCI to generate a single-bit reset signal for other Nexus blocks. It is used to reset all debug functions without affecting any system level functions. The DCI provides the reset signal for all Nexus blocks, with the PD DCI providing the reset signal to all PD debug blocks, and the BD DCI providing the reset signal for all BD debug blocks. This reset is controlled using the following signals:

- Power-on reset/Low voltage detect (applicable to both PD and BD)
- JCOMP (applicable to both PD and BD)
- SoC Debug_Enable signal (applicable to PD only)
- System PLL is out of lock state and not bypassed (applicable to PD only)

Asserting power-on reset, negating JCOMP, or negating the Debug_Enable signal, results in asynchronous entry into reset state. Following negation of power-on reset, the BD Nexus reset signal remains asserted until the JCOMP signal is asserted. Any subsequent loss of PLL lock does not generate a Nexus reset. This reset signal is unaffected by other sources of reset.If Debug_Enable is de-asserted by the SoC security modules, then the DCI holds the debug and trace logic in reset. The Debug_Enable signal is generated by security logic after functional or destructive reset based on the device censored mode, the PASS_LOCK3[DBL] control, and the device life cycle. If the Debug_Enable signal is asserted by the security logic, it is possible to trace through functional resets

The following table shows how the Nexus reset is asserted/negated based on various signals.

Table 2448. Nexus reset truth table

POR	JCOMP	Debug_Enable	PLL lock	Nexus reset
Assert	X	X	X	Assert
X	Negate	X	X	Assert
X	X	Disable	X	Assert
X	X	X	Unlocked	Assert
Negate	Assert	Enable	Locked	Negate

102.1.3.5 System watchdog control

The BD DCI does not include the Software Watchdog Timer (SWT) disable feature

102.1.3.6 Reset control

The BD DCI includes control bits to allow a connected tool to force a reset of the connected device. The description is as follows:

- Force BD Reset—This reset is generated by programming DCI_CR[FBD_RESET] bit. When set it forces reset. This bit has to be reset by the tool to negate the reset signal.

These reset signals are directly connected to the respective DCI_CR bits. The DCI does not reset the DCI_CR bits after generation of the respective reset signals. The tool must specifically reset the appropriate DCI_CR bits to remove the reset signals. These reset control bits can be set/reset through JTAG programming, and only reset through JCOMP assertion or Power-on-reset.

102.1.3.7 Security

The SoC security logic provides a Debug_Enable signal to the PD DCI. The PD DCI uses the Debug_Enable signal as a control input to generate debug module reset. When debug disable is requested, the PD DCI keeps all debug blocks in the reset state. This is a protection mode used to disable debug accesses to the peripheral when the device is in a secured state.

102.2 External signal description

The JTAGC module used inside DCI module defines the interface with external signals. It consists of five signals (TMS/TDI/TDO/TCK/JCOMP) that connect to off chip development tools and allow access to test support functions. For more details for JTAG signals, refer to JTAGC documentation.

102.3 Register description

The PD DCI module contains two registers, DCI_CR and DCI_PINCR, which are present inside the JTAGC module. The BD DCI only contains the DCI_CR and does not contain the

DCI_PINCR. These registers are only controlled using the JTAGC TAP and no memory-mapped registers are present.

102.3.1 DCI control register (BD_DCI_CR)

The BD_DCI_CR is a 32-bit data register. This register is programmed using JTAG shift with the corresponding enable instruction described in the JTAGC chapter. *Figure 2570* shows the format of the BD_DCI_CR.

Note: The BD version of this register is refered to as DCI_CR in the other sections of this chapter.

In an ED which has both PD and BD instances of the DCI, all bits controlling the switch between LFAST and JTAGM must be programmed on the BD DCI instance. The only bits of this register that do not follow this rule are EVTI1_PAD_EN and EVTI0_PAD_EN, which must be programmed in both DCI instances.

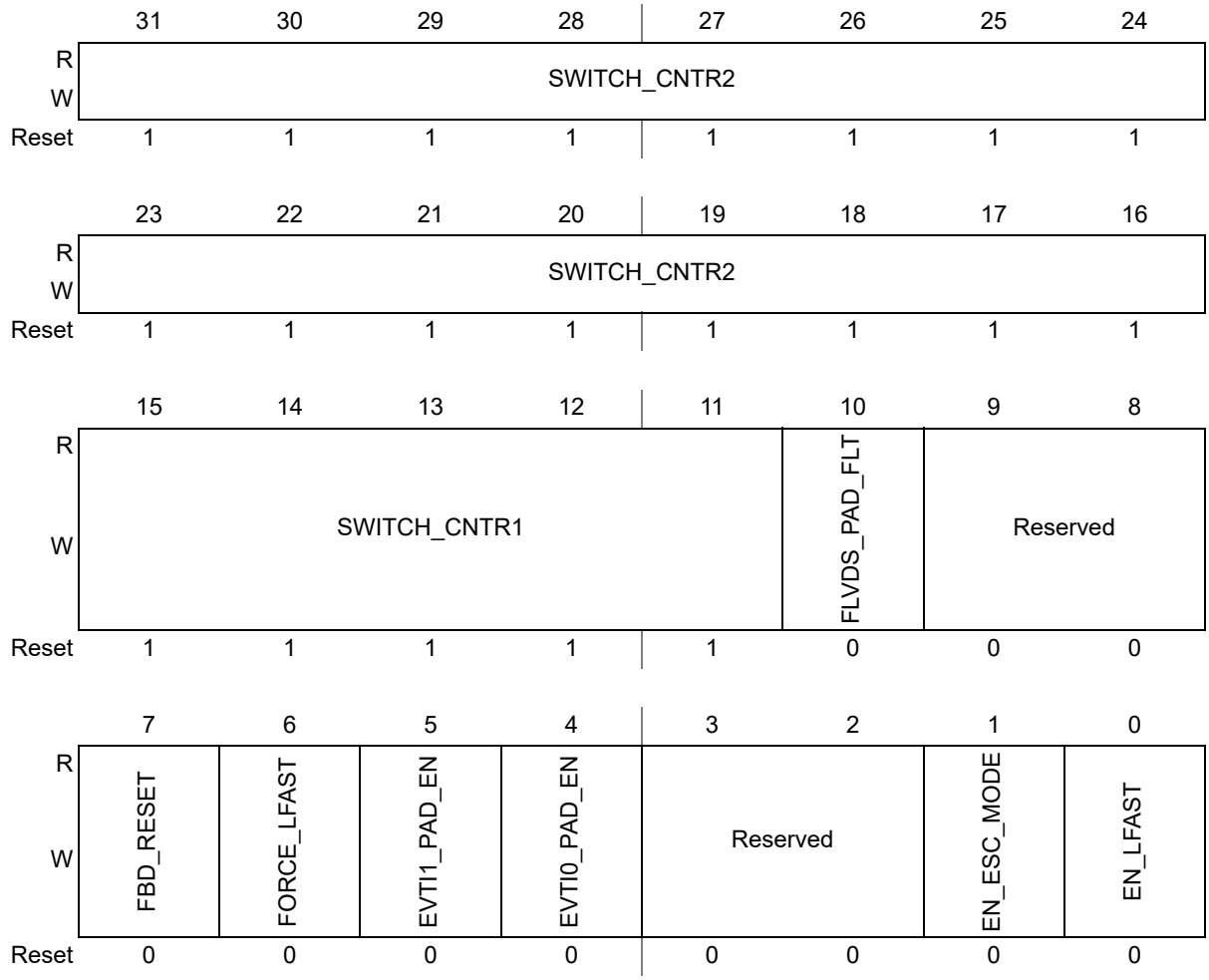


Figure 2570. DCI control register (DCI_CR)

The DCI_CR is described in *Table 2449*.

Table 2449. DCI_CR field descriptions

Field	Description
31:16 SWITCH_CNTR2	Width of Switch_CNTR2 Decides the second timeout value while switching from JTAG to LFAST mode. The maximum timeout duration = 32,768×T_{lfast_sysclk}
15:11 SWITCH_CNTR1	Width of Switch_CNTR1 Decides the first timeout value while switching from JTAG to LFAST mode, and the timeout value while switching from LFAST mode to JTAG mode. The maximum timeout duration = 32×T_{lfast_sysclk}
10 FLVDS_PAD_FLT	Force LVDS Pad Fault Forces the pad fault condition to tune the tool without actually having the pad fault on LFAST Rx pads. This bit is effective only when DCI_CR[EN_ESC_MODE] is set. 0 Do not force Rx LVDS Pad Fault 1 Force Rx LVDS Pad Fault if DCI_CR[EN_ESC_MODE] is set
7 FBD_RESET	Force Reset 0 Do not force Reset 1 Force Reset
6 FORCE_LFAST	Force enable for LFAST 0 LFAST not forced to enable and is controlled using DCI_CR[EN_LFAST] bit 1 LFAST forcefully enabled (without using DCI_CR[EN_LFAST] bit)
5 EVTI1_PAD_EN(1)	$\overline{\text{EVTI}}$[1] feature enable 0 $\overline{\text{EVTI}}$[1] disabled (default) 1 $\overline{\text{EVTI}}$[1] enabled
4 EVTI0_PAD_EN(2)	$\overline{\text{EVTI}}$[0] feature enable 0 $\overline{\text{EVTI}}$[0] disabled (default) 1 $\overline{\text{EVTI}}$[0] enabled
1 EN_ESC_MODE	Enable Escape Mode Fault detect feature enable for LFAST Rx pads 0 Fault detect feature not present for LFAST Rx pads (default) 1 Fault detect feature present for LFAST Rx pads and are enabled after the successful switching to LFAST mode
0 EN_LFAST(2)	Enable for LFAST-based tool selection This bit can be reset either by programming by tool, or pad fault condition in which JTAGC block is reset. 0 LFAST tool not enabled. DCI operates in JTAG mode (default) 1 LFAST tool enabled

1. In an ED that has both PD and BD, EVTI1_PAD_EN and EVTI0_PAD_EN must be programmed in both BD and PD instances of the DCI in order to allow events generated on the BD to pass through the BD DCI and reach the PD DCI.
2. DCI_CR is a JTAG data register and is not accessible for software read. However, the DCI_CR[EN_LFAST] bit is mapped to JTAGM register bit JTAGM_SR[1] to allow for software read access and a mechanism for software to be aware of an active debug tool interface.

102.3.2 DCI EVTx pin multiplexing control register (DCI_PINCR)

The DCI_PINCR is a 32-bit data register. This register is programmed using JTAG shift with the corresponding enable instruction described in the JTAGC chapter. For a description of the DCI_PINCR functionality, see the Calibration and Debug configuration chapter (or the Debug and Trace chapter) which describes how the modules are configured.

102.4 Functional description

102.4.1 DCI mode transition

After power-on reset, the BD DCI starts in JTAG mode. To enable LFAST mode, the external tool has to first operate in JTAG mode, configure DCI and then switch to LFAST mode. This requires certain careful steps so as to avoid any contention on double-bonded pads for JTAG/LFAST. *Table 2450* and *Table 2451* show the steps and expected states for the MCU and tool for JTAG to LFAST switching. *Table 2452* and *Table 2453* contains the different steps, and the MCU and tool expected states for the LFAST to JTAG switching.

Table 2450. MCU operation (JTAG to LFAST switching)

MCU pin	JCOMP low	JCOMP high (JTAG mode)	DCI_CR [EN_LFAST]=1	Run-Test-Idle state	1st Timeout	2nd Timeout	LFAST ICLC enable command received
TDI	Logic Input	Logic Input	Logic Input	Hi-Z	Hi-Z	Hi-Z	LVDS TxN
TMS	Logic Input	Logic Input	Logic Input	Hi-Z	Hi-Z	Hi-Z	LVDS TxP
TDO	Hi-Z	Logic Output	Logic Output	Hi-Z	LVDS Rx	LVDS Rx	LVDS RxP
JCOMP	Logic Input	Logic Input	Logic Input Latch	Hi-Z	LVDS Rx	LVDS Rx	LVDS RxN
TCK	Logic Input	Logic Input	Logic Input	Hi-Z	Logic Output	Logic Output	Logic Output
Mode Escape	Inhibit	Inhibit	Inhibit	Inhibit	Inhibit	Enable	Enable
LFAST_ SYSCL K _EN	Inhibit	Inhibit	Inhibit	Inhibit	Enable	Enable	Enable

Table 2451. Tool operation (JTAG to LFAST switching)

Tool	JCOMP low	JCOMP high (JTAG mode)(1)	DCI_CR [EN_ LFAST]=1	Run-Test-Idle state	TCK detected	Internal timer elapse
TDI	Don’t Care	Logic Output	Hi-Z	Hi-Z	LVDS Rx	LVDS TxN
TMS	Don’t Care	Logic Output	Hi-Z	Hi-Z	LVDS Rx	LVDS TxP
TDO	Hi-Z	Logic Input	Hi-Z	Hi-Z	LVDS Tx	LVDS RxP

Table 2451. Tool operation (JTAG to LFAST switching) (continued)

Tool	JCOMP low	JCOMP high (JTAG mode)(1)	DCI_CR [EN_ LFAST]=1	Run-Test-Idle state	TCK detected	Internal timer elapse
JCOMP	Hi-Z	Logic Output (High)	Hi-Z	Hi-Z	LVDS Tx	LVDS RxN
TCK	Don't Care	Logic Output	Logic Input	Logic Input	Logic Input	Logic Input

1. This state is used to set DCI_CR[EN_LFAST] bit.

Table 2452. MCU operation (LFAST to JTAG switching)

MCU pin	LFAST mode	DCI_CR[EN_LFAST]=0	First timeout
TDI	LVDS Tx	Logic Input	Logic Input
TMS	LVDS Tx	Logic Input	Logic Input
TDO	LVDS Rx	Hi-Z	Logic Output
JCOMP	LVDS Rx	Logic Input Latch	Logic Input (sample JCOMP pin)
TCK	Logic Output	Logic Input	Logic Input
Mode Escape	Enable	Inhibit	Inhibit
LFAST_SYSCLK_EN	Enable	Inhibit	Inhibit

Table 2453. Tool operation (LFAST to JTAG switching)

Tool	Send command to clear DCI_CR[EN_LFAST]	LFAST transmits idle after command sent	TCK input clock stops or LVDS Rx both low	Internal timeout: start sending JTAG messages
TDI	LVDS Rx	LVDS Rx	Logic Output	Logic Output
TMS	LVDS Rx	LVDS Rx	Logic Output	Logic Output
TDO	LVDS Tx	Hi-Z	Logic Input	Logic Input
JCOMP	LVDS Tx	Hi-Z	Logic Output (High)	Logic Output (High)
TCK	Logic Input	Logic Input	Logic Output	Logic Output

102.4.2 LFAST LVDS pad fault

It is possible that when the tool is operating in LFAST mode, the tool may get disconnected due to external disturbances, which could put the MCU in an undetermined state. It is imperative to keep the MCU in a safe state. Thus the BD LFAST Rx pads have a sensing logic which detects if the Rx pad voltages are not equal to the common mode voltage when the BD DCI is in LFAST mode. If it is found that the LFAST Rx pads are not corresponding to the common mode voltage, the pad logic sends a signal to the DCI which then puts the DCI into JTAG mode. *Table 2454* and *Table 2455* contain the different steps and expected states for the MCU and tool for LFAST to JTAGRST switching.

Table 2454. MCU operation (LFAST to JTAGRST switching)

MCU pin	LFAST mode	JCOMP detected low (mode escape)
TDI	LVDS TxN	Logic Input
TMS	LVDS TxP	Logic Input
TDO	LVDS RxP	Hi-Z
JCOMP	LVDS RxN	Logic Input
TCK	Logic Output	Logic Input
Mode Escape	Enable	Inhibit
LFAST_SYSCLK_EN	Enable	Inhibit
Internal TRST	Negated	Asserted

Table 2455. Tool operation (LFAST to JTAGRST switching)

Tool	LFAST mode	Tool forces JCOMP low
TDI	LVDS TxN	Don’t Care
TMS	LVDS TxP	Don’t Care
TDO	LVDS RxP	Hi-Z
JCOMP	LVDS RxN	Logic Output (Low)
TCK	Logic Input	Don’t Care

103 Buddy Device—JTAG Controller (JTAGC)

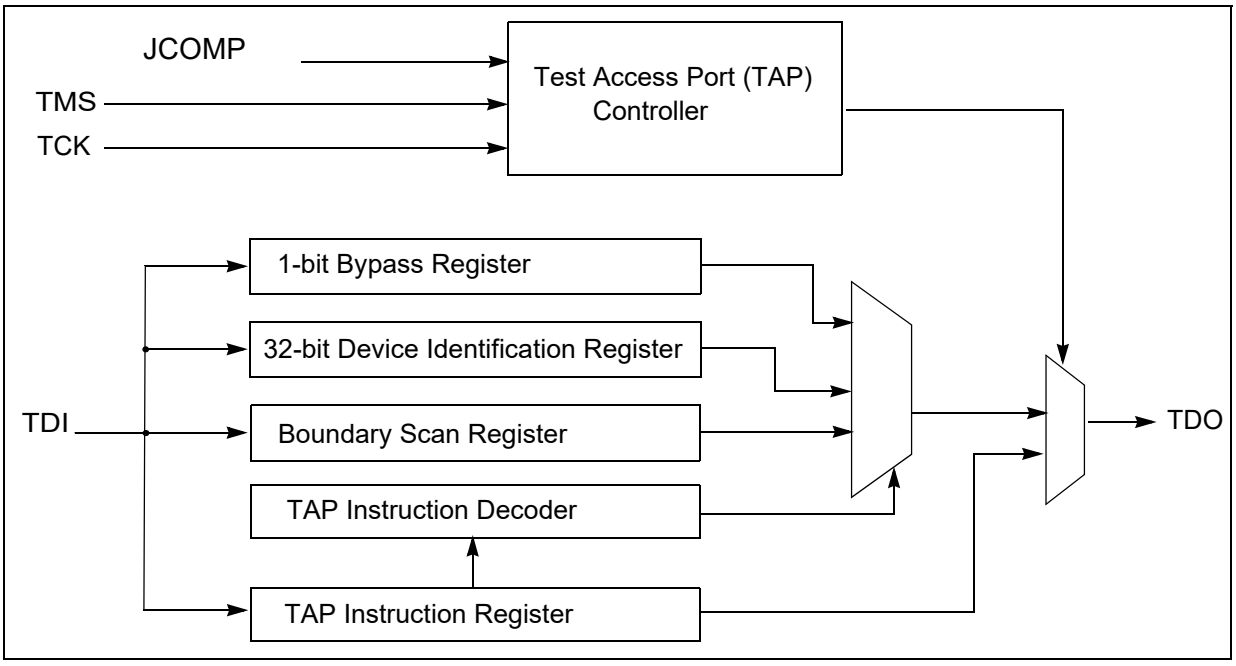
103.1 Introduction

The JTAGC block provides the means to test chip functionality and connectivity while remaining transparent to system logic when not in test mode. Testing is performed via a boundary scan technique, as defined in the IEEE 1149.1-2001 standard. All data input to and output from the JTAGC block is communicated in serial format.

103.1.1 Block Diagram

The following is a block diagram of the JTAG Controller (JTAGC) block.

Figure 2571. JTAG (IEEE 1149.1) block diagram



103.1.2 Features

The JTAGC block is compliant with the IEEE 1149.1-2001 standard, and supports the following features:

- IEEE 1149.1-2001 Test Access Port (TAP) interface
  - 4 pins (TDI, TMS, TCK, and TDO)
- JCOMP input that provides reset control and the ability to share the TAP.
- Instruction register that supports several IEEE 1149.1-2001 defined instructions as well as several public and private device-specific instructions. Refer to *Table 2459* for a list of supported instructions.
- Sharing of the TAP with other TAP controllers via ACCESS_AUX_x instructions
- JTAG_PASSWORD register
- Data registers, bypass register, boundary scan register, and device identification register.
- TAP controller state machine that controls the operation of the data registers, instruction register and associated circuitry.

103.1.3 Modes of operation

The JTAGC block uses JCOMP and a power-on reset indication as its primary reset signals. Several IEEE 1149.1-2001 defined test modes are supported, as well as a bypass mode.

103.1.3.1 Reset

The JTAGC block is placed in reset when either power-on reset is asserted, JCOMP is negated, or the TMS input is held high for enough consecutive rising edges of TCK to sequence the TAP controller state machine into the Test-Logic-Reset state. Holding TMS high for five consecutive rising edges of TCK guarantees entry into the Test-Logic-Reset state regardless of the current TAP controller state. Asserting power-on reset or setting JCOMP to a value other than the value required to enable the JTAGC block results in asynchronous entry into the reset state. While in reset, the following actions occur:

- The TAP controller is forced into the Test-Logic-Reset state, thereby disabling the test logic and allowing normal operation of the on-chip system logic to continue unhindered
- The instruction register is loaded with the IDCODE instruction

103.1.3.2 IEEE 1149.1-2001 defined test modes

The JTAGC block supports several IEEE 1149.1-2001 defined test modes. A test mode is selected by loading the appropriate instruction into the instruction register while the JTAGC is enabled. Supported test instructions include EXTEST, HIGHZ, CLAMP, SAMPLE and SAMPLE/PRELOAD. Each instruction defines the set of data register(s) that may operate and interact with the on-chip system logic while the instruction is current. Only one test data register path is enabled to shift data between TDI and TDO for each instruction.

The boundary scan register is enabled for serial access between TDI and TDO when the EXTEST, SAMPLE or SAMPLE/PRELOAD instructions are active. The single-bit bypass register shift stage is enabled for serial access between TDI and TDO when the BYPASS, HIGHZ, CLAMP or reserved instructions are active. The functionality of each test mode is explained in more detail in *Section 103.4.4: JTAGC block instructions*.

103.1.3.3 Bypass mode

When no test operation is required, the BYPASS instruction can be loaded to place the JTAGC block into bypass mode. While in bypass mode, the single-bit bypass shift register is used to provide a minimum-length serial path to shift data between TDI and TDO.

103.2 External signal description

The JTAGC consists of five signals that connect to off chip development tools and allow access to test support functions. The JTAGC signals are outlined in the following table and described in the following sections.

Table 2456. JTAG signal properties

Name	I/O	Function	Reset State	Pull
TCK	Input	Test Clock	—	Down
TDI	Input	Test Data In	—	Up
TDO	Output	Test Data Out	High Z(1)	—
TMS	Input	Test Mode Select	—	Up
JCOMP	Input	JTAG Compliancy	—	Down

1. TDO output buffer enable is negated when the JTAGC is not in the Shift-IR or Shift-DR states. A weak pull may be implemented at the TDO pad for use when JTAGC is inactive.

103.2.1 TCK—Test clock input

Test Clock Input (TCK) is an input pin used to synchronize the test logic and control register access through the TAP.

103.2.2 TDI—Test data input

Test Data Input (TDI) is an input pin that receives serial test instructions and data. TDI is sampled on the rising edge of TCK.

103.2.3 TDO—Test data output

Test Data Output (TDO) is an output pin that transmits serial output for test instructions and data. TDO is three-stateable and is actively driven only in the Shift-IR and Shift-DR states of the TAP controller state machine, which is described in *Section 103.4.3: TAP controller state machine*.

103.2.4 TMS—Test mode select

Test Mode Select (TMS) is an input pin used to sequence the IEEE 1149.1-2001 test control state machine. TMS is sampled on the rising edge of TCK.

103.2.5 JCOMP—JTAG compliancy

The JCOMP signal provides IEEE 1149.1-2001 compatibility and provides the ability to share the TAP. The JTAGC TAP controller is enabled when JCOMP is set to the JTAGC enable encoding, otherwise the JTAGC TAP controller remains in reset.

103.3 Register description

This section provides a detailed description of the JTAGC block registers accessible through the TAP interface, including data registers and the instruction register. Individual bit-level descriptions and reset states of each register are included. These registers are not memory-mapped and can only be accessed through the TAP.

103.3.1 Instruction register

The JTAGC block uses a 6-bit instruction register as shown in the following figure. The instruction register allows instructions to be loaded into the block to select the test to be performed or the test data register to be accessed or both. Instructions are shifted in through TDI while the TAP controller is in the Shift-IR state, and latched on the falling edge of TCK in the Update-IR state. The latched instruction value can only be changed in the Update-IR and Test-Logic-Reset TAP controller states. Synchronous entry into the Test-Logic-Reset state results in the IDCODE instruction being loaded on the falling edge of TCK. Asynchronous entry into the Test-Logic-Reset state results in asynchronous loading of the IDCODE instruction. During the Capture-IR TAP controller state, the instruction shift register is loaded with the value 000001b, making this value the register's read value when the TAP controller is sequenced into the Shift-IR state.

	0	1	2	3	4	5
R	0	0	0	0	0	1
W	Instruction Code					
Reset:	0	0	0	0	0	1

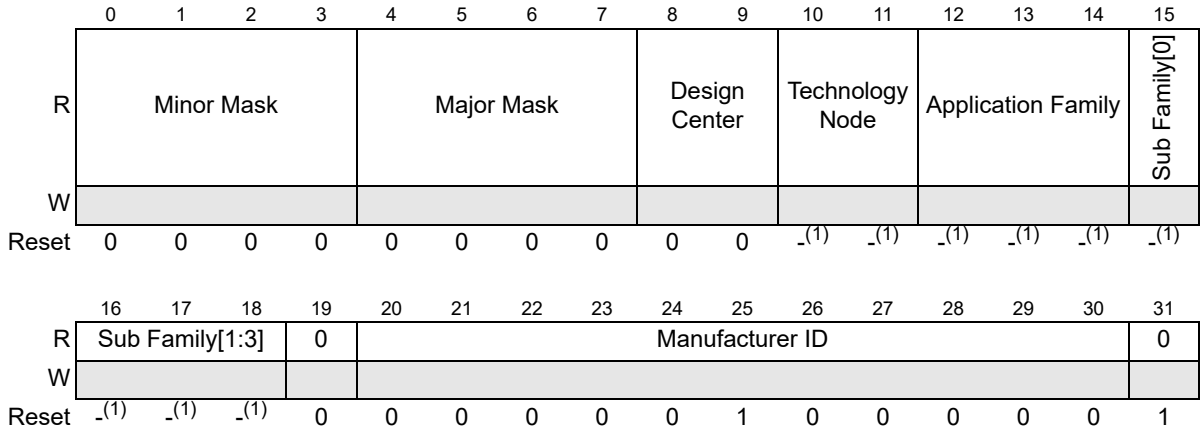
Figure 2572. 6-bit instruction register

103.3.2 Bypass register

The bypass register is a single-bit shift register path selected for serial data transfer between TDI and TDO when the BYPASS, CLAMP, HIGHZ or reserve instructions are active. After entry into the Capture-DR state, the single-bit shift register is set to a logic 0. Therefore, the first bit shifted out after selecting the bypass register is always a logic 0.

103.3.3 Device identification register

The device identification (JTAG ID) register, shown in the following figure, allows the revision number, part number, manufacturer, and design center responsible for the design of the part to be determined through the TAP. The device identification register is selected for serial data transfer between TDI and TDO when the IDCODE instruction is active. Entry into the Capture-DR state while the device identification register is selected loads the IDCODE into the shift register to be shifted out on TDO in the Shift-DR state. No action occurs in the Update-DR state.



1. The value is shown in *Table 2457*

Figure 2573. Device identification register

The following table describes the device identification register functions.

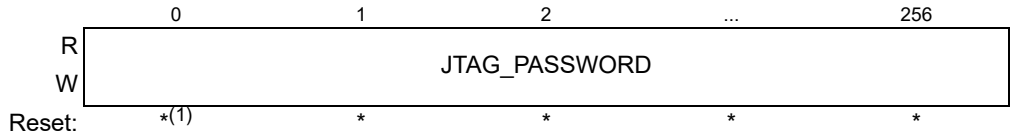
Table 2457. Device identification register field descriptions

Field	Description
0:3 Minor Mask	Minor Mask These bits contain the minor mask of the part
4:7 Major Mask	Major Mask These bits contain the major mask of the part
8:9 Design Center	Design Center These bits indicate the device design center
10:11 Technology Node	Technology Node These bits indicate the device technology node 00 40nm
12:14 Application Family	Application Family These bits indicate the device application family 101 SPC58N - Chassis & Safety 40nm
15:18 Sub Family	Sub Family These bits indicate the device sub family 0100 SPC58NE84C3 , SPC58NE84E7
20:30 Manufacturer ID	Manufacturer Identity Code These bits contain the reduced Joint Electron Device Engineering Council (JEDEC) ID 00000100000: STMicroelectronics

103.3.4 JTAG_PASSWORD register

The JTAG_PASSWORD register is a 256-bit shift register path from TDI to TDO selected when the ENABLE_JTAG_PASSWORD instruction is active. The default reset value of the

JTAG_PASSWORD register is 256'b0. The JTAG_PASSWORD register transfers its value to a parallel hold register on the rising edge of TCK when the TAP controller state machine is in the Update-DR state. Once the ENABLE_JTAG_PASSWORD instruction is executed, the register value remains valid until a JTAG reset occurs. The operation of this register is described in the security documentation.



1. The reset value of JTAG_PASSWORD is 256'b0.

Figure 2574. JTAG_PASSWORD register

The following table describes the JTAG_PASSWORD register functions.

Table 2458. JTAG_PASSWORD register field descriptions

Field	Description
JTAG_PASSWORD	JTAG Password. The JTAG_PASSWORD bits are used to provide the JTAG password for security.

103.3.5 Boundary scan register

The boundary scan register is connected between TDI and TDO when the EXTEST, SAMPLE or SAMPLE/PRELOAD instructions are active. It is used to capture input pin data, force fixed values on output pins, and select a logic value and direction for bidirectional pins. Each bit of the boundary scan register represents a separate boundary scan register cell, as described in the IEEE 1149.1-2001 standard and discussed in *Section 103.4.5: Boundary scan*. The size of the boundary scan register and bit ordering is device-dependent and can be found in the device BSDL file.

The BD boundary scan chain includes registers connected to all PD/BD interface signals at the BD side. These registers are controlled by the EXTEST, SAMPLE or SAMPLE/PRELOAD instructions applied to the BD JTAG controller. Similarly, the PD boundary scan chain also contains registers connected to the interface signals at the PD side, which are controlled by instructions applied to the PD JTAG controller.

103.4 Functional description

This section explains the JTAGC functional description.

103.4.1 JTAGC reset configuration

While in reset, the TAP controller is forced into the Test-Logic-Reset state, thus disabling the test logic and allowing normal operation of the on-chip system logic. In addition, the instruction register is loaded with the IDCODE instruction.

103.4.2 IEEE 1149.1-2001 (JTAG) Test Access Port

The JTAGC block uses the IEEE 1149.1-2001 TAP for accessing registers. This port can be shared with other TAP controllers on the MCU. Ownership of the port is determined by the value of the currently loaded instruction. For more detail on TAP sharing via JTAGC instructions refer to *Section 103.4.4.8: ACCESS_AUX_x instructions*.

Data is shifted between TDI and TDO through the selected register starting with the least significant bit, as illustrated in the following figure. This applies for the instruction register, test data registers, and the bypass register.

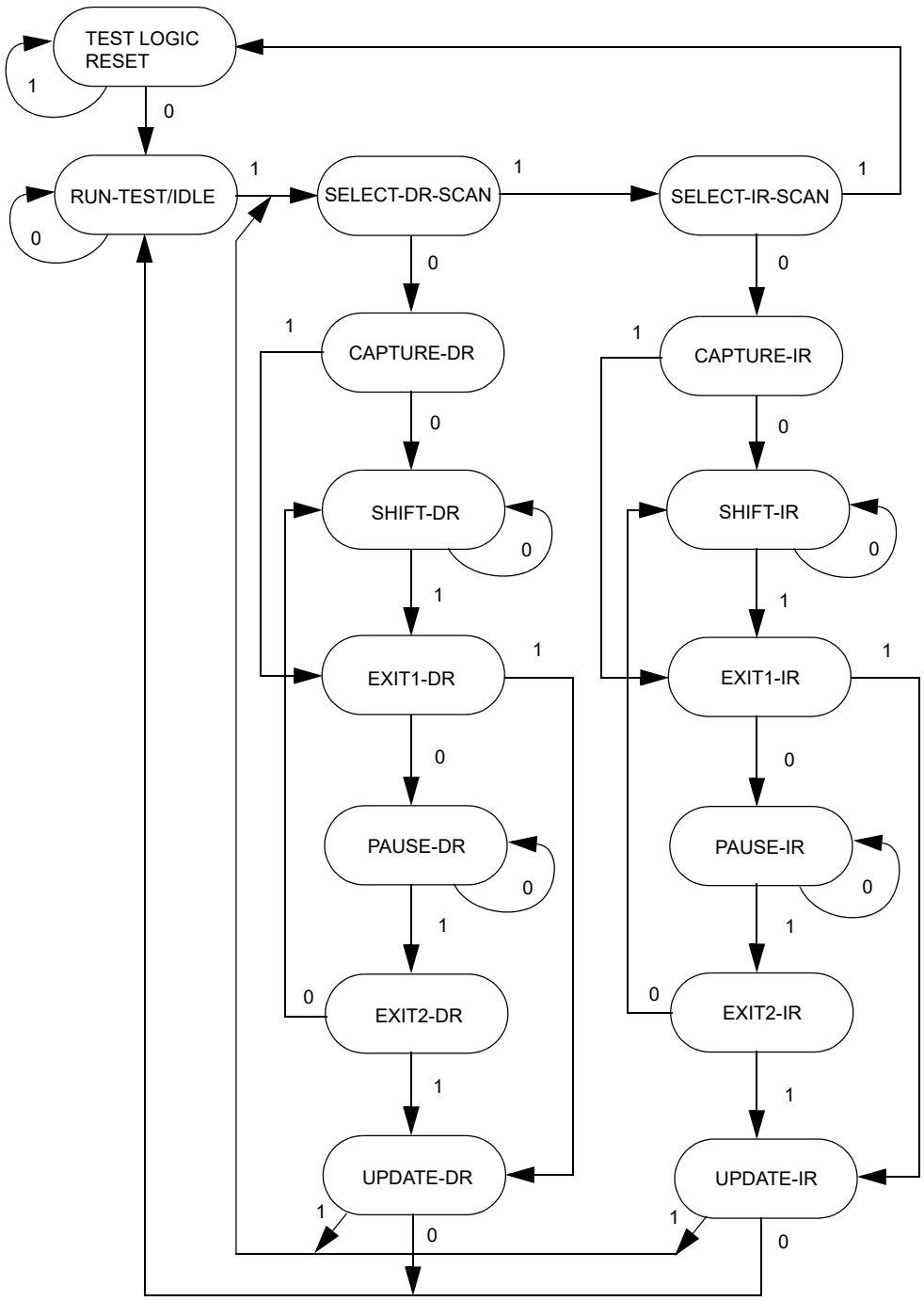
Figure 2575. Shifting data through a register



103.4.3 TAP controller state machine

The TAP controller is a synchronous state machine that interprets the sequence of logical values on the TMS pin. The following figure shows the machine's states. The value shown next to each state is the value of the TMS signal sampled on the rising edge of the TCK signal. As the following figure shows, holding TMS at logic 1 while clocking TCK through a sufficient number of rising edges also causes the state machine to enter the Test-Logic-Reset state.

Figure 2576. IEEE 1149.1-2001 TAP controller finite state machine



Note:

1. The value shown adjacent to each state transition represents the value of TMS at the time of a rising edge of TCK.

103.4.3.1 Enabling the TAP controller

The JTAGC TAP controller is enabled by setting JCOMP to a logic 1 value.

103.4.3.2 Selecting an IEEE 1149.1-2001 register

Access to the JTAGC data registers is achieved by loading the instruction register with any of the JTAGC block instructions while the JTAGC is enabled. Instructions are shifted in via the Select-IR-Scan path and loaded in the Update-IR state. At this point, all data register access is performed via the Select-DR-Scan path.

The Select-DR-Scan path is used to read or write the register data by shifting in the data (LSB first) during the Shift-DR state. When reading a register, the register value is loaded into the IEEE 1149.1-2001 shifter during the Capture-DR state. When writing a register, the value is loaded from the IEEE 1149.1-2001 shifter to the register during the Update-DR state. When reading a register, there is no requirement to shift out the entire register contents. Shifting may be terminated once the required number of bits have been acquired.

103.4.4 JTAGC block instructions

The JTAGC block implements the IEEE 1149.1-2001 defined instructions listed in the following table. This section gives an overview of each instruction; refer to the IEEE 1149.1-2001 standard for more details. All undefined opcodes are reserved.

Table 2459. 6-bit JTAG instructions

Instruction	Code[5:0]	Instruction Summary
IDCODE	000001	Selects device identification register for shift
SAMPLE/PRELOAD	000010	Selects boundary scan register for shifting, sampling, and preloading without disturbing functional operation
SAMPLE	000011	Selects boundary scan register for shifting and sampling without disturbing functional operation
EXTEST	000100	Selects boundary scan register while applying preloaded values to output pins and asserting functional reset
Reserved	000101	Reserved
Reserved	000110	Reserved
ENABLE_JTAG_PASSWORD	000111	Selects JTAG_PASSWORD register
HIGHZ	001001	Selects bypass register while three-stating all output pins and asserting functional reset
Reserved	001010	Reserved
CLAMP	001100	Selects bypass register while applying preloaded values to output pins and asserting functional reset
ENABLE_DCI_CR	001110	Enables access to DCI CR register
ENABLE_DCI_PINCR	001111	Enables access to DCI PINCR register
ACCESS_AUX_x	100000-111110	Grants one of the auxiliary TAP controllers ownership of the TAP as shown in the cells below.
ACCESS_AUX_NR_BD	100000	Enables access to the NAR TAP controller on BD
Reserved	100001	Reserved
ACCESS_AUX_NAL	100010	Enables access to the NAL TAP controller

Table 2459. 6-bit JTAG instructions (continued)

Instruction	Code[5:0]	Instruction Summary
ACCESS_AUX_RWA_BD	100011	Enables access to the RWA TAP controller
Reserved	100100	Reserved
Reserved	100101	Reserved
Reserved	100110	Reserved
Reserved	100111	Reserved
Reserved	101000	Reserved
Reserved	101001	Reserved
Reserved	101010	Reserved
Reserved	101011	Reserved
Reserved	101100	Reserved
Reserved	101101	Reserved
Reserved	101110	Reserved
Reserved	101111	Reserved
Reserved	110000	Reserved
Reserved	110001	Reserved
Reserved	110010	Reserved
Reserved	110011	Reserved
Reserved	110100	Reserved
Reserved	110101	Reserved
Reserved	110110	Reserved
Reserved	110111	Reserved
Reserved	111000	Reserved
Reserved	111001	Reserved
ACCESS_AUX_DWPU	111001	Reserved
Reserved	111010	Reserved
Reserved	111011	Reserved
ACCESS_AUX_PARALLEL(1)	111100	Enables access to all cores in parallel
Reserved	111101	Reserved
ACCESS_AUX_DCI_PD	111110	Enables access to pass control from BD to PD
BYPASS	111111	Selects bypass register for data operations
Reserved(2)	All other opcodes	Decoded to select bypass register

1. The use of this instruction is limited to the synchronous exit of debug mode via the execution of the GO+EXIT OnCE command. The execution of other instructions or commands is not supported
2. The manufacturer reserves the right to change the decoding of reserved instruction codes in the future

103.4.4.1 IDCODE instruction

IDCODE selects the 32-bit device identification register as the shift path between TDI and TDO. This instruction allows interrogation of the MCU to determine its version number and other part identification data. IDCODE is the instruction placed into the instruction register when the JTAGC block is reset.

103.4.4.2 SAMPLE/PRELOAD instruction

The SAMPLE/PRELOAD instruction has two functions:

- First, the SAMPLE portion of the instruction obtains a sample of the system data and control signals present at the MCU input pins and just before the boundary scan register cells at the output pins. This sampling occurs on the rising edge of TCK in the Capture-DR state when the SAMPLE/PRELOAD instruction is active. The sampled data is viewed by shifting it through the boundary scan register to the TDO output during the Shift-DR state. Both the data capture and the shift operation are transparent to system operation.
- Secondly, the PRELOAD portion of the instruction initializes the boundary scan register cells before selecting the EXTEST or CLAMP instructions to perform boundary scan tests. This is achieved by shifting in initialization data to the boundary scan register during the Shift-DR state. The initialization data is transferred to the parallel outputs of the boundary scan register cells on the falling edge of TCK in the Update-DR state. The data is applied to the external output pins by the EXTEST or CLAMP instruction. System operation is not affected.

103.4.4.3 SAMPLE instruction

The SAMPLE instruction obtains a sample of the system data and control signals present at the MCU input pins and just before the boundary scan register cells at the output pins. This sampling occurs on the rising edge of TCK in the Capture-DR state when the SAMPLE instruction is active. The sampled data is viewed by shifting it through the boundary scan register to the TDO output during the Shift-DR state. There is no defined action in the Update-DR state. Both the data capture and the shift operation are transparent to system operation.

103.4.4.4 EXTEST—External test instruction

EXTEST selects the boundary scan register as the shift path between TDI and TDO. It allows testing of off-chip circuitry and board-level interconnections by driving preloaded data contained in the boundary scan register onto the system output pins. Typically, the preloaded data is loaded into the boundary scan register using the SAMPLE/PRELOAD instruction before the selection of EXTEST. EXTEST asserts the internal system reset for the MCU to force a predictable internal state while performing external boundary scan operations.

103.4.4.5 ENABLE_JTAG_PASSWORD instruction

The ENABLE_JTAG_PASSWORD instruction selects the JTAG_PASSWORD register for connection as the shift path between TDI and TDO.

103.4.4.6 HIGHZ instruction

HIGHZ selects the bypass register as the shift path between TDI and TDO. While HIGHZ is active all output drivers are placed in an inactive drive state (e.g., high impedance). HIGHZ also asserts the internal system reset for the MCU to force a predictable internal state.

103.4.4.7 CLAMP instruction

CLAMP allows the state of signals driven from MCU pins to be determined from the boundary scan register while the bypass register is selected as the serial path between TDI and TDO. CLAMP enhances test efficiency by reducing the overall shift path to a single bit (the bypass register) while conducting an EXTEST type of instruction through the boundary scan register. CLAMP also asserts the internal system reset for the MCU to force a predictable internal state.

103.4.4.8 ACCESS_AUX_x instructions

The JTAGC is configurable to allow other TAP controllers on the device to share the port with it. This is done by providing ACCESS_AUX_x instructions for each of these TAP controllers. When this instruction is loaded, control of the JTAG pins are transferred to the selected TAP controller. Any data input via TDI and TMS is passed to the selected TAP controller, and any TDO output from the selected TAP controller is sent back to the JTAGC to be output on the pins. The JTAGC regains control of the JTAG port during the UPDATE-DR state if the PAUSE-DR state was entered. Auxiliary TAP controllers are held in RUN-TEST/IDLE while they are inactive. Instructions not used to access an auxiliary TAP controller on a device are treated like the BYPASS instruction.

103.4.4.9 BYPASS instruction

BYPASS selects the bypass register, creating a single-bit shift register path between TDI and TDO. BYPASS enhances test efficiency by reducing the overall shift path when no test operation of the MCU is required. This allows more rapid movement of test data to and from other components on a board that are required to perform test functions. While the BYPASS instruction is active the system logic operates normally.

103.4.5 Boundary scan

The boundary scan technique allows signals at component boundaries to be controlled and observed through the shift-register stage associated with each pad. Each stage is part of a larger boundary scan register cell, and cells for each pad are interconnected serially to form a shift-register chain around the border of the design. The boundary scan register consists of this shift-register chain, and is connected between TDI and TDO when the EXTEST, SAMPLE, or SAMPLE/PRELOAD instructions are loaded. The shift-register chain contains a serial input and serial output, as well as clock and control signals.

PD and the BD have their own separate boundary scan chains, each one controlled by the respective JTAG controller. Besides the package I/O pins, these chains also encompasses the interface signals between BD and PD. Testing of the interface can be done by applying EXTEST, SAMPLE or SAMPLE/PRELOAD instructions to both PD and BD JTAG.

103.5 Initialization/Application information

The test logic is a static logic design, and TCK can be stopped in either a high or low state without loss of data. However, the system clock is not synchronized to TCK internally. Any mixed operation using both the test logic and the system functional logic requires external synchronization.

To initialize the JTAGC block and enable access to registers, the following sequence is required:

1. Set the JCOMP signal to the JTAGC enable value, thereby enabling the JTAGC TAP controller
2. Load the appropriate instruction for the test or action to be performed

104 Buddy Device—JTAG Master (JTAGM)

104.1 Introduction

The JTAG Master (JTAGM) is a module that is able to act as JTAG master inside the device. The module has a parallel interface that can exchange data with another serial communication module (such as the LFAST module) or via customer software.

The data transferred to this module is transformed to produce TCK, TMS, TDI and TRST outputs and to accept TDO inputs. The JTAGM is connected in the device to allow these five signals to connect to the JTAGC as if the JTAG data is coming from an outside tool. The JTAGM generates all required JTAG scan chains to allow software and high speed serial communication access to all JTAG mapped resources.

104.1.1 Overview

The JTAGM is the master to drive JTAG signals from within the device. It has options to receive parallel data from software through the IPS interface or from the LFAST through the parallel interface. The JTAGM has the capability to differentiate between data from software and LFAST. It then sends this data on TDI, TMS and TCK to the DCI. The JTAGM also receives serial data through TDO from the DCI and transfers this data to the LFAST through another parallel interface. The JTAGM has the following registers:

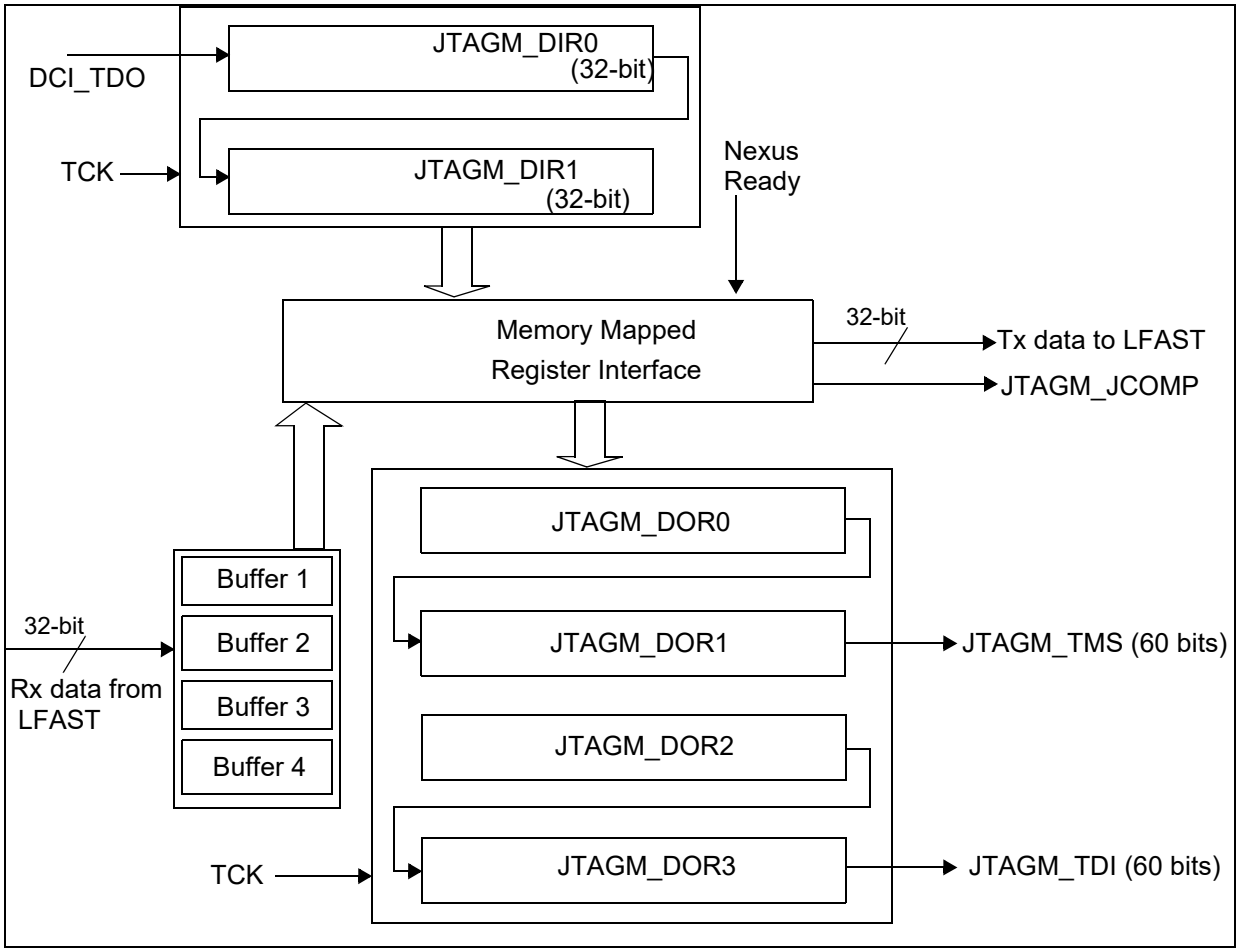
- Configuration register
- Status register
- Four data output registers
- RxCRC receive register
- Two data input registers

All these registers, described in detail in *Section 104.4: Memory map and register definition*, are memory mapped and can be accessed by software.

The clocks for the JTAGM are derived from the system clock. The JTAGM also samples the ready signal from the Nexus, to provide a more efficient handshake to the external tool to read and write any memory mapped address location within the device.

Figure 2577 shows the block diagram of the JTAGM.

Figure 2577. JTAGM block diagram



104.1.2 Feature description

The main features of the JTAGM are the following:

- Provides efficient handshake to the external tool to read any memory mapped address location within the device
- Provides software the option to write data for driving JTAG
- Receives/provides JTAG data from/to the LFAST

104.2 Functional description

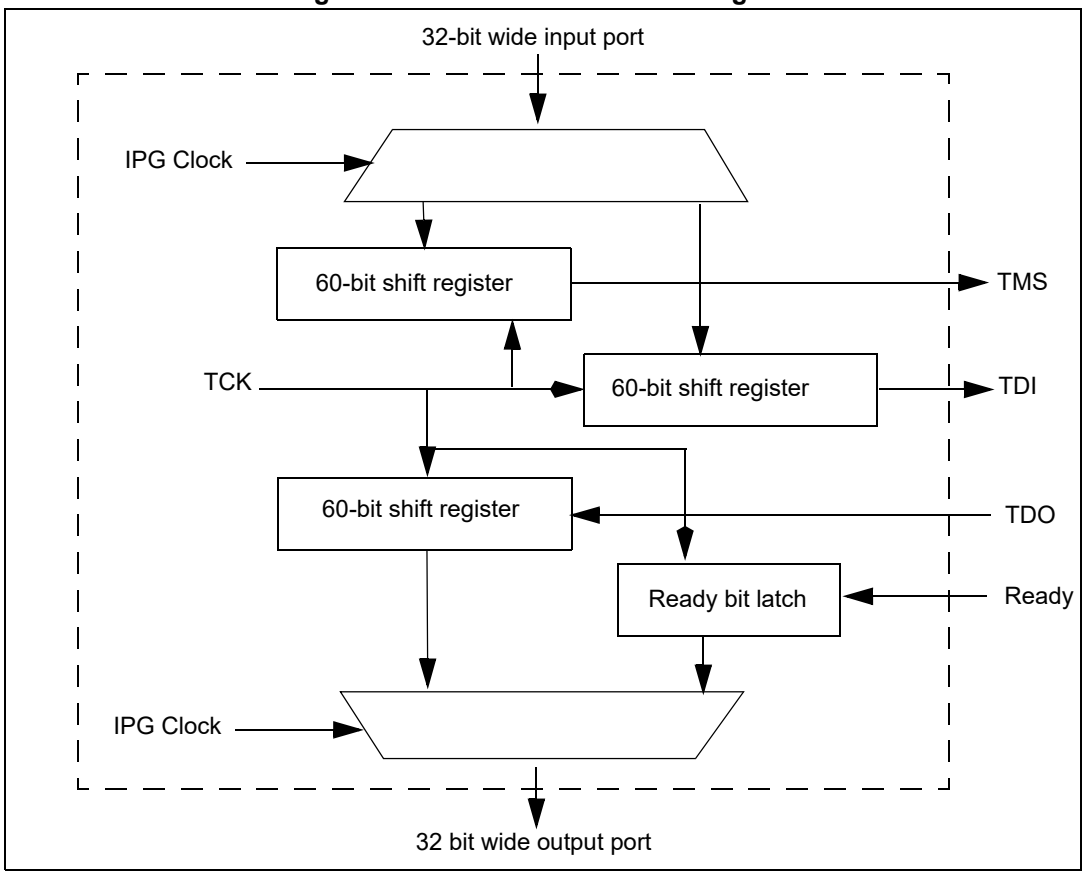
104.2.1 Module architecture

The JTAGM is simple in concept. TCK is generated from the device system clock. Data is pushed to the modules via a 32-bit wide parallel interface. The data is in the form of 60 bits of TMS data and 60 bits of TDI data. This data is the logical state to be driven onto the TMS and TDI output pins on each of the 60 TCK cycles. During these 60 TCK cycles, the TDO pin is sampled and the 60 bits of sampled data is transferred to the 32-bit wide module interface. This concept allows complete flexible generation on TMS and TDI data and

capture of TDO data. The only limitation is that JTAG signals can only be generated in 60 TCK cycles packets. Extra cycles are wasted in idle cycles at the end of a scan chain.

Figure 2578 shows the flow diagram of the LFAST to DCI connection.

Figure 2578. LFAST-to-DCI flow diagram



104.2.2 JTAGM JCOMP usage in LFAST

JTAGM is used to drive JTAG sequences in software mode and in LFAST mode. The JTAGM_MCR[jtagm_JCOMP] bit is provided to reset all TAP controllers by generating a JCOMP signal to JTAGC.

The guidelines to be considered for the usage of JTAGM_MCR[jtagm_JCOMP]:

- Do not clear the JTAM_MCR[jtagm_JCOMP] bit while in LFAST mode as this operation is not guaranteed
- If for some reasons during the LFAST mode there is a loss of communication, one of the following should be done (instead of clearing the jtagm_JCOMP bit):
  - Generate the escape sequence and then, once the device is back in JTAG mode, generate a test-logic-reset state via the TCK and TMS signals. This resets all TAP controllers but does not affect the client debug configuration
  - Generate the escape sequence and then, once the device is back in JTAG mode, force the JCOMP pin low. This resets all TAP controllers as well as the client debug configuration

Note: The escape sequence is generated by forcing LFAST LVDS ports RX+ and RX- to zero.

104.2.3 JTAGM data error detection

To improve the integrity of the JTAG operation, data passed to and from the JTAGM includes an 8-bit CRC only in LFAST mode. The data sent to the JTAGM consists of 60 bits of TMS data, 60 bits of TDI and an 8-bit CRC. The JTAGM recalculates the 8-bit CRC for the first 120 bits of data and compares the result with the received CRC. If the two match, the data is assumed to be valid and the JTAGM converts this data into a valid JTAG frame.

If the CRCs do not match, the module sets the CRC error status bit (CRC_err) in the JTAGM status register (JTAGM_SR) and sends a standard 96-bit response frame to the LFAST channel B, with all 60 data bits set to 0. The CRC error status bit is automatically cleared as soon as the response frame has been sent to the LFAST.

Data sent from the JTAGM to the LFAST also includes an 8-bit CRC. The data sent from the JTAGM is in the form of a 96-bit frame consisting of 60 bits of TDO data + 4 bits of zero-padding + 32 bits of the status register. The 8-bit CRC value is calculated for the first 88 bits of the 96-bit frame and inserted into the least significant 8 bits of the status register; which is also the last 8 bits of the 96-bit frame. The tool is responsible for checking the CRC of the received data and taking corrective action.

104.2.4 JTAGM message tagging

The JTAGM, in conjunction with the LFAST, is pipelined to allow several JTAGM messages to be sent from the tool before any response is received back. This allows better utilization of the LFAST bandwidth. To enable the tool to associate individual messages with the correct response, the messages are tagged. The tool sends data messages on LFAST channels E, F, G and H in sequence. The JTAGM module returns responses on the same channel they were received. For example, if the tool sends a data command on channel E, JTAGM returns the response on channel E; if the tool sends a data command on channel F, the JTAGM returns the response on channel F; and so on. This way, the tool can ensure a response was generated by a specific command.

Configuration commands to write/read the configuration and status registers within the JTAGM are sent on channel A and response data is also sent back on channel A.

104.2.5 JTAGM Ready signal

The Nexus Read/Write Access (RWA) module allows an external tool to read and write any memory mapped address location within the device via JTAG commands. Read accesses to memory take a finite amount of time and a JTAG read of the data register too soon results in erroneous data being read back. There is a bit in a JTAG register to show the read data is available. In conventional JTAG mode, the status of this ready bit is also provided back to the tool via a dedicated pin to allow the tool to know data is ready and the JTAG read register can be accessed. With the LFAST interface providing pipelined data to the JTAGM, this technique is not available. Two interlocked mechanisms are provided within the JTAGM that use this ready bit to allow optimized data reads without ready bit errors.

104.2.5.1 Status register ready bit

A bit is provided in the JTAGM status register to indicate the status of the ready signal. To provide this ready signal functionality, the ready signal is monitored. If the ready signal has toggled between the start of the previous 60 TCK shift period and start of the current 60 TCK shift period, the ready bit in the JTAGM status register is set. Otherwise the ready bit in the status register is cleared.

The entire content of the 32-bit status register is appended to the 60 bits of JTAG TDO data captured along with 4 bits of 0-padding and sent as output on the parallel output port as a 96-bit payload on channel A. This provides back to the tool, an indication that the data read in the current transaction is valid.

104.2.5.2 Inter-JTAG frame gap timer

To operate in conjunction with the status register ready bit, a configurable timer is provided to force a programmable gap between the end of one 60-bit JTAG frame and the start of the next. This timer is configured via the JTAGM configuration register from 1 to 64 TCKs in 1 TCK intervals. Furthermore, if the ready signal asserts during this inter-frame gap period, the gap timer is aborted and the next JTAG frame starts immediately.

104.2.6 $\overline{\text{EVTO}}$ and $\overline{\text{EVTI}}$ signals

The standard Nexus EVTI (Event In) and EVTO (Event Out) signals are optionally present on dedicated package pins and are used by an external tool in conjunction with the JTAG interface to pass information, events and triggers between the tool and the MCU.

The JTAGM as well as providing a higher speed alternative to the standard JTAG interface, also allows control and status of the EVTI and EVTO signals to be embedded into the serial protocol. The EVTI0 and EVTI1 signals can be asserted by writing control bits in the Module Configuration Register (JTAGM_MCR). The JTAGM can be configured by bits in the Module Configuration Register (JTAGM_MCR) to respond to a rising/falling/both edge transition of either EVTO0 or EVTO1. The response is either an unsolicited serial message to the tool when in LFAST mode (DTM = 0); or a CPU interrupt when in software mode (DTM = 1).

The unsolicited LFAST message is sent on LFAST channel B to allow the tool to distinguish the message from normal JTAG response data (channels E, F, G & H) and Config Command Responses (channel A). The CPU interrupt is gated by an Interrupt enable bit, in MCR and reported by Interrupt Status flags in the Status Register (JTAGM_SR). The status bits stay set until cleared by writing to the clear bits in the JTAG_SR.

Asserting the evti0_assert bit in the JTAGM_MCR register can trigger debug mode in the DCI. Asserting the evti1_assert bit in the JTAGM_MCR register can trigger debug mode in the DCI. The DTM bit in the JTAGM_MCR register can put the DCI in LFAST or SW mode.

Note: EVTI generation and EVTO detection bits of JTAGM can be used by debugger to detect EVTI assertion and to generate EVTO. This method is useful when EVTI/O pins are not available on SoC PADs.

104.2.7 JTAGM configuration and status monitoring

The JTAGM module requires some level of configuration and status monitoring. The JTAGM includes a 32-bit configuration register (JTAGM_MCR) and a 32-bit status register (JTAGM_SR, which includes the ready bit mentioned in *Section 104.2.5.1: Status register*

ready bit). The configuration register is accessed via the parallel input port by a 128-bit payload arriving on channel A. The 128-bit payload is made from the following:

- 32-bit mask of which bits within the configuration register should be written
- 32-bit value to be written to the configuration register
- 24-bit mask of which bits within the status register should be written
- 8 bits of zero-padding
- 24 bits of data to be written to the status register
- 8-bit CRC (The least significant 8 bits of the status register cannot be written because they contain automatically generated CRC data)

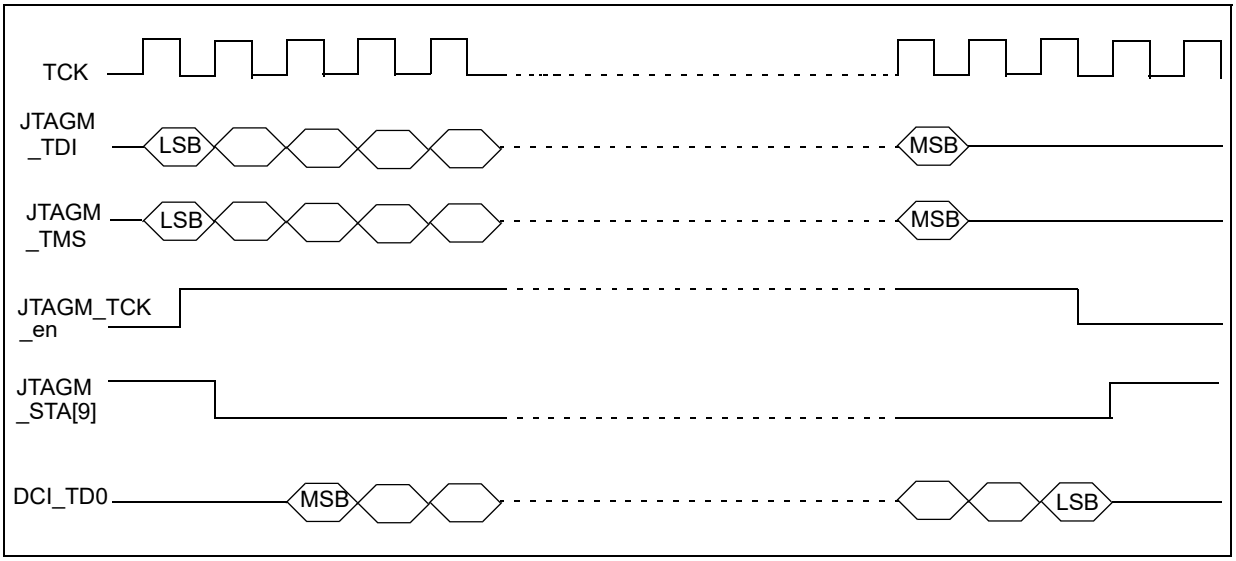
The masked data is transferred to the 32-bit configuration register and 32-bit status register. The JTAGM module responds to this configuration/status write by outputting a 96-bit payload to the parallel output port to channel number A. The 96-bit payload is made up of the 32-bit value of the configuration register plus 32 bits of zero-padding plus 32-bit value of the status register (including the least significant 8 bits which are an 8-bit CRC calculated on the previous 88 bits).

[BLG_JTAGM_0008][Covers: ADD23.035]While the current messages are being transmitted from the second buffering (memory mapped registers), the first buffers are continuously filled up. The status register also includes status bits provided by the LFAST module and the DCI module. The JTAGM requests data from the LFAST only when at least one internal buffer is free.

104.2.8 JTAGM to DCI serial interface

Figure 2579 shows the data transfer timings between JTAGM and DCI.

Figure 2579. JTAGM to DCI serial interface diagram



104.2.9 JTAGM data handling and CRC calculation in LFAST mode

Figure 2580 shows the bit ordering when a data frame is transmitted from LFAST to JTAGM for subsequent transmission to DCI.

Figure 2580. Bit ordering for LFAST to JTAGM transfer

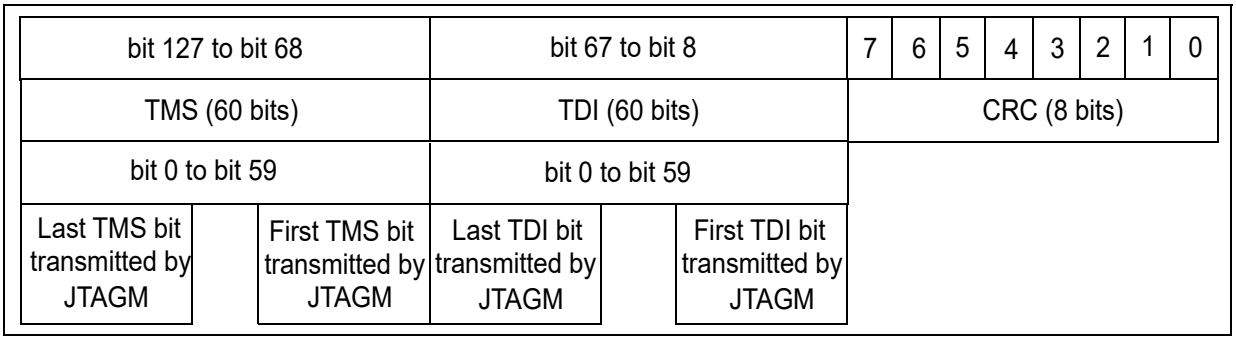
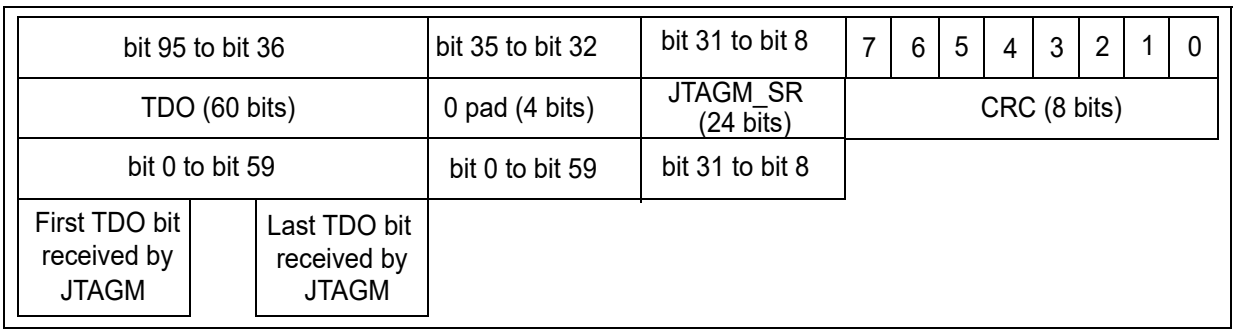


Figure 2581 shows the bit ordering when a data frame is transmitted from JTAGM to LFAST consisting of TDO data received from DCI.

Figure 2581. Bit ordering for JTAGM to LFAST transfer



The CRC calculation is described in following example.

The data transferred from LFAST to JTAGM (in four frames) is as follows:

- D0—F7B3_D591
- D1—E6A2_C486
- D2—A2C4_80F7
- D3—B3D5_9183

These are supplied to CRC block starting from D0 to D3. The CRC calculation begins on D0 with the MSB (left bit) going first. D3 is supplied as B3D5_9100 (after masking CRC).

JTAGM extracts TMS and TDI information from the data received as follows:

- TMS = F7B3_D591_E6A2_C48
- TDI = 6A2C_480F_7B3D_591

JTAGM transmits the data to DCI starting from LSB (right bit going first). Corresponding to this TMS and TDI transmission, JTAGM receives TDO data from DCI and stores it as first bit to most significant location in TDO frame (shown in *Figure 2581*).

104.2.10 Software interface

It is possible for software to write the 120 bits of JTAG data to be output and to read the 60 bits of data. The JTAGM module has an IPS interface.

104.3 Modes of operation

There are no special modes of operation. The JTAGM works normally in the debug mode.

104.4 Memory map and register definition

Table 2460 contains the memory map of the JTAGM registers.

Table 2460. Memory map

Offset	Register	Location
0x00	Module Configuration Register (JTAGM_MCR)	*104.4.1.1 on page 4153*
0x04	Status Register (JTAGM_SR)	*104.4.1.2 on page 4155*
0x08	Data Out Register 0 (JTAGM_DOR0)	*104.4.1.3 on page 4158*
0x0C	Data Out Register 1 (JTAGM_DOR1)	*104.4.1.4 on page 4158*
0x10	Data Out Register 2 (JTAGM_DOR2)	*104.4.1.5 on page 4159*
0x14	Data Out Register 3 (JTAGM_DOR3)	*104.4.1.6 on page 4159*
0x18	Receive CRC Register (JTAGM_RxCRC)	*104.4.1.7 on page 4160*
0x1C	Data Input Register 0 (JTAGM_DIR0)	*104.4.1.8 on page 4160*
0x20	Data Input Register 1 (JTAGM_DIR1)	*104.4.1.9 on page 4161*

104.4.1 Register descriptions

104.4.1.1 Module configuration register (JTAGM_MCR)

This register contains the status bits for the current transfer. *Figure 2582* shows the format of the JTAGM_MCR. In the Emulation and Debug Device (ED), all bits of this register need to be programmed in the Buddy Device instance of the JTAGM. The only exception is the SIE bit, which always needs to be programmed in the Production Device instance of the JTAGM, regardless of being ED or not.

Offset 0x00 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	SWRESET	evto0_sense		evto1_sense		evto_IE	evti0_assert	evti1_assert	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	inter_jtag_frame_timer						0	SIE	IIE	TCKSEL			jtagm_JCOMP	DTM
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0

Figure 2582. JTAGM_MCR register

The JTAGM_MCR register is described in *Table 2461*.

Table 2461. JTAGM_MCR register field descriptions

Field	Description
0 SWRESET	Software Reset. Writing a 1 resets the state machine and counters inside JTAGM. It is a self clearing bit. Writing a 0 has no effect. **Note:** After being written to “1” , this bit is auto cleared within 3 TCK cycles.
1:2 evto0_sense	Determines JTAGM response to activity on $\overline{\text{EVTO}}$0 00 No action is taken by the JTAGM in response to any $\overline{\text{EVTO}}$0 edges (default setting) 01 evto0_edge status bit is set whenever a falling edge is detected on $\overline{\text{EVTO}}$0 10 evto0_edge status bit is set whenever a rising edge is detected on $\overline{\text{EVTO}}$0 11 evto0_edge status bit is set whenever a falling or rising edge is detected on $\overline{\text{EVTO}}$0
3:4 evto1_sense	Determines JTAGM response to activity on $\overline{\text{EVTO}}$1 line 00 No action is taken by the JTAGM in response to any $\overline{\text{EVTO}}$1 edges (default setting) 01 evto1_edge status bit is set whenever a falling edge is detected on $\overline{\text{EVTO}}$1 10 evto1_edge status bit is set whenever a rising edge is detected on $\overline{\text{EVTO}}$1 11 evto1_edge status bit is set whenever a falling or rising edge is detected on $\overline{\text{EVTO}}$1
5 evto_IE	Enables $\overline{\text{EVTO}}$ triggered JTAGM interrupt 0 No JTAGM interrupt is generated In response to any $\overline{\text{EVTO}}$0 or $\overline{\text{EVTO}}$1 activity 1 JTAGM interrupt is generated when evto0_edge or evto1_edge are set
6 evti0_assert	EVTI0 Assert 0 Set jtagm_evti0 low/asserted 1 Set jtagm_evti0 high/de-asserted
7 evti1_assert	EVTI1 Assert 0 Set jtagm_evti1 low/asserted 1 Set jtagm_evti1 high/de-asserted **Note:** In the SW mode(when DTM =1) , regardless of the state of “evti1_assert” bit the internall “evti” signal remains de-asserted . Though this bit can be written, writing to this bit has no effect .

Table 2461. JTAGM_MCR register field descriptions (continued)

Field	Description
18:23 inter_jtag_frame_timer	TCK delay 000000 0 TCK cycle 000001 1 TCK cycle … 111111 63 TCK cycles
25 SIE	SPU Interrupt Enable(1) 0 JTAGM does not generate an interrupt to the CPU 1 JTAGM generates an interrupt to the CPU if the SPU interrupt request is asserted
26 IIE	Idle Interrupt Enable 0 JTAGM does not generate an interrupt to the CPU upon completion of a 60-bit JTAG transfer 1 JTAGM generates an interrupt to the CPU upon completion of a 60-bit JTAG transfer
27:29 TCKSEL	TCK Divider division factor control value. (TCK Divider divides the clock input to JTAGM IP to generate TCK out(input to DCI)). 000 TCK Divider division factor is 1 001 TCK Divider division factor is 2 010 TCK Divider division factor is 3 011 TCK Divider division factor is 4 100 TCK Divider division factor is 5 101 TCK Divider division factor is 6 110 TCK Divider division factor is 7 111 TCK Divider division factor is 8 **Note:** It should be taken care that the maximum frequency of TCK input to DCI, should be equal or less than half of system clock.
30 jtagm_JCOMP	JTAG reset 0 JCOMP low/asserted 1 JCOMP high/not asserted (default)
31 DTM	Data Transfer Mode. 0 Data for JTAG transferred by LFAST 1 Data for JTAG transferred by software

1. The BD JTAGM instance does not receive signals from the SPU. If required, this feature needs to be enabled in the PD instance of the JTAGM, even when JTAG generation in LFAST mode is handled by the BD JTAGM.

104.4.1.2 Status register (JTAGM_SR)

104.4.1.2.1 LFAST mode

Figure 2583 shows the format of the JTAGM_SR in LFAST mode. In the ED, all bits of this register need to be accessed in the BD instance of the JTAGM. The exceptions are the bits related to the SPU interrupts (SPU_INT, SPU_INT_CLR), which always need to be accessed from the PD JTAGM, regardless of being ED or not.

Offset 0x04 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	overrun	0	lfast_status						dci_status							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	evto0_edge	evto1_edge	SPU_INT_CLR	SPU_INT	CRC_err	Nexus_err	Idle	NR	CRC							
W																
Reset	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0

Figure 2583. JTAGM_SR register in LFAST mode

104.4.1.2.2 SW mode

Figure 2584 shows the format of the JTAGM_SR in SW mode.

Offset 0x04 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	lfast_status						dci_status							
W																
Reset	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	evto0_edge	evto1_edge	SPU_INT_CLR	SPU_INT	CRC_err	Nexus_err	Idle	NR			0	0	0	0	0	0
W									evto0_clr	evto1_clr						
Reset	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0

Figure 2584. JTAGM_SR register in SW mode

The JTAGM_SR register is described in *Table 2462*.

Table 2462. JTAGM_SR register field descriptions

Field	Description
0 overrun	Set by hardware when the JTAGM overwrites the first Tx data with the second Tx data. Writing 1 clears this bit. This bit is set only in LFAST mode.
2:7 lfast_status	Status information from the LFAST to be used by software. The composition of these bits are as follows: bit 2 Indicates either data frame or ping frame was transmitted successfully bit 3 Indicates Tx data Interface not enabled and a frame is ready to be transmitted bit 4 Indicates Rx data FIFO overflow bit 5 Indicates reception of frame with invalid frame payload size bit 6 Indicates reception of frame with invalid ICLC code bit 7 Indicates reception of illegal LCT frame
8:15 dci_status	Status information from the DCI. The composition of these bits is as follows: bit 8 Reserved to 0 bit 9 DCI LVDS port enable bit 10 DCI JTAG port enable bit 11 DCI LVDS safe mode bit 12 DCI JTAG safe mode bit 13 DCI LVDS escape mode bit 14 DCI enable LFAST bit 15 DCI tool present , set when a debug tool is connected
16 evto0_edge	In LFAST mode (DTM = 0) a channel B message is sent on LFAST and the bit is automatically cleared. In software mode (DTM = 1), the bit stays set until cleared by writing to evto0_clr bit 0 No activity sensed on $\overline{\text{EVTO}}$0 1 Activity defined by evto0_sense detected on $\overline{\text{EVTO}}$0
17 evto1_edge	In LFAST mode (DTM = 0) a channel B message is sent to the LFAST and the bit is automatically cleared. In software mode (DTM = 1), the bit stays set until cleared by writing to evto1_clr bit 0 No activity sensed on $\overline{\text{EVTO}}$1 1 Activity defined by evto1_sense detected on $\overline{\text{EVTO}}$1
18 SPU_INT_CLR	Write 1 clears the SPU_INT(1) This is a self clearing bit. If SPU_INT is not set then SPU_INT_CLR remains set until SPU_INT is set, and both are cleared.
19 SPU_INT	This bit is set when the SPU interrupt request input rising edge is detected. Writing a 1 to SPU_INT_CLR resets this bit. (1)
20 CRC_err	CRC error bit. Set when the calculated CRC does not match the last received CRC. This bit is cleared when the error packet is sent to the LFAST on channel B.
21 Nexus_err	Nexus error bit. It is set when there is an error, else it is 0.
22 Idle	Idle bit. Idle bit is cleared while a JTAG frame is in progress. This bit is set to ‘1’ when a JTAG frame ends. Writing a 1 to the Idle bit clears the Idle_interrupt in SW Mode. A read always gives the JTAG frame status but does not give the write value.

Table 2462. JTAGM_SR register field descriptions (continued)

Field	Description
23 NR	Status of Ready bit from Nexus RWA. This bit is automatically cleared when a new JTAG message is started. 0 RDY has not asserted indicating Nexus RWA is not ready 1 RDY has asserted indicating a Nexus RWA has completed bus access and is ready for the data register to be read **Note:** In LFAST mode(when DTM =0) , this bit has no effect and acts as a “dont care” . This bit should not be used by the application in the LFAST mode
24:31 CRC	Calculated CRC. CRC calculated on the information received. LFAST Mode: Bit 24:31 reads the CRC. Write has no effect. SW Mode: – Bit 24:31 reads as 0x00. Writing a 0 to any bit have no effect. – Writing 1 on bit 24 clears evto0_edge – Writing 1 on bit 25 clears evto1_edge

1. The BD JTAGM instance does not receive signals from the SPU. If required, this bit needs to be read from the PD instance of the JTAGM, even when JTAG generation in LFAST mode is handled by the BD JTAGM.

104.4.1.3 Data out register 0 (JTAGM_DOR0)

Figure 2585 shows the format of the JTAGM_DOR0.

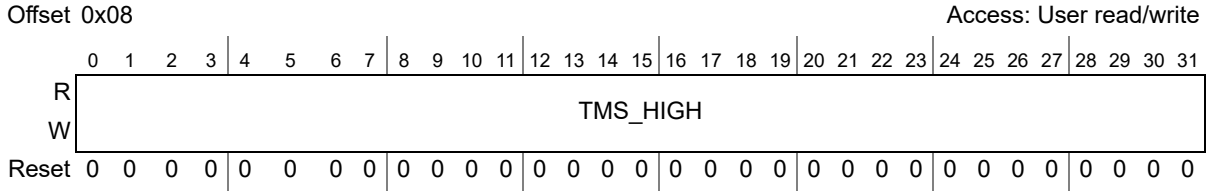


Figure 2585. JTAGM_DOR0 register

The JTAGM_DOR0 register is described in *Table 2463*.

Table 2463. JTAGM_DOR0 register field descriptions

Field	Description
0:31 TMS_HIGH	Higher word of data for TMS, bits 59-28. Writable by software only when DTM = 1

104.4.1.4 Data out register 1 (JTAGM_DOR1)

Figure 2586 shows the format of the JTAGM_DOR1.

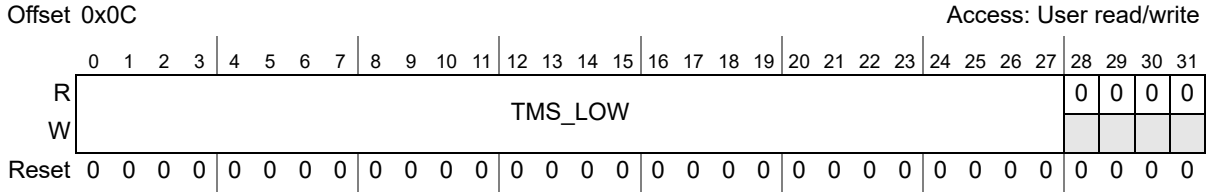


Figure 2586. JTAGM_DOR1 register

The JTAGM_DOR1 register is described in *Table 2464*.

Table 2464. JTAGM_DOR1 register field descriptions

Field	Description
0:27 TMS_LOW	Lower word of data for TMS, bits 27–0. Writable by software only when DTM = 1. TMS_LOW[0] is shifted out first

104.4.1.5 Data out register 2 (JTAGM_DOR2)

Figure 2587 shows the format of the JTAGM_DOR2.

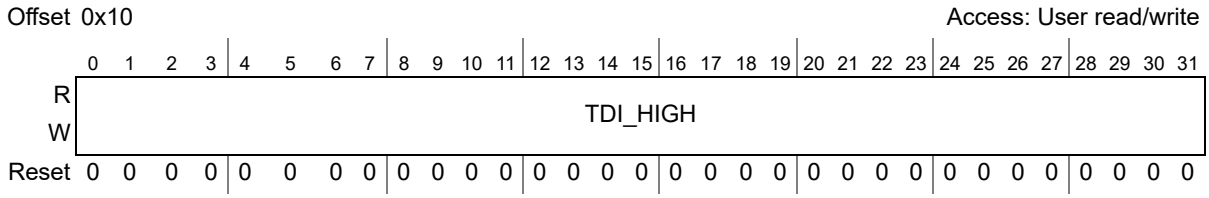


Figure 2587. JTAGM_DOR2 register

The JTAGM_DOR2 register is described in *Table 2465*.

Table 2465. JTAGM_DOR2 register field descriptions

Field	Description
0:31 TDI_HIGH	Higher word of data for TDI, bits 59–28. Writable by software only when DTM = 1

104.4.1.6 Data out register 3 (JTAGM_DOR3)

Figure 2588 shows the format of the JTAGM_DOR3.

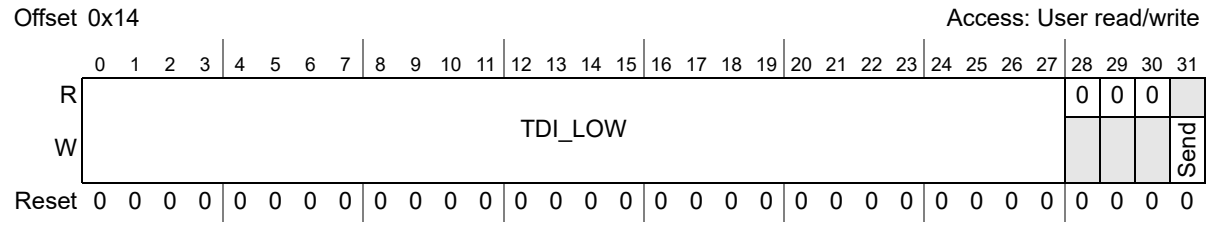


Figure 2588. JTAGM_DOR3 register

The JTAGM_DOR3 register is described in *Table 2466*.

Table 2466. JTAGM_DOR3 register field descriptions

Field	Description
0:27 TDI_LOW	Lower word of data for TDI, bits 27–0. Writable by software only when DTM = 1. TDI_LOW[0] is shifted out first
31 Send	Send bit. When this bit is set, data is sent to the DCI. It is a self-clearing bit and is always read as 0. Writable by software only when DTM = 1

104.4.1.7 Receive CRC register (JTAGM_RxCRC)

Figure 2589 shows the format of the JTAGM_RxCRC register.

Offset 0x18 Access: User read/write

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

Figure 2589. JTAGM_RxCRC register

The JTAGM_RxCRC register is described in *Table 2467*.

Table 2467. JTAGM_RxCRC register field descriptions

Field	Description
24:31 CRC	Received CRC. Refers to the CRC of the received LFAST message.

104.4.1.8 Data input register 0 (JTAGM_DIR0)

Figure 2590 shows the format of the JTAGM_DIR0.

Offset 0x1C Access: Read

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

Figure 2590. JTAGM_DIR0 register

The JTAGM_DIR0 register is described in *Table 2468*.

Table 2468. JTAGM_DIR0 register field descriptions

Field	Description
0:31 TDO_LOW	Lower word of data received on TDO, bits 0–31

104.4.1.9 Data input register 1 (JTAGM_DIR1)

Figure 2591 shows the format of the JTAGM_DIR1.

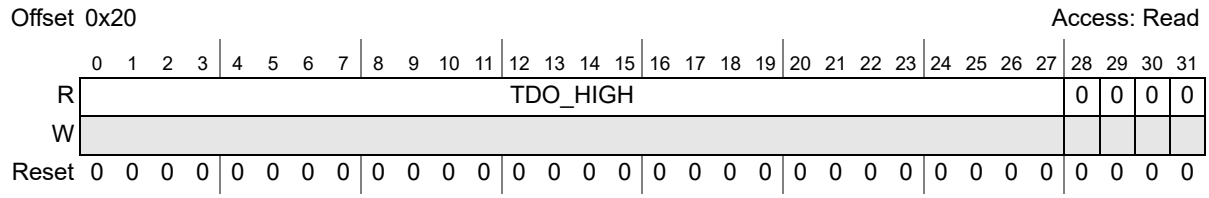


Figure 2591. JJTAGM_DIR1 register

The JTAGM_DIR1 register is described in *Table 2469*.

Table 2469. JTAGM_DIR1 register field descriptions

Field	Description
0:27 TDO_HIGH	Higher word of data received on TDO, bits 32–59

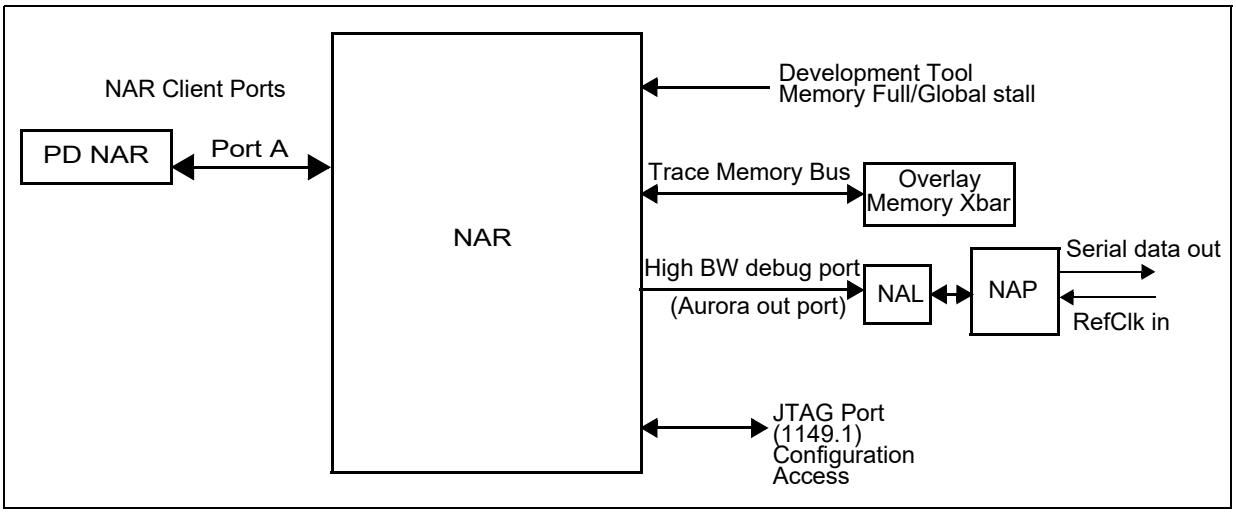
105 Buddy Device—Nexus Aurora Router (NAR)

105.1 Introduction

On the device, there are multiple blocks that require development interface support. The blocks that are Nexus-compliant (based on the IEEE-ISTO 5001 standard) are expected to interface with the development tool through a dedicated high-bandwidth debug port. The Nexus Aurora Router (NAR) controls the usage of the output port in a manner that allows all the individual development interface blocks to share the port, and appear to the development tool to be a single block.

Top-level architecture is shown in *Figure 2592*. Multiple clients send messages by sharing the high-bandwidth port through the NAR. For a list of the specific clients that are connected to the NAR, see the Calibration and Debug configuration chapter which describes how the modules are configured. The Nexus Aurora Link (NAL) takes the 32-bit Nexus message beats from the NAR and processes them in Aurora-ready data octets for the Nexus Aurora Phy (NAP) block, which interfaces directly with the high-bandwidth port. The Aurora output port of NAR can be fed as a client to another NAR.

Figure 2592. NAR top level diagram



The trace memory bus port provides added flexibility to the channel trace data to memory on the Nexus Aurora crossbar. Setup and configuration of the NAR is performed by accessing the configuration registers via a JTAG (1149.1) port.

Each Nexus client has a dedicated receive queue to allow seamless switching between sources at Nexus message boundaries. Incoming Nexus data is parsed into the receive queues so that end-of-message can be determined. Messages are then transferred into the main queue. An arbitration block determines the order in which clients are serviced, using an optional client request level bus to prioritize the data (lacking request level data, a simple round-robin algorithm is used). The NAR has two output ports: the high bandwidth debug port (Aurora out) and the trace memory bus. If both output ports are enabled at the same time, the main queue is logically partitioned inside NAR. These two partitions of the main queue are supported for better throughput each having 50% size. The two NAR output ports can be in operation simultaneously, with the output arbitration block splitting out messages by type or by source. Data going through the Aurora out port or the Trace port goes through an additional translation block to put the data in the correct format. The actual Aurora

formatting is done in the NAL. Additional blocks for message generation and timestamp generation are described in more detail in the following sections.

The Buddy Device (BD) NAR supports the following client interface:

- Aurora type interface with clock domain transfer at the same frequency

105.2 Overview

The NAR serves as an arbiter for the high-bandwidth debug port (HBDP). The NAR pulls Nexus messages from multiple clients, queues the data, and forwards it (based on the user-controlled configuration) to the high-speed Aurora Out port or the Trace port, or both The trace memory bus controller allows the NAR to transmit data via the normal system interface. NAR control and configuration registers are accessed via JTAG (1149.1) port.

105.3 Features

The NAR supports the following features:

- Connects 10 client(s) with dedicated receive queue(s). Clients can transmit data as long as there is space in its associated receive queue, but only the active port receive queue empties into the main queue.
- Aurora output data format for dedicated high-bandwidth serial transmission
- Aurora output format support at client interface
- Trace Memory port (64-bit address and data) can be used to write Nexus messages directly to external overlay memory. Block transfer mode provides highest-bandwidth option.
- Configurable Start Address and Range Selection for Trace Memory port write operation provides flexibility to choose Overlay memory Section and range to store Trace data.
- Both output ports may operate simultaneously. Filtering logic sorts messages to determine destination port.
- Partitionable main queue makes it possible to simultaneously service multiple output ports. It can also be used to devote more bandwidth to high-priority message-types or split trace stream between on-chip memory and trace port.
- Re-allocatable receive queues allow queue-space of an inactive client to be reassigned to an active client
- Stall detection and handling allows NAR to continue when a client stops sending data in the middle of a message
- Programmable Global stall signal on debug tool indication of memory overflow
- Event generation on External Trace Memory (overlay) programmable full
- Suppress mode can be used to shut down message delivery and relieve back-pressure to clients without turning off Nexus messaging in the clients.

105.4 Register definition

The NAR configuration registers provide configuration and control for all NAR operations. They are accessible through the JTAG (1149.1) port. Addressing (within the local NAR address space) for the NAR registers is shown in the following table. If the NAR clock is turned off, the NAR registers cannot be accessed.

Table 2470. NAR registers

Register name	Read/ Write	JTAG index (NAR relative)	Location
NAR_CR—NAR Control Register	R/W	0	on page 4164
NAR_ST—NAR Status Register	R	1	on page 4166
NAR_SFR—Source Filter Register	R/W	2	on page 4168
NAR_TFR—Type Filter Register	R/W	3	on page 4170
NAR_TBALO—Trace memory port base address, low	R/W	4	on page 4171
NAR_TBAHI—Trace memory port base address, high	R/W	5	on page 4171
NAR_TCR—Trace Memory bus control register	R/W	6	on page 4172
NAR_STCR—Suppress Trigger Control Registers 1–4	R/W	7 (Trigger 1) 8 (Trigger 2) 9 (Trigger 3) A (Trigger 4)	on page 4173
NAR_CSSR–Client Suppress Status Register	R	B	on page 4174
NAR_CDR—Client Disable Register	R/W	C	on page 4175
NAR_NAPCR—Nexus Aurora Phy Control Register	R/W	D	on page 4177
NAR_AHFPAR—Trace memory fill level and partition configuration	R/W	E	on page 4177

105.4.1 Register descriptions

105.4.1.1 NAR control register (NAR_CR)

The NAR control register provides all the main control and enable bits for the NAR and its output ports. The following figure shows the format of the NAR_CR.

Register index: 0

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	NEN	NCD	TEN	TRS	TSG		SED	SRD	TBW	AQP		0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	WPE			0	0	MSP		0	ALT		0	AFA	0	FSD	PUS	NSR
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 2593. NAR control register (NAR_CR)

The NAR_CR is described in the following table.

Table 2471. NAR_CR field descriptions

Field	Description
31 NEN	NAR Enable. 0 NAR is disabled unless HBDP is independently enabled. See *Section 105.5.2.2: Disabled* for more information. 1 NAR is enabled.
30 NCD	NAR Communication Disable. 0 NAR monitors clients for activity 1 NAR communication with clients is disabled
29 TEN	Timestamp Enable. 0 Timestamp packet is not included in messages 1 Timestamp packet is included in messages from all client including NAR generated messages
28 TRS	Timestamp Reset. 0 to 1 transition: reset timestamp counter in NAR
27–26 TSG	Timestamp Granularity (Mode). *Note: Some clients may only support timestamps on every message.* 00 Add timestamp to every message 01 Add timestamp to every 4th message 10 Add timestamp to every 16th message 11 Add timestamp to every 32nd message
25 SED	Stall Error Disable. 0 NAR injects dummy complete message when active client stalls 1 NAR does not inject dummy complete message
24 SRD	Stall Recovery Disable. 0 Stalled receive queue resumes operation when its client comes back on line 1 Stall error recovery is disabled
23 TBW	Tracebuffer Wrap. 0 NAR does not capture new data when main queue is full 1 NAR begins overwriting main queue from the beginning upon queue overflow
22–21 AQP	Alternate Queue Partition. 00 Queue partitioning is disabled 01 Reserved 10 1/2 of queue is dedicated to alternate partition 11 Reserved
20	Reserved and must be written with 0
19–16	Reserved and must be written with 0s
15–13 WPE	Watchpoint Enable 000 NAR watchpoint messaging is disabled xx1 NAR generates a watchpoint message upon a sync event x1x NAR generates a watchpoint message upon a trace memory bus or tracebuffer block full/wrap event 1xx NAR generates a watchpoint message upon a suppress-mode event
12	Reserved
11	Reserved. This bit must be written with 0.

Table 2471. NAR_CR field descriptions (continued)

Field	Description
10–9 MSP	Message Select Port. Defines which port is the target for message type/source filtering. Only active in two-port applications. 00 No message select port. 0–1 port is active. 01 HBDP is the MSP 10 Trace memory bus is the MSP 11 Reserved
8	Reserved
7–6 ALT	Alternate Port. Defines which port is the target for non-filtered messages; main port in single-port applications. 00 No alternate port. Running in trace-buffer mode. 01 HBDP is the ALT 10 Trace memory bus is the ALT 11 Reserved
5	Reserved
4 AFA	Automatic Flush Accumulators. 0 Output queue accumulators do not flush stranded beats 1 Accumulators automatically flush stranded beats
3	Reserved. This bit must be written with 0.
2 FSD	Frame Serial Data. 0 Framing disabled. Data is streamed through the Aurora out port. 1 Framing enabled. Start/End frame symbol inserted between every 256 Aurora words.
1 PUS	Power-Up Suppressed. 0 NAR powers up active 1 NAR powers up in global suppressed mode
0 NSR	NAR Soft Reset. 1 Resets all configuration registers and sets queue address pointers to the top of their respective queues. Also resets itself after a single-cycle delay. (This does not affect other blocks)

105.4.1.2 NAR status register (NAR_ST)

The NAR_ST register provides information on NAR activity and queue status, as well as HBDP status and errors. It is a read-only register. The following figure shows the format of the NAR_ST.

Register index: 1

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	NEN	NUM			NCE				MPF	MLV		APF	ALV		0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	HPR	HPE	MPE	MOV	0	TOV	QWA									
W		w1c	w1c	w1c		w1c										
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

Figure 2594. NAR status register (NAR_ST)

The NAR_ST register is described in the following table.

Table 2472. NAR_ST field descriptions

Field	Description
31 NEN	NAR Enabled. 0 NAR is disabled and inactive 1 NAR is enabled with clocks on
30–28 NUM	NAR Mode. 000 NAR is in disabled mode 001 NAR is in global suppressed mode 010 NAR is in communication-disabled mode 011 NAR is in tracebuffer mode 100 NAR is in active transmit mode 101 Reserved 110 Reserved 111 NAR is in run-control mode
27–24 NCE	NAR Config Error. See *Table 2487* for coding
23 MPF	MSP Partition Full. 0 MSP partition not full 1 MSP partition full
22–21 MLV	MSP Partition Queue Level. This field is available only when both output ports are enabled that is the main queue is partitioned. 00 0% < queue level < 25% 01 25% < queue level < 50% 10 50% < queue level < 75% 11 75% < queue level <100%
20 APF	ALT Partition Full. 0 ALT partition not full 1 ALT partition full
19–18 ALV	ALT Partition Queue Level. 00 0% < queue level < 25% 01 25% < queue level < 50% 10 50% < queue level < 75% 11 75% < queue level <100%

Table 2472. NAR_ST field descriptions (continued)

Field	Description
17–16	Reserved
15 HPR	HBDP Ready. 0 HBDP (Aurora Out port) is not ready 1 HBDP is ready and accepts data for transmission
14 HPE	HBDP Error. 0 No HBDP error 1 Error condition exists on HBDP
13 MPE	Trace memory bus Error. 0 No Trace memory bus error 1 Error condition detected on trace transaction
12 MOV	Trace memory bus block overflow. 0 No Trace memory bus overflow 1 Targeted external memory block is full. Trace memory bus writes are halted (or wrapped).
11	Reserved
10 TOV	Tracebuffer Overflow. 0 No tracebuffer overflow 1 The tracebuffer has overflowed/wrapped at least once
9–0 QWA	Queue Write Address. Current value of the output queue write pointer of ALT partition (last valid write + 1)

105.4.2 Message filtering registers

The two output ports (HBDP and trace memory bus) are independently controllable. In the BD NAR, two ports can be active simultaneously. When multiple ports are active, user-programmable filters are determine which port receives a message, based on the message source or type. Filters apply to the designated Message Select Port (MSP), and are used to select messages for that port. All other messages go to the other (alternate) port. If multiple ports are used, an MSP must be designated; otherwise a configuration error results. See *Section 105.5.6: Output arbitration*, for details on how the NAR arbitrates between multiple output ports. These registers should only be set in the BD NAR because partitioning is not supported in the PD NAR.

105.4.2.1 Source filter register (NAR_SFR)

The MSP can be configured to accept or exclude messages from up to four sources. The sources which are to be filtered are programmed in the source filter register (NAR_SFR) as shown in the following figure. This register should only be set in the BD NAR because partitioning is not supported in the PD NAR.

Register index: 2

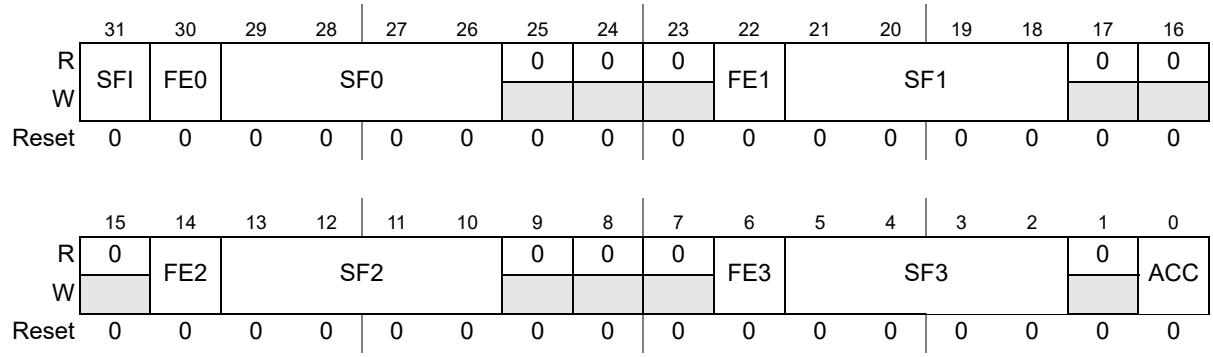


Figure 2595. Source filter register (NAR_SFR)

The NAR_SFR is described in the following table. The source trace IDs, that are compared to the source filter fields (SFn), are device specific. For a list of the clients that are connected to the NAR and their corresponding Nexus source trace IDs, see the Calibration and Debug configuration chapter which describes how the modules are configured.

Table 2473. NAR_SFR field descriptions

Field	Description
31 SFI	Source Filter Invert. 0 Match if message source equals any of the enabled filters 1 Match if message source equals none of the enabled filters
30 FE0	Enable Filter 0. 0 Filter 0 disabled. SF0 ignored 1 Filter 0 enabled and contained in SF0
29–26 SF0	Source Filter 0. The message source trace ID is compared with this field. See *Section 105.5.6: Output arbitration* for an example.
25–23	Reserved.
22 FE1	Enable Filter 1. 0 Filter 1 disabled. SF1 ignored 1 Filter 1 enabled and contained in SF1
21–18 SF1	Source Filter 1. The message source trace ID is compared with this field. See *Section 105.5.6: Output arbitration* for an example.
17–15	Reserved
14 FE2	Enable Filter 2. 0 Filter 2 disabled. SF2 ignored 1 Filter 2 enabled and contained in SF2
13–10 SF2	Source Filter 2. The message source trace ID is compared with this field. See *Section 105.5.6: Output arbitration* for an example.
9–7	Reserved
6 FE3	Enable Filter 3. 0 Filter 3 disabled. SF3 ignored 1 Filter 3 enabled and contained in SF3

Table 2473. NAR_SFR field descriptions (continued)

5–2 SF3	Source Filter 3. The message source trace ID is compared with this field. See *Section 105.5.6: Output arbitration* for an example.
1	Reserved
0 ACC	OR source and type filters. 0 Match if source AND type filters match 1 Match if source OR type filters match

105.4.2.2 Message type filter register (NAR_TFR)

The MSP can be configured to accept or exclude messages of up to four types, as encoded in the message TCODE field. The TCODEs which are to be filtered are programmed in the message type filter register (NAR_TFR) as shown in the following figure. This register should only be set in the BD NAR because partitioning is not supported in the PD NAR.

Register index: 3

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	TFI	FE0	TF0						0	FE1	TF1					
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	FE2	TF2						0	FE3	TF3					
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 2596. Message type filter register (NAR_TFR)

The NAR_TFR is described in the following table.

Table 2474. NAR_TFR field descriptions

Field	Description
31 TFI	Type Filter Invert. 0 Match if message TCODE equals any of the enabled filters 1 Match if message TCODE equals none of the enabled filters
30 FE0	Enable Filter 0. 0 Filter 0 disabled. TF0 ignored 1 Filter 0 enabled and contained in TF0
29–24 TF0	Type Filter 0. The message TCODE is compared with this field. See *Section 105.5.6: Output arbitration* for an example.
23	Reserved
22 FE1	Enable Filter 1. 0 Filter 1 disabled. TF1 ignored 1 Filter 1 enabled and contained in TF1
21–16 TF1	Type Filter 1. The message TCODE is compared with this field. See *Section 105.5.6: Output arbitration* for an example.

Table 2474. NAR_TFR field descriptions (continued)

15	Reserved
16 FE2	Enable Filter 2. 0 Filter 2 disabled. TF2 ignored 1 Filter 2 enabled and contained in TF2
13–8 TF2	Type Filter 2. The message TCODE is compared with this field. See *Section 105.5.6: Output arbitration* for an example.
7	Reserved
6 FE3	Enable Filter 3. 0 Filter 3 disabled. TF3 ignored 1 Filter 3 enabled and contained in TF3
5–0 SF3	Type Filter 3. The message TCODE is compared with this field. See *Section 105.5.6: Output arbitration* for an example.

105.4.3 Trace memory bus configuration registers

The trace memory bus can be used to transfer debug messages out to a dedicated block of external memory space. The trace memory bus configuration registers are used to specify the location and size of the external debug space.

105.4.3.1 Trace memory bus base address registers (NAR_TBALO and NAR_TBAHI)

The trace memory bus base address registers determine where Nexus data sent out through the trace memory bus goes. The NAR sends out blocks of data on the trace memory bus, writing to an address block defined by the specified base address. Data can transmitted either on an as-available, word-wide basis, or in larger blocks. In block transfer mode, the size of the data block is determined by the BTM field of the Control Register; the default is 64 bytes. Even if block transfer mode is not used, the 5, 6, 7, or 8 (depending on the BTS field in NAR_TCR register) LSBs of the base address are not accessible. This forces the debug memory space to be aligned to a block boundary.

When read, the address registers return the current address, allowing the user to monitor the progress of a trace memory transfer. The following figures show the format of the trace memory base address registers (NAR_TBALO and NAR_TBAHI). For a description of the allowed address for these registers, see the Calibration and Debug configuration chapter which describes how the modules are configured.

Register index: 5

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

Figure 2597. Trace memory base address register NAR_TBAHI

Register index: 4

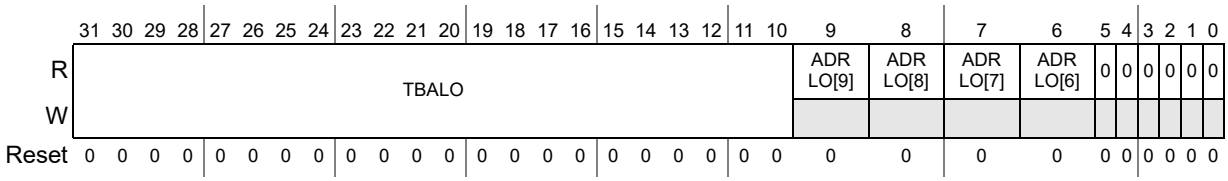


Figure 2598. Trace memory base address register NAR_TBALO

The NAR_TBALO and NAR_TBAHI registers are described in the following tables.

Table 2475. NAR_TBAHI field descriptions

Field	Description
31–0 TBAHI	Trace Debug Memory Base Address on AHB-EW port, upper part. For 32-bit address applications this field is all zeros.

Table 2476. NAR_TBALO field descriptions

Field	Description
31–10 TBALO	Trace Debug Memory Base Address on AHB-EW port, lower part
9 ADRLO[9]	Base Address LSB for BTS = 11 (384-byte block)
8 ADRLO[8]	Base Address LSB for BTS = 10 (256-byte block)
7 ADRLO[7]	Base Address LSB for BTS = 01 (128-byte block)
6 ADRLO[6]	Base Address LSB for BTS = 00 (64-byte block)
5–0	Reserved

105.4.3.2 Trace memory bus control register (NAR_TCR)

The trace memory bus control register is used to control the attributes of the trace memory bus port, such as block transfer size and message priority. Since the trace memory bus transfers are expected to go to a static block of external memory, a maximum transfer size must be specified to prevent the NAR from overwriting memory it is not authorized to access. This is done via the MXFR max transfer size register field. The memory allocation is specified in units of the trace memory bus block transfer size (64, 128, 256, or 384 bytes), so the actual size of the reserved memory block depends on the BTS setting. The following figure shows the format of the NAR_TCR.

Register index: 6

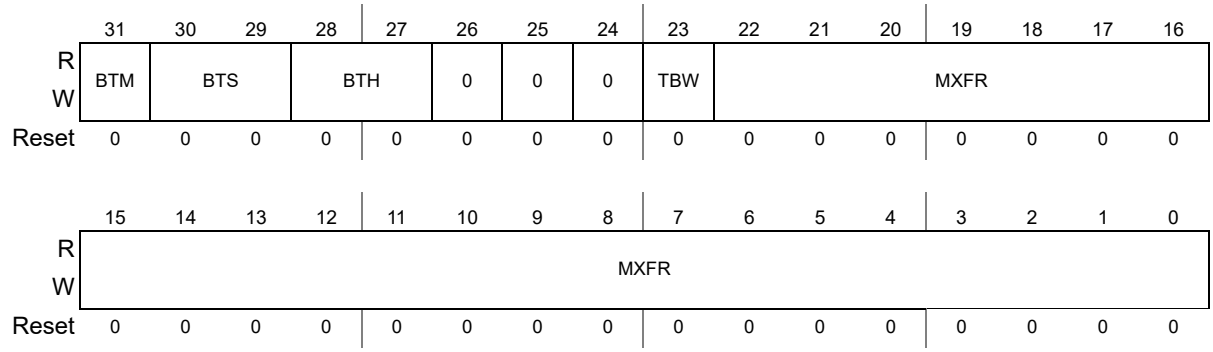


Figure 2599. Trace memory bus control register (NAR_TCR)

The NAR_TCR is described in the following table.

Table 2477. NAR_TCR field description

Field	Description
31 BTM	Block Transfer Mode. 0 Data for the Trace Memory bus port is transmitted one word at a time, upon first availability 1 Data to the Trace Memory bus port is bundled into blocks before transmitting
30–29 BTS	Block Transfer Size. 00 Block transfer size = 64 bytes 01 Block transfer size = 128 bytes 10 Block transfer size = 256 bytes 11 Block transfer size = 384 bytes
28–27 BTH	Block Transfer Threshold. 00 Full threshold. Number of entries in partition must equal block transfer size before block transfer starts 01 Begin transfer when block is 25% full 10 Begin transfer when block is 50% full 11 Begin transfer when block is 75% full
26–24	Reserved. These bits must be written with 0.
32 TBW	Trace Memory Bus Block Wrap. 0 NAR stops writing to the Trace Memory bus debug memory space once Max Size has been reached 1 NAR overwrites Trace Memory bus debug memory from the beginning upon Trace Memory bus block overflow.
22–0 MXFR	Trace Memory Bus Max Transfer Size. Total size of the dedicated debug memory space, in Trace Memory bus data transfer blocks.

105.4.4 Suppress mode triggers register (NAR_STCR)

Four suppress mode triggers are provided for precise control of the NAR suppress mode feature described in *Section 105.5.4: Suppress mode*. The suppress mode triggers are only available on the PD NAR and do not apply to the BD NAR. The registers are present in the BD NAR, but input suppress triggers are not connected. Each trigger can be programmed to suppress or wake clients individually or the entire NAR. All receive queues can be suppressed on the occurrence of a selected trigger, or the full NAR can be brought out of suppress mode upon the occurrence of the appropriate trigger. A trigger can also be programmed to toggle one or more clients between the suppressed and active states. Each

Suppress Trigger Control Register, shown in the following figure, is associated with one of the triggers, and are used to determine the action taken, if any, upon the assertion of that trigger. The suppress triggers are connected to the Sequence Processing Unit (SPU) action unit.

Register index: 7–A

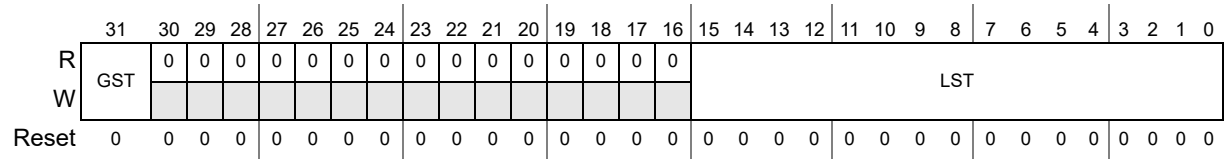


Figure 2600. Suppress trigger control registers (NAR_STCR1–4)

The NAR_STCR are described in the following table. The set of clients connected to the NAR is device specific. For a list of the clients that are connected to the NAR and their corresponding NAR client index for the local suppress trigger, see the Calibration and Debug configuration chapter which describes how the modules are configured.

Table 2478. NAR_STCR field descriptions

Field	Description
31 GST	Global Suppress Trigger. 0 The associated trigger cannot cause NAR to enter global suppress mode 1 The associated trigger is enabled for global suppress mode
29–16	Reserved
15–0 LST	Local Suppress Trigger. 0000000000000000The associated trigger is disabled for local suppress mode for all clients xxxxxxxxxxxxxxx1Trigger is enabled for Client 1 local suppress xxxxxxxxxxxxxx1xTrigger is enabled for Client 2 local suppress xxxxxxxxxxxxx1xxTrigger is enabled for Client 3 local suppress xxxxxxxxxxxx1xxxTrigger is enabled for Client 4 local suppress xxxxxxxxxxx1xxxxTrigger is enabled for Client 5 local suppress xxxxxxxxxx1xxxxxTrigger is enabled for Client 6 local suppress xxxxxxxxx1xxxxxxTrigger is enabled for Client 7 local suppress xxxxxxxx1xxxxxxxTrigger is enabled for Client 8 local suppress xxxxxxx1xxxxxxxxTrigger is enabled for Client 9 local suppress xxxxxx1xxxxxxxxxTrigger is enabled for Client 10 local suppress xxxxx1xxxxxxxxxxTrigger is enabled for Client 11 local suppress xxxx1xxxxxxxxxxxTrigger is enabled for Client 12 local suppress xxx1xxxxxxxxxxxxTrigger is enabled for Client 13 local suppress xx1xxxxxxxxxxxxxTrigger is enabled for Client 14 local suppress x1xxxxxxxxxxxxxxTrigger is enabled for Client 15 local suppress 1xxxxxxxxxxxxxxxTrigger is enabled for Client 16 local suppress All other options Reserved

105.4.4.1 Client suppress status register (NAR_CSSR)

Clients can be individually suppressed, leaving the rest of the NAR unaffected. When enabled for local suppress, assertion of one of the suppress triggers causes one or more client receive queues to stop queueing messages. A subsequent pulse of the same trigger causes the client(s) to awaken. Local suppress mode is enabled by setting the appropriate bit in one of the NAR_STCRs. An additional status register, Client Suppress Status

(NAR_CSSR), shown in the following figure, can then be read to determine which clients are suppressed, and which are active.

Register index: B

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

Figure 2601. Client suppress status register (NAR_CSSR)

The NAR_CSSR is described in the following table. The set of clients connected to the NAR is device specific. For a list of the clients that are connected to the NAR and their corresponding NAR client index for the local suppress status, see the Calibration and Debug configuration chapter which describes how the modules are configured.

Table 2479. NAR_CCSR field descriptions

Field	Description
31–16 LSS	Local Suppress Status. 0000000000000000No client is in local suppress mode xxxxxxxxxxxxxxx1Client 1 is suppressed xxxxxxxxxxxxxx1xClient 2 is suppressed xxxxxxxxxxxxx1xxClient 3 is suppressed xxxxxxxxxxxx1xxxClient 4 is suppressed xxxxxxxxxxx1xxxxClient 5 is suppressed xxxxxxxxxx1xxxxxClient 6 is suppressed xxxxxxxxx1xxxxxxClient 7 is suppressed xxxxxxxx1xxxxxxxClient 8 is suppressed xxxxxxx1xxxxxxxxClient 9 is suppressed xxxxxx1xxxxxxxxxClient 10 is suppressed xxxxx1xxxxxxxxxxClient 11 is suppressed xxxx1xxxxxxxxxxxClient 12 is suppressed xxx1xxxxxxxxxxxxClient 13 is suppressed xx1xxxxxxxxxxxxxClient 14 is suppressed x1xxxxxxxxxxxxxxClient 15 is suppressed 1xxxxxxxxxxxxxxxClient 16 is suppressed All other optionsReserved
15–0	Reserved

105.4.5 Receive queue client disable register (NAR_CDR)

Disabling a client reallocates its receive queue to the next active client, allowing that memory space to be put to use. The Client Disable Register, shown in the following figure, allows the user to individually disable or enable clients as needed. The status bits also inform the user when the client has new data available.

Note: Even though the BD NAR is described as only having one client, there is actually a second dummy client, client 2. To re-allocate the dummy RX queue to client 1 (the PD NAR), set the BD NAR client 2 RQD bit.

Register index: C

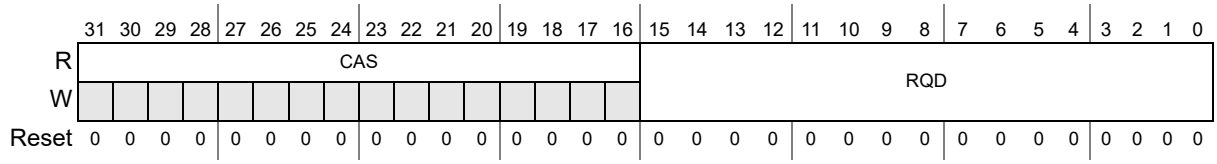


Figure 2602. Client disable register (NAR_CDR)

The NAR_CDR is described in the following table. The set of clients connected to the NAR is device specific. For a list of the clients that are connected to the NAR and their corresponding NAR client index for the client activity status and receive queue disable, see the Calibration and Debug configuration chapter which describes how the modules are configured.

Table 2480. NAR_CDR field descriptions

Field	Description
31–16 CAS	Client Activity Status. xxxxxxxxxxxxxxx1 Client 1 is active with new data xxxxxxxxxxxxxx1x Client 2 is active with new data xxxxxxxxxxxxx1xx Client 3 is active with new data xxxxxxxxxxxx1xxx Client 4 is active with new data xxxxxxxxxxx1xxxx Client 5 is active with new data xxxxxxxxxx1xxxxx Client 6 is active with new data xxxxxxxxx1xxxxxx Client 7 is active with new data xxxxxxxx1xxxxxxx Client 8 is active with new data xxxxxxx1xxxxxxxx Client 9 is active with new data xxxxxx1xxxxxxxxx Client 10 is active with new data xxxxx1xxxxxxxxxx Client 11 is active with new data xxxx1xxxxxxxxxxx Client 12 is active with new data xxx1xxxxxxxxxxxx Client 13 is active with new data xx1xxxxxxxxxxxxx Client 14 is active with new data x1xxxxxxxxxxxxxx Client 15 is active with new data 1xxxxxxxxxxxxxxx Client 16 is active with new data All other optionsReserved
15–0 RQD	Receive Queue Disable. xxxxxxxxxxxxxxx1 Disable receive queue for Client 1 xxxxxxxxxxxxxx1x Disable receive queue for Client 2 xxxxxxxxxxxxx1xx Disable receive queue for Client 3 xxxxxxxxxxxx1xxx Disable receive queue for Client 4 xxxxxxxxxxx1xxxx Disable receive queue for Client 5 xxxxxxxxxx1xxxxx Disable receive queue for Client 6 xxxxxxxxx1xxxxxx Disable receive queue for Client 7 xxxxxxxx1xxxxxxx Disable receive queue for Client 8 xxxxxxx1xxxxxxxx Disable receive queue for Client 9 xxxxxx1xxxxxxxxx Disable receive queue for Client 10 xxxxx1xxxxxxxxxx Disable receive queue for Client 11 xxxx1xxxxxxxxxxx Disable receive queue for Client 12 xxx1xxxxxxxxxxxx Disable receive queue for Client 13 xx1xxxxxxxxxxxxx Disable receive queue for Client 14 x1xxxxxxxxxxxxxx Disable receive queue for Client 15 1xxxxxxxxxxxxxxx Disable receive queue for Client 16 All other optionsReserved

105.4.6 Nexus Aurora Phy control register (NAR_NAPCR)

The Nexus Aurora Phy control register provides configuration control to number of HBDP lanes of Nexus Aurora Phy and LVDS pad control signals. The following figure shows the format of the NAR_NAPCR.

Register index: D

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	NUM_LANE			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0		0	LVDS_TXOPT				LVDS_RX_ TERM_EN	LVDS_BIAS_ PRG	LVDS_RXOPT					
W																
Reset	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0

Figure 2603. Nexus Aurora Phy control register (NAR_NAPCR)

The NAR_NAPCR is described in the following table.

Table 2481. NAR_NAPCR field descriptions

Field	Description
19–16 NUM_LANE	TX Lane Configuration. Number of enabled TX lanes. This field is device specific. For a description of this field see the Calibration and Debug configuration chapter which describes how the modules are configured.
11–8 LVDS_TXOPT	This is an internal characterization field.
7 LVDS_RX_TERM_EN	This is an internal characterization field.
6 LVDS_BIAS_PRG	This is an internal characterization field.
5–0 LVDS_RXOPT	This is an internal characterization field.

105.4.7 Trace Memory fill level and partition configuration register (NAR_AHFPAR)

This register provides the configuration to set the fill level marker for debug trace memory, connected at the trace memory bus, to generate the partial full event indication and the partition related configuration for main queue (internal central queue). The following figure shows the format of the NAR_AHFPAR.

Register index: E

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	GSE	0	0	0	GLOBAL_STALL_COUNT				
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0				

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	AHB65kEN	AHB01kEN	PART_CFG		AHB_65K_Marker				AHB_01K_Marker			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 2604. Trace memory fill level and partition configuration register (NAR_AHFPAR)

The NAR_AHFPAR is described in the following table.

Table 2482. NAR_AHFPAR field description

Field	Description
31-25	Reserved and must be written with 0s
24 GSE	Global Stall Enable. Setting this bit enables the NAR to act on dci_evti[0] event to generate a global stall
23-21	Reserved and must be written with 0s
20-16 GLOBAL_STALL_COUNT	Global Stall Count. Value at this field determines how many cycles, in the NAR's operating clock, the global stall remains asserted on a dci_evti event. This is a five bit field so the maximum value is 31 cycles. See *Section 105.5.13: Global (Development tool client) stall control*
15-12	Reserved and must be written with 0s
11 AHB65kEN	Enable 65K Marker Event
10 AHB01kEN	Enable 1K Marker Event
9-8 PART_CFG	Partition Configuration. 00 No partition 01 Reserved 10 50% partition (in two partition two configuration case) 11 Reserved
7–4 AHB_65K_Marker	Value at this field is used to compare a 4-bit slice of internal Trace Memory address bus which works at boundary of 65K. A match with this value generates an event to SPU to indicate the fill status of External Overlay Trace memory. See *Section 105.5.14: Overlay and internal trace memory full status generation* **Note:** This is used in BD as it has memory ranges from 65K to 983K
3–0 AHB_01K_Marker	Value at this field is used to compare a 4-bit slice of internal Trace Memory address bus which works at boundary of 1K. A match with this value generates an event to SPU to indicate the fill status of External Overlay Trace memory. See *Section 105.5.14: Overlay and internal trace memory full status generation* **Note:** This is used in PD as it has memory ranges from 1K to 15K

105.5 Functional description

High-level descriptions of the operation of the NAR and its main functional blocks are given in the following sections.

105.5.1 Reset/Startup

Assertion of global reset causes the NAR operation to immediately cease. All queues and configuration registers are reset. Upon exiting reset, the Nexus Port Controller (NPC) block inside the NAR is disabled. Holding TMS high for five consecutive rising edges of TCK puts TAP controller into the Test-Logic-Reset state regardless of the current state. This applies reset to only TAP controller and JTAG related logic. Applying soft reset through the NAR_CR resets the entire NAR including the TAP controller and configuration registers.

The typical startup procedure for running a trace through the NPC is as follows:

1. Reset the block
2. Upon exiting reset, the Aurora Phy configuration is determined by sampling the configuration pin values from the Nexus Aurora Phy (NAP). If the NAP is enabled, then the NAR is enabled. The Aurora communication is started only when the NAP is ready to accept data. (If any debug lanes are enabled, the Aurora Link begins training at this time.)
3. Debugger/external user sets the NAR enable (NEN) bit in the NAR_CR and configures the NAR. The configuration includes setting up message selection/filtering when two output ports are enabled, enabling timestamp and so on.
4. User programs additional device level debug facilities, such as event-processing, performance-monitoring, and so on.
5. User sets up trace filtering.
6. Enable Nexus transmission in clients.

To enable NAR operation, either the Aurora Link must be enabled at reset, or the user must set the NEN bit in the NAR control register.

Note: The HBDP may already be active by the time the NAR comes up. The NAR only tries to initialize the HBDP if it is currently inactive.

The NAR can be enabled to begin immediate operation, or to start in suppress mode, see *Section 105.5.4: Suppress mode*. Once the NAR and desired ports are enabled, the user should wait until the NAR-ready and HBDP port-ready status bits are set in the NAR status register before beginning Nexus messaging in the clients.

105.5.2 Operation modes

The NAR operating mode is determined by the state of the main control signals: reset and Phy enable. In its enabled state, operation is controlled by its configuration registers. A truth table for all the NAR states is given in this table.

Table 2483. NAR operational modes truth table

Reset	NAR enable	NAR comm disable	Global suppress	HBDP enable	ALT partition	MSP partition	Mode
0	x	x	x	x	x	x	RESET
1	0	x	x	0	x	x	Disabled

Table 2483. NAR operational modes truth table (continued)

Reset	NAR enable	NAR comm disable	Global suppress	HBDP enable	ALT partition	MSP partition	Mode
1	1	1	x	0	x	x	Trace-Disable
1	x	1	x	1	x	x	Trace-Disable
1	0	x	x	1	x	x	Trace-Disable
1	1	0	1	x	x	x	Suppressed
1	1	0	0	x	Trace Memory bus	x	NAR Transmit
1	1	0	0	1	AUR	x	NAR Transmit
1	1	0	0	1	Trace Memory bus	AUR	NAR Transmit
1	1	0	0	1	AUR	Trace Memory bus	NAR Transmit

There can be overlap between modes as modes are not mutually exclusive.

105.5.2.1 Reset

During reset all internal resources and queues are reset and all output ports are disabled. All inputs are ignored. Upon exiting reset mode, the NAR is in a fully disabled state.

105.5.2.2 Disabled

In the disabled state, all message traffic is ignored, and all output ports are disabled. The NAR is disabled if and only if the NAR enable bit of the NAR control register is not set and the Aurora Link is inactive.

105.5.2.3 Suppressed

The suppressed state differs from the disabled state in that the NAR is on. The NAR pulls out any messages that come up at the clients, keeping the Message Start-End Out (MSEO) bits (so that it can keep track of message boundaries) and discarding the rest.

In the suppressed state, the NAR is waiting for a trigger to begin normal operation. Once that trigger is received, the NAR begins queueing messages that started after the trigger arrived. Incomplete messages that were in transit to the NAR when the trigger arrived are discarded. The NAR can be put into suppressed mode by enabling one of the global suppress states.

105.5.2.4 Trace disable

In trace disable mode, the NAR is powered up but all client-trace is disabled, and no messages are pulled. This mode is also useful for port initialization (to prevent any activity from occurring before the outputs are ready) or for debug/trace-buffer mode, to ensure that, during data retrieval, the NARs queues are stable.

The NAR goes into trace disable (communication-disable) mode when the NAR-enable and NAR communication-disable bits are set or when the HBDP channel is active while the NAR-enable bit is not set.

105.5.2.5 NAR transmit

This is the standard operational mode of the NAR when it is transferring Nexus messages from multiple clients to one or two enabled output port(s). The NAR performs four main operations as described in this table on the incoming messages before transmitting to one of the output ports.

Table 2484. NAR transmit operations

Operation	Description
Data reception	NAR accepts data from each client based on the status of the client receive queue status. The maximum data rate is 8 × 32-bit data words (where 8 is the number of Nexus message beats in a full entry and 32 is the number of bits in the Nexus word) per cycle per client. MSEO bits of incoming data are monitored to identify end-of-message boundaries.
Input Arbitration	Active receive queue unloads into main queue(s), 8 × 32 bits per cycle (where 8 is the number of Nexus message beats in a full entry and 32 is the number of bits in the Nexus word). If end-of-message detected, new active receive queue is determined based on client request level or round-robin algorithm.
Output Arbitration	If multiple output ports are enabled, route messages from top of queue to appropriate output port based on message type or source. If split queue is used, outputs can run simultaneously, with each sub-queue communicating directly with a port.
Translation	Data going out the Aurora or trace memory bus ports must be translated into the correct format prior to transmission.

Enabling the NAR and one or more output ports without setting the communication-disable bit puts the NAR into transmit mode.

105.5.3 Data reception and arbitration

The BD NAR supports 10 client (although a second dummy client is used to increase the RX buffer size for client 1). Each client has a dedicated input port on the NAR and communicates with its associated input port via a dedicated data bus. A data transmission must consist of an integer number of 32-bit Nexus words (32-bit word is called a beat). The number of beats varies from client to client. The number of beats (Bn) for each client are described in the Calibration and Debug configuration chapter which describes how the modules are configured.

There is one type of client supported at NAR client interface. One Nexus word consists of 30 Message Data Out (MDO) bits plus two Message Start/End Out (MSEO) control bits.

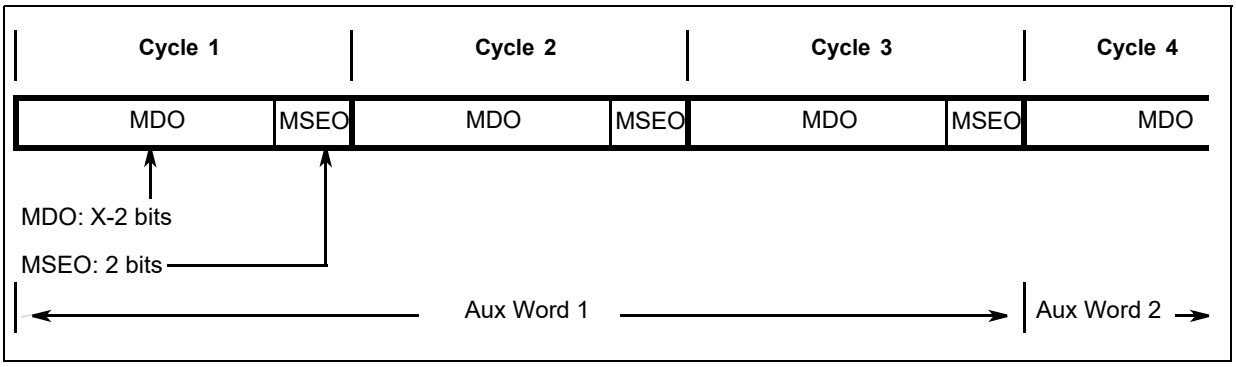
When new data is available in the core, it is transferred over to the asynchronous gasket in the internal NAR clock domain. The gasket then asserts enough data valids to fill internal receive queue, using a receive queue available signal to track available space in the associated queue, under the assumption that the receive queue available signal is initialized with the size of the queue. If the internal arbiter consumes any of its queued entries, it responds with a data accept signal, which increments the receive queue available signal and allows the gasket to send the next piece of data. Under certain circumstances, like receive queue reallocation, it is possible for the internal arbiter to send out surplus data accept signals such that the receive queue available signal can become larger than the size of the queue. If the receive queues are sized correctly, when NAR is actively servicing a particular client, then asserted data accept signals allow the client to continually stream data until the NAR switches to a new client. The data tracking at the gasket assures that it never

sends more data than the receive queue can accept and inactive queues are always full and ready for when they are switched to.

The client gaskets must always be in sync with the internal logic of NAR in terms of the number of available entries in each receive queue. For this reason, the internal NPC and its client gaskets need to be reset by the same signal. Any soft-reset of the NAR (via the SRESET configuration bit) is communicated to all gaskets and used to reset every thing inside it.

Each input port has a dedicated receive queue with a minimum of two entries, where an entry is wide enough to store an entire Bn $\times$ 32 data word (where 32 is the number of bits in the Nexus word or the beat). The receive queue can be made larger to allow the NAR to continue transmitting during switches from one client to another in a pipelined environment. Every cycle, if there is space in a particular receive queue, it latches in one full AUX word from its client. Because the client may be running at a much higher frequency than the NAR, several cycles worth of Nexus messages are contained in a single transaction, as shown in the following figure for a 3-beat per word transaction.

Figure 2605. AUX bus parsing



Each AUX word consists of Bn cycles of Nexus data, along with an optional request-level sideband signal consisting of a priority field (bit 0) and a weight field (bits 1 and 2). Request level priority indicates to the NAR the clients queue status, allowing it to give preferential treatment to clients that are closer to queue overflow. Clients with a set priority bit are always chosen ahead of clients without the bit. For this reason, priority should be used sparingly, only when a client is in danger of overflowing. Request-level weighting also gives a client preferential treatment, but without entirely shutting out the other clients. Normally, the NAR attempts to switch to a new client every time it sees an end-of-message boundary.

As the AUX word is pulled into the receive queue, the MSEO bits are monitored to determine the current Nexus state of the data flow; since Nexus is a variable-length protocol, messages can end in any cycle. If the start message or end message state is detected on any data beat, the beat is marked as containing an end-of-message boundary.

Any received beat which is in the idle state, indicating null data content, is discarded (see the IEEE-ISTO 5001-2003 standard for details of the Nexus protocol). At the same time, if two outputs are enabled, then the source and TCODE fields are extracted from each new message and compared with the user-programmed filters. Messages that generate a match are tagged with an MSP bit, which allows the NAR to route it to the correct port.

Every cycle, the NAR transfers 8 beats of data from the receive queues into the main output queue. Only one port (the active port) is allowed to empty into the main queue at a time: the active port unloads its receive queue into the main queue until the end of the current Nexus message is detected. At that point, a new active port is selected based on the latest request

level received at each port. In the case where multiple ports have the same request level (or when request level is not used), then the active port is rotated amongst those ports. If the end of message occurs in the middle of an AUX word, then all the beats up to the end of message are transferred from the old active port, while the rest of the beats come from the new active port. For example, if the message ended at beat two of a three-beat AUX word, then the first two beats into the main queue would come from the original active port, while the last beat would come from the new active port.

105.5.4 Suppress mode

The suppress mode feature enables the user to quickly turn Nexus messaging through the NAR on or off, based on the four dedicated suppress-mode triggers. Suppress mode is only supported in the PD NAR, it is not supported in the BD NAR. A suppress can either be global, meaning messaging is turned off for all clients, or local, which only applies to selected clients. In suppress mode, the NAR pulls messages from the suppressed client(s), but does not queue or deliver them. After extracting the MSEO bits to keep track of message boundaries, the rest of the message is discarded. This allows the NAR to better keep up with fast-running Nexus clients without having to turn off messaging in the client.

Typically, a client would be suppressed during less interesting periods, and would be awakened when some preset condition is detected. This is communicated to the NAR via the one of the suppress triggers, which can be sourced by a client or an external Event Processing block. When an enter suppress trigger is received, the NAR output queue finishes servicing the current active ports until the end of the message. If the end of the message occurs in a middle beat of a Nexus word, then the remainder of the word is padded with null messages. Once the last message has been loaded into the output queue, the receive queues are all emptied. New messages are pulled from clients as they appear, go through their respective receive queues and then are discarded. In this way, the NAR can naturally monitor the MSEO bits and keep track of each client's Nexus state. In suppress mode, the output ports remain active until the output queue is emptied, meaning there is a delay between the assertion of the suppress trigger and the end of messaging.

When the NAR is in suppress mode and receives a wake-up trigger, it continues discarding any partial messages still in the receive queues. Any message that begins after the trigger is queued normally, insuring that only complete messages are sent to the output queue.

Global suppress mode applies to all clients. Any of the suppress triggers can be programmed to cause the NAR to toggle between suppressed and awake by setting the GST bit of the associated Suppress Trigger Control Register (NAR_STCR). Additionally, the NAR starts up in global suppress mode if the PUS bit of the Control Register is set when the NAR is enabled. All clients are then suppressed until a wake-up trigger is received. If the PUS bit is set without enabling any of the triggers for global suppress, a configuration error results and the NAR powers up in active mode.

Clients can go also into local suppress mode without putting the entire NAR to sleep. Each NAR_STCR has a 16-bit LST field. Setting one or more of these bits causes the trigger assigned to that NAR_STCR to toggle the associated clients between suppressed and awake. All other clients are unaffected. Thus, suppress trigger one can be assigned to one set of clients, while suppress trigger two can be assigned to another set of clients, and different parts of the device can be ignored at different times, all based on different, programmable events. Global suppress always has higher priority than local suppress. That is, if the NAR is in global-suppressed mode, any local suppress wake-up triggers are ignored. Also, if some clients are in local suppress when a global suppress trigger hits, then

all go into suppress mode; however, upon the subsequent global wake-up, the clients that were previously in local suppress remain so.

105.5.5 Stall detection

Since the NAR can only switch between clients on an end-of-message boundary, every message that goes into the NAR must complete. If a client happens to die or stall in the middle of transmitting a message, this uncompleted message could cause the NAR to stall, at a time when other clients might still be able to deliver useful information. To prevent this from occurring, the NAR monitors each client for a stall condition, which is defined as occurring when data is not valid in the middle of a Nexus message for a set number of cycles.

When a stall condition is detected, the NAR can do one thing. If the client stalls, then the NAR must complete the current stalled message to continue operating. This scenario is termed a stall error. On a stall error, the NAR injects a dummy message beat into the message stream after the last valid beat of data received from the stalled client. The dummy message is a single beat injected at the output of the receive queue into the main queue. The MSEO bits of the data payload are set to 2b11 regardless of the previous Nexus state. This forces the dummy message into the End Message state. When the NAR injects a dummy message, it also generates an error message to notify the user what has happened.

Injecting a dummy message is an intrusive act, which alters the data stream from the client. If a stalled client is thought likely to recover cleanly, it might be advisable to suppress stall errors to prevent the data stream from being corrupted. This can be done by setting the SED bit of the configuration register, which carries the risk of a truly dead client causing the NAR to hang. Otherwise, the default operation is for the NAR to inject the dummy message and then monitor the client for signs of recovery. If the client does come back online, the NAR attempts to wait until it sees an end-of-message boundary before capturing new data. At best, if the client recovers smoothly, only the interrupted message is lost; if the client does not recover smoothly, it might not be possible for the client and NAR to get back in sync in terms of their Nexus states, which would lead to nothing but garbled data coming out. For this reason, it is also possible to prevent the NAR from trying to recover from a stall error, by setting the SRD bit of the NAR_CR. If the SRD bit is set, a stalled client is not allowed to recover until the SRD bit is unset or the NAR is reset.

105.5.6 Output arbitration

The NAR allows two output ports to be active at the same time. This feature is only available on the BD NAR. Typically, this feature is only useful if one of the ports is used to service messages of a specific type or messages from a particular client, or both, while the other port supports all the rest of the message traffic. For example, the NAR could be set up to send in-circuit trace (ICT) traffic out the HBDP and to reserve the trace memory bus port for all other Nexus messages.

If two output ports are enabled, one of them must be designated as the Message Select Port (MSP). The MSP is the destination for those messages that satisfy a set of user-defined requirements as to the source and type of message. All other traffic is sent to the other enabled port. Ports are enabled by setting the port enable bits in the BD NAR_CR. The MSP is designated by setting bits [10–9] and ALT is designated by setting bits [7–6]. Once the MSP has been specified, other bits in the NAR_CR, along with the Source Filter Register (NAR_SFR) and Type Filter Register (NAR_TFR), are used to determine which messages go to the MSP and which do not. These facilities can be used to build a logical expression that can filter messages based on up to four types or four sources, or both.

The NAR_SFR can be programmed with up to four source codes, while the NAR_TFR can hold up to four message-type codes. Each filter also includes an enable bit to turn it on or off. A source match occurs if a messages source code matches one of the source filters or (if the SFI bit is set) if it does not match any of the source filters. Similarly, a type match occurs if a messages type code matches one of the type filters or (if the TFI bit is set) if it does not match any of the type filters. Finally, the ACC bit of the NAR_SFR determines when both source and type filters are used, whether the two are ANDed together (ACC = 0) or ORed together (ACC = 1). For example, to send ICT messages out the HBDP and all other traffic to the trace memory bus port:

- Set ALT = 10 in the NAR_CR—sets trace memory bus port as the alternate port
- Set MSP = 01 in the NAR_CR—sets HBDP as the message select port
- Set FE0 = 1 & TF0 = 100010 (0x22) in the NAR_TFR—enables type filter 0 to match ICT messages

To send everything but data trace from source 13 to the trace memory bus port, while all other traffic goes to the HBDP:

- Set ALT = 01 in the NAR_CR—sets HBDP as the alternate port
- Set MSP = 10 in the NAR_CR—sets trace memory bus port as the message select port
- Set TFI = 1—inverts type-filter logic
- Set FE0 = 1 & TF0 = 000101 (0x05) in the NAR_TFR to select data trace of size 8 bits
- Set FE1 = 1 & TF1 = 000110 (0x06) in the NAR_TFR to select data trace of size 16 bits
- Set FE2 = 1 & TF2 = 001101 (0x0d) in the NAR_TFR to select data trace of size 32 bits
- Set FE3 = 1 & TF3 = 001110 (0x0e) in the NAR_TFR to select data trace size of 64 bits
- Set FE0 = 1 & SF0 = 001101 (0x0d) in the NAR_SFR to select source trace ID = 13
- Set ACC = 0 in NAR_SFR so the final filter is ~(data_trace) AND (source = 13)HBDP

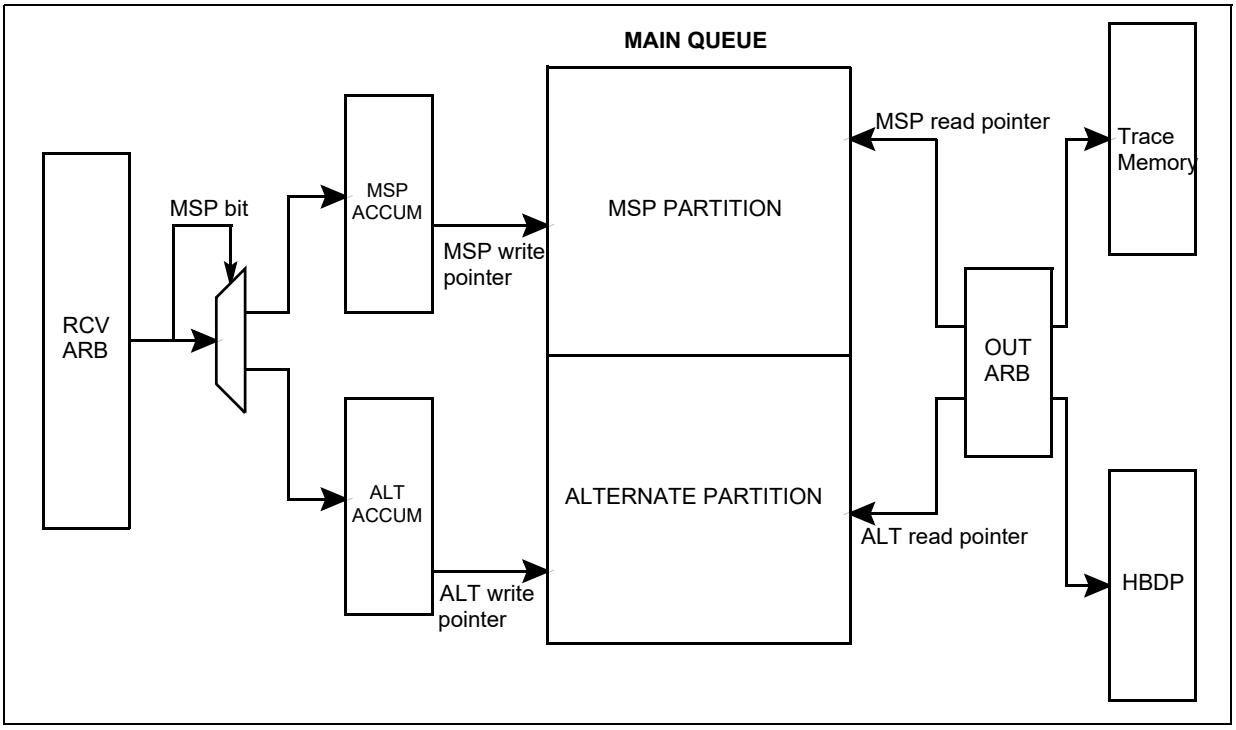
Data is written into the output queue at a maximum rate of one entry per cycle. If the current messages source and type pass the filters, then the message is marked for the MSP. When the entry is later pulled from the queue, it is then sent to the MSP. Unmarked entries are sent to the alternate port. Message destination must be recalculated on every end-of-message boundary, using the fixed-width SRC and TYPE fields at the beginning of each Nexus message.

105.5.7 Queue partitioning

It is possible to partition the queue into two sections, one that supports the MSP, and the other that supports the alternate port as shown in *Figure 2606*. Messages are routed, based on the MSP bit field, to either the MSP queue partition or the alternate queue partition. This allows both ports to be active simultaneously, since messages can be taken directly from both partitions. Each partition uses an accumulator to store a full central queue width beats of data before writing into memory, since on any given cycle the full Nexus word out of receiver arbitration may be split between the two partitions. Queue partitioning is enabled by setting the AQP bits of the NAR_CR.

The main advantage of using a partitioned queue is that it can improve data throughput and performance when multiple ports are active. It allows the trace memory bus port to utilize block transfer mode, which also improves performance. The only drawback is that queue utilization efficiency can be negatively impacted.

Figure 2606. Queue partitioning



105.5.8 Message translation and formatting

The NAR performs the formatting on Nexus messages before sending them to an output port. The exact translation performed depends on the destination port.

105.5.8.1 Aurora format

Nexus messages are pre-translated before sending to the HBDP using the Aurora protocol. The NAR output is a stream of 16 × 4-bit words (where 4 is the width of the NAR Aurora out port) that are further encoded by an external Aurora block before being sent for transmission on the Aurora (HBDP) port.

Entries from the output queue are chopped into chunks of 4 × 16-bit words. The amount of data that can be transmitted per Aurora cycle depends on the number of lanes available in the HBDP (transmitted HBDP word = 16 × number of lanes).

105.5.8.2 Trace memory bus format

The trace memory bus works on AHB-EW protocol. Data destined for the trace memory bus port is split back out into 32-bit beats (Nexus word). Depending on the relative sizes of the beat and the width of the trace memory bus port, one or more Nexus words are packed into a trace memory bus data word, with any remaining bits padded to 0.

For example, if the size of the beat = 32 and the width of the trace memory bus = 64, then each trace memory bus data packet contains two beats of Nexus data. Since the data is stored in a regular format (MSEO bits in same place at every location), this makes data interpretation straightforward, regardless of where the trace memory bus port is sending the data. The trace memory bus translation block performs the necessary accumulation and

resizing in the likely case where the trace memory bus width does not equal the queue entry size.

105.5.9 Aurora interface

The external Nexus Aurora Link (NAL) contains all of the link-layer, PCS, and control logic for the HBDP channel. The Aurora data is transmitted from the NAR to the NAL for sending to the HBDP. Typically, the NAL powers up coming out of reset, but it can also be enabled by the NAR by turning on the NAR with the HBDP enabled. The reverse is true as well. If the NAL is active while the NAR is disabled, then the NAR powers up to communication-disabled mode.

Data flow between the NAR and the NAL uses a simple, synchronous interface. If the NAL is active, it indicates this by a ready signal from the NAL. If Aurora is configured as output it is indicated by a signal from the NAR.

The Aurora interface can either be operated in framing or streaming mode. Streaming mode provides the highest performance and lowest communication overhead through the link. Whereas framing mode provides greater robustness and improved error-handling at the receive side of the link. When framing mode is enabled, the NAR encapsulates the Aurora data in frames consisting of 256 Aurora chunks; that is, after every 256 Aurora transmissions, the NAR asserts end of frame and start of frame to mark the end of the last frame and the beginning of the next frame. In either mode, the NAR always asserts a start-of-frame when beginning transmission, and an end-of-frame when its output queue empties, to notify the channel partner of the end of valid data.

105.5.10 Trace memory bus output port usage

The NAR can communicate directly to external memory by sending Nexus data out onto the trace memory bus port, using the AHB-EW (Early Write) protocol. This provides a convenient, secondary area for data in applications or in situations where the HBDP is not available. The trace memory bus access are generated internally by trace memory bus controller.

Because in many systems it is desirable to have the trace memory bus interface run at a divided clock rate relative to the main platform clock, the trace memory bus port operates off of a separate AHB clock. This clock is assumed to be free-running (that is unaffected by NAR-enable and the status of the Aurora Link), synchronous to the platform clock used by the rest of the NAR, and either running at the same rate as the platform clock or an integer divide of that rate.

The trace memory bus interface protocol is a slightly modified version of AMBA AHB specification. The write data comes in the address phase instead of data phase and it is termed as Early Write (EW). When NAR uses the trace memory bus port as a destination for Nexus data, it performs a series of writes to a predefined block of memory space.

This memory space is specified through the trace memory bus base address and trace memory bus max transfer configuration registers. The size of the block is set by the max transaction count (MXFR field in NAR_TCR) times the size of a transaction (trace memory

bus-width, 64-byte, 128-byte, 256-byte, or 384-byte, depending on block transfer mode and block transfer size). For example, if:

- Base address = 0x0D00_0000 (that is TBAHI = 0x0, TBALO = 0x0D00_0000)
- NAR_TCR[BTM] = 1 (Block-transfer mode enabled)
- NAR_TCR[BTS] = 00 (Block-transfer size = 64-byte)
- NAR_TCR[MXFR] = 0x100 (max transaction count)

then the total block size would be 0x100 $\times$ 0x40 = 0x4000 bytes. The dedicated debug address range would then be 0x0D00_0000 – 0x0D00_3FFF.

The NAR can either write to the debug space one (trace memory bus-width) word at a time, or for improved performance, it can use a block transfer. If the BTM bit of the trace memory bus Control Register (NAR_TCR) is set, then the NAR uses block transactions to send data through the trace memory bus port, where the size of the block is determined by the setting of the BTS field of the NAR_TCR.

In block transfer mode, the NAR waits until the main queue partition dedicated to the trace memory bus port reaches a specified threshold point, determined by the BTH field of the NAR_TCR, before initiating a block transfer. The default threshold is the size of the block transfer. That is, a block transfer does not start until there is enough data in the associated partition to fill the block. But it might be desirable, for bandwidth reasons or because the associated partition is smaller than the block transfer size, to begin the transfer sooner. Hence, based on BTH, a transfer can begin when the partition is at 25%, 50%, or 75% of the block transfer size. This capability should be used with caution. If the Nexus client does not generate enough new data during the transfer to fill the block, then a block transfer error results.

The NAR generates Nexus trace write transactions for the trace memory bus port using the address from the base address register for the first transaction and then incrementing through the memory space. Once the programmed max number of transactions have been sent, the NAR sets the trace memory bus overflow status bit and, if desired, asserts the NAR event-out. At this point, the NAR can either shut down the trace memory bus port for trace, or, if the TBW bit of the NAR_TCR is set, reset the address to the beginning of the debug space and continue sending trace data. This allows the debugger to view the most recent data, in cases where the buffer overflows.

When TBW bit of NAR_TCR is not set then NAR shuts down the trace memory bus port. To restart the trace memory write again from trace memory base address, the user should write either the NPC trace memory bus base address (TBALO/TBAHI) or the transaction count (MXFR). The user does not have to actually change the value, but can write the same target base address again. The NAR sees any write to one of these registers as a reconfiguration of the trace memory bus port, and resets its internal address pointer to the base address along with clearing the target-space-full flag.

105.5.11 Internal message generation

Generally, the NAR simply conveys Nexus messages from client to port, and does not generate messages of its own. There are, however, certain situations when the NAR needs to communicate directly with the debugger. Internally generated messages are added to the queue the same as external messages. If the queue is partitioned, then internal messages are always channeled to whichever partition is dedicated to the HBDP.

The NAR generates the following message types:

- Device_ID—The first message transmitted when the NAR comes up. It consists of a 6-bit TCODE (TCODE = 1) and the 32-bit device ID. The first beat of the message is always an IDLE (MSEO = 2b11, MDO = all 0s) to allow the debugger Nexus state machine to correctly initialize (by moving to end message or idle state). Device_ID is also generated if the queue is repartitioned (by changing either the ALT or MSP setting in the NAR_CR) while it is not empty. This serves the dual purpose of insuring that the user always gets at least one device_ID message and providing a message boundary in the case where there might have been corruption due to the lost data.
- Watchpoint—The NAR generates its own watchpoint match message under the following circumstances as shown in *Table 2488*.
- NAR_Error (TCODE = 8)—If an error condition is detected inside the NAR, it generates its own error message. An NAR_Error message consists of the 6-bit TCODE, the 6-bit NAR source ID, and a 32-bit data field which corresponds to the NAR status register. An NAR error can be triggered by:
  - HBDP error
  - Illegal configuration
  - Stall Error
  - Trace memory bus read/write access error

This table describes the NAR message formats.

Table 2485. NAR message formats

Message name	Min packet size (bits)	Max packet size (bits)	Packet type	Packet name	Packet description
Device ID	6	6	fixed	TCODE	TCODE value = 1
	32	32	fixed	DID	DID register value
	0	24	variable	TSTAMP	Optional, globally synchronized timestamp
Error Message(1)	6	6	fixed	TCODE	TCODE value = 8
	6	6	fixed	SRC(2)	NAR source ID
	4	4	fixed	ETYPE	See *Table 2486*
	12	12	fixed	ECODE	See *Table 2487*
	0	24	variable	TSTAMP	Optional, globally synchronized timestamp
Watchpoint Hit Message(3)	6	6	fixed	TCODE	Value = 15
	6	6	fixed	SRC2	NAR source ID
	6	6	fixed	WPHIT	See *Table 2488*
	0	24	variable	TSTAMP	Optional, globally synchronized timestamp

1. The Error message conforms to the IEEE-ISTO 5001-2011 standard.
2. Tools should treat the NAR SRC field as a 4-bit SRC field and a 2-bit unused field. The extra 2-bit field (the two most significant bits of the 6-bit TCODE) is always 0b00. This operation is not compliant to the IEEE-ISTO 5001 Nexus standard.
3. The Watchpoint Hit message does not conform to the IEEE-ISTO 5001-2011 standard, it conforms to the IEEE-ISTO 5001-2003 standard. The WPHIT should be a variable length packet, but is implemented as a fixed length packet. In most cases, tools will not see a difference, however, a difference will be seen if a timestamp field is appended to the message.

105.5.11.1 Error conditions

NAR errors can occur upon events such as illegal memory accesses, block-transfer overloads, communication errors, or when the NAR is configured incorrectly. If the configuration registers are programmed incorrectly, it is possible to put the NAR into an illegal state. The NARs response varies depending on the error. For a minor error, the NAR reverts to a default mode and continues operation, while major errors result in complete shutdown.

In either case, a Nexus error message is sent to the Aurora out port, if its enabled, otherwise to the trace memory bus port. The Nexus error message contains both ETYPE and ECODE fields:

- ETYPE is used to denote the basic type of error that has occurred
- ECODE carries specific additional information about the error

All NAR error types are defined in this table.

Table 2486. NAR error types

ETYPE	Description	Classification	NAR action	Comment
0000 (h'0)	Receive queue overrun	Minor	Dropped message	If a client sends a data word that the NAR cannot accept (due to a full receive queue), the new data word is dropped.
0001 (h'1)	Internal message contention	Minor	Dropped message	Internal errors/watchpoints occurred faster than NAR could generate messages for them, causing lower-priority message(s) to be dropped. The type of message dropped is conveyed in the ECODE field, see *Table 2487*.
0011 (h'3)	Trace memory bus port read/write access error	Minor	Access fails	Unable to send an Aurora-to-trace memory bus request. Typical causes would be writes larger than the data buffer or sending in a new request before the previous one completes.(1)
0101 (h'5)	Invalid NRR	Minor	Access fails	Tried to access an un-implemented NRR.
1001 (h'9)	Config error	Major	See *Table 2487*	The specific configuration error is conveyed in bits 0–3 of the ECODE field (*Table 2487*), as well as the NCE field of Status Register.
1100 (h'C)	HBDP error	Minor	N/A	A hard error is automatically reset the Aurora Link and causes it to re-initialize. Although the Link is down, the error message is queued and released as soon as the Link comes back up.
1110 (h'E)	Block transfer error	Minor	Dummy fill	If the Trace memory bus port becomes data-starved during a block transfer, it fills the rest of the block with IDLE messages.
1111 (h'F)	Stall error	Minor	—	—

1. In the case where a request was successfully translated to a Trace memory bus transaction which itself returned an error, that is conveyed in the STATUS field of the Nexus Target Response message subsequently returned to the debugger.

Typically, the ECODE field is used to communicate which type of message was dropped in queue overflow or message contention situation, but for the NAR it is also used to denote the type of configuration error that occurred. If a configuration error is detected, the NAR generates a Config Error Nexus message and transmits it, with the actual type of error encoded in the four MSBs of ECODE, as well as the NCE field of the Status Register. The ECODE field encodings are listed in this table.

Table 2487. Configuration error codes

ECODE	ETYPE	Description	NAR action	Comment
0000xxxxxxx1	0001	Watchpoint message dropped	N/A	Occurs with internal message overflow error.
0000xxxxxx1x	0001	Data trace message dropped	N/A	Occurs with internal message overflow error; MMA response message was dropped.
00001xxxxxxx	0001	Response message dropped	N/A	Occurs with internal message overflow error; target response (memory access or NRR) message was dropped.
0000xxx1xxxx	0001	Status message dropped	N/A	Occurs with internal message overflow error.
000100000000	1001	Config error: multiply defined port	Communication disable mode	Both MSP and ALT were assigned to the same port. NPC waits until the configuration is fixed before proceeding.
001000000000	1001	Config error: no ALT defined	Tracebuffer	An MSP was defined without defining the ALT. The NPC reverts to tracebuffer mode.
001100000000	1001	Config error: cannot enable port	Communication disable mode	User has attempted to enable a port that does not exist in this implementation of the NPC.
010000000000	1001	Config error: cannot support message filtering	Message filtering or queue partitioning disabled, or both	User has attempted to enable message filtering or queue partitioning with only one output enabled. Filtering/partitioning is turned off.
100100000000	1001	Config error: no wakeup trigger	Power up in active mode	Occurs if none of the defined clients in the system can cause a wakeup (PUS is set in NAR_CR, but no NAR_STCRs have WUT set).

105.5.11.2 Watchpoint messages

The NAR generates its own watchpoint match message under the circumstances shown in the following table. The value of the Watchpoint Hit field of the Watchpoint Message is also shown in this table.

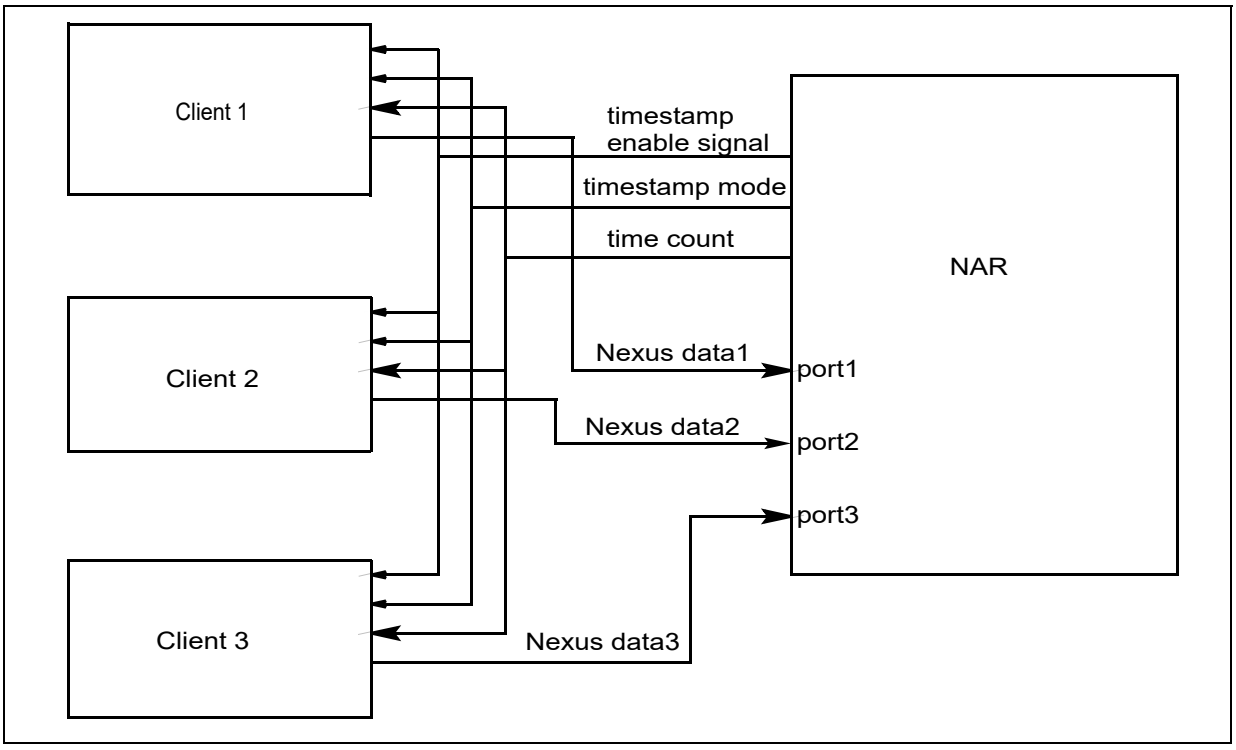
Table 2488. Watchpoint Hit values

WPHIT	Description	Comment
b010000	Entering Suppress Mode	If (local or global) suppress mode is enabled and the appropriate trigger is received, the NAR generates a watchpoint message. The type of event (in this case, a suppress event) is encoded in the WPHIT field of the watchpoint message (see the Nexus standard, sec. 5.3.21).
b100000	Client Sync	If sync messaging is enabled, the NAR generates a sync watchpoint message whenever a client sync event occurs.
b000100	Trace Memory bus Block Overflow (TBO)	If the trace memory bus block is filled, causing it to either disable trace memory bus data trace or wrap back to the beginning of the dedicated debug memory block and if trace memory bus overflow messaging is enabled, then an TBO message is generated.
b001000	Tracebuffer Wrap	If the NAR is being used in tracebuffer mode with the wrap-around feature, then a watchpoint message is generated every time the tracebuffer wraps back to address 0. This would be useful in informing the user that the buffer had wrapped and that older data had been overwritten.

105.5.12 Timestamp function

The NAR and all of its clients are required to transmit timestamps, to allow precise ordering of events. All clients synchronous or asynchronous uses NAR timer value broadcasted over a bus, this ensures that all timestamps are in sync as shown in *Figure 2607*. The NAR uses enable signal to centrally control the enabling of time stamping, which informs the clients to start adding the timestamp field to their message traffic.

Figure 2607. Timestamp architecture



Timestamping is enabled by setting the TEN bit of the NAR_CR. This causes the NAR to assert the time stamp enable signal to clients. Typically, the user would also set the TRS bit at this time to reset the NAR Time stamp counter. Internally the NAR has a 30-bit counter external to the NPC block. This 30-bit counter is broadcast to all clients after encoding its value in Gray code. This minimizes the error in synchronization by the clients. The NPC uses an additional internal 24-bit counter to send the timestamp inside its messages. The client converts the gray coded timestamp to binary to append to the Nexus messages.

The Time stamp mode, a 2-bit field, is broadcasted to all client which tells the granularity of sending time stamps into message. This is controlled by the TSG bits of the NAR_CR.

Asynchronous clients, those on a separate clock domain from the NAR, synchronize the enable, mode and timestamp value in their domain. The error introduced by this is assumed to be within tolerable range by application.

A timestamp counter reset can be forced by causing a 0 to 1 transition in the TRS bits of the NAR_CR. The NAR deasserts the enable signal to clients while the reset occurs, and then the counter is restarted (assuming the TEN bit is set). Data throughput can be increased (at the loss of timestamp granularity) by controlling the TSG bits of the NAR_CR. Rather than adding a timestamp to every message, clients can be told to only stamp every 4th, 16th, or 32nd message, easing the bandwidth requirements on the output ports.

The timestamp block maintains the timestamp counter for NAR. It is the central time source for all synchronous/asynchronous clients.

105.5.13 Global (Development tool client) stall control

The external development tool has finite memory. To prevent data loss when this memory is full, the tool gives an event to the device. The DCI block processes this and forwards to NAR block. The dci_evti[0] carries this information. NAR detects the falling edge of dci_evti[1] and loads a counter with a configurable value programmed in GLOBAL_STALL_COUNT field of NAR_AHFPAR. This counter starts decrementing immediately, reaches to zero and stops. During the running period of the counter the Global Stall signal remains asserted thus indicating to stall the message interface to the NAL by that many cycles. The GLOBAL_STALL_COUNT field is five bits so client can be stalled up to 31 cycles.

105.5.14 Overlay and internal trace memory full status generation

In certain events, such as main memory (internal trace memory) full or external overlay trace memory full, it is required to control the clients to stop sending data, to avoid any loss of messages from the clients.

The NAR provides three outputs (triggers) to indicate three different situations as described below:

- Internal trace memory (the main queue) is full
- External debug memory on trace memory bus is full
- External debug memory on trace memory bus is full up to a programmable watermark set by the configuration register

The last two triggers are used by SPU to generate stall and suppress triggers. Typically the stall request should be generated whenever space in the external memory connected to trace memory bus has been exhausted. The NAR starts writing from the first location (wrap) as specified in trace memory bus address register in this condition if overwrite trace memory bus debug memory (TBW bit) is set in the NAR_TCR.

The user should configure the partial full watermark register such that it can give around 90% full indication of external debug memory on the trace memory bus. The emulator tool memory full indication is typically generated from the DCI block which actually communicates with the tool.

For example, the following steps can be used to generate event at around 90% fill level for trace memory connected in AHB-EW and to set the watermark in the configuration register.

- From the configuration, the programmer knows the following:
  - AHB start (base) address—Base address = 0x0D00_0000 (that is TBAHI = 0x0, TBALO = 0x0D00_0000
  - Block transfer size—NAR_TCR[BTM] = 1 (Block-transfer mode enabled)
  - Max transfer count—NAR_TCR[MXFR] = 0xC0 (maximum transaction count)
- Thus, the total block size is (0xC0 $\times$ 0x40) = 0x3000 bytes. The dedicated debug address range is then 0x0D00_0000 – 0x0D00_2FFF.
- To generate an event at nearly 90% fill level of the trace memory at 1 KB boundary, the value to be programmed in the register NAR_AHFPAR[3:0] is calculated as:
  - Number of 1K blocks in 0x3000 bytes = (3000 right shift by 10 bits) = C = 12
  - So the value to be programmed should be 11 as 90% of 12 is almost 11.
- NAR_AHFPAR[3:0] = (NAR_TBALO[13:10] + 0xB) = (0x7+0xB) 0x2 (upper bits truncated due to width) and enable this with NAR_AHFPAR[10] = 1.

This assumes that NAR starts writing from base address and stops at maximum address and no reading is done in overlay memory trace data area during this operation. Whenever the address bus bits [13:10] match with this value, the partial event is generated (toggled).

Similarly for 65 KB boundary matching NAR_AHFPAR[7:4] needs to be programmed and the corresponding enable bit needs to be set.

105.5.15 Configuration/debug interface JTAG (1149.1)

The configuration and status registers of the NAR are accessed through the standard IEEE 1149.1 JTAG interface. The JTAG frequency (TCK) must be 1/2 of the NAR operating clock. The access mechanism through JTAG is described in this section.

The NAR block uses the IEEE 1149.1-2001 TAP for accessing registers. TAP signals include TCK, TDI, TMS, and TDO. The NAR implements a TAP controller state machine that transitions based on the state of the IEEE 1149.1-2001 16-state state machine. It also implements Nexus controller state machine as defined by the IEEE-ISTO 5001-2001 standard.

The instructions implemented by the NAR TAP controller are listed in *Table 2489*. Each unimplemented instruction acts like the BYPASS instruction. The size of the NAR instruction register is 4 bits.

Figure 2608. Nexus controller state machine

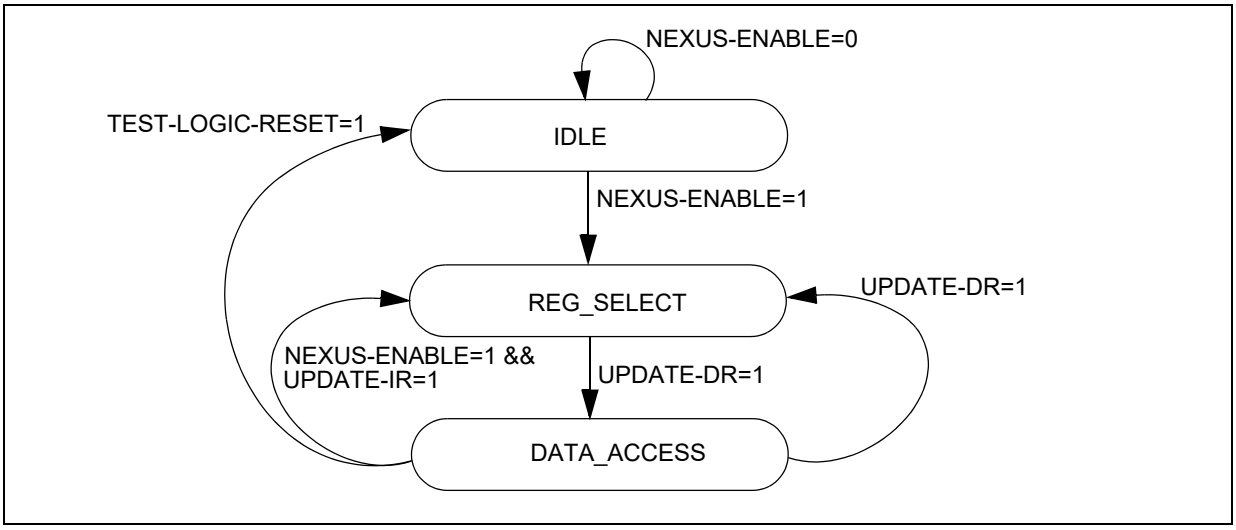
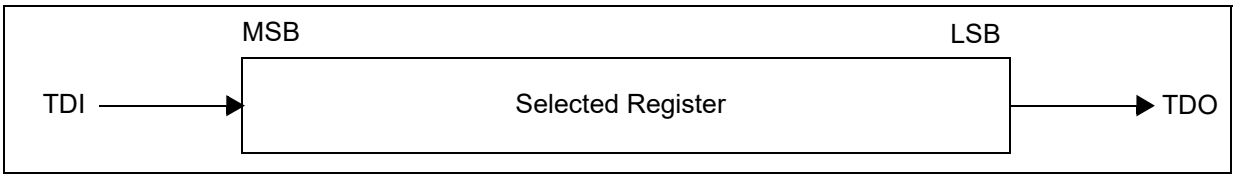


Table 2489. Implemented instructions

Instruction Name	Private/ Public	Opcode	Description
NEXUS_ENABLE	Public	0x0	Activate Nexus controller state machine to read and write NAR registers.
BYPASS	Private	0x0F	NAR BYPASS instruction. Also the value loaded into the NAR IR upon exit of reset.

Data is shifted between TDI and TDO starting with the least significant bit as illustrated in the following figure. This applies for the instruction register and all Nexus tool-mapped registers.

Figure 2609. Shifting data into register



105.5.15.1 Enabling the NAR TAP controller

Assertion of the power-on reset signal resets the NAR TAP controller. When not in power-on reset the NAR TAP controller is enabled by driving JCOMP with the NAR enable value and exiting the Test-Logic-Reset state. Loading the NEXUS-ENABLE instruction then grants access to Nexus debug.

105.5.15.2 Loading NEXUS-ENABLE instruction

Access to the NAR registers is enabled when the TAP controller instruction register is loaded with the NEXUS-ENABLE instruction. This instruction is shifted in via the SELECT-IR-SCAN path and loaded in the UPDATE-IR state. At this point, the Nexus controller state machine, shown in *Figure 2608*, transitions to the REG_SELECT state. The Nexus

controller has three states: idle, register select, and data access. The following table illustrates the IEEE 1149.1 sequence to load the NEXUS-ENABLE instruction.

Table 2490. Loading NEXUS-ENABLE instruction

Clock	TMS	IEEE 1149.1 state	Nexus state	Description
0	0	RUN-TEST/IDLE	IDLE	IEEE 1149.1-2001 TAP controller in idle state
1	1	SELECT-DR-SCAN	IDLE	Transitional state
2	1	SELECT-IR-SCAN	IDLE	Transitional state
3	0	CAPTURE-IR	IDLE	Internal shifter loaded with current instruction
4	0	SHIFT-IR	IDLE	TDO becomes active, and the IEEE 1149.1-2001 shifter is ready. Shift in all but the last bit of the NEXUS-ENABLE instruction.
5 TCKS				
12	1	EXIT1-IR	IDLE	Last bit of instruction shifted in
13	1	UPDATE-IR	IDLE	NEXUS-ENABLE loaded into instruction register
14	0	RUN-TEST/IDLE	REG_SELECT	Ready to be read/write Nexus registers

105.5.15.3 Selecting a JTAG register to create a bus (memory or peripheral) access

When the NEXUS-ENABLE instruction is decoded by the TAP controller, the input port allows the development tool access to all NAR registers. Each register has a 7-bit address index.

All register access is performed via the SELECT-DR-SCAN path. The Nexus controller defaults to the REG_SELECT state when enabled. Accessing a register requires two passes through the SELECT-DR-SCAN path: one pass to select the register and the second pass to read/write the register.

The first pass through the SELECT-DR-SCAN path is used to enter an 8-bit Nexus command consisting of a read/write control bit in the LSB followed by a 7-bit register address index, as illustrated in *Figure 2610*. The read/write control bit is set to 1 for writes and 0 for reads. The index for registers are shown in *Table 2470*.

Figure 2610. Nexus command format for register selection

MSB	LSB
7-bit register index	R/W

The second pass through the SELECT-DR-SCAN path is used to read or write the register data by shifting in the data (LSB first) during the SHIFT-DR state.When reading a register, the register value is loaded into the IEEE 1149.1-2001 shifter during the CAPTURE-DR state.When writing a register, the value is loaded from the IEEE 1149.1-2001 shifter to the register during the UPDATE-DR state. When reading a register, there is no requirement to shift out the entire register contents. Shifting may be terminated once the required number of bits have been acquired.

This table illustrates a sequence which writes a 32-bit value to a register.

Table 2491. Writing a 32-bit value to a register

Clock	TMS	IEEE 1149.1 State	Nexus State	Description
0	0	RUN-TEST/IDLE	REG_SELECT	IEEE 1149.1-2001 TAP controller in idle state
1	1	SELECT-DR-SCAN	REG_SELECT	First pass through SELECT-DR-SCAN path
2	0	CAPTURE-DR	REG_SELECT	Internal shifter loaded with current value of controller command input.
3	0	SHIFT-DR	REG_SELECT	TDO becomes active, and write bit and 4 bits of register index shifted in.
7 TCKs				
11	1	EXIT1-DR	REG_SELECT	Last bit of register index shifted into TDI
12	1	UPDATE-DR	REG_SELECT	Controller decodes and selects register
13	1	SELECT-DR-SCAN	DATA_ACCESS	Second pass through SELECT-DR-SCAN path
14	0	CAPTURE-DR	DATA_ACCESS	Internal shifter loaded with current value of register
15	0	SHIFT-DR	DATA_ACCESS	TDO becomes active, and outputs current value of register while new value is shifted in through TDI
31 TCKs				
47	1	EXIT1-DR	DATA_ACCESS	Last bit of current value shifted out TDO. Last bit of new value shifted in TDI.
48	1	UPDATE-DR	DATA_ACCESS	Value written to register
49	0	RUN-TEST/IDLE	REG_SELECT	Controller returned to idle state. It could also return to SELECT-DR-SCAN to write another register.

105.5.16 Trace memory bus controller

The purpose of the trace memory bus is to transfer trace data from main queue to external overlay memory. This port works on a modified AHB protocol that supports early write. It has AHB interface signals as defined in AMBA specification (Rev 2.0). In addition, there are side-band signals for added functionality.

Key features of the module include:

- AHB-EW Interface
  - Early Write (for higher BW in special module)
  - Configurable 32/64-bit address bus width
  - 64-bit data bus width
  - Transfer Types
  - IDLE, NONSEQ, SEQ
  - Single Transfers
  - Burst Transfers
  - Configurable support for INCR
  - Configurable support for INCR4, INCR8, INCR16, WRAP4
  - Response Type
  - OKAY, ERROR
  - Transfer Size
  - Byte, Half Word, Word, Double-Word
  - Protection Control
  - Cacheable/ Non Cacheable Access Control
  - User/Privilege Access
- Endianess Support
  - Configurable Little Endian/Big Endian support on AMBA-AHB interface
- Pipeline Support
  - Total of two transactions (One plus one ongoing)

105.5.16.1 Operation

The trace memory bus controller allows all read/write operations to be cut through. The controller has a pipeline depth of two. While the data phase of one transaction is ongoing, it can accept one more read/write request from the internal bus master and initiate the address phase on the AHB interface during the last data cycle of the ongoing transaction.

105.5.16.2 Write operation

When the trace memory bus initiator asserts a request for a write operation, it can find the acknowledge signal parked asserted if the controller is ready to accept the transaction. If the pipeline is full or in case of an ongoing read transaction, the controller inserts wait states by keeping the acknowledge low. The trace memory bus controller initiates the address phase on AHB interface once it accepts the transaction by asserting the acknowledge on the trace memory bus interface. The write data from the trace memory bus initiator is accepted by the gasket by asserting the acknowledge only on completion of address phase at AHB interface, which is signified by receiving the ready signal asserted. Simultaneously, the data is sent to the data bus. Once the requisite number of bytes have been accepted from the trace memory bus initiator, the gasket generates end of data and transfer according to the Trace memory bus protocol.

The AHB protocol supports data transactions of byte, half-word, word and double-word (when both trace memory bus and AHB data bus width is 64 bits). A write transaction from the trace memory bus initiator take place as single/multiple transactions on the AHB side.

105.5.16.3 Read operation

When the trace memory bus initiator asserts a request signal to request for a read operation, it can find the acknowledge signal parked asserted if the gasket is ready to accept the transaction. If the pipeline is full, the controller inserts wait states by keeping the acknowledge low. The trace memory bus controller initiates the address phase on AHB-EW interface once it accepts the transaction by asserting the acknowledge on trace memory bus interface. The controller reads the data from the bus when the ready signal is sensed asserted. The data is transferred on the trace memory bus data bus with the data acknowledge asserted. The gasket asserts end of data and transfer at the end of transaction as per trace memory bus protocol.

The AHB protocol supports data transactions of byte, word, half-word and double-word (when both trace memory bus and AHB data bus width is 64-bits). A read transaction from trace memory bus initiator takes place as single/multiple transaction on the AHB interface before the read data is merged and sent on the trace memory bus read data bus.

In case of variable data bus width on trace memory bus and AHB interface (32-bit data bus on one interface and 64-bit data bus on the other interface), a maximum of 64 bits is read on each beat.

105.5.17 Aurora client controller

This block converts the Aurora client protocol from the PD NAR into an internal client protocol that is accepted by the NPC block. The Nexus beat size must be same at both interfaces. The relation between the Aurora client data width and the NPC client interface data width must be maintained as described here.

The Aurora data width is defined as the Aurora width $\times$ 16 bits, where 16 bits comprise one Aurora word. The Nexus client data width is defined as Bn $\times$ 32, where 32 = MDO bits + 2 MSEO bits and Bn is the number of beats.

Thus:Aurora data width = the Nexus client data

Aurora width $\times$ 16 = Bn $\times$ 32

Aurora width = 2 $\times$ Bn.

This block works in a similar way to the legacy client controller block except for the interface level control. The Aurora client has the information about the FIFO size. It sends as many data as the FIFO size then waits for the data accept signal from this block. The controller gives the data accept only when a read occurs and sends data to the NPC interface. The FIFO act as a storage and serves the purpose of clock domain transfer as well.

105.6 Initialization information

The NAR is initialized by asserting the block reset signal for a minimum of one cycle of the slowest clock. Reset clears all latches to 0, with the only exception being the output queue, whose initialization state depends on the reset behavior of the compiled memory used. Upon exiting reset, the NAR is in the disabled state, as described in *Section 105.5.2.1: Reset*.

Pointers and state machines are initialized by the starting of NAR functional clocks (which occurs either upon the setting of the NAR enable bit or upon the assertion of the Phy enable signal) or by an NAR soft reset.

105.7 Application information

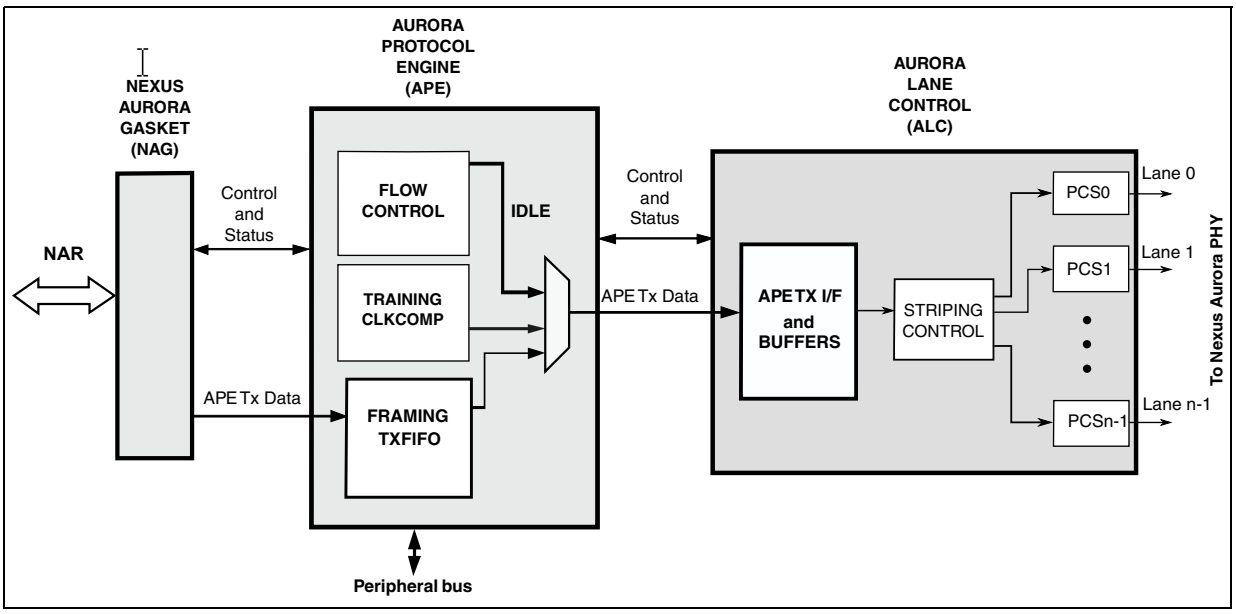
The base application for the NAR is collecting and collating Nexus trace data. Programming examples for how to set up the NAR for trace are given in *Section 105.5.6: Output arbitration*.

106 Buddy Device—Nexus Aurora Link (NAL)

106.1 Introduction

The Aurora Protocol is used to send debug information over a high-speed serial link out of the device. The Nexus Aurora Link (NAL) consists of all the logic on the MCU needed to implement the connection up to the physical-layer SerDes and transceivers; it includes Aurora control and data multiplexing, data-encoding and striping, and the Physical Coding Sublayer (PCS) portion of the protocol. *Figure 2611* details the NAL block diagram. The number of lanes, n, is 4.

Figure 2611. NAL block diagram



Debug data in the Nexus format from multiple clients is collected by the Nexus Aurora Router (NAR); messages are then packetized and sent across an asynchronous clock boundary-crossing gasket to the Aurora Protocol Engine, which performs all the logical layer functions and flow control; data from the protocol engine is sent on to the Aurora Lane Control (ALC) block, which performs data-striping and sizing, based on the number of lanes selected for debug; it also performs the PCS functions of the link. The PCS drives the Nexus Aurora Physical (NAP) interface. The NAP, in conjunction with the NAL, supports one-way simplex high-speed serial communication with an external debugger.

The NAL is composed of three main blocks: the NAR/Aurora Gasket, the Aurora Protocol Engine, and Aurora Lane Control. *Table 2492* lists NAL functionality by block and lists the high level features for the Nexus Aurora PHY.

Table 2492. NAL and NAP feature summary

Sub-component	Feature
NAR Aurora Gasket (NAG)	– Transmit FIFO to buffer data as it crosses the NAR / NAL clock-domain boundary and Transmit FIFO space available indication to NAR. The NAR uses Transmit FIFO space available indication(s) to determine if data can be sent to the NAL. – Platform to Domain clock domain crossing. The NAR is on the core clock domain while the NAP is on a domain sourced from the external PHY clock input signals.
Aurora Protocol Engine (APE)	– Link training – Per the Aurora Protocol specification, clock-compensation, and data framing – Cyclic Redundancy Check (CRC) generator
Aurora Link Control (ALC)	– Striping of data stream from the Aurora core across all available lanes in two-byte chunks – PCS functionality for NAL
Nexus Aurora PHY (NAP)	– Physical transmission of serial data to device pins – External clock source for SerDes clock domain

106.2 Transmit operation

During transmit operation, the NAR collects debug data from Nexus clients over a 32-bit interface: 30 bits are used for MDO and 2 bits are used for MSEO information. The NAR places MDO data into a payload (maximum payload size depends on the number of lanes, each lane has a maximum of 7424 symbols before terminating the payload) and adds the following:

- 16-bit start symbol
- 16-bit end symbol
- 16-bit CRC if enabled before end of previous symbol

Each packet is transferred from the NAR to the NAR/NAL gasket, through the Aurora Protocol Engine, the Aurora Lane Control and finally out to the device SerDes/LVDS pins through the NAP.

After each payload transfer, the NAL adds a clock compensation sequence needed for downstream SerDes devices to remain synchronized to the MCU.

Note: Symbol counting for clock compensation starts just prior to the verification symbols; therefore, training symbols are excluded when counting.

106.3 Nexus Aurora formatting

The Aurora physical interface is the transport mechanism for Nexus messages. The Nexus messages are formatted by the Aurora formatter and striped into Aurora lanes. The Nexus messages themselves encapsulated into a virtual Nexus port that consists of a 30-bit Message Data Output (MDO) port and a 2-bit Message Start/End Output (MSEO) signal. The MSEO portion of the port is the two least significant bits (LSB) of the 32-bit Aurora message. The rest of the 32-bit Aurora frame consists of 30 MDO bits that are added LSB first.

106.4 Static training

The NAL supports static training. The NAL performs Static training by using programmable timers to determine how long to stay in each stage of the training process. In static training, there is no communication between the channel partners to exchange training status; rather the NAL sends the align/bond/verify sequence to its partner for a set amount of time sufficient for the receive partner to complete the training stage.

Note: Prior to initialization, the NAL must be reset to its POR state; to do this, assert the block reset. See the RST bit field description in the NAL General Control Register (NAL_GCR).

The following steps enable static training:

1. Program the counters in the NAL_TCR and set the STE bit
2. Set the RST bit

This resets the channel and start the training routine. The time spent in each stage can range from 16 to 3072 cycles; once a counter times out, a counter time-out signal is asserted to the Aurora Protocol Engine (APE) to prompt it to move to the next stage. Additionally, it is possible to hold the training procedure in any of the stages indefinitely, by setting one of the three hold bits (AHD/BHD/VHD). When set, these bits cause the channel to sit in Align/Bond/Verify until unset, allowing a user to interactively step through the training procedure by monitoring the status at the debugger side and then stepping the device side to the next stage.

During the Aurora training sequence, the NAL generates a specific set of character sequences per the Aurora protocol specification which are to be transmitted simultaneously on all lanes.

106.5 Programmable registers

The Nexus Aurora Link (NAL) module provides a Nexus parallel to Aurora formatting for the MCU trace information. These registers are not memory-mapped and can only be accessed via the JTAG interface.

Table 2493. NAL memory map

Offset	Register name	Width (in bits)	Access	Reset value
0x0	NAL General Status Register (NAL_GSR)	32	R	0x0000_8000
0x8	NAL General Control Register (NAL_GCR)	32	R/W	0x0000_0000
0xC	NAL Training Control Register (NAL_TCR)	32	R/W	0x0000_0000

106.5.1 Nexus Aurora Link General Status Register (NAL_GSR)

Access: Read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	TXONLY	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	TXCFG			0	0	0	0	0	0	0	0	0	AS
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 2612. Nexus Aurora Link General Status Register (NAL_GSR)

Table 2494. NAL_GSR field descriptions

Field	Description
0:11	Reserved
12 TXONLY	Transmit Only mode. Indicates that Aurora is configured in a transmit-only mode. 0 Reserved 1 Aurora is configured for transmit-only simplex communication. Training method is configured to use Static Training.
13:18	Reserved
19:21 TXCFG	TX Lane Configuration Number of enabled TX lanes. These bits are read-only and are controlled external to the NAL. 000 Reserved 001 2 TX lanes 010 Reserved 011 4 TX lanes 100 Reserved 101 Reserved 110 Reserved 111 Reserved
22:30	Reserved
31 AS	Aurora Status 0 Aurora is not enabled. 1 Aurora is enabled.

106.5.2 NAL General Control Register (NAL_GCR)

The NAL_GCR configures the behavior of the APE. It can be programmed to reset the Aurora channel, or to configure simplex/duplex initialization and to enable debug features.

Access: Read/Write

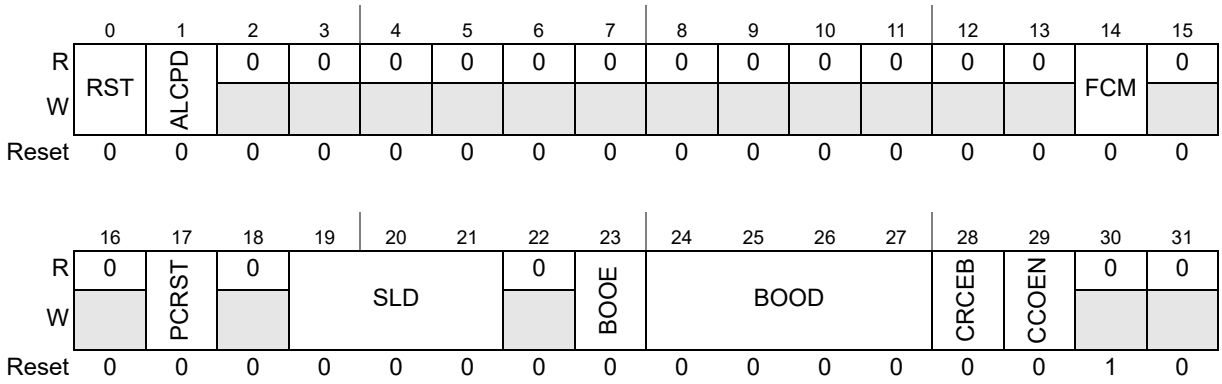


Figure 2613. NAL General Control Register (NAL_GCR)

Table 2495. NAL_GCR field descriptions

Field	Description
0 RST	Reset the Aurora Channel 0 to 1 transition resets the channel and begins the training sequence. 0 normal operation 1 transition from 0 to 1 resets channel and starts training sequence
1 ALCPD	ALC/NAL Power Down Gate off clocks to the NAL blocks (ALC and APE). 0 NAL is enabled with all clocks running provided other mechanisms of powerdown (such as alc_num_lanes) are not in effect. 1 NAL is powered down and inactive.
2:13	Reserved This read-only bitfield is reserved and always has the value zero.
14 FCM	Flow Control Mode 0 Completion Mode 1 Immediate Mode
15:16	Reserved and must be written with zero.
17 PCRST	Protocol Converter Reset Resets the ALC when asserted.
18	Reserved This read-only bit is reserved and always has the value zero.
19:21 SLD	Symbol Lock Delay 000 Allow 3 characters to accumulate in elastic buffer before starting to read 001 Allow 4 characters to accumulate in elastic buffer before starting to read 010 Allow 5 characters to accumulate in elastic buffer before starting to read 011 Allow 6 characters to accumulate in elastic buffer before starting to read 100 Allow 7 characters to accumulate in elastic buffer before starting to read 101 Allow 8 characters to accumulate in elastic buffer before starting to read 110 Allow 9 characters to accumulate in elastic buffer before starting to read 111 Allow 10 characters to accumulate in elastic buffer before starting to read
22	Reserved This read-only bit is reserved and always has the value zero.

Table 2495. NAL_GCR field descriptions (continued)

Field	Description
23 BOOE	Bond Offset Override Enable 0 ALC channel bond attempts to determine channel bond offset. 1 ALC uses user override setting for channel bond offset.
24:27 BOOD	Bond offset override data Only valid when BOOE is set. 0000/1000 use 0-cycle offset 0001 use -7 cycle offset 0010 use -6 cycle offset 0011 use -5 cycle offset 0100 use -4 cycle offset 0101 use -3 cycle offset 0110 use -2 cycle offset 0111 use -1 cycle offset 1001 use +1 cycle offset 1010 use +2 cycle offset 1011 use +3 cycle offset 1100 use +4 cycle offset 1101 use +5 cycle offset 1110 use +6 cycle offset 1111 use +7 cycle offset
28 CRCEB	CRC Enable 0 CRC generation and checking is disabled for TX and RX data. 1 CRC is inserted for TX frames and CRC is checked for RX paths.
29 CCOEN	Clock Compensation Override Enable This is a debug feature which allows the duration between clock compensation sequences to be reduced to 1280 cycles. 0 Clock compensation counter times out at 7424 cycles 1 Clock compensation counter times out at 1280 cycles
29	Reserved and must be written with zero.
30:31	Reserved This read-only bitfield is reserved and always has the value zero.

106.5.3 NAL Training Control Register (NAL_TCR)

The NAL_TCR supports static training as described in *Section 106.4: Static training*.

Access: Read/Write

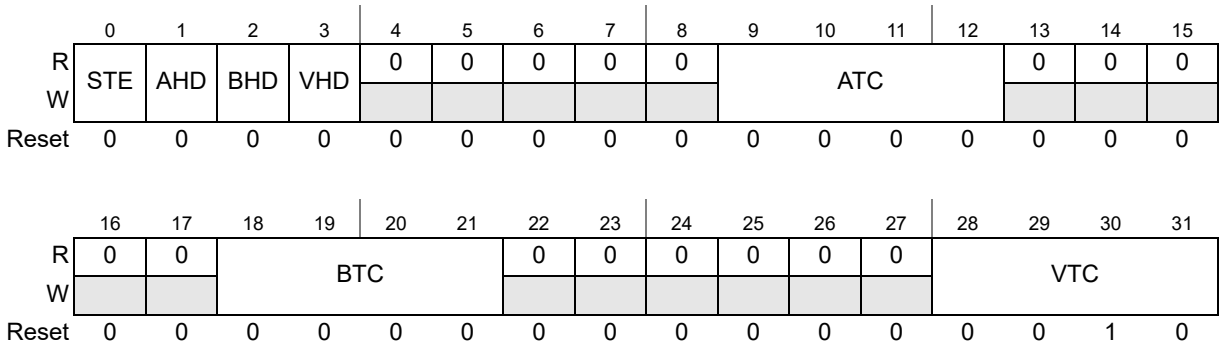


Figure 2614. NAL Training Control Register (NAL_TCR)

Table 2496. NAL_TCR field descriptions

Field	Description
0 STE	Static Training Enable 0 Reserved 1 Timer-based training method is used, ie. RX data is ignored and training is established based on ATC/BTC/VTC timers.
1 AHD	Hold in Align 0 Follow Aurora training sequence 1 Remain in align until this bit is cleared (applies even during duplex-mode training)
2 BHD	Hold in Bond 0 Follow Aurora training sequence 1 Remain in bond until this bit is cleared (applies even during duplex-mode training)
3 VHD	Hold in Verify 0 Follow Aurora training sequence 1 Remain in verify until this bit is cleared (applies even during duplex-mode training)
4:8	Reserved and must be written with zero.
9:12 ATC	Align Timer Count 0000 Remain in align for 16 cycles 0001 Remain in align for 24 cycles 0010 Remain in align for 32 cycles 0011 Remain in align for 48 cycles 0100 Remain in align for 64 cycles 0101 Remain in align for 96 cycles 0110 Remain in align for 128 cycles 0111 Remain in align for 192 cycles 1000 Remain in align for 256 cycles 1001 Remain in align for 384 cycles 1010 Remain in align for 512 cycles 1011 Remain in align for 768 cycles 1100 Remain in align for 1024 cycles 1101 Remain in align for 1536 cycles 1110 Remain in align for 2048 cycles 1111 Remain in align for 3072 cycles
13:17	Reserved

Table 2496. NAL_TCR field descriptions (continued)

Field	Description
18:21 BTC	Bond Timer Count 0000 Remain in bond for 16 cycles 0001 Remain in bond for 24 cycles 0010 Remain in bond for 32 cycles 0011 Remain in bond for 48 cycles 0100 Remain in bond for 64 cycles 0101 Remain in bond for 96 cycles 0110 Remain in bond for 128 cycles 0111 Remain in bond for 192 cycles 1000 Remain in bond for 256 cycles 1001 Remain in bond for 384 cycles 1010 Remain in bond for 512 cycles 1011 Remain in bond for 768 cycles 1100 Remain in bond for 1024 cycles 1101 Remain in bond for 1536 cycles 1110 Remain in bond for 2048 cycles 1111 Remain in bond for 3072 cycles
22:27	Reserved
28:31 VTC	Verify Timer Count 0000 Remain in verify for 16 cycles 0001 Remain in verify for 24 cycles 0010 Remain in verify for 32 cycles 0011 Remain in verify for 48 cycles 0100 Remain in verify for 64 cycles 0101 Remain in verify for 96 cycles 0110 Remain in verify for 128 cycles 0111 Remain in verify for 192 cycles 1000 Remain in verify for 256 cycles 1001 Remain in verify for 384 cycles 1010 Remain in verify for 512 cycles 1011 Remain in verify for 768 cycles 1100 Remain in verify for 1024 cycles 1101 Remain in verify for 1536 cycles 1110 Remain in verify for 2048 cycles 1111 Remain in verify for 3072 cycles

107 Buddy Device—Nexus Aurora PHY (NAP)

107.1 Introduction

The Nexus Aurora PHY (NAP), in conjunction with the Nexus Aurora Link (NAL), supports one-way simplex high-speed serial communication with an external debugger. Configuration settings in the Nexus Aurora Router (NAR) Nexus Aurora Phy control register (NAR_NAPCR) specify the NAP and NAL lane configuration. The NAL performs all logical layer functions, flow control, data-striping and sizing based on the number of data lanes available for use, and physical coding sublayer functions on the data being sent to the NAP for transmission. The NAP receives 8b/10b encoded data from the NAL then serializes and transmits this data through a given PHY lane's Low Voltage Differential Signaling (LVDS) buffers to an external debugger.

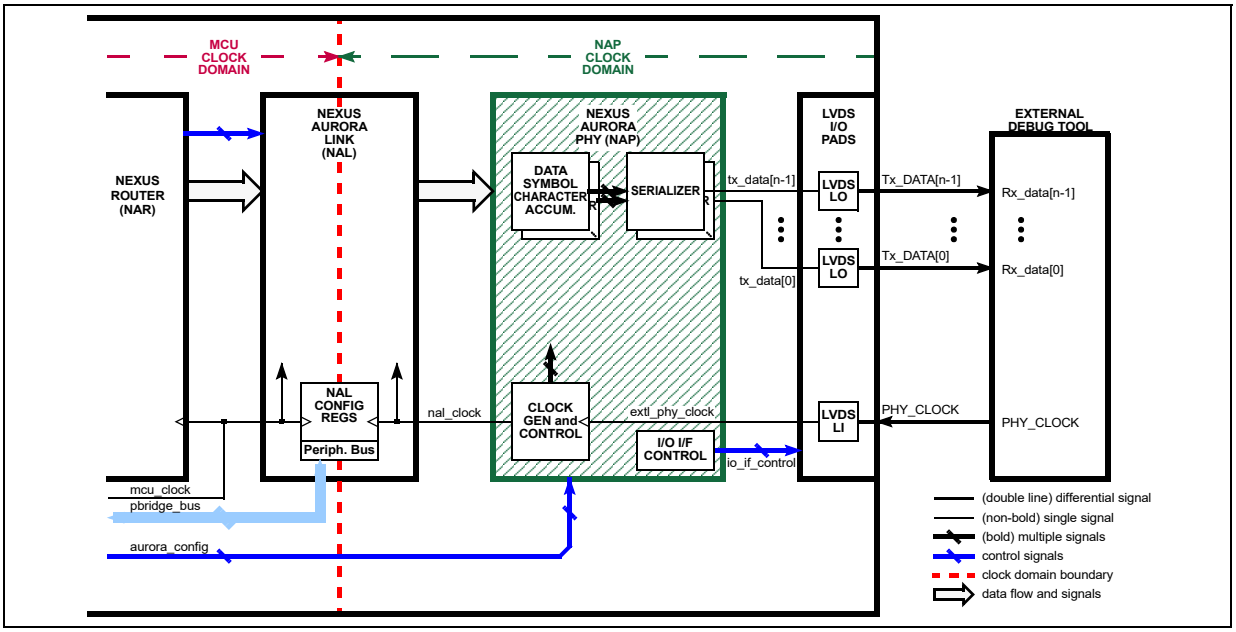
Each NAP lane consists of a data symbol character accumulator which registers the 10-bit character for that lane from the NAL on the rising edge of the NAL clock. Next, a serializer takes the 10-bit character and serializes it into the individual data bits that are then transmitted at the PHY clock rate via the LVDS output drivers to the external debugger receive channel.

Note: The NAL performs static link training for an attached debugger's PHY. The NAP is not directly aware of the link training operation, since it merely transmits the data presented to it by the NAL.

On the receive side of the external debugger, it is the responsibility of the external debugger to perform elastic buffering, symbol detection, symbol locking (called Lane Alignment in Aurora terminology), 10b/8b decoding, decode and disparity error tracking in the received data stream.

Figure 2615 shows a high level block diagram of the High Speed Nexus Interface and the NAP.

Figure 2615. High speed Nexus interface block diagram



107.1.1 Features

The features of the NAP are as follows:

- Xilinx Aurora Protocol Specification V2.x compliant
- 1 Gb/s per channel trace data payload bandwidth with 1.25 GHz clock
- Simplex mode with static (timer-based) channel training

107.1.2 Modes of operation

The operating mode of the NAP is determined by the device control signals. The functional configuration is determined by the NAR_NAPCR setting; functional operation is enabled by the NAR.

107.1.2.1 Reset

During Nexus Reset all internal NAP resources are reset, all output ports are disabled, and all inputs are ignored. Since there is no device clock domain logic in the NAP itself, the NAP is unaffected by device system reset. Depending on the NAP configuration settings, the NAP may activate upon exiting Nexus Reset if any Aurora lanes are enabled during this reset mode, else it remains disabled and inactive.

107.1.2.2 Link training

Before beginning trace output mode operation, the Aurora Link must go through a training process conducted by the NAL to ensure that an external debugger can correctly reconstruct the transmitted data stream. Training proceeds once the link powers up, and must complete successfully before data transmission can occur.

The NAL initializes and trains the Aurora Link in a simplex configuration using a timer based static-training method. The initialization and training sequence described in Aurora Protocol Specification progresses from one step to the next via the use of programmable timers contained in the NAL. A timer is programmed for each stage of training and as each timer expires, the training sequence proceeds to the next stage until complete. The NAP is not directly aware of the link training operation, since it merely transmits the data presented to it by the NAL.

107.1.2.3 Transmit

During Nexus trace output operation, it is the responsibility of the NAR to push data into the Aurora Link at a sufficiently high rate to ensure that the link does not become data-starved. The NAL and NAP support up to 4 Transmit lanes. The Aurora encapsulated data is modified by the NAL to ensure the appropriate number of symbols are transmitted through the NAP.

107.2 External signal description

The external signals of the NAP are listed in *Table 2497*. Each of the NAP signals consists of a positive and negative differential pair.

Table 2497. NAP signals

Signal Name	Direction	Description
PHY_CLOCK (CLKP/CLKN)	Input	NAP clock input from external debug tool
TX Data0 (TX0P/TX0N)	Output	NAP Tx data for lane 0
TX Data1 (TX1P/TX1N)	Output	NAP Tx data for lane 1
TX Data2(TX2P/TX2N)	Output	NAP Tx data for lane 2
TX Data3(TX3P/TX3N)	Output	NAP Tx data for lane 3

107.3 Memory map and register definition

There are no control, configuration or status registers within the NAP. All control, configuration and status options for number of lanes and LVDS pads are handled in the NAR_NAPCR which is described in the NAR Chapter.

107.4 Functional description

High-level descriptions of the operation of the NAP and its main functional blocks are given in the following sections.

107.4.1 Data symbol character accumulator

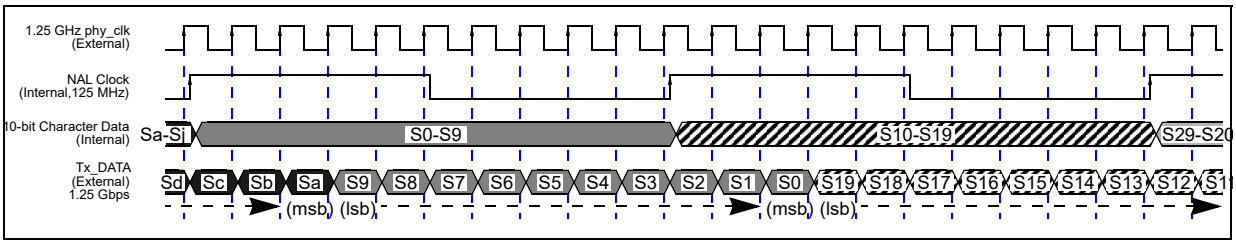
The data symbol character accumulator portion of the NAP for each lane is made up of a symbol character register whose inputs are connected to the 10-bit NAL character bus. The data symbol character accumulator essentially registers a 10-bit character for that lane simultaneous with the rising edge of the NAL clock and presents the 10-bit quantity to the serializer at the appropriate times.

107.4.2 Serializer

The serializer portion of the NAP for each lane takes the data captured by that lane's Data Symbol Character Accumulator and demuxes the data to provide a single serial stream of symbol data to be output by the LVDS LO buffer for that lane at the PHY clock rate.

Figure 2616 shows a high level timing diagram that demonstrates the operation of the serializer for a NAP lane, where S0–S9 represents a 10-bit symbol, S10–S19 represent the next symbol, and so on.

Figure 2616. Serializer timing diagram



107.4.3 Clock generation and control

The Clock Generation and Control block is responsible for generating the necessary clocks and state signals needed by the rest of the NAP. The Data Symbol Character Accumulators and serializers are the main consumers of the signals generated by the Clock Generation and Control block.

107.5 Initialization information

The NAP does not require initialization other than the Nexus Reset and programming of the NAR_NAPCR to configure the number of lanes and LVDS pads.

Appendix A Acronyms and abbreviations

Table 2498 lists acronyms and abbreviations used in this document.

Table 2498. Acronyms and abbreviations

Term	Meaning
AHB	Advanced high-performance bus
AIC	PBRIDGE (AIPS) integrity checker
AMBA	Advanced microcontroller bus architecture
APU	– Advanced protocol unit – Auxiliary processing unit
ASILD	Automotive SIL D
AUTOSAR	Automotive open system architecture
AV	Audio video
BAF	Boot assist flash
BIST	Built-in self test
BIU	Bus interface unit
CAN	Controller area network
CF	Critical fault
CGM	Clock generation module
CMU	Clock monitoring unit
CPU	Central processing unit
CQM	Clock quality monitor
CRC	Cyclic redundancy check
CSI	Combined serial interface. DSPI configuration that alternates DSI and SPI frames.
CTU	Cross triggering unit
DCB	Data center bridging
DCF	Device configuration format
DMA	Direct memory access
DMAMUX	Direct memory access controller multiplexer
DSPI	Deserial serial peripheral interface
DSI	Deserial serial interface. DSPI configuration that serializes and deserializes registers or signals to allow for pin reduction.
ECC	Error correction code
ECSM	Error correction status module
EIM	Error injection module
EMI	Electromagnetic interference
EOUT	Error out

Table 2498. Acronyms and abbreviations (continued)

Term	Meaning
ESD	Electromagnetic static discharge
FCCU	Fault collection and control unit
FMPLL	Frequency-modulated phase-locked loop
FOSU	FCCU output supervision unit
FPEC	Flash program erase controller
FSM	Finite state machine
FTTI	Fault tolerant time interval
GPIO	General-purpose I/O
HVD	High voltage detector
HW	Hardware
ID	LIN identifier field
IEEE	Institute of electrical and electronics engineers
INTC	Interrupt controller
INTF	Interface
IP	Intellectual propriety
IPS	– Internal peripheral system – IPI slave bus
IRQ	Interrupt request
ISR	Interrupt service routine
JDC	JTAG data communication
JEDEC	Joint electron device engineering council
JTAG	Joint test action group
KGD	Known good die
LBIST	Logic built-in self test
LIN	Local interconnect network
LINFlexD	Local interconnect network controller
LLDP	Link layer discovery protocol
LVD	Low voltage detector
LWB	Late-write buffer
MB	Message buffer
MBIST	Memory built-in self test
MBM	Message buffer management
ME	Mode entry module (MC_ME)
MEMU	Memory error management unit
MFD	Multiplication factor divider

Table 2498. Acronyms and abbreviations (continued)

Term	Meaning
Mux	Multiplex
NCF	Non-critical fault
NMI	Non-maskable interrupts
NVM	Non-volatile memory
OPP	Over-program protection
OSC	Oscillator
OTP	One time programmable
PASS	Password and device security module
PBRIDGE	Peripheral bridge
PCM	Platform configuration module
PCU	Power control unit
PFLASH	Platform flash controller
PIT	Periodic interrupt timer
PLL	Phase-locked loop
PMC	Power management controller
PRAM	Platform RAM controller
PWD	Password
RC	– Resistance-capacitance – Resistor-capacitor
RCC	Redundancy control checkers
RCCU	Redundancy control checker unit
RCOSC	RC oscillator
RF	Recoverable fault
RFD	Reduced frequency divider
RGM	Reset generation module
RMW	Read-modify-write
RTL	Register transfer language
RWW	Read-while-write
RX	Receive
SAG	Safety application guide
SAR ADC	Successive approximation register analog-to-digital converter
SIL	Safety integrity level
SIUL	System integration unit lite
SMPU	System memory protection unit
SoC	System-on-chip

Table 2498. Acronyms and abbreviations (continued)

Term	Meaning
SOR	Sphere of redundancy
SPI	Serial peripheral interface
SSCM	System status and configuration module
STCU	Self-test control unit
STM	System timer module
SW	Software
SWT	Software watchdog timer
TBD	To be defined
TCD	Transfer control descriptor
TSENS	Temperature sensor
TX	Transmit
UART	Universal asynchronous/synchronous receiver transmitter
UF	Unrecoverable fault
VCO	Voltage controlled oscillator
VREG	Voltage regulator
WDG	Watchdog timer
WKPU	Wakeup unit
WS	Wait state
XBAR	Crossbar
XBIC	XBAR integrity checker

Revision history

Table 2499. Document revision history

Date	Revision	Changes
04-May-2015	1	Initial release.
08-Feb-2016	2	Throughout document: replaced “LINFlex” to “LINFlexD”; editorial and formatting changes *Chapter 1: Preface* *Section 1.4: Reference documents*: – Replaced “http://www.power.org/” by “Power Architecture web site”. *Chapter 2: Introduction* Removed instance of “Checker Core 1”. *Section 2.1.2: Overall architecture*: – “...two pairs of CPUs...” updated to “...one pair of CPU...”. – added “one CPU without lock-step, a DMA move engine”. – “DMA” modified to “a DMA move engine, fast local memories inside the CPU, fast system memories”. *Section 2.1.3: Core features*: – Operating frequency updated to “180 MHz”. – Removed “They include enhancements....(DSP) algorithms”. – Programmable CPU updated from 7 to 6. – CPU frequency updated from “100MHz” to “90MHz”. *Section 2.1.4: Core_2*: – Number of processor cores updated from 7 to 6. – Operating frequency updated from “200MHz” to “180MHz”. – “64-bit address bus and a 32-bit data bus” updated to “64-bit data bus and a 32-bit address bus”. *Section 2.1.5: Memory hierarchy*: – Operating frequency updated from “200MHz” to “180MHz”. – Peripheral frequency updated to “180 MHz or 45MHz”. *Section 2.1.6: Features*: – Replaced “Boot assist Flash (BAF) supports factory programming using a serial bootload through the asynchronous CAN or FlexRay” by “Boot assist Flash (BAF) supports factory programming using a serial bootload through the asynchronous CAN or LIN/UART”. – Added bullet “Standby power domain with smart wake-up sequence”. – Removed last bullet “6582 KB on-chip Flash” as it is redundant. – Added Ethernet controller feature. – Instance of “DSP” updated to read “DSP on Core_2”. – SRAM updated to 608KB. – “64+32 channels per eDMA” updated to “One eDMA with 64 channels; One eDMA with 32 channels”. – Removed “one INTC per computing shell”. – Number of INTC updated to “1”. – “GTM 303” updated to “GTM 343”. – LinFlexD modules updated to “Eighteen”.

Table 2499. Document revision history (continued)

Date	Revision	Changes
08-Feb-2016	2 (cont.)	*Table 3: Features List*: – “Number of checker cores” updated to “1”. – “SIMD” updated to “SIMD (LSP)”. – “System RAM” updated to “608 KB (including 256KB of Standby RAM)”. – Removed instance of “External bus”. – “LinFlexD” updated to “18”. – “MCAN/CAN-FD/TTCAN” updated to “M_CAN supporting CAN-FD according to ISO 11898-1 2015 (instances supporting also TTCAN)”. Its value also updated from “7/8/1” to “8 (1)”. – “Microsecond channel downlink” updated to “2”. – “Ethernet” updated to “2”. – “Interrupt controller” updated to “710 sources”. – “integrated switch mode voltage regulator” updated to “yes”. *Section 2.1.8: Packages*: – Replaced “The SPC58NE84/SPC58EG84 is also....device footprint” by “The eLQFP176 and LFBGA292 packages....device footprint”. – 100-pin package removed (eTQFP100). Updated *Figure 2: Block Diagram*. Updated *Figure 3: Periphery allocation*. *Chapter 3: Embedded memories* Added *Section 3.2.2: Standby SRAM regulator*. Updated *Section 3.2.4: Processor core local RAM*. *Section 3.3.2.1: Features*: – Updated ECC bit detection and correction. Modified *Figure 7: Flash memory segmentation and read-while-write partitioning*. Updated error detection and correction setting in *Section 3.3.2.4: Chip-specific C40FMC Flash register information* and *Section 3.4: End-to-end Error Correction Code (e2eECC)*. Updated *Section 3.5: Security features*. *Chapter 4: Signal Description* *Section 4.2: Emulation package*: – Text “A 216 pin.... on the die” updated to “The eLQFP176....device footprint”. *Chapter 5: Memory Map* *Table 7: Overview memory map*: – Used size of “Large Flash Blocks - no overlay” updated to “5888 KB”. – Used size of “Large Flash Blocks - overlay enabled mapping” updated to “5888 KB”. – Footnote “This memory range is applicable on PFLASH_1 and is not accessible by NAR” removed. – Used size of “System RAM” updated to “608 KB”. – Used size of “HSM” updated to “40 KB”. – For “Processor Core local SRAM”, used size updated to “160 KB DMEM, 48 KB IMEM”.

Table 2499. Document revision history (continued)

Date	Revision	Changes
08-Feb-2016	2 (cont.)	*Table 8: Peripheral (PBRIDGE_0, PBRIDGE_1, PBRIDGE_2) memory map*: – Corrected the end address of the reserved section before MC_CGM – Added footnotes *Table 9: Flash memory map*: – For “HSM Data Flash Blocks”, Partition ID updated to 4. – Flash 64 KB Data Block 0 merged into a single line. – Flash 64 KB Data Block 1 merged into a single line. – For Flash 64 KB Data Blocks 2 and 3, Partition ID updated to 3. – For Flash 16 KB Code Block 3, Partition ID updated to 1. – For Flash 16 KB Code Block 2, Partition ID updated to 0. – For “Flash 256 KB Code Block 14 to 21”, Partition ID updated to 5. – Added new section “Mirror Low & Mid Flash Blocks”. – Added new section “Mirror Large Flash Blocks”. *Table 10: Emulation and debug map*: – Columns “BD1M” and “BD2M” removed. – Buddy Overlay RM merged into a single row. Updated *Table 11: RAM memory map*. Updated *Table 12: Processor local RAM memory map*. Updated *Table 13: UTest Flash memory map*. *Chapter 6: Functional Safety* Updated *Table 16: Module classification*. Updated *Table 17: FCCU failure inputs*. *Chapter 7: Device configuration* Extensive updates throughout the chapter. *Chapter 8: Reset and Boot* Re-worked the whole chapter according to 40 nm product specifications. *Chapter 9: Device Configuration Format (DCF) Records* Updated *Section 9.1: Introduction*. *Table 132: UTEST Miscellaneous DCF Client field descriptions*: – Updated OPT_RGM_ESR0_HW, ESR0 GATE field descriptions *Table 130: DCF client list*: – Fully checked and updated – Renamed PMC_VREG1P2_EN by PNREG_ENB. – Updated 0x0211 User Serial Boot – Updated 0x0212-0x0214 Reserved – Updated 0x0215 Boot CPU IVPR – Updated 0x0212-0x7FFF Reserved – Removed ESR0_CONFIG Updated *Section 9.4.3: Miscellaneous DCF registers*: – Replaced bit 22 “Reserved” status with “XOSC_AUTOSWITCHB”.

Table 2499. Document revision history (continued)

Date	Revision	Changes
08-Feb-2016	2 (cont.)	*Chapter 10: Power management* *Section 10.1: Overview*: – Added bullet Smart Standby Wake-up Unit (SSWU). – Removed tables “SYSTEM modes” and “USER modes”, figure “Mode relationships” as these information are present in MC_ME Chapter, added cross-reference to this chapter. *Figure 56: Device power management framework*: – Removed boxes ‘FUSE’ and ‘STANDBY_ECTU_0’; replaced boxes ‘Processor core n’ and ‘IOP’ by ‘Core 0’, ‘Core 1’, ‘Core 2’. – Added PD2 and PD3 power domains. *Section 10.1.3: Power management controller overview*: – Updated bullet “Internal regulator” and added: ‘An internal Switching Mode Power Supply (SMPS) regulator, using external PMOS, NMOS and inductance’. – Updated bullet “Low-power regulator”. – Added in bullet “Auxiliary and clamp regulators“: “Auxiliary and clamp regulators can not be enabled in SMPS regulator mode”. *Table 138: Device power management controller external signals*: – Updated VDD_HV_FLA type and description. – Removed External low voltage power supply for VDD_LV. *Figure 58: Voltage Monitor Coverage*[(1)(2)]: added MVD082_B and MVD270_SB, removed MVD240T_C; fixed Footnote. *Section 10.2: Low power mode support*: replaced sentence “Any register leading to low power entry is triple-voted” by “Inadvertent entry into low power mode is prevented by triple voting of the control flops, both SW readable and otherwise”. *Section 10.2.1: Low power mode (HALT/STOP/STANDBY)*: Section title renamed and content reworked. Added *Section 10.2.1.2: Checker core disabling*. *Table 139: Voltage monitors description*: – Added POR031_C and POR200_C. – Removed VDD_HV_PMC (as well as in *Section 10.6: Power sequence*) – Added rows MVD082_B, MVD270_SB, updated row LVD400_IM with ‘VD14’ (was ‘VD13’), removed row MVD240T_C. *Figure 60: Voltage monitors functionality*: added MVD082_B and MVD270_SB, removed MVD240T_C. *Table 140: Voltage monitors configurability*: added LVD100_SB. *Figure 64: VDD_HV monitored voltages during power-up*: renamed VDD_HV_PMC by VDD_HV_IO. *Chapter 11: Smart Stand-by Wake-up Unit (SSWU)* Renamed the title of *Section 11.4: Clocking*. Renamed the title of *Figure 68: SSWU clock architecture*. *Table 160: CTUCR field descriptions*: Renamed CTU_ADC_reset to CTU_FSM_reset. Removed the following register descriptions: CTUEFR, CTU_CFGSR1, CTU_EXPECTED, CTU_CNT_RANGE. Updated register descriptions for: THCR1, THCR2, CLR1-CLR32 (ADC client), CLR1-CLR32 (PDC client), CLR1-CLR32 (TU client).

Table 2499. Document revision history (continued)

Date	Revision	Changes
08-Feb-2016	2 (cont.)	Section 1.5.2.1: PDC channel handling and Section 1.5.2.2: OPC channel handling: inserted reference to “the Excel pinout file attached to the SPC58xEx/SPC58xGx Reference Manual (I/O Signal Table)”. *Section 11.7.2.1: SSWU registers Configuration*: Added CTU FSM reset before next stand-by event. *Section 11.7.2.2: Other IP registers configuration*: – Added trigger mode selection and changed the PWDN to 1. – Removed “for Analog Comparator Block”. – Added Caution. *Chapter 14: e200z4d Core Complex Overview* Changed number of cores from 6 to 5 *Chapter 15: Core (z420n3) Description* Chapter title changed from “Core (z420n3) Overview” to “Core (z420n3) Description”. *Figure 101: Core (z420n3) Supervisor Mode Programmer’s Model SPRs*: Updated with the correct picture (was Model DCRs and PMRs). *Chapter 16: Core (z425n3) Description* Chapter title changed from “Core (z425n3) Overview” to “Core (z425n3) Description”. *Figure 126: Core (z425n3) Supervisor Mode Programmer’s Model SPRs*: Updated z425n3 picture from z420n3 picture. *Figure 127: Core (z425n3) Supervisor Mode Programmer’s Model DCRs and PMRs*, *Figure 128: Core (z425n3) User Mode Programmer’s Model SPRs*, and *Figure 129: Core (z425n3) User Mode Programmer’s Model PMRs*: Added Core (z425n3). *Chapter 17: System Integration Unit Lite2 (SIUL2)* *Section 17.1.1: Overview*: – Changed DMA inputs from ‘9’ to ‘16’ in the sentence “There are 16 external interrupts and 16 DMA inputs to the SIUL2 that map to EIRQ pins on the device.”. – Updated *Figure 156: SIUL2 block diagram* *Section 17.2.1: Memory map*: *Table 261: SIUL2 memory map*: – Removed Width, Access and Reset value columns. – Updated registers PGPFO, PGPDI, MPGPDO (n=0 to 12 instead of n=0 to 31). *Section 17.1.2: Features*: – Removed the number of independant DMA channels in the bullet “Independent DMA channels for each EIRQ pin”. – Added “Details about the electrical parameters and setting of functional pads can be found ...”. *Section 17.2.2.1: MCU ID Register #1 (SIUL2_MIDR1)*: – In “PKG” field description, removed unused package settings *Section 17.2.2.5: DMA/Interrupt Request Select Register 0 (SIUL2_DIRSR0)*: Bits 16 to 31 updated to DIRSR[16:31]. *Table 272: SIUL2_SCR0 field descriptions*: – Bit 0 updated from “FEC_0_MODE” to “ETHERNET_0_MODE”. – Bit 1 updated from “FEC_1_MODE” to “ETHERNET_1_MODE”. – Description of Bit 0 updated to “Ethernet Controller mode setting”.

Table 2499. Document revision history (continued)

Date	Revision	Changes
08-Feb-2016	2 (cont.)	*Section 17.2.2.11: Soc Configuration Register 0 (SIUL2_SCR0)*: – Replaced FEC_0 with ETHERNET_0 in the introduction sentence. – Bit 0 updated from “FEC_0_MODE” to “ETHERNET_0_MODE”. – Bit 1 updated from “FEC_1_MODE” to “ETHERNET_1_MODE”. – New bitfields created. – Added paragraph “MASK_OPCn is used to prevent the SSWU OPC channel n to drive the pad during standby. PADn_MUX_SEL is used to maintain the Pad value at standby exit (PADn_MUX_SEL = 0)” and added note “PADn_MUX_SEL must be set to 1 to be able to change the pads value and direction after stand-by while system is in the RUN mode.”. *Section 17.2.2.12: I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IOn)*: – Removed bitfield “HYS”. – Set to ‘0’ the “IBE” and “WPUE” field reset values – updated description of ILS *Table 274: I/O MSCR Reset State Exceptions* – Removed HYS column. – Added PA[4] line – Updated reset values *Section 17.2.2.14: GPIO Pad Data Output Registers (SIUL2_GPDOn)* and *Section 17.2.2.15: GPIO Pad Data Input Registers (SIUL2_GPDIn)*: Added Note “For the exact number... to this Reference Manual)”. *Chapter 20: Peripheral Bridge (PBRIDGE)* *Table 292: Peripheral bridge memory map*: – Added address offset reserved areas. – Added OPACRn registers. – Added footnotes “Access to this address space will terminate in an error” and “The following address spaces when programmed... these address spaces.” *Table 301: OPACRx field descriptions*: Updated fields names to OPACRa-OPACRh (was PACRAa-PACRh). *Chapter 21: System Memory Protection Unit (SMPU)* Updated RGDn registers instances from n=0 to 11. *Section 21.4.2.5: Region Descriptor n, Word 0 (RGDn_WORD0)*, *Section 21.4.2.6: Region Descriptor n, Word 1 (RGDn_WORD1)*, *Section 21.4.2.7: Region Descriptor n, Word 2 Format 0 (RGDn_WORD2_FMT0)*, and *Section 21.4.2.8: Region Descriptor n, Word 3 (RGDn_WORD3)*: Changed register occurences from 12 to 24. *Chapter 24: Interrupt Controller (INTC)* *Section 24.5.2.7: INTC Priority Select Register n (INTC_PSRn)*: Changed the implementation of the PRC_SEL bitfield to PRC_SEL*n* where *n* is the number of the processor(s). *Section 24.4.1: Software vector mode*: “HVEN_PRCn bit” updated to “HVENn bit”. *Chapter 25: Enhanced Direct Memory Access (eDMA)* Updated *Table 331: DMA memory map*

Table 2499. Document revision history (continued)

Date	Revision	Changes
08-Feb-2016	2 (cont.)	*Chapter 26: DMA channel multiplexer (DMACHMUX)* Updated register name to CHCFG (was CHCONFIG) throughout the chapter. *Table 379: DMACHMUX memory map*: Corrected register name to CHCFG (was CHCNFG). *Figure 265: DMACHMUX block diagram*: Replaced Trigger #2 by Trigger #3. Updated *Section 26.1.2: Features*. *Chapter 27: Clocking* Updated *Figure 270: Clock generation(1)(2) (part 1 of 2)*. *Table 382: MC_CGM relationship to clocks*: Removed “1s” from “Output clock” column of CGM_SC_DC0 register. Added CGM_SC_DC4 register row. Removed CGM_AC0_DC4 CGM_AC0_SC, CGM_AC10_DC0 CGM_AC10_SC, CGM_AC10_DC1 CGM_AC10_SC rows. Updated “Output clock” column of CGM_AC12_DC2 CGM_AC12_SC register *Table 383: Maximum system level clock frequencies*: – Replaced value of “180” with “200” for System clock selector. – Updated nominal programmed divider max output frequencies of Aux Clock 8 Selector and of Aux Clock 11 Selector. – Added PFBRIDGE_CLK to System clock selector. – Removed ECTU_CLK from Aux Clock 0 Selector. – Removed Aux Clock 10 Selector. Split the table “Maximum system level clock frequencies” into two tables *Table 383: Maximum system level clock frequencies* and *Table 384: Maximum auxiliary level clock frequencies* and updated them. *Table 385: JTAG frequencies*: updated frequencies for PD and ED devices *Table 387: Dual PLL Digital Interface register reset values*: Set offset 0x0034 to Reserved. Updated *Section 27.9: Progressive clock switching*. *Figure 273: Clock distribution*: – Updated the figure – r/w_data has been renamed BIU in the PIT/RTI box. – Replaced FEC_RXCLK and FEC_TXCLK with ETHERNET_RXCLK and ETHERNET_TXCLK respectively. *Section 27.3: System clock frequency limitations*: Deleted text “In order....specifications”. *Section 27.6.1: HSM clock divider* : Added sentence “The minimum possible divider after reset is 'divide by 2”. *Section 27.6.4: Ethernet clocking*: Reworked contains and added new *Figure 275: Ethernet clocking*. *Section 27.6.8: System Clock/GTM output clock*: Removed “Ethernet clock pin muxing” from the heading. Removed FEC_REF_CLK from this section. Updated *Figure 277: SYSCLK/EXTCLK/REF_CLK pin muxing*. *Table 391: Clock input sources*: Replaced “CMU_13_ECTU” with “CMU_13_PFBRIDGE” and “ECTU” with “PFBRIDGE_CLK”. *Chapter 28: Dual PLL Digital Interface (PLLDIG)* *Section 28.3: Features*: – Added “The frequency range after the prescaler is required to be 8-20Mhz”. – Removed the information about inout and output clock frequency range. Added a note to consult the Device data sheet for the same.

Table 2499. Document revision history (continued)

Date	Revision	Changes
08-Feb-2016	2 (cont.)	*Table 394: Dual PLL Digital Interface memory map*: Removed “Access” and “Reset Value (hex)” columns. Reformated all end address of reserved spaces. *Figure 285: PLL1 Control Register (PLL1CR)* and *Figure 288: PLL1 Frequency Modulation Register (PLL1FM)*: added “default” to figure footnote. *Figure 283: PLL0 Status Register (PLL0SR)*. Updated reset value. *Section 28.5.2.3: PLL0 Divider Register (PLL0DV)* Updated the description of bitfields RFDPHI and MFD. *Chapter 30: Clock Generation Module (MC_CGM)* *Section 30.4.1.4.1: System clock divider synchronization*: Updated the section. Removed figure “MC_CGM auxiliary clock 10 generation overview”. *Section 30.4.2.1: Auxiliary clock dividers*: Updated the section. Removed register “Auxiliary Clock 0 Divider 4 Configuration Register (CGM_AC0_DC4)”. *Chapter 31: OSC Digital Interface (XOSC)* Updated *Table 485: XOSC configurations*. *Chapter 33: LPRC Digital Interface* *Section 33.3.1.1: Low Power RC Control register (CTL)*: Changed register reset value from 0x00000300 to 0x00000700. *Chapter 35: Platform RAM controller (PRAM)* *Section 35.3.1.2: Optional read wait-state*: Removed note. *Chapter 36: Flash memory controller* *Section 36.2: Features*: Added feature “double bit” to the exisitng featue “Support for reporting of single- and multi-bit flash ECC events on a 64-bit doubleword boundary”. *Section 36.5.15: e2eECC on data Flash accesses*: Added instance of “double bit correction”. *Table 500: PFCR1 field descriptions*: Modified APC bitfield description. *Section 36.4.1.7: PFlash calibration region descriptor n (PFCRDn)*: Renamed PFCRD registers PFCRDn.Word0-3 into PFCRDn_Word0-3. *Chapter 37: Embedded Flash Memory* Updated *Figure 399: Flash memory module structure*. Updated *Section 37.2.1.2: Data Flash*. *Table 516: UTEST block memory map*: Updated UTEST base address. Added *Section 37.2.1.4: RWR1_UTEST virtual block memory map*. *Table 518: Flash User Registers memory map*: Added registers ALTSELn, ELOCKn, PLOCKn and ADR2. *Table 518: Flash User Registers memory map*, *Figure 414: Program Address register (PAR)* and *Table 534: PAR field descriptions*: Changed register name from Parallel Address register (PADR) to Program Address regsiter (PAR).

Table 2499. Document revision history (continued)

Date	Revision	Changes
08-Feb-2016	2 (cont.)	*Table 535: ADR field descriptions*: Updated description of bitfield ADDR. Updated *Table 536: ADR update mode list* and *Table 537: ADR content: priority list*. *Table 544: UT0 field descriptions*: Updated description of bitfields CPA, CPR and CPE. *Section 37.3.2.18: User Multiple Input Signature registers*: Changed the bitfields representation of all registers in this section. *Section 37.4.5.1.2: Over-programming protection (OPP) enable*: Replaced doubleword by quad-word in the sentence “In an OPP enabled block...cannot be programmed again.” *Chapter 38: Flash Memory Programming and Configuration* In *Section 38.1: Selection of Flash memory blocks for erase*, – Removed page footnotes “The table lists the LOWLOCK register field...” and “Register bits in Power Architecture-based microcontrollers...” – Removed tables “Flash memory partition 1 memory map” and “Flash memory partition 4 memory map” – Changed value to write to SEL0 register to select for erase the blocks in *Table 510* from *0x0030_0000* to *0x0005_0000* – Changed value to write to LOCK0 register to select for erase the blocks in *Table 510* from *0x0030_0000* to *0x0005_0000* – Changed the Data[31:0] value of DCF record from *0x0030_0000* to *0x0005_0000* In *Section 38.8.2: Passcode-protected test mode disable seal*, changed value to program into the Test Mode Disable Seal location from *0x2D3C_4B5A* to *0x5A4B_3C2D* *Chapter 40: Analog-to-Digital Converters (ADC) Configuration* *Table 581: ADC implementation*: updated SAR_ADC_12bit_2 from ‘15’ to ‘13’ *Table 574: External analog input pins*: removed row ‘216 FusionQuad ED’ *Section 40.2.2.1.1: 12-bit Fast SAR ADC* and *Section 40.2.2.1.3: 10-bit Fast SAR ADC*: added “4 internal test channels...up to the external circuitry. This includes: four internal voltage...impedance” *Section 40.2.2.5: SAR ADC triggering*: added in ‘The bCTU...’ paragraph ‘or from EMIOS’. *Section 40.2.2.9.2: Internal reference*: updated ‘The SAR_ADC_12bit_B0’ by ‘All SAR_ADC’s’. *Figure 478: Analog input — discharge switch to VSS_HV_ADV*: updated ‘SAR_ADC_4’ by ‘SAR_ADC_n’. *Table 591: SAR_ADC analog test channel assignment*: from 124 to 127 added ‘SAR_ADC_12bit_0 to 3, SAR_ADC_10bit_0/1’ in column Fast SAR and Fast SAR channel numbers. *Section 40.2.2.11: SAR ADC register definitions* added. *Chapter 41: Sigma-Delta Analog-to-Digital Converter (SDADC) Digital Interface* *Section 41.3: Features*: Added new feature “Cascaded COMB...data path”. *Section 41.6.2: Module Configuration Register (SDADC_MCR)*: Updated description of PDR field. *Section 41.6.8: FIFO Control Register (SDADC_FCR)*: Updated description of “FTHLD” field. *Figure 479: SDADC block diagram*: Added “PSI” digital interface.

Table 2499. Document revision history (continued)

Date	Revision	Changes
08-Feb-2016	2 (cont.)	*Chapter 42: Successive Approximation Register Analog-to-Digital Converter (SARADC) Digital Interface* *Section 42.4.1.1: Start of normal conversion*: Text “If the trigger mode...conversion request” updated to read “If the trigger mode....MCR[NSTART] bit”. *Chapter 43: Temperature Sensor* *Section 43.2: Functional description*: Updated the section. *Chapter 44: Body Cross Triggering Unit (BCTU)* *Section 44.5: Functional description*: Removed table “Trigger source” as it is redundant with the assignment between eMIOS outputs and BCTU trigger inputs of the device that is defined in the *Chapter 7: Device configuration*. *Chapter 47: GTM343 Integration (GTMINT) Module* replaces the Chapter GTM303 Integration *Chapter 48: Generic Timer Module (GTM343)* replaces the Chapter Generic Timer Module (GTM303) *Chapter 49: GTM Development Interface (GTMDI)* *Table 725: GTMDI_DS field descriptions*: Description of field “HLT” updated. *Section 49.5.9: Fetch trace*: “Fetch trace ties...MCS channel” updated to “Fetch trace tries...in the MCS pipeline” *Section 49.5.9.2.1: Fetch trace start state*: Added text “When an instruction...valid fetch access”. Added new *Section 49.2: Overview*. Updated *Section 49.1: Introduction*. *Chapter 52: CAN Subsystem* Removed Section “Shared Memory”. Removed Section “Time Triggered Modular CAN (M_TTCAN) core”. *Section 52.1: Introduction*: In *Table 820: IP user manual versions*, changed version from 3.0.1 to 3.2.1 for M_TTCAN IP. *Section 52.3.5.2: Register map*: – Renamed register DBTP (was FBTP) – Renamed register NBTP (was BTP) – Added Transmitter Delay Compensation Register (TDCR) at address offset 0x0048 – Added Rx Buffer / FIFO Element Size Configuration (RXESC) at address offset 0x00BC – Added Tx Buffer Element Size Configuration (TXESC) at address offset 0x00C8 *Section 52.3.5.2.3: Data Bit Timing and Prescaler Register (DBTP)*: – Renamed register DBTP (was FBTP) – Added field TDCO – Renamed register DBRP (was FBRP) – Renamed bitfields DTSEG1, DTSEG2, and DSJW (was FTSEG1, FTSEG2, and FSJW) – Increased configuration range for data bit timing

Table 2499. Document revision history (continued)

Date	Revision	Changes
08-Feb-2016	2 (cont.)	*Section 52.3.5.2.4: Test Register (TEST)*: – Removed field TDCV and moved to register PSR *Section 52.3.5.2.6: CC Control Register (CCCR)* – Added 1 bitfield NISO – Replaced field FDBS by EFBI – Replaced field FDO by PXHD – Change bitfield FDBS by new control bit EFBI – Change bitfield FDO by new control bit PXDH – Removed bitfield CMR. – Replaced bitfield CME by BRSE and FDOE bitfields *Section 52.3.5.2.7: Nominal Bit Timing and Prescaler Register (NBTP)*: – Renamed bitfield NBRP (was BRP) – Renamed bitfields NTSEG1, NTSEG2, and NSJW (was TSEG1, TSEG2, and SJW) *Section 52.3.5.2.13: Protocol Status Register (PSR)*: – Added bitfield TDCV – Added bitfield PXE – Renamed bitfield DLEC (was FLEC) Added *Section 52.3.5.2.14: Transmitter Delay compensation Register (TDCR)* *Section 52.3.5.2.15: Interrupt Register (IR)*: Replaced bitfields STE, FOE, ACKE, BE and CRCE by ARA, PED and PEA. *Section 52.3.5.2.16: Interrupt Enable Register (IE)*: Replaced bitfields STEE, FOEE, ACKEE, BEE and CRCEE by ARAE, PEDE and PEAE. *Section 52.3.5.2.17: Interrupt Line Select Register (ILS)*: Replaced bitfields STEL, FOEL, ACKEL, BEL and CRCEL by ARAL, PEDL and PEAL *Section 52.3.6: Message RAM*: added note “Because the Message RAM is equipped with ECC functionality...” just after the first paragraph. *Section 52.3.6.1: Rx Buffer and FIFO element*: – Renamed bit EDL to FDF *Section 52.3.6.2: Tx buffer element*: – Added bits ESI, FDF and BRS Updated *Section 52.3.7.3: CAN FD Operation*. Updated *Section 52.3.7.4: Transceiver Delay Compensation*. Added *Section 52.9: External Time Stamp* and *Figure 863: CAN external time base* *Section 52.3: Modular CAN (M_CAN) cores*: changed caution “Implemented CAN FD mode is...” to note “Implemented CAN nodes are...”. Added *Section 52.3.5.2.14: Transmitter Delay compensation Register (TDCR)*. *Section 52.3.6.1: Rx Buffer and FIFO element*: Renamed bit ELD to FDF. *Section 52.3.6.2: Tx buffer element*: Added bit ESI (Error State Indicator), bit FDF (FD format) and bit BRS (Bit Rate Switching). Updated text in *Section 52.3.7.3: CAN FD Operation*. Updated text in *Section 52.3.7.4: Transceiver Delay Compensation*. *Section 52.4: Time Triggered Modular CAN (M_TTCAN) core*: – Removed “with data fields of up to eight bytes length” from paragraph “In addition the M_TTCAN supports...”. Changed caution “Implemented CAN FD mode is...” to note “Implemented TTCAN nodes are...”.

Table 2499. Document revision history (continued)

Date	Revision	Changes
08-Feb-2016	2 (cont.)	*Section 52.3.5.2.16: Interrupt Enable Register (IE)*: – in *Figure 713: Interrupt Enable Register (IE)* updated DRXE field description *Section 52.8: CAN nodes 1 and 2 I/Os sharing*: In the second sentence, changed M_CAN_0 to M_CAN_1. Updated *Figure 861: Hardware control for M_TTCAN cycles management* *Figure 765: Calibration Configuration Register (CCFG)* and *Table 881: CCFG field descriptions*: Changed SWR field access to RW (was R only). Added values 0x00–0xFF to OPCM field description. Added values 0x04–0x19 to TQBT field description. *Section 52.11: Shared memory map*: Removed note sentence“The write access protection is only for CPU access to the message memory.”. *Table 820: IP user manual versions*: Renamed (was “References”). *Figure 706: Timestamp Counter Value Register (TSCV)*, *Figure 708: Timeout Counter Value Register (TOCV)*, *Figure 712: Interrupt Register (IR)*, *Figure 706: Timestamp Counter Value Register (TSCV)* and *Figure 708: Timeout Counter Value Register (TOCV)*: Changed register access to R/W (was w1c). *Figure 712: Interrupt Register (IR)*: Apply w1c to all register fields. Updated *Figure 861: Hardware control for M_TTCAN cycles management*. *Chapter 53: Serial Interprocessor Interface (SIPI)* *Table 969: SIPI memory map*: Set End-offset value to all reserved spaces. *Table 977: Channel data Register address offsets*: Reworked register names from SIPI_CDR2[0] -SIPI_CDR2[7] to SIPI_CDR2*i* and offset addresses to “0x005C + *i**0x4 (*i* = 0 to 7)”. *Chapter 56: Ethernet* replaces Chapter “Fast Ethernet Controller” *Chapter 58: Inter-Integrated Circuit (I2C)* *Section 58.2.4: Features*: Removed bullet “Operation at higher baud rates...”. *Chapter 59: SENT Receiver (SRX)* *Section 59.1.1: Features*: Changed the maximum supported bit rate from 36 µs to 90 µs. *Section 59.3.2.1: Global Control Register (GBL_CTRL)*: Updated TSPRSC field from value from [1 to 255] to [0 to 255]. *Section 59.3.2.16: DMA Slow Serial Message Bit2 Read Register (DMA_SMSG_BIT2)*: Replaced field "DATA[11:8]" with "ID3_0_DATA11_8” in register figure and updated field description table accordingly. *Section 59.3.2.25: Channel n Serial Message Bit2 Read Register (CHn_SMSG_BIT2)*: – Replaced field "DATA[11:8]" with "ID3_0_DATA11_8” in register figure and updated field description table accordingly. – Replaced field "DATA[11:8]" with "ID3_0_DATA11_8” in register figure and updated field description table accordingly. *Figure 1426: Channel n Fast Message Data Read Register (CHn_FMSG_DATA)*: Updated the Offset address. *Figure 1408: Global Status Register (GBL_STATUS)* and *Figure 1424: Channel n Status Register (CHn_STATUS)*: Changed register access to R/W (was w1c).

Table 2499. Document revision history (continued)

Date	Revision	Changes
08-Feb-2016	2 (cont.)	*Chapter 62: Deserial Serial Peripheral Interface (DSPI)* *Figure 1578: DSPI block diagram*: Removed Slave Bus Interface and Clock/Reset arrows. *Chapter 63: LINFlexD* *Section 63.1: Introduction*: Updated with “2.2” version at the end of sentence starting with: “The LINFlexD supports LIN protocol version...”. *Chapter 64: Reset Generation Module (MC_RGM)* Updated the naming of all registers Peripheral Reset Status Register (RGM_PSTATn) by adding the word “Status”. *Table 1695: RGM_DES field descriptions*: Added Note to bitfield F_VOR_STDBY. *Table 1699: RGM_FES field descriptions*: Added Note to bitfield F_JTAG_FUNC. *Section 64.3.1.14: Peripheral Reset Register 1 (RGM_PRST1)*, *Section 64.3.1.18: Peripheral Reset Register 5 (RGM_PRST5)*, *Section 64.3.1.22: Peripheral Reset Status Register 1 (RGM_PSTAT1)*, *Section 64.3.1.26: Peripheral Reset Status Register 5 (RGM_PSTAT5)*: Updated ‘ADCSD_x’ by ‘SD_ADC_x’ to be consistent with other chapters (MC_ME PS*n* registers, Memory Map, ADC Config). *Section 64.3.1.15: Peripheral Reset Register 2 (RGM_PRST2)*: – CAN_RAM_SUB_0_CTR_RST bitfield description has been updated. – Added bitfield LINFLEXD_16_RST. *Table 1710: RGM_PRST3 register field descriptions* and *Table 1714: RGM_PRST7 register field descriptions*: Added note “This peripheral is coupled with RGM_PRST4.SAR_ADC_*_SEQ_RST therefore they have to be identically set” for register fields SAR_ADC_*. *Section 64.3.1.17: Peripheral Reset Register 4 (RGM_PRST4)*: Removed SAR_ADC_xxx_SEQ bitfields. *Section 64.3.1.18: Peripheral Reset Register 5 (RGM_PRST5)*: Removed bitfield PSI5_S_0_RST. *Section 64.3.1.19: Peripheral Reset Register 6 (RGM_PRST6)*: – CAN_RAM_SUB_1_CTR_RST bitfield description has been updated. – Added bitfield LINFLEXD_17_RST. *Section 64.3.1.20: Peripheral Reset Register 7 (RGM_PRST7)*: – Removed bitfields ECTU_0_RST and ECTU_1_RST. – Added bitfields ETHERNET_0_RST and ETHERNET_1_RST. – Removed bitfield FLEXRAY_1_RST. *Section 64.3.1.23: Peripheral Reset Status Register 2 (RGM_PSTAT2)*: Added bitfield LINFLEXD_16_STAT. *Section 64.3.1.25: Peripheral Reset Status Register 4 (RGM_PSTAT4)*: Removed SAR_ADC_xxx_SEQ bitfields. *Section 64.3.1.27: Peripheral Reset Status Register 6 (RGM_PSTAT6)*: Added bitfield LINFLEXD_17_STAT. *Section 64.3.1.28: Peripheral Reset Status Register 7 (RGM_PSTAT7)*: – Removed bitfields ECTU_0_STAT and ECTU_1_STAT. – Added bitfields ETHERNET_0_STAT and ETHERNET_1_STAT. – Removed bitfield FLEXRAY_1_STAT. *Section 64.3.1.10: ‘Functional’ Reset Escalation Threshold Register (RGM_FRET)*: Updated FRET bitfield description. *Section 64.3.1.11: ‘Destructive’ Reset Escalation Threshold Register (RGM_DRET)*: Updated DRET bitfield description.

Table 2499. Document revision history (continued)

Date	Revision	Changes
08-Feb-2016	2 (cont.)	*Chapter 65: Boot Assist Flash (BAF)* Updated *Section 65.1: Introduction*. *Section 65.2: Memory map*: Replaced the sentence “The BAF is located in a 16 KB block of flash that is mapped adjacent to the UTEST flash memory block“ by “For the BAF location in flash, refer to the Embedded memories chapter”. *Section 65.3.6: Production disable activation protocol and implementation*: updated flag value from ‘7FFF_FFFFh’ to ‘FFFF_FFFEh’. Updated *Figure 1765: Flow of control*. *Section 65.3.8: Optionally perform a serial boot*: Added the sentence “The BAF is executed via the XOSC as clock source”. *Section 65.4: Resources accessed by BAF code execution*: – Renamed “Clocks” to “Magic Carpet (clocks)”. – Added SSCM, SIUL, PASS and short description of each to resource list. *Application code execution*: Deleted the sentence “The boot header contains flags that enable/disable individual cores and their reset vectors”. *Figure 1771: Production Disable callback function flow chart* *Chapter 66: System Status and Configuration Module (SSCM)* *Section 66.2.1.4: HSM and User Option Status Register (UOPS)*: Added register bitfields BAF_FA, FL_STDBY and FL_RUN. *Table 1753: Life Cycle Slots*: Renamed table title and added two rows. *Section 66.3.1: DCF mechanism* – *Table 1752: Life Cycle Slots in Memory* *Chapter 67: Power management controller digital interface (PMC_Dig)* *Table 1754: Power Management Controller Memory Map*: Removed columns “Access” and “Reset value”. *Table 1755: 32-bit registers LV0, LV1, HV0, HV1 arrangement table*: Updated VD9 from “MVD2407T_C” to “Reserved”. *Table 1756: VD number versus VD name table*: Removed VD9. *Table 1759: RES_LV0 field descriptions*: Added Note to bitfield RES3_SB. *Table 1763: REE_LV1 field descriptions*: In description of bit[REE6_C], “RES_LV1 bit” replaced with “RES_LV1[RES6_C]”. *Table 1764: RES_LV1 field descriptions*: In description of bit[REE6_C], “REE_VD3 bit” replaced with “REE_LV1[REE6_C]”. *Section 67.5.24: Voltage selection of IO (VSIO)*: – Description “This register controls the....input and output buffers” added. – Reset value of bits “VSIO_IM0_IM1”, “VSIO_IF” and “VSIO_IJ” changed from 0 to 1. *Table 1781: PNREG_S field descriptions*: For fields “PREG_ENB” and “NREG_ENB” description, the “0” and “1” description reversed. *Figure 1819: Miscellaneous CTRL REG (MISC_CTRL_REG)*: Bit 29, “SIRC_ENB” replaced with “RCOSC1M_ENB”. *Table 1793: BIST_CTRL field descriptions*: Description of “STATUS” bit updated. *Table 1797: BIST_DEBUG field description*: Description of “VD_MON” bit updated. *Section 67.5.25: PMOS/NMOS Regulator Status Register (PNREG_S)*: Description updated to “This register shows....external regulator mode”. *Section 67.5.39: User BIST Time3 and Time2 register (BIST_TIME32)*: Reset value of bit “TIME_2” changed from “00F9” to “03E7”.

Table 2499. Document revision history (continued)

Date	Revision	Changes
08-Feb-2016	2 (cont.)	*Chapter 68: Power Control Unit (MC_PCU)* Added *Section 68.3.2.3: Power Domain #2 Configuration Register (PCU_PCONF2)* and *Section 68.3.2.4: Power Domain #3 Configuration Register (PCU_PCONF3)*. *Section 68.3.2.5: Power Domain Status Register (PCU_PSTAT)*: added fields “PD2” and “PD3” *Chapter 69: Mode Entry Module (MC_ME)* *Section 69.3.2.26: Peripheral Control registers (ME_PCTLn)*: Removed bitfield DBG_F. *Section 69.3.2.27: Core Status register (ME_CS)*: Added fields - ChkINTC_Stat, ChkDMA1_Stat, ChkDMA0_Stat, S_CHK_CORE5, and S_CHK_CORE2 to bits 21, 22, 23, 26, and 29 respectively. Also added description to the fields. *Table 1803: MC_ME register description*: – Added register PCTL84 at offset 0x114 (S_LINFlexD_16 Control). – Added register PCTL212 at offset 0x194 (S_LINFlexD_17 Control). *Figure 1860: Peripheral Status register 2 (ME_PS2)*: Added field S_LINFlexD_16 *Figure 1864: Peripheral Status register 6 (ME_PS6)*: Added field S_LINFlexD_17 at bit 11. *Chapter 72: Debug and Calibration Interface (DCI)* Added JDC and SPU blocks. Clarified the text in *Section 72.1.3.6: Break control* after . : Renamed pd_evti_b to evti_b. *Chapter 75: JTAG Data Communication (JDC)* *Section 75.3.1.2: Module status register (JDC_MSR)*: Changed access type of register JDC_MSR to w1c (was RW). *Section 75.2: Overview*: removed sentence “The JTAG accessible registers...“. *Section 75.3.1.3: JTAG output IPS data register (JDC_JOUT_IPS)*: renamed field “JOUT_IPS Data“ to “DATA“. *Section 75.3.1.4: JTAG input IPS data register (JDC_JIN_IPS)*: renamed field “JIN_IPS Data“ to “DATA“. *Chapter 78: Debug Zipwire* *Figure 2042: Block diagram showing connections for Emulation Device* and *Figure 2043: Block diagram showing connections for the Emulation Device* : Replaced “FEC” with “Ethernet”. *Chapter 86: Cyclic Redundancy Check (CRC) Unit* *Table 2170: CRC unit memory map*: Changed CRC registers per context from (n=1 to 4) to (n=0 to CRC_CNTX_NUM - 1 with CRC_CNTX_NUM = 4). *Figure 2277: CRC computation flow*: Reworked picture (context values). *Chapter 88: Indirect Memory Access (IMA)* *Table 2205: IMA_READ_DATA_n field descriptions*: Changed field name to READ_n (was WRITE_n).

Table 2499. Document revision history (continued)

Date	Revision	Changes
08-Feb-2016	2 (cont.)	*Chapter 89: Fault Collection and Control Unit (FCCU)* All signal names (such as ipg_clk) are replaced by thier respectives description names (System clock) throughout the document. Changed register names from NCF/UF (Noncritical / Unrecovarable Faults) to CF/RF (Critical / Recoverable Faults). *Section 89.6.21: FCCU XTMR Register (FCCU_XTMR)*: Updated the second sentence with “loads (a value corresponding to T_min=250 us + value programmed in FCCU_DELTA_T register) ...which has EOUT signaling enabled.”. *Figure 2334: FCCU RF Status register n (FCCU_RF_Sn)*: Changed reset value from 0x0 to the note “The reset value is chip-specific; see the chapter that describes how modules are configured and connected.” *Figure 2335: FCCU RF Key register (FCCU_RFK)* and *Figure 2346: FCCU RF Fake Register (FCCU_RFF)*: Replaced access “Write” by “Write-only”. *Chapter 90: Self-Test Control Unit (STCU2)* *Figure 2408: STCU2 MBIST Control k Register (STCU_MB_CTRLk)*: Updated offset value to “Offset: 0x0600 + n * 0x4 (n = 0 to 100)”. *Chapter 92: Password and Device Security Module (PASS)* – Renamed bitfield L_128LCK into A256KLOCK. – Renamed bitfield U_256LCK into A256KLOCK. *Table 2310: PASS memory map*: – Changed LOCK3_PG*n* reset values to 0xXFXXXXXX (was 0xXXXXXXXX). – Changed LCSTAT reset value to 0xX0000007 (was 0x000000XX). *Chapter 93: Tamper Detection Module (TDM)* *Section 93.5.3: Diary Base Address (TDM_DBA)*: – changed reset value to 0xFFFF_FFFF and added a footnote. *Chapter 94: ASC Modeling Unit, Version 2 (AMU2)* *Section 94.4.2: Single calculation using SDMA for data load, CPU for post-processing*: “5X16-byte 4-beat bursts” updated to “5X8-byte 8-beat bursts”. Added Footnote(a) in *Section 94.1.1: Overview*. *Chapter 96: Real Time Clock / Autonomous Periodic Interrupt (RTC/API)* *Section 96.4.2: RTC Control Register (RTCC)*: Added bitfield TRIG_EN. *Figure 2507: RTC/API block diagram*: Added rtc_clk. *Table 2402: RTC/API register map*: Removed Base Address. *Chapter 97: Emulation and Debug Device Introduction* *Section 97.2: Feature summary*: Updated on-chip SRAM size from 4 x 256 KB to 4 x 512 KB in the sentence “Implemented as 4 × 512 KB partitions where...”.

Table 2499. Document revision history (continued)

Date	Revision	Changes
03-May-2017	3	*Chapter 2: Introduction* Created two sub chapters, for SPC58xEx and for SPC58xGx. *Chapter 3: Embedded memories* *Figure 7: Flash memory segmentation and read-while-write partitioning*: centered in the figure RWW Partition 4 and 5. *Section 3.3.2: Flash memory array*, update size of double word from 64 to 128-bits;added “For code Flash sectors” at the beginning of 2nd paragraph. *Section 3.3.2.1: Features*, updated ECC bullet *Chapter 5: Memory Map* *Table 8: Peripheral (PBRIDGE_0, PBRIDGE_1, PBRIDGE_2) memory map*: updated *Table 9: Flash memory map*: updated header cells. *Table 10: Emulation and debug map* added with information merged from *Table 9: Flash memory map* (Buddy Device and Overlay RAM) and *Table 10: Emulation and debug map* removed. *Table 11: RAM memory map*: Reviewed the standby RAM partitioning for PRAMC_2. (8K+120K+128K) *Table 13: UTest Flash memory map* updated. *Section 5.7: Security module memory*: Added this section *Chapter 6: Functional Safety* *Table 16: Module classification*: Replaced ‘EBI’ by ‘External memory interface *Section 6.4.1.7.1: External error indication*: Added paragraph ‘In case of functional reset, the application... during functional reset.” *Table 18: ECC RAM implementations*, ‘Ethernet’ Location: updated ‘FIFO’ by ‘FIFO RX’, ‘MIB’ by ‘FIFO TX’ and value in cell ‘React to uncorrectable error’ to‘Y’. *Table 17: FCCU failure inputs*: – put table in landscape orientation, – updated column “Failure description” for rows 63, 83, 88 to 91, 103 to 106, 114, 123 – removed rows115 to 122. – Copied content from fault 85 row and updated faults 111, 112, 113 and 114 row with the same content. *Chapter 7: Device configuration* Added *Section 7.5.2: Platform RAM Controler (PRAMC) configuration* Updated *Table 39: DMAMUX Map*

Table 2499. Document revision history (continued)

Date	Revision	Changes
03-May-2017	3 (cont.)	*Chapter 8: Reset and Boot* *Section 8.1: Introduction*: After reset, Core_2 is woken-up, Checker_Core_2 and HSM may be woken-up (see details in DCF Record chapter). *Figure 41: Reset Generation Module Reset Sequence* reworked. *Section 8.3.12.2: Boot CPU start-up*: replaced “BAF DCF” by “BAF configuration DCF” within this section and through the chapter. *Figure 42: Boot CPU start-up sequence*: – updated text box “Found Bypass Mode DCF?” by “Found BAF Configuration DCF?” – updated text box “Bypass Mode DCF” by “BAF_BYPASS selection” *Section 8.3.12.2.3: SW boot record search (BAF search)*: deleted in 2nd bullet “not” in sentence “the BAF configuration DCF does bypass the BAF code and no valid ...” *Section 8.1.1: TEST flash memory block*: Replaced following list bullet: “Selection between HPREG or SMPS in internal regulator mode” with this one: “Power management configuration, for the devices supporting different regulation schemes” *Chapter 9: Device Configuration Format (DCF) Records* *Section 9.1: Introduction*: Added Note “Because the Flash memory programming resolution is 128 bits then 2 DCF records must be written within 1 Flash programming operation.”. *Section 9.3: DCF records*: Updated this section. *Table 130: DCF client list*: – Removed Life Cycle (LCSTAT) to DCF client description and this row to Reserved. – Added “Write Once” and “Refer to Boot Assist Flash (BAF) chapter” to all Security Watchdog in BAF ‘Soft’ Clients subsection. – Added “Support for 7 LBIST partitions (NLBIST); 102 Memory Cuts (NMCUT); 32-bit PRPG; 64-bit MISR” to STCU subsection. – separated the DCF address [16:2] into [16:10] and [9:2] – changed all BAF ‘Soft’ Client DCF address values from hexadecimal to binary – changed the value of UTEST miscellaneous to 0x00049E01 – Marked address 0_0000_1000_ 0101_01 (0x0215) as reserved *Section 9.4: DCF client table*: Completed the phrase - DCF order dependency in flash memory: this column details whether or not there are special conditions on the location of a particular DCF record in flash memory *Table 132: UTEST Miscellaneous DCF Client field descriptions*: updated the field descriptions for XOSC_FREQ_SEL and XOSC_LOAD_CAP_SEL *Figure 50: UTEST Miscellaneous DCF Client* and *Table 132: UTEST Miscellaneous DCF Client field descriptions*: added CORE_CLK_ONLINE_SELF_TEST bit field *Table 136: PNREG_ENB field descriptions*: – inverted the descriptions for the NREG_ENB bit – inverted the descriptions for the PREG_ENB bit

Table 2499. Document revision history (continued)

Date	Revision	Changes
03-May-2017	3 (cont.)	*Chapter 10: Power management* *Table 140: Voltage monitors configurability*: simplified the table *Table 138: Device power management controller external signals*: removed 3.3V from VDD_HV_IO_FLEX description *Section 10.1.2: Power management supply description*: changed the Warning to a Caution *Table 139: Voltage monitors description*: removed “low range” from the description of MVD094_C *Section 10.2.1: Low power mode (HALT/STOP/STANDBY)*: Added a note with 2 cross-references. *Section 10.5.3.1: LVD and HVD implementation*: Updated the fifth bullet. *Chapter 11: Smart Stand-by Wake-up Unit (SSWU)* Extensive updates throughout the chapter. *Chapter 13: Calibration and Debug* *Section 13.4: Debug over CAN*: Replaced M_CAN_0 with CAN_SUB_0_M_CAN_1 and M_CAN_1 with CAN_SUB_0_M_CAN_2 *Chapter 15: Core (z420n3) Description* *Figure 106: L1 Cache Control and Status Register 0 (L1CSR0)*: Updated bit 12 from “reserved” to “DCWA” *Section 15.8: MPU* Created *Figure 115: MPU Read Entry Instruction (mpure)*, *Figure 116: MPU Write Entry Instruction (mpuwe)*, *Figure 117: MPU Synchronize Instruction (mpusync)* *Figure 110: L1 Flush/Invalidate Register 0 (L1FINV0)*: Updated the figure title with a missing “0” *Figure 111: L1 Flush/Invalidate Register 1 (L1FINV1)*: Updated the figure title with the register’s full name *Chapter 17: System Integration Unit Lite2 (SIUL2)* *Table 273: SIUL2_MSCR_IOn field description*: Updated ILS bitfield description with the following note“(CMOS for LP pads)” related to TTL. *Table 262: SIUL2_MIDR1 field descriptions*, updated PARTNUM description. *Table 263: SIUL2_MIDR2 field description*, updated FAMILYNUM description. *Table 273: SIUL2_MSCR_IOn field description*, updated ILS description. *Chapter 18: Crossbar switch (XBAR)* *Figure 180: XBAR Control Register (XBAR_CRSn)* and *Table 284: XBAR_CRSn field descriptions*: Changed field HPEx to 8 bitfields HPE7–HPE0 (bits 8 to 15). *Chapter 19: Crossbar Integrity Checker (XBIC)* *Table 287: XBIC_EIR field descriptions*: added footnotes for fields ‘Target Slave Port’ and ‘Target Master ID’. *Chapter 24: Interrupt Controller (INTC)* *Section 24.5.2.6: INTC Software Set/Clear Interrupt Register n (INTC_SSCIRn)* : Removed text “Unlike the SWT bit in the PSRs,” from the first paragraph.

Table 2499. Document revision history (continued)

Date	Revision	Changes
03-May-2017	3 (cont.)	*Chapter 26: DMA channel multiplexer (DMACHMUX)* updated *Figure 267: DMACHMUX triggered channels* *Chapter 27: Clocking* *Figure 270: Clock generation(1)(2) (part 1 of 2)*: updated. *Figure 273: Clock distribution*: updated. *Table 382: MC_CGM relationship to clocks*: Updated 'Output clock' column of CGM_SC_DC2 into 'PBRIDGE_CLK (All peripheral bridges and peripherals not connected to any other system clock divider)' *Section 27.5: Clock sources*: updated "Slow internal RC oscillator ... (SIRC)" by "Low power RC oscillator ... (LPRC)". *Table 382: MC_CGM relationship to clocks*: – CGM_SC_DC1: added SWT and STM in Output Clock, – CGM_SC_DC3: removed SWT, STM and INTC. *Table 383: Maximum system level clock frequencies*: – HPBM_CLK: updated "SYS_CLK / 2" by 90, – PBRIDGE_CLK: updated "SYS_CLK / 4" by 45. – FBRIDGE_CLK: updated "SYS_CLK" by 180, – HSM_CLK: updated "SYS_CLK / 2 or SYS_CLK / 4" by 90. *Section 27.4: Default clock configuration*: updated sentence "Under the sofware responsability...set in the Magic Carpet documentation..." by "Under the software responsibility, the value chosen for the divisors CGM_SC_DC0, CGM_SC_DC1 and CGM_SC_DC2 shall respect the ratios shown in ...." *Section 27.5.2: External oscillator (XOSC)*: updated clock range to "4 MHz - 40 MHz". *Section 27.5.2.1: XOSC Start-up*: – rewrote 1st sentence and added cross-reference to DCF Records chapter – updated UTEST Miscellaneous[x] bit numbers – updated sentence "The XOSC has the ability to be started with either... " by "The XOSC has the ability to be started with 4 MHz - 40 MHz crystal. The default value of this field is for an 35 MHz - 40 MHz:" – updated "XOSC EN_40MHz" by "XOSC_FREQ_SEL" – updated default value for "UTEST Miscellaneous[19:21] = 111b (XOSC_FREQ_SEL)" *Section 27.6.8: System Clock/GTM output clock*: updated "The two system clocks..." by "The system clock...", removed "CGM_AC7_SC" and "SYSCLK1". *Chapter 28: Dual PLL Digital Interface (PLLDIG)* *Section 28.6.2: Clock configuration*: Added text "refer to the device datasheet"; Added 2 notes. *Table 402: PLL1FD field descriptions*: Updated DTHDIS field description; Added note to FRCDIV field description. *Section 28.6.3: Frequency modulation*: – Changed feedback divider f_{ref} by $f_{pll1ref}$ – Move text "PLLFM[MODPRD] and PLLFM[INCSTP] are subject to the following restriction:" to next equation.

Table 2499. Document revision history (continued)

Date	Revision	Changes
03-May-2017	3 (cont.)	*Chapter 30: Clock Generation Module (MC_CGM)* *Table 411: MC_CGM memory map*: Swapped address offsets of CGM_PCS_DIVS1 and CGM_PCS_DIVE1 registers. *Figure 311: PCS Divider End Register 1 (CGM_PCS_DIVE1)*: Changed offset address to 0x0108 (was 0x010C). *Figure 312: PCS Divider Start Register 1 (CGM_PCS_DIVS1)*: Changed offset address to 0x010C (was 0x0108). *Figure 327: System Clock Divider 3 Configuration Register (CGM_SC_DC3)*: Changed title from Table to Figure. *Figure 329: Auxiliary Clock 0 Select Control Register (CGM_AC0_SC)*, *Table 449: CGM_AC1_SC field descriptions*, *Table 454: CGM_AC3_SC field descriptions*, *Table 456: CGM_AC4_SC field descriptions*, *Table 461: CGM_AC6_SC field descriptions*, *Table 464: CGM_AC7_SC field descriptions*, *Table 467: CGM_AC8_SC field descriptions*, *Table 470: CGM_AC9_SC field descriptions*, *Table 473: CGM_AC11_SC field descriptions* and *Table 476: CGM_AC12_SC field descriptions* : Added Note “Status of respective clock source must be ensured before selection.” to field SELCTL. *Table 411: MC_CGM memory map*: Swapped address offsets of CGM_PCS_DIVS2 and CGM_PCS_DIVE2 registers. Swapped address offsets of CGM_PCS_DIVS4 and CGM_PCS_DIVE4 registers. *Figure 314: PCS Divider End Register 2 (CGM_PCS_DIVE2)*: Changed offset address to 0x0114 (was 0x0118). *Figure 315: PCS Divider Start Register 2 (CGM_PCS_DIVS2)*: Changed offset address to 0x0118 (was 0x0114). *Figure 317: PCS Divider End Register 4 (CGM_PCS_DIVE4)*: Changed offset address to 0x012C (was 0x0130). *Figure 318: PCS Divider Start Register 4 (CGM_PCS_DIVS4)*: Changed offset address to 0x0130(was 0x012C). *Figure 297: MC_CGM block diagram*: updated “SIRC” by “LPRC”. *Section 30.4.1: System clock generation*: figure “MC_CGM system clock generation overview” removed, added cross-ref to “Clock generation” figure from Clocking chapter. *Section 30.4.2: Auxiliary clock generation*: figures “MC_CGM auxiliary clock *n* *generation* overview” removed (and their cross-refs), added cross-ref to “Clock generation” figure from Clocking chapter. *Chapter 33: LPRC Digital Interface* *Section 33.2: Low Power RC Oscillator*: Renamed MISC_CTRL_REG[SIRC_ENB] into MISC_CTRL_REG[RCOSC_1M_ENB]; Changed RCOSC_1M_ENB to RCOSC1M_ENB *Chapter 36: Flash memory controller* Updated *Figure 384: Platform-centric simple block diagram with Flash memory controllers*: – removed “+” signs, – replaced “p” with “P” and “s” with “S”, – aligned port numbers with Device configuration chapter’s Crossbar switch section.

Table 2499. Document revision history (continued)

Date	Revision	Changes
03-May-2017	3 (cont.)	*Chapter 37: Embedded Flash Memory* *Section 37.3.2.21: HSM Alternate selector registers*: Reworked introduction sentence. Field N8KH has been replaced with N128KL in *Table 522: MCRE field descriptions* Corriged typo: “last” instead of “fast” in *Section 37.3.2.15: Program Address register (PAR)* Added*Table 542: Flash Memory EEPROM data partition 3 addresses mapping* Corrected typo in *Chapter 37.3.2.20: Test Mode Disable Password Check register (TMD)*: UTEST location for password was not correct and TMD Seal was not updated wit *Table 559: TMD field descriptions*: Added Note “PWD bitfield is byte-reversed...” *Chapter 40: Analog-to-Digital Converters (ADC) Configuration* *Section 40.2.2.11: SAR ADC register definitions* updated *Section 40.2.2.5: SAR ADC triggering*: Added (SAR_ADC parameter CTSEN = 2) for instance SAR_ADC_10bit_STDBY. *Section 40.2.2.1: SAR ADC configuration overview*: Removed sentence “This file is used as a text inset in ADC Configuration chapter”. *Figure 478: Analog input — discharge switch to VSS_HV_ADV* added the weak pullup/pulldown *Section 40.2.2.9.4: Analog input pin switch to $V_{SS_HV_ADV}$*: Updated this section. *Chapter 41: Sigma-Delta Analog-to-Digital Converter (SDADC) Digital Interface* *Figure 479: SDADC block diagram*: updated VAREF2 by VDD_HV_ADR_D, VAGND2 by VSS_HV_ADR_D. *Table 607: Signal properties*: – appended in cell ‘VREFP’ ‘= VDD_HV_ADR_D’, in cell ‘VREFN’ ‘= VSS_HV_ADR_D’ – replaced cells named ‘VDD_HV_ADR_D’ by ‘VDD_LV’, ‘VSS_HV_ADR_D’ by ‘VSS_LV’ – Added table note “See Section: Sigma-Delta ADC integration diagram.” to column name. *Table 610: SDADC_MCR field descriptions*, ‘EN’ bitfield: removed “Channel and conversion configuration values” sentence, added “Refer to *Section 41.8: .Initialization information*...”. *Chapter 42: Successive Approximation Register Analog-to-Digital Converter (SARADC) Digital Interface* *Table 622: CTU modes*: added table footnote to CTSEN “Refer to Section 1.2.2.5: SAR ADC triggering of Chapter 1: Analog-to-Digital Converters (ADC) Configuration for CTSEN value applicable on each SAR_ADC instance.” *Section 42.4.4.1: Conversion timings*: – “PRECHG must be greater than...” replaced with “PRECHG is the configured precharging phase duration” – Removed “In case the value of...inside SARADC” *Table 625: MCR field descriptions*, added note in the description of CTUEN

Table 2499. Document revision history (continued)

Date	Revision	Changes
03-May-2017	3 (cont.)	*Chapter 49: GTM Development Interface (GTMDI)* *Table 734: GTMDI_SPEB_WPC1 field descriptions*: Changed description of value '000' to "SPE1" (was "SPE0"), '001' to Reserved (was "SPE1") and '111' to "SPE0" (was Reserved). *Figure 557: GTMDI block diagram*: Added arrow "Restore" between GTM IP and Halt Control blocks. *Chapter 50: Periodic Interrupt Timer (PIT)* Extensive updates throughout the chapter. *Chapter 52: CAN Subsystem* Extensive updates throughout the chapter. *Chapter 53: Serial Interprocessor Interface (SIPI)* *Section 53.15.1.7: SIPI Channel Data Register (SIPI_CDRn)*: – Added CDR0, CDR1 and CRDR3 offsets in *Figure 900: SIPI Channel Data Register (SIPI_CDRn)* – Added *Section 53.15.1.7.1: SIPI Channel Data Register (SIPI_CDR2i)* included *Figure 901: SIPI Channel Data Register (SIPI_CDR2i)* and *Table 978: SIPI_CDR2i field descriptions* *Figure 895: Channel Status Register (CSRn)*: Changed TID field at bit 25:27 to read-only access (was w1c). *Figure 888: SIPI data stream model*: Replaced a "Bus Master DMA" box with a "Bus AMBA lite" one. *Figure 890: SIPI data stream model - Target*: Replaced a "Bus Master DMA" box with a "Bus AMBA lite" one. *Chapter 56: Ethernet* Extensive updates throughout the chapter. *Chapter 59: SENT Receiver (SRX)* *Figure 1419: DMA Fast Message Time Stamp Read Register (DMA_FMSG_TS)* and *Figure 1428: Channel n Fast Message Time Stamp Read Register (CHn_FMSG_TS)*: Changed register access from R/W to Read-only. *Chapter 60: Peripheral Sensor Interface (PSI5)* *Figure 1461: Overwrite Status Register (CH0_OWSR)*: Changed register access to User read/write (was User w1c). *Figure 1463: Set New Data Status Register (PSI5_CH0_SNDSR)*, *Figure 1464: Set Overwrite Status Register (PSI5_CH0_SOWSR)*, *Figure 1465: Set Error Status Register (PSI5_CH0_SEISR)* and *Figure 1466: Set SMC Error Status Register (PSI5_CH0_SSESR)*: Changed register access to User read/write (was User w1s). *Chapter 62: Deserial Serial Peripheral Interface (DSPI)* *Figure 1581: Hardware Configuration Register (DSPI_HCR)* and *Table 1600: DSPI_HCR field descriptions*: Updated register CMDFR bitfield lenght to 4 bits (was 5 bits); moved register bitfields MSTR_MODE, SLV_MODE, TSB_EN and ITSB_EN; updated reset value; updated description of CMDFR

Table 2499. Document revision history (continued)

Date	Revision	Changes
03-May-2017	3 (cont.)	*Chapter 63: LINFlexD* *Section 63.4.2.9: LIN Time-Out Control Register (LINTOCR)*: Changed the reset value of bitfield HTO from 0x1C to 0x2C. *Figure 1690: LIN Time-Out Control Register (LINTOCR)*: Deleted footnote. *Chapter 64: Reset Generation Module (MC_RGM)* *Section 64.1.3: Reset sources*: – Removed “and the memory content must be considered to be unknown except in the software ‘destructive’ reset and external power-on reset cases, which do preserve the system memory content if the chip is not configured to execute a self-test on power-on, ‘destructive’, and external resets.” – Removed “as well as the system memory content is preserved, except on an external reset in the case where the chip is configured to execute a self-test on power-on, ‘destructive’, and external resets”. *Table 1695: RGM_DES field descriptions*: Added a table note to bit 30’s description. description. *Figure 1741: ‘Functional’ Event Reset Disable Register (RGM_FERD)*: Set bit 25, D_FCCU_SOFT, as read only (used to be R/W) *Figure 1742: ‘Functional’ Event Alternate Request Register (RGM_FEAR)* : Set bit 25, AR_FCCU_SOFT, as read only (used to be R/W) and other minor editorial changes *Chapter 65: Boot Assist Flash (BAF)* *Table 1726: BAF image header*: Added *Table 1729: BAF DCF client list*: Set addresses 0x0212, 0x213 and 0x215 to Reserved *Section 65.3.8: Optionally perform a serial boot* Replaced “If the Life Cycle is greater” with “If the Life Cycle is less”. *Chapter 67: Power management controller digital interface (PMC_Dig)* Extensive updates throughout the chapter. *Chapter 68: Power Control Unit (MC_PCU)* *Figure 1838: Power Domain #2 Configuration Register (PCU_PCONF2)*: Updated the Standby RAM value from 20KB to 120KB in the paragraph under the figure. *Figure 1839: Power Domain #3 Configuration Register (PCU_PCONF3)*:Updated the Standby RAM value from 224KB to 128KB in the paragraph under the figure. *Chapter 69: Mode Entry Module (MC_ME)* Extensive updates throughout the chapter. *Chapter 76: Sequence Processing Unit (SPU)* *Figure 2011: Trace group configuration register format (DTMTGC, PTMTGC, OTMTGC, WTMTGC)* – Added (DTMTGC, PTMTGC, OTMTGC, WTMTGC) to figure title *Table 1957: DTMTGC, PTMTGC, OTMTGC, WTMTGC field descriptions* – Updated table title with “DTMTGC, PTMTGC, OTMTGC, WTMTGC”

Table 2499. Document revision history (continued)

Date	Revision	Changes
03-May-2017	3 (cont.)	*Chapter 87: Memory Error Management Unit (MEMU)* *Section 87.1: Introduction* Updated info related to correctable and uncorrectable errors. *Section 87.6.2.1: Control register (CTRL)* Added new field “FLASH_ERR_SELECT[1:0] *Table 2176: CTRL field descriptions*: Added description of new field “FLASH_ERR_SELECT” *Section 87.6.2.14: Flash memory correctable error reporting table status register (FLASH_CERR_STSn)*: ; Added Note “MEMU captures flash ecc error address on 64bit boundary”. *Table 2189: FLASH_CERR_STSn register field descriptions*: in [BAD_BIT] description, “RAM” updated to read “FLASH”. *Chapter 88: Indirect Memory Access (IMA)* *Figure 2311: IMA RAM Write Data register n (IMA_WRITE_DATA_n)* and *Table 2204: IMA_WRITE_DATA_n field descriptions*: Changed register fields WRITE_n to WRITE. *Figure 2312: IMA RAM Read Data register n (IMA_READ_DATA_n)* and *Table 2205: IMA_READ_DATA_n field descriptions*: Changed register fields READ_n (n=4 to 0) to READ. *Table 2201: IMA_SLCT field descriptions*: Removed “word” from ROW_SLCT field description. *Section 88.3.3.7: IMA RAM Write Data register n (IMA_WRITE_DATA_n)*: Changed text “IOs for the RAM selected.” by text “bits for the RAM selected.” in long description paragraph. *Section 88.3.3.8: IMA RAM Read Data register n (IMA_READ_DATA_n)*: Changed text “IOs for the RAM selected” by text “bits for the RAM selected.” in long description paragraph. *Section 88.3: Accessing memory via the IMA module*: removed bullet ‘ECC must be disabled...’, added the 2 bullets ‘When a bus master performs an IMA access...’ and ‘Before any IMA access...’. *Chapter 89: Fault Collection and Control Unit (FCCU)* *Section 89.5.7: FAULT interface*: removed “in order to generate fake faults directly in the FAULT root” in sentence “To support a fault injection mechanism ... optional signal is available.” *Section 89.5.6: NMI/WKPU interface*, removed bullet about of Wake-up indication *Chapter 90: Self-Test Control Unit (STCU2)* *Figure 2400: STCU2 Offline LBIST MISR Expected Low n Register (STCU_LB_MISRELn)*: Changed MISRE[31:0] field to MISRE31-MISRE0 fields. *Figure 2401: STCU2 LBIST MISR Expected High n Register (STCU_LB_MISREHn)*: Changed MISRE[63:32] field to MISRE63-MISRE32 fields. *Chapter 92: Password and Device Security Module (PASS)* *Figure 2424: Clock Jitter Enable (CJE)*: Moved bitfield CJE from bit 0 to bit 31.

Table 2499. Document revision history (continued)

Date	Revision	Changes
03-May-2017	3 (cont.)	*Chapter 93: Tamper Detection Module (TDM)* *Table 2341: TDM memory map*: Added TDM_ prefix to STO_KEY*n* registers. [All projects] From *Section 93.4.1: Diary Base Address (DBA) DCF client* to *Section 93.4.9: Tamper Region Assignment DCF client (TDRn_LOCK2)*: changed the Warning to a Caution *Table 2334: OTPEN1 field descriptions*: Added a reference to the Embedded memories chapter. *Chapter 95: Wakeup unit (WKPU)* *Table 2392: WKPU Memory Map*: Removed reserved space 0x000C–0x03FF). *Section 95.4: Functional Description*: Removed Note “An edge enable for a given input must be asserted before the NMI/external interrupt edge in order for a system wakeup or interrupt to be generated.”. *Section 95.3.2.9: Wakeup/Interrupt Pullup Enable Register (WIPUER)*: Added note “This register shall be left at '0' default value, as:...”. *Chapter 96: Real Time Clock / Autonomous Periodic Interrupt (RTC/API)* *Section 96.2: Features*, *Figure 2507: RTC/API block diagram* and *Figure 2508: Clock gating for RTC clocks*: Removed XOSC (4-40 MHz) Removed register RTC_ANLCMP_CNT and its relative description. *Figure 2507: RTC/API block diagram*: – Changed CNTEN from 10 to 32 – Changed 32 bit counter output from 12 to 32 – Removed RTC rollover wakeup block. *Table 2404: RTCC register field descriptions*: Removed wakeup related text. *Section 96.5: RTC functional description* – Rephrased the sentence starting by “When the counter value...” to change from 12 to 32 bits and to remove the example. – Removed rollover wakeup occurrences as it is not supported *Section 96.6: API functional description*: Replaced 10-bit by 32-bit in the second sentence. *Table 2408: RTC_RTCVAL register field descriptions*: added Note “Minimum supported value of RTCVAL is 4. This is due to synchronization issues.” *Chapter 97: Emulation and Debug Device Introduction* *Table 2411: BD memory map*: – Changed Extended overlay RAM allocated size to 2048KB at space address 0x0C00_0000–0x0C1F_FFFF. – Updated address 0x0C80_C000 by “Reserved” (was “DWPU”), updated address 0x0C81_0000 by “BD XBAR” (was “Reserved”). *Figure 2515: Emulation and debug device architecture*: Updated the figure to suppress the lock-step core associated to Core_1 *Chapter 101: Buddy Device—System Integration Unit Lite (BD_SIUL2)* Added registers *Section 101.4.1.5: BD_SIUL2 Multiplexed Signal Configuration Registers (BD_SIUL2_MSCR1, BD_SIUL2_MSCR2)*

Table 2499. Document revision history (continued)

Date	Revision	Changes
06-Aug-2018	4	Following are the changes in this version of the Reference manual: *Chapter 2: Introduction*: *Section 2.1.9: Software debug and calibration*: added 32 KB overlay RAM. Canceled referenced to ED (Emulation Device) not present on Chorus devices. *Chapter 5: Memory Map* *Table 7: Overview memory map*: Updated HSM. Updated *Table 14: HSM memory map*. *Table 13: UTest Flash memory map*: Address 0x0000036F replaced by 0x0000035F and 0x00000370 replaced by 0x00000360. *Chapter 6: Functional safety* *Table 15: PMHF module categories*: Updated definition of Non-safety modules *Section 6.4.4.2: All-x words and ECC*: – Added sentence starting with “Memories that include addresses in the ECC...” – Reworked the paragraph starting with “In case of not valid codeword...” *Section 6.4.4.3: ECC failure handling*, *Section 6.4.4.4: Memory Error Management Unit (MEMU)* and *Section 6.4.4.4.1: Interface to ECC units*: Added “with same (optional) syndrome” *Table 16: Module classification*: – Moved Concentrator DMA from NoSaMos to ViMos. – Added STANDBY SAR ADC in NoSaMos Classification. *Section 6.4.1.5.1: BIST during boot (off-line)*: Updated Logic BIST bullet *Table 17: FCCU failure inputs*: – Set “Not testable” as Error reaction path for Failures LVD_ERROR and HVD_ERROR – Replaced Failure descriptions for FCCU channel 93, 99, 104 - 109 – Replaced PRAM with “FLASH controller internal RAM” in FCU channel 115 – Filled Error reaction path, Default reaction, Reference Manual reference and Recovery mechanism *Section 6.4.4.1: ECC for storage*: Changed introduction. *Chapter 8: Reset and Boot* *Section 8.3.3*, *Section 8.3.4*, *Section 8.3.5*, *Section 8.3.8* and *Section 8.3.9*: Deleted all indications of duration of each phase (“This phase is completed in...”) *Chapter 10: Power management* *Table 137: Device power management controller external signals*. Added VDD_HV_OSC and VSS_HV_OSC *Chapter 11: Smart Stand-by Wake-up Unit (SSWU)* *Section 11.3: Architecture*: Added the paragraph starting with “In STANDBY mode, only RTC is powered on....”

Table 2499. Document revision history (continued)

Date	Revision	Changes
06-Aug-2018	4 (cont'd)	*Chapter 17: System Integration Unit Lite2 (SIUL2)* *Figure 166: SIUL2 MCU ID Register #1 (SIUL2_MIDR1)*, in the footnote removed the default value of PKG bitfiled. *Table 269: SIUL2_MIDR1 field descriptions*. PARTNUM for Chorus6M changed from 0x5848 to 0x5846 *Chapter 21: System Memory Protection Unit (SMPU)* *Figure 197: Control/Error Status Register 1 (CESR1)*, changed reset value of NRGD field to 0110 *Chapter 27: Clocking* *Section 27.1: Introduction*: changed divider value from 512 to 1024 *Section 27.5.1.4: PLL register write protection*: added cross ref to Device configuration chapter *Table 392: PLL register write protection*: deleted "Protection" column *Section 27.6.2: PSI5 clock dividers*: deleted "three" and replaced by "two" PSi5 modules *Section 27.6.6.2: M_CAN subsystem 1*: "/M_TTCAN" deleted from section title *Section 27.7.1.3: CMU register write protection*: – Added cross ref to Device configuration chapter – Deleted "Protection" column *Table 388: MC_CGM relationship to clocks*, Deleted "Bridge clock" from Output clock column of CGM_SC_DC1 *Chapter 28: Dual PLL Digital Interface (PLLDIG)* *Table 397: PLL0DV field descriptions* and *Table 400: PLL1DV field descriptions*: Updated division factors of RFDPHI. *Section 28.6.3: Frequency modulation*, replaced PLLnCR[MFD] with PLL1DV[MFD]. *Section 28.7: Initialization information*: added footnote linked to "procedure" *Table 395: PLL0CR field descriptions*: EXPEDIE bitfield description enhanced Changed "MC_ME_<mode>" by "<ME_mode>, "MC_ME_" or "ME" by "<ME>" and "MC_CGM_ACn" by "<CGM_ACn>" throughout all this chapter. *Section 28.4: Modes of operation*: – Changed "the operating modes" by "the operating mode" – Changed "The various modes of operation for the PLL are" by "The mode of operation for the PLL is defined below" *Chapter 29: Clock Monitor Unit (CMU)* *Table 411: CMU_CSR field descriptions* – SFM: added sentences "MDR[MD] ... CSR[SFM]=1" and "Software ... bit" – RCDIV: added sentence "Ensure ... [RCDIV]" – CME: added sentence "CMU_HFREF ... (CSR[CME=1])" *Section 29.5.2: CLKMN0_RMT supervisor*: deleted Note "f_{CLKMN0_RMT} ··· *monitoring"*

Table 2499. Document revision history (continued)

Date	Revision	Changes
06-Aug-2018	4 (cont'd)	*Chapter 31: OSC digital interface (XOSC)* *Figure 382: XOSC Control register (XOSC_CTL)*: Changed I_OSC field access type to R/w1c (was RW). *Chapter 37: Embedded Flash Memory* *Table 522: UTEST block memory map*: For Customer OTP data, the size changed from 4096 bytes to 12288 bytes. *Figure 409: Flash memory module structure*: Updated to depict RWR0 and RWR1 Flash arrays *Section 37.3.2.3: Extended Module Configuration Register (MCRE)*: Bitfield "HT" replaced by Reserved bitfield *Section 37.4.4.1: Read While Read*: Removed lines "RWW P6" and "RWW P7" Updated *Table 518: Flash memory partition 5 memory map* *Section 37.3.2.24: Parallel Address register (ADR2)*: Added the sentence "For the address mapping, refer to *Section 37.3.2.16: Address register (ADR)* description." Added *Section 37.4.4.2.1: Multi-bits error management (more than 3 bits error >=4)* *Section 37.3.2.1: Module Configuration Register (MCR)*: updated description of bitfield EEE *Chapter 40: Analog-to-Digital Converters (ADC) Configuration* *Section 40.2.2.1.2: 12-bit SAR ADC B0*: Removed references to SUPERVISOR *Table 588: ADC implementation*: updated SAR_ADC_12bit_2 from '13' to '15' *Section 40.2.2.1.1: 12-bit Fast SAR ADC* Specified 124-127 as Test channels *Section 40.2.2: Successive Approximation Register Analog-to-Digital Converter (SAR ADC)*; Removed SUPERVISOR referred to 12 bit SAR ADC. *Chapter 41: Sigma-Delta Analog-to-Digital Converter (SDADC) Digital Interface* Throughout the chapter, renamed all occurrences of – AN[x] to AIN[x] – EMDATA to BSEXT – EMCLK to CLKEXT *Figure 489: Converted Data Register (SDADC_CDR)*: Changed register access type to Read (was RW). *Chapter 42: Successive Approximation Register Analog-to-Digital Converter (SARADC)* *Table 616: SARADC digital interface register map*: Removed the "Register classification" column. *Table 617: MCR field descriptions*: removed note in CTUEN bitfield Replaced SDADC by SARADC in FRZ bitfield description (was a typo) *Section 42.4.5.1: CTU trigger mode*: Replaced JABORT by JABORTCHAIN (was a typo) *Section 42.4.6: Test channel connection with internal analog channel*: Minor changes *Section 42.5.1.16: Internal Channel Data Register n (ICDRn)*: Removed reference to REFSEL bitfield. *Section 42.5.1.41: External Channel Data Register n (ECDRn)*: Removed reference to REFSEL bitfield

Table 2499. Document revision history (continued)

Date	Revision	Changes
06-Aug-2018	4 (cont'd)	*Chapter 46: Software Watchdog Timer (SWT)* *Section 46.1.1: Overview*: Added "Note: The SWT runs over all device mode except the STANDBY mode where it is not powered." *Chapter 50: Periodic Interrupt Timer (PIT)* *Section 50.2.1.6: PIT RTI Timer Control Register (RTI_TCTRL)*: Removed bitfield CHN *Chapter 53: Serial Interprocessor Interface (SIPI)* *Section 53.11.3: Maximum count reached error*: updated blank cross reference to SIPI_ACR. *Chapter 57: FlexRay communication controller (FlexRay)* *Table 1300: FlexRay memory map*: Added Strobe Port Control Register (FR_STBPCR) at 0x000A offset. *Section 57.5.2.7: Strobe Port Control Register (FR_STBPCR)*: Added this register. *Figure 1324: Stop Watch Count High Register (FR_STPWHR)* and *Figure 1325: Stop Watch Count Low Register (FR_STPWLR)*: Changed register and STPW field accesses to Read-only (was Read-write). *Chapter 59: SENT Receiver (SRX)* *Table 1464: CHNL_EN field descriptions*, *Table 1466: FMSG_RDY field descriptions*, *Table 1467: SMSG_RDY field descriptions*, *Table 1470: FDMA_CTRL field descriptions*, *Table 1471: SDMA_CTRL field descriptions* and *Table 1472: FRDY_IE field descriptions*: Changed <field>[15:0] to <field>[(CH-1):0] Changed bit position 16:31 to (32-CH):31 *Table 1473: SRDY_IE field descriptions*: Changed bit position 16:31 to (32-CH):31. *Chapter 60: Peripheral Sensor Interface (PSI5)* *Table 1494: PSI5 memory map*: Removed the "Register classification" column. *Chapter 61: Peripheral Sensor Interface-Support module (PSI5-S)* *Figure 1544: PSI5-S global control register (PS_GLCR)*: Changed field accesses of bit 21:23 to write-only and always read as 0. *Figure 1557: PSI5-S channel 0 message configuration register A (PS_MSGA_CH0)*: Changed field access of bit 28 to write-only and always read as 0. *Figure 1560: PSI5-S channel message configuration register A (PS_MSGA_CHn)*: Changed field access of bit 28 to write-only and always read as 0. *Figure 1563: PSI5-S channel watchdog configuration register (PS_WD_CFGR_CHn)*: Changed field access of bit 4 to read-write. *Table 1546: PSI5-S memory map*: Removed the "Register classification" column. *Table 1582: PS_WD_CFGR_CHn field descriptions*: Removed text "This bit is one shot. it is always read as 0." at bit 4. *Figure 1566: PSI5-S ECU to sensor control register (PS_E2SCR_CHn)*: Changed field access of bit 18 to write-only and always read as 0.

Table 2499. Document revision history (continued)

Date	Revision	Changes
06-Aug-2018	4 (cont'd)	*Figure 1569: PSI5-S channel1 ECU to sensor downstream data shift register low (PS_DDSR_L_CHn)* and *Table 1588: PS_DDSR_L_CHn field descriptions*: Changed field names to DDSR_L1 for bits 0:7 and DDSR_L0 for bits 8:31. *Chapter 63: LINFlexD* *Figure 1679: Master node: DMA Tx FSM (concept scheme)* and *Figure 1683: Slave node: DMA Tx FSM (concept scheme)*: Added note "DTF Flag gets auto cleared by Hardware if Software does not clear it within 3 ipg_clk cycles." *Table 1661: LINFlexD memory map*: Removed the "Register classification" column. *Figure 1700: LIN Time-Out Control Register (LINTOCR)*: Updated HTO reset value by adding a note (1) on bits 26:27. *Figure 1704: LIN Control Register 2 (LINCR2)*: Updated IOPE reset value note (3). *Chapter 65: Boot Assist Flash (BAF)* *Section 65.3.8: Optionally perform a serial boot*: updated text. *Figure 1781: Production Disable callback function flow chart*: Replaced "PIT" by "Periodic Timer". *Chapter 66: System Status and Configuration Module (SSCM)* Updated *Table 1745: Life Cycle slots*, *Table 1746: Life Cycle Slots in Memory* and *Table 1747: Resulting Life Cycle* Removed "Glossary" section. *Chapter 67: Power management controller digital interface (PMC_Dig)* *Section 67.3: IPS bus interface*: Updated the section. *Section 67.5: Registers description*: Replaced "Access: User read/write" with "Access: User read/write, Supervisor read/write, Test read/write" in all the relevant register figures. *Figure 1804: Reset Event Enable Register (REE_LV0)*: Updated the access of all the fields as read/write. *Figure 1805: Reset Event Select Register (RES_LV0)*: Updated the access of all the fields as read/write. *Figure 1806: Interrupt Enable Register (IE_LV0)*: Updated the access of all the fields as read/write. *Figure 1825: Global Interrupt Enable Register (IE_G)*: Updated the access of all the fields as read/write. *Table 1771: GR_S field descriptions*: Updated the description of fields "TEMP_2", "TEMP_1", and "TEMP_0". *Section 67.1: Introduction*: Updated the text "The PMU digital block generates a transfer error event ....." *Section 67.3: IPS bus interface*: Updated the bullet point "The PMU digital block generates a transfer error event ....." *Chapter 69: Mode Entry Module (MC_ME)* *Figure 1874: Peripheral Status register 7 (ME_PS7)*: Added fields S_ETHERNET_0 to bit 21 and S_ETHERNET_1 to bit 22.

Table 2499. Document revision history (continued)

Date	Revision	Changes
06-Aug-2018	4 (cont'd)	*Chapter 76: Sequence Processing Unit (SPU)* For each occurrence of "Calibration and Debug" in the chapter, added "(or Debug and Trace)". *Chapter 55: LVDS Fast Asynchronous Serial Transmission – Interprocessor Communications* Removed "Line Driver digital connections" section *Table 995: PLLCR field descriptions*: Updated description of FBDIV *Chapter 86: Cyclic Redundancy Check (CRC) Unit* Fixed *Equation 95*. *Chapter 87: Memory Error Management Unit (MEMU)* *Section 87.6.2.14: Flash memory correctable error reporting table status register (FLASH_CERR_STSn)* changed boundary from 64bit to 128 bit. *Figure 2278: MEMU reporting table uniqueness check flow chart*. Added arrow from [Error dropped] *Section 87.1: Introduction*: Replaced "Peripheral System RAM" by "Peripheral RAM" in the second bullet *Table 2179: SYS_RAM_UNCERR_ADDR field descriptions*, *Table 2184: PERIPH_RAM_UNCERR_ADDR field descriptions* and *Table 2189: FLASH_UNCERR_ADDR field descriptions*: Added the sentence "This register must be READ ONLY when its corresponding valid bit is not asserted." to ERR_ADD field description. *Chapter 89: Fault Collection and Control Unit (FCCU)* *Table 2212: FCCU memory map*: Removed the "Register classification" column. *Table 2215: FCCU_CFG field descriptions*: updated description of bitfield FCCU_SET_CLEAR. *Figure 2336: FCCU CTRL Key Register (FCCU_CTRLK)* and *Table 2214: FCCU_CTRLK field descriptions*: Changed CTRLK field name to CTRLK_VAL. *Figure 2341: FCCU RF Key register (FCCU_RFK)* and *Table 2341: FCCU RF Key register (FCCU_RFK)*: Changed RFK field name to RFK_VAL. *Figure 2355: FCCU XTMR Register (FCCU_XTMR)* and *Table 2239: FCCU_XTMR field descriptions*: Changed XTMR field name to XTMR_VAL. *Figure 2359: FCCU Delta T Register (FCCU_DELTA_T)* and *Table 2244: FCCU_DELTA_T field descriptions*: Changed XTMR field name to XTMR_VAL. *Table 2213: FCCU_CTRL field descriptions*: – Added field numbers – Updated description of DEBUG field – Updated description of OPR field *Section 89.6.2: FCCU CTRL Key Register (FCCU_CTRLK)*: updated Note.

Table 2499. Document revision history (continued)

Date	Revision	Changes
06-Aug-2018	4 (cont'd)	*Chapter 90: Self-Test Control Unit (STCU2)* *Table 2257: STCU_WDG field descriptions*: added table note "STCU2 is being clocked from PBRIDGE Clock. Please refer Clocking chapter for PBRIDGE clock frequency value." *Section 90.4.2.1: Offline reset generation*: added sentence "In case of unrecoverable fault during offline self-test execution, the STCU2 generates a request of destructive reset to the MC_RGM module." *Table 2251: STCU2 memory map*: Renamed table title (was register set). *Section 90.5.8: STCU2 CRC Expected Status Register (STCU_CRCE)*: Removed equation number. *Figure 2375: STCU2 Error FM Register (STCU_ERR_FM)*: Set reset value of bits 0:15 to all 0. Removed this "Register map" section. *Chapter 92: Password and Device Security Module (PASS)* *Table 2307: LCSTAT register field descriptions*: Updated description of LIFE bitfield with value 110 (ST production). *Figure 2431: Password Group n - Lock 0 Status register (LOCK0_PGn)*: Added ATSL field at bit 1 and set undefined reset with figure note. *Table 2312: LOCK0_PGn register field descriptions*: Added ATSL field and its description at bit 1. *Section 92.3.1.4: Challenge Input register (CINn)*; added the following sentence "Valid password criteria is that it should not be all Zeros or all Ones".: *Chapter 93: Tamper Detection Module (TDM)* *Table 2340: TDM_STO_KEYn field descriptions*, added a second note. *Figure 2454: Diary implementation*; changed Diary_TDR0 size from 256x64 to 256x128 bits. *Section 93.6.3: Diary*, replaced "is 8 bytes" with "depends on the Flash sector, 8 bytes for Data sector and 16 bytes for the remaining sector." *Section 93.6.4: Specifying the diary base address*, replaced "is 8 bytes" with "depends on the Flash sector, 8 bytes for Data sector and 16 bytes for the remaining sector." Deleted "Therefore the diary can hold a maximum of 256 × 8 byte entries." *Section 93.6.5.1: Diary management*, replaced "64-bit double word "with "record". *Chapter 94: ASC Modeling Unit version 2 (AMU2)* *Figure 2469: AMU V Pointer Channel n Register (AMU2_VPTRCHn)*: Changed register and fields accesses to Read/Write (was Read). *Figure 2458: AMU P8 Channel n Register (AMU2_P8CHn)*: Changed register and fields accesses to Read (was Read/Write).

Table 2499. Document revision history (continued)

Date	Revision	Changes
30-Jul-2020	5	Following changes are done throughout the document: Updated the subtitle, in the cover page. *Chapter 3: Embedded memories* *Section 3.2.1: System SRAM*: Updated the text from, “The SRAM can be configured....PFCR1.....controller.” to “The SRAM can be configured... PRCR1.....controller.” *Chapter 5: Memory Map* *Table 9: Flash memory map*: – Added footnote to PFLASH Controller Assignment (RWR Partition ID) in header column. – Added footnote to reserved space with assignment to PFLASH 0 at start address 0x015C0000. *Table 13: UTest Flash memory map*: Added a footnote, “Factory could program.....find it” to the “UTest DCF records” for Start offset of 0x00000370. *Chapter 6: Functional safety* *Table 16: Module classification*: – Moved Concentrator HSM to ViMos classification (was NoSaMos) – NoSaMos: Changed SEMA4 by SEMA42. – Below Modules have been moved Vimos to NoSaMoS: – Performance Core (Core_1), – D-Cache Performance Core (Core_1), – I-Cache Performance Core (Core_1), – other local memories of Performance Core (Core_1) – Overlay RAM – PIT – Below Modules to be moved from PeMoS to ViMoS: – PBRIDGEn – Below Modules to be moved from ViMoS to SuMoS – STCU2 *Section 6.4.4.1: ECC for storage*: Updated *Table 17: FCCU failure inputs*: – Updated the table. – Added note to “COMPENSATION CELLS” in “Failure description” column. – Updated failure description and Error reaction path check for FCCU channel 17-20. – Updated failure description for FCCU channel 30 – Updated column Reference Manual related information for FCCU channel 58, 60, 63, 64 and 66 to Section: Safety considerations *Table 18: ECC RAM implementations*: – Updated column-head from memory classification to MEMU classification – Added footnote for column-headed ECC for System RAM for location Platform – Added footnote for column-headed ECC in location Core0, Core1 and Core2 for all I-Dem and D-Mem

Table 2499. Document revision history (continued)

Date	Revision	Changes
30-Jul-2020	5 (cont.)	– Updated MEMU Classification for Location HSM for devices – Updated column-headed ECC for Nexus for location Platform *Figure 8: Simplified view of E2E ECC*: Updated this picture. *Chapter 7: Device configuration*: Updated the chapter. *Chapter 8: Reset and Boot* *Table 119: Module status during reset phases*: Removed note from Module “System watchdog” *Section 8.3.1: Power-on and the Reset Generation Module*: Updated the text from, “The STCU is reset on any power-on.....(external reset in this case is ESR0)”. to “The STCU is reset and offline self-test (if enabled) is triggered on any power-on.....(external reset in this case is ESR0)”. *Table 121: SSCM boot record search locations*: Updated: – The location for address (0x00FC_0000) is updated from Flash 16 KB Code Block 0 to Flash 16 KB Code Block 3. – The location for address (0x00FC_8000) is updated from Flash 16 KB Code Block 2 to Flash 16 KB Code Block 4. – The location for address (0x00FC_C000) is updated from Flash 16 KB Code Block 3 to Flash 16 KB Code Block 2. *Table 124: BAF boot record search locations*: Updated: – The location for address (0x00FC_0000) is updated from Flash 16 KB Code Block 0 to Flash 16 KB Code Block 3. – The location for address (0x00FC_8000) is updated from Flash 16 KB Code Block 2 to Flash 16 KB Code Block 4. – The location for address (0x00FC_C000) is updated from Flash 16 KB Code Block 3 to Flash 16 KB Code Block 2. *Table 125: SSCM boot record search locations*: Updated: – The location for address (0x0060_C000) is updated from Flash 16 KB Code Block 0 to Flash 16 KB HSM Code Block 5. – The location for address (0x0061_0000) is updated from Flash 64 KB Code Block 0 to Flash 64 KB HSM Code Block 2. – The location for address (0x0062_0000) is updated from Flash 64 KB Code Block 1 to Flash 64 KB HSM Code Block 3. *Section 8.3.5: PHASE2[DEST] Phase: flash initial configuration*: Updated the note. *Section 8.3.8: PHASE1[FUNC] Phase: temporization and monitoring setup*: Added the latest sentence “During this phase, test modules are available.” *Figure 41: Reset Generation Module Reset Sequence*: – Updated the figure – Updated footnote, “No self-tests are executed.” – Removed a block of “no self test”

Table 2499. Document revision history (continued)

Date	Revision	Changes
30-Jul-2020	5 (cont.)	*Chapter 9: Device Configuration Format (DCF) Records*: *Figure 50: UTEST Miscellaneous DCF Client*: – Renamed bit 12 from CORE_CLK_ONLINE_LBIST to CORE_CLK_ONLINE_SELF_TEST – Refer to DCF client list for reset value. *Table 131: UTEST Miscellaneous DCF Client field descriptions*: – Renamed bit 12 from CORE_CLK_ONLINE_LBIST to CORE_CLK_ONLINE_SELF_TEST – Updated description of bit field XOSC_EXT_CLOAD *Table 134: PMC_REE_BUS field descriptions*: Updated description for each bit field. *Chapter 11: Smart Stand-by Wake-up Unit (SSWU)*: *Table 159: CTUCR field descriptions*: Updated note for bit field 24. *Section 11.6.4: OPC*: Added second and third bullets to the set of pads behavior. *Chapter 15: Core (z420n3) Description*: *Section 15.9: Local memories*: Added this section. *Section 15.9.2.3: DMEM Control Register 1 (DMEMCTL1)*: Added. *Section 15.9.2.4: IMEM Configuration Register 0 (IMEMCFG0)*: Added. *Section 15.9.2.5: IMEM Control Register 0 (IMEMCTL0)*: Added. *Section 15.9.2.6: IMEM Control Register 1 (IMEMCTL1)*: Added. *Section 15.10: End-to-End ECC support*: Added this section. *Figure 102: Core (z420n3)Supervisor Mode Programmer's Model DCRs and PMRs*: Updated. *Chapter 16: Core (z425n3) description*: *Figure 135: Core (z425n3) Supervisor Mode Programmer's Model SPRs*: Added L1CSR0 / SPR 1010 in cache registers. *Table 235: Exceptions and conditions*: Deleted TBD line. *Section 16.9.2.1: DMEM Configuration Register 0 (DMEMCFG0)*: Updated. *Figure 157: DMEM Configuration Register 0 (DMEMCFG0)*: Updated DMSIZE. *Table 260: DMEMCFG0 Field Descriptions*: Updated row-headed DMEM BASE ADDR. *Section 16.9.2.4: IMEM Configuration Register 0 (IMEMCFG0)*: Updated. *Table 263: IMEMCFG0 Field Descriptions*: Updated row-headed IMEM BASE ADDR. *Section 16.10.1.1: End-to-End ECC Control Register 0 (E2ECTL0)*: Added note. *Figure 136: Core (z425n3) Supervisor Mode Programmer's Model DCRs and PMRs*: Updated. *Chapter 17: System Integration Unit Lite2 (SIUL2)*: *Table 280: SIUL2_MSCR_IOn field description*: – ILS bitfield description changed for value 00 (Automotive for SPC58xEx, Reserved for SPC58xGx). – Changed WPDE and WPUE field descriptions.

Table 2499. Document revision history (continued)

Date	Revision	Changes
30-Jul-2020	5 (cont.)	*Section 17.2.2.12: I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR_IOn)*: Changed the latest sentence by referring to the device I/O_Definition document. *Chapter 18: Crossbar switch (XBAR)*: *Table 289: XBAR memory map*: Added MASTER channel n (n = 0 to 7) title and XBAR_MGPCRn registers at 0x0800 +n*0x100 address offsets. *Section 18.3.3: Master General Purpose Control Register (XBAR_MGPCRn)*: Added this section. *Table 290: XBAR_PRSn field descriptions*: Added description of bitfield M3. *Table 291: XBAR_CRSn field descriptions*: Updated the note of PARK field from “Only select master.....behavior may occur.” to “Only select master.....parking configuration.” *Chapter 19: Crossbar Integrity Checker (XBIC)*: *Table 295: XBIC_EIR field descriptions*: In the second note, replaced “SMPU logical bus master assignments” with “AHB Master ID Assignments”. *Chapter 23: Semaphores2 (SEMA42)*: *Section 23.2: Memory map/register definition* *Table 322: Memory map/register definition*: – Updated register name for 0x0040 – Deleted line for 0x0042 – Updated Address offset for Reserved line *Section 23.2.2: Semaphores2 (Secure) Reset Gate (SEMA4_RSTGT)*: Replaced all occurrences of SEMA4_RSTGT_W by SEMA4_RSTGT – *Figure 216: Semaphores2 (Secure) Reset Gate (SEMA4_RSTGT)*: Added RSTGSM and RSTGMS. – *Table 324: SEMA4_RSTGT field descriptions*: Added RSTGSM and RSTGMS. *Table 23.2.2: Semaphores2 (Secure) Reset Gate (SEMA4_RSTGT)*: Deleted this section. *Chapter 25: Enhanced Direct Memory Access (eDMA)*: *Figure 233: Control Register (eDMA_CR)*: – Changed GRP3PRI and GRP2PRI field access type to RW. – Updated register reset value with a specific figure footnote. *Table 338: eDMA_CR field descriptions*: Added note to GRP3PRI and GRP2PRI field descriptions. *Chapter 27: Clocking*: *Section 27.1: Introduction*: – Deleted text “as a backup clock” – Updated text “The internal 16 MHz...chapter for details.” *Section 27.2: Clock generation*: – *Figure 280: Clock generation(1)(2) (part 1 of 2)*: – Updated the figure. – Updated dividers for FRAY_CLK (AUX2).

Table 2499. Document revision history (continued)

Date	Revision	Changes
30-Jul-2020	5 (cont.)	*Section 27.2.1: MC_CGM registers*: *Table 388: MC_CGM relationship to clocks*: SEM4 replaced by SEMA42. *Table 390: Maximum auxiliary level clock frequencies*: – Added note “DSPI_CLK frequency should always be greater than or equal to 1.25 times PBRIDGE_CLK” to DSPI_CLK. – Row “FRAY_CLK”: Updated ‘40’ by ‘80’. – Added note “80 MHz maximum frequency is at aux divider output. The maximum frequency of FRAY_CLK after the multiplexer is 40 MHz” to FRAY_CLK for Aux Clock 2 Selector. *Section 27.5.1.6: Loss of clock detection*: Deleted the last sentence “Software may enable or disable an optional backup clock in the PLL”. *Section 27.8.1: Loss of PLL/XOSC clock*: Updated this section. *Section 27.8.2: Loss of IRCOSC clock*: – Updated the section. – Deleted text “and backup.” – Removed the note. *Figure 283: Clock distribution*, MEMU block: updated “Memory Clock” by “MEMU Clock” and added Note “The frequency of the MEMU clock...”. *Chapter 28: Dual PLL digital interface (PLLDIG)*: *Figure 291: Dual PLL digital interface block diagram*: – Backup Clock Selection” block is made blue and note added “Backup Clock Selection” block in blue is not implemented”. – Updated this figure by adding a divider by 2 in PLL0 block. *Section 28.4.1: Normal mode with reference, PLL0 or both PLLs enabled*: Added in Note cross-ref to *Section 28.7: Initialization information*. *Section 28.5.2.1: PLL0 control register (PLL0CR)*, *Section 28.5.2.4: PLL1 control register (PLL1CR)*: Updated note for field LOLIE. *Chapter 29: Clock Monitor Unit (CMU)*: *Section 29.4.1.1: CMU Control Status Register (CMU_CSR)*: Updated the description of RCDIV from, “Ensure clock.....CMU_CSR[RCDIV].” to “Ensure clock.....otherwise the operation can lead to a spurious OLR event.” *Section 29.4.1.5: CMU Interrupt Status Register (CMU_ISR)*: Updated Note. *Table 411: CMU_CSR field descriptions*: Updated the descriptions of SFM and CME fields. *Chapter 31: OSC digital interface (XOSC)*: *Section 31.2.4.1: Automatic Level Control (ALC)*: Added. *Chapter 32: IRCOSC digital interface*: *Table 496: IRCOSC_NT field descriptions*: Updated description of bitfields FT_CODE and RCTRIM. *Chapter 35: Platform RAM controller AHB (PRAMC_AHB)*: Changed chapter name to PRAMC_AHB.

Table 2499. Document revision history (continued)

Date	Revision	Changes
30-Jul-2020	5 (cont.)	*Chapter 36: Flash memory controller*: *Section 36.5.14.1.1: e2eECC and Flash accesses*: Added the latest sentence. *Section 36.5.15: e2eECC on data Flash accesses*: Added the latest sentence. *Chapter 37: Embedded Flash Memory*: *Table 522: UTEST block memory map*: Added note to offset 0x140. *Table 525: MCR field descriptions*: – Improved wording of DWEE both for code and data memory. – Add sentence “Note that EDC post ECC reporting is suppressed for Data Flash and HSM Data Flash” at EEE field description. *Table 528: MCRE field descriptions*: – Updated description of fields N64KL and N16KL. – Removed the row of field “N256KH”. *Table 539: MCR2 field descriptions*: Same done for PDWEE but only for code memory *Table 541: ADR field descriptions*: Improved wording for bit field ADDR *Table 576: ADR2 field descriptions*: Improved wording for bit field PADR *Section 37.3.2.1: Module Configuration Register (MCR)*: Updated PEEE field description. *Section 37.4.1: Reset*: Sentence has been highlighted with Warning style. *Section 37.4.4.2: Error correction code*: Add sentence “EDC post ECC reporting is suppressed for Data Flash and HSM Data Flash” at the end of the fourth sentence. *Table 529: LOCK0 field descriptions*: Updated. *Table 530: LOCK1 field descriptions*: Updated. *Table 531: LOCK2 field descriptions*: Updated. *Table 532: LOCK3 field descriptions*: Updated. *Table 533: LOCK0A field descriptions*: Updated. *Table 534: Alternate LOCK1A field descriptions*: Updated. *Table 535: SEL0 field descriptions*: Updated. *Table 536: SEL1 field descriptions*: Updated. *Table 537: SEL2 field descriptions*: Updated. *Table 538: SEL3 field descriptions*: Updated. *Chapter 38: Flash Memory Programming and Configuration*: *Section 38.6.2: Creating the tamper detect diary*: – The maximum size of Diary updated from 2 KB to 4 KB. – Multiplied by 2 KB is updated to 4 KB. *Example 4: Overriding secure write protection*: Updated paragraph list 2. *Example 7: Creating the tamper detect diary*: Updated. *Figure 464: Sample DCF record for secure write protection*: Updated the value of the figure. *Chapter 40: Analog-to-Digital Converters (ADC) Configuration*: *Section 40.2.2.1.1: 12-bit Fast SAR ADC*: Updated “analog input channel for each SARADC instances *Section 40.2.2.1.2: 12-bit SAR ADC B0*: Minor format change. SAR ADC register definitions: Deleted the section.

Table 2499. Document revision history (continued)

Date	Revision	Changes
30-Jul-2020	5 (cont.)	*Chapter 42: Successive Approximation Register Analog-to-Digital Converter (SARADC)*: *Table 616: SARADC digital interface register map*: Removed “Register classification” column. *Section 42.4.4.1: Conversion timings*: Changed SAMPSEL by CTSEL field name. *Table 617: MCR field descriptions*: – Added note to NSTART, JSTART and PWDN field descriptions. – Updated the note for NSTART. *Table 636: PDEDR field descriptions*: Added latest paragraph in PDED field description. *Section 42.4.6: Test channel connection with internal analog channel*: Removed temperature sensor and bandgap from the first sentence of the first paragraph. *Section 42.5.1.11: Watchdog Threshold Register n (WTHRHLRn)*: Replaced in 1st note MSB=0 bits numbers by LSB=0 bit numbers. User access updated for: – *Section 42.5.1.32: External Channel Interrupt Pending Register n (ECIPRn)* – *Section 42.5.1.39: External Channel Analog Watchdog Out of Range Register n (ECAWORRn)* *Chapter 50: Periodic Interrupt Timer (PIT)*: *Section 50.5: Example Configuration for the Lifetime Timer*: Updated *Equation 29:* From, “current_uptime = PIT_LTMR64H << 64” to “current_uptime = PIT_LTMR64H << 32” *Chapter 51: Enhanced Modular IO Subsystem (eMIOS)*: *Table 797: EMIOSMCR field descriptions*: Updated note in GTBE field description. *Chapter 52: CAN Subsystem*: *Table 823: TSCC field descriptions*: – TCP: Removed the note, “With CAN FD ..... (TSS = ‘10’).” – TSS: Added a note, “In case of CAN FD mode..... generation.” – TSS: Updated the value of 10 from “External timestamp counter value used” to “External timestamp counter value used” only for SPC58xEx (“Reserved” for SPC58xGx).” *Section 52.3.5.2.3: Data Bit Timing and Prescaler Register (DBTP)*: Corrected time quanta max number in the first sentence. *Table 836: XIDFC field descriptions*: Corrected max enumerated value of LSE field to 0x7F. *Table 841: RXF0C field descriptions*: Corrected max enumerated value of F0WM and FOS fields to 0x7F. *Section 52.3.5.2.8: Timestamp Counter Configuration Register (TSCC)*: Updated the description of TSS field. *Chapter 53: Serial Interprocessor Interface (SIPI)*: *Section 53.1.1: Scalability*:Updated *Figure 875: SIPI module*: Updated figure note by removing “Channel 0 has the highest priority for transmission”

Table 2499. Document revision history (continued)

Date	Revision	Changes
30-Jul-2020	5 (cont.)	*Section 53.4.1: Main features*: – Updated the eleventh feature bullet. – Removed bullet “Target node...commands” – Updated bullet starting by “Target node contains a set of nine” *Figure 881: Streaming write request format*: Updated the Note below the figure. *Figure 889: Streaming transfer with data*: Added CR2[0], CR2[1] and CR2[7]. *Section 53.5: SIPI operation from reset*: Updated. *Section 53.6.2: Frame format*: Added Note at the end of the section. *Section 53.6.2.5: Streaming write request*: Updated in *Note:* *Section 53.6.2.8: Payload field*: Updated this section. *Section 53.7.2: Register read answer transfer*: – Added “big endian” in brackets to the note. – Added *Note:* and *Table 963: Target Memory content dependent from CAR content* *Section 53.7.3.2.1: Streaming write transfer with address*: – Added first sentence “The related registers...”. – Updated the second paragraph. *Section 53.8: Transfer API and flow charts*: – Updated this section. – Updated *5.* in list Multiple Write Transfer Request generation. – Updated Multiple Write Transfer Request generation sequence and Single Read Transfer Request generation sequence. *Section 53.9: DMA handling in transfer initiator mode*: – Re-named the title. – Added definition of DMA handling of transfer initiator and of transfer target *Figure 900: SIPI data stream model - Target*: Replaced “8-bit data” by “Data”. *Figure 901: SIPI data stream with acknowledge trigger*: – Replaced ‘Bus Master DMA’ by ‘Bus Master AHB’ – Moved the arrow that was between CRC Checker and ACK Frame Generation so that it is between Bus Master AHB and ACK Frame Generation. *Figure 904: SIPI Channel Control Register (SIPI_CCRn)*: Deleted footnote “Only transfers with channel 2...” *Figure 909: SIPI Channel address Register (SIPI_CARn)*: Deleted figure footnote. *Table 961: Payload size of LFAST channel frame*: – Added footnote. – Updated this table. *Table 962: Converted coding of LFAST channel code for SIPI headers*: Updated this table. *Table 965: SIPI_CCRn field descriptions*: – Updated DEN field description. – Updated row-headed TC *Table 970: SIPI_CARn field descriptions*: – Updated row-headed CAR. – Updated CAR bit field definition.

Table 2499. Document revision history (continued)

Date	Revision	Changes
30-Jul-2020	5 (cont.)	*Table 974: SIPI_MCR field descriptions*: – Updated AID and SR fields. – Updated row-headed AID. *Figure 886: Data allocation*: Added 32-bit data allocation. *Figure 893: SIPI single register write API*: Changed TC name by CPU. *Section 53.9: DMA handling in transfer initiator mode*: Updated single write transfer request generation list. *Section 53.10: Modes of operation*: Updated this section. *Section 53.10.2: Normal Mode (NM)*: Removed text “SIPI needs...Register (SIPI_MCR)).” *Section 53.11.1: Timeout error*: – Added *Note:* – Removed Note: In case of timeout...being sent now. *Section 53.15.1.1: SIPI Channel Control Register (SIPI_CCRn)*: – Added “streaming command is 9 words long”. – Updated CHEN and DEN fields. – Updated *Note:* *Section 53.15.1.6: SIPI Channel Address Register (SIPI_CARn)*: Updated this section. *Table 975: SIPI_SR field descriptions*: – Updated row-headed TE. – Updated STATE field. *Section 53.15.1.12: SIPI Address Count Register (SIPI_ACR)*: – Updated this section. – Updated *Note:* *Figure 889: Streaming transfer with data*: Updated. *Figure 895: SIPI multiple register write API*: Updated. *Figure 896: SIPI single register read API*: Updated. *Figure 897: SIPI multiple register read API*: Updated. *Figure 898: SIPI data stream model*: Updated. *Chapter 54: Zipwire*: Deleted the entire section “Zipwire performance”. *Chapter 55: LVDS Fast Asynchronous Serial Transmission – Interprocessor Communications*: *Table 985: TMCR field descriptions*: Updated description of bitfields CLKTST and LPON *Table 990: PICR field descriptions*: Updated description of bitfield PNGREQ *Table 997: UNSTCR field descriptions*: Updated description of bitfield USNDRQ *Table 995: PLLCR field descriptions*: Updated the description of FBDIV. *Figure 920: LFAST frame structure*: Added “(bits)” to “Payload size” and “Frame size” columns headers. *Table 1003: RISR field descriptions*: Updated ICPFF.

Table 2499. Document revision history (continued)

Date	Revision	Changes
30-Jul-2020	5 (cont.)	*Table 1003: RISR field descriptions*: Updated ICPFF. *Figure 959: Rx Loopback mode* to *Figure 964: Tx LVDS Loopback (external) mode with automatic frame generation*: Added note regarding CSR. *Table 994: RIIER field descriptions*: Updated ICPFIE. *Table 1018: Frames supported by LFAST interfaces*: Value changed to “1 x 8” in Tx Buffer/ ICLC frame. *Section 55.7.1.1: LFAST master interface startup procedure*: Updated bullets 1, 13, 23, 26, and 28. *Chapter 56: Ethernet*: *Figure 1025: MMC Rx Interrupt mask Register (MMC_RX_INTERRUPT_MASK)*: Updated the register access type from User Read to Software read/write. Access type for all the bits ranging from [4:31] changed from Read only to Read Write. *Figure 1026: MMC Tx Interrupt Mask Register (MMC_TX_INTERRUPT_MASK)*: Updated the register access type from User Read to Software read/write. Access type for all the bits ranging from [4:31] changed from Read only to Read Write. *Chapter 57: FlexRay communication controller (FlexRay)*: *Table 1300: FlexRay memory map*: Corrected not implemented and reserved spaces. *Figure 1302: Protocol Configuration Register 9 (FR_PCR9)*: Corrected minislot_exists and symbol_window_exists field names. *Figure 1303: Protocol Configuration Register 10 (FR_PCR10)*: Corrected wakeup_channel field name. *Figure 1304: Protocol Configuration Register 11 (FR_PCR11)*: Corrected key_slot_used_for_startup field name. *Figure 1315: Protocol Configuration Register 22 (FR_PCR22)*: Updated bit field ‘R’ by ‘Reserved’. *Chapter 59: SENT Receiver (SRX)*: *Figure 1436: Channel n Fast Message Data read register (CHn_FMSG_DATA)*, *Figure 1441: Channel n Slow Serial Message Time Stamp read register (CHn_SMSG_TS)*: Added the following footnote and a cross ref to it “The value of the CH is device-specific. Refer to the Device Configuration chapter for details.” *Chapter 60: Peripheral Sensor Interface (PSI5)*: *Table 1494: PSI5 memory map*: Removed “Register classification” column. *Figure 1454: PSI5 Channel Control Register (PSI5_CH0_PCCR)*: Updated the access of bit 15 from “Read only” to “RW” and added a note “This is a redundant bit and RW will not have any effect”. *Chapter 61: Peripheral Sensor Interface-Support module (PSI5-S)*: *Section 61.1.2: Features*: Removed “(big endian)” from the first bullet. *Chapter 62: Deserial Serial Peripheral Interface (DSPI)*: *Section 62.3.6: DSPI Status Register (DSPI_SR)*: changed reset value from 0xDEAD_BEEF to 0x0201_0000.

Table 2499. Document revision history (continued)

Date	Revision	Changes
30-Jul-2020	5 (cont.)	*Table 1594: DSPI_MCR field descriptions*: added note to HALT field description. *Section 62.5.6.3: Modified SPI/DSI Transfer Format (MTFE = 1, CPHA = 0)*: Added a second note. *Chapter 63: LINFlexD*: *Section 63.2.2: UART mode features*: Updated the 3 latest bullets. *Table 1659: TCD setting: UART - Tx mode*: Removed note at DADDR[31:0]. *Table 1660: TCD setting: UART - Rx mode*: Removed note at SADDR[31:0]. *Table 1661: LINFlexD memory map*: – Removed “Register classification” column. – Removed note that refer to the Device Configuration chapter and all cross-references to this note. *Section 63.4: Memory map and register description*: added note that refer to the LinFlexD configuration of the Device Configuration chapter. *Table 1686: DMATXE field descriptions*: field editorial changes to remove the table note. *Table 1687: DMARXE field descriptions*: field editorial changes to remove the table note. *Chapter 64: Reset Generation Module (MC_RGM)*: *Section 64.4.3: External reset*: – Updated the text from, “The ESR0.....ESR0 output being asserted.....the chip.” to “The ESR0.....ESR0_B output being asserted.....the chip.” – Added content “The correct timing between PORST and ESR0..... reset release.” *Section 64.4.7: ‘Destructive’ reset escalation*: added footnote to “each ‘destructive’ reset”. *Chapter 65: Boot Assist Flash (BAF)*: *Section 65.1: Introduction*: updated the fourth paragraph. *Section 65.3.8: Optionally perform a serial boot*: Deleted text “All Tx-channels have to be disabled during this phase” *Chapter 66: System Status and Configuration Module (SSCM)*: *Table 1739: ERROR field descriptions*: Removed “This feature...application code.” for row-headed PAE and RAE *Figure 1791: HSM and User Option Status Register (UOPS)*: – Added fields FL_RUN, BAF_FA and FL_STDBY and their respective descriptions. – Changed HSB bit field length to 3 bits and updated its description. *Table 1740: UOPS field descriptions*: – Added fields FL_RUN, BAF_FA and FL_STDBY and their respective descriptions. – Changed HSB bit field length to 3 bits and updated its description. *Section 66.3.3: Boot mode functionality*: updated Standby-Boot section. *Table 1746: Life Cycle Slots in Memory*: Updated.

Table 2499. Document revision history (continued)

Date	Revision	Changes
30-Jul-2020	5 (cont.)	*Chapter 67: Power management controller digital interface (PMC_Dig)*: *Section 67.5.26: HPREG SMPS SELECT STATUS (HPREG_SMPS_SEL_S)*: Changed HPRE to HPREG. *Table 1752: REE_LV0 field descriptions* and *Table 1762: REE_HV0 field descriptions*: Changed “REE” by “RES” in REE_XX field descriptions. *Table 1753: RES_LV0 field descriptions* and *Table 1763: RES_HV0 field descriptions*: Changed “RES” by “REE” in RES_XX field descriptions. *Table 1776: HPREG_SMPS_SEL_S field descriptions*: Changed “SMPS_HP_N_SEL” by “HPREG_SEL pad” in all field descriptions. *Table 1785: BIST_FLAGS_PHASE1 field descriptions*, *Table 1786: BIST_FLAGS_PHASE2 field descriptions* and *Table 1791: BIST_DEBUG field description*: Replaced “Status will be ignored” by “Status should be ignored” *Section 67.5.38: User BIST Time1 and Time0 register (BIST_TIME10)*,*Section 67.5.39: User BIST Time3 and Time2 register (BIST_TIME32)* and *Section 67.5.40: User BIST Time6 and Time5 register (BIST_TIME65)*: – Added a cross-reference to Voltage monitor BIST programming section. – Updated Size of bitfields, TIME_x are updated to 12 bits from of 13 bits. *Section 67.9: Flash interface (DCF client usage)*: Added 2 paragraphs relative to monitors reactions. *Section 67.10: Voltage monitor BIST programming*: Updated this section. *Chapter 69: Mode Entry Module (MC_ME)*: *Section 69.4.1: Mode transition request*: “Added a note, “If CMU reveals..... executed correctly”. *Table 1804: ME_<mode>_MC field descriptions (continued)*: Removed table footnote in XOSCON field description. *Chapter 72: Debug and Calibration Interface (DCI)*: *Section 72.1.3.5: System watchdog control*: Added note “Debugger should...the SWT.” *Chapter 75: JTAG Data Communication (JDC)*: *Figure 1983: Module Status Register (JDC_MSR)*: corrected register access to read/write. *Chapter 77: Buddy Device—JTAG Master (JTAGM)*: *Table 1961: JTAGM_MCR register field descriptions*: Updated row-headed IIE *Table 1963: JTAGM_DOR0 register field descriptions*: Updated. *Table 1964: JTAGM_DOR1 register field descriptions*: Updated. *Table 1965: JTAGM_DOR2 register field descriptions*: Updated. *Table 1966: JTAGM_DOR3 register field descriptions*: Updated. *Chapter 79: Debug Serial Interprocessor Interface (SIPI)* *Section 79.5: Debug SIPI operation from reset*: Updated. *Section 79.6.2: Frame format*: Added Note at the end of the section. *Section 79.6.2.7: Payload field*: Updated this section.

Table 2499. Document revision history (continued)

Date	Revision	Changes
30-Jul-2020	5 (cont.)	*Figure 2066: Data allocation*: Added 32-bit data allocation *Table 1973: Payload size of LFAST channel frame*: – Added footnote. – Updated this table. *Table 1974: Converted coding of LFAST channel code for Debug SIPI headers*: Updated this table. *Section 79.7.2: Register read answer transfer*: Added “big endian” in brackets to the note. *Table 1977: SIPI_MCR field descriptions*: Updated SR field. *Section 79.7.2: Register read answer transfer*: – *Note:* Added. – *Table 1975: Target Memory content dependent from CAR content*: Added. *Chapter 80: LVDS Fast Asynchronous Serial Transmission – High Speed Debug* *Table 1979: Header payload sizes*: Added “(bits)” to “Payload size” and “Frame size” columns headers. *Table 1984: TMCR field descriptions*: Updated description of bitfields CLKTST and LPON *Table 1988: PICR field descriptions*: Updated description of bitfield PNGREQ *Table 1990: UNSTCR field descriptions*: Updated description of bitfield USNDRQ *Table 1996: RIISR field descriptions*: Updated ICPFF. *Figure 2076: LFAST frame structure*: Added “(bits)” to “Payload size” and “Frame size” columns headers. *Figure 2107: Rx Loopback mode* to *Figure 2112: Tx LVDS Loopback (external) mode with automatic frame generation*: Added note regarding CSR. *Figure 2006: Frames supported by LFAST interfaces*: Value changed to “1 x 8” in Tx Buffer/ ICLC frame. *Chapter 86: Cyclic Redundancy Check (CRC) Unit*: *Figure 2278: CRC Configuration Register n (CRC_CFGn)*: Swapped reset values versus function of n in the figure footnote of POLYG reset. *Chapter 87: Memory Error Management Unit (MEMU)*: *Table 2179: SYS_RAM_UNCERR_ADDR field descriptions*: Removed to “This register is not writable when the corresponding VLD bit is zero.” *Table 2184: PERIPH_RAM_UNCERR_ADDR field descriptions* and *Table 2189: FLASH_UNCERR_ADDR field descriptions*: Deleted the sentence “This register is not writable when the corresponding VLD bit is zero.)” *Chapter 88: Indirect Memory Access (IMA)*: *Table 2193: E2EECSR0 field descriptions*: Corrected register name in title. *Figure 2307: Cache Debug Access Control register (CDACNTL)*: Updated. *Figure 2308: Cache Debug Access Data register (CDADATA)*: Updated. *Section 88.2.2.1: e2eECC Error Control/Status register 0 (E2EECSR0)*: Updated.

Table 2499. Document revision history (continued)

Date	Revision	Changes
30-Jul-2020	5 (cont.)	*Chapter 89: Fault Collection and Control Unit (FCCU)*: *Figure 2319: FCCU block diagram*: Updated. *Table 2203: FCCU submodules*: Updated row-headed “REG intf”. *Table 2212: FCCU memory map*: Removed “Register classification” column. *Table 2213: FCCU_CTRL field descriptions*: Updated row-headed OPR. *Figure 2340: FCCU RF Status register n (FCCU_RF_Sn)*: Removed “Refer to for the errors description.” from table footnote 1. *Table 2231: FCCU_STAT field descriptions*: Added a note to PhysicErrorPin field description. *Section 89.7: FCCU Output Supervision Unit*: Updated this section. *Table 2235: F2A_STATUS field descriptions*: – Changed the table title from F2AF_STATUS to F2A_STATUS – Description of value 0xFF of bitfield FAFS: replaced word “critical” by word “recoverable.” *Section 89.6.1: FCCU Control Register (FCCU_CTRL)*: Updated the bullet “configure the glitch filter present on location given by FILTER_POSITION parameter” to “configure the glitch filter present on EIN0 input (FCCU failure input #96)” and replaced text “(FCCU_RF_Sn[RFSx] where y =FILTER_POSTN)” to “FCCU failure input #96”. *Section 89.6.8: FCCU RF Enable Register n (FCCU_RF_En)*: Deleted text “for debugging purposes”. *Section 89.6.13: FCCU Status Register (FCCU_STAT)*: Deleted text “for debugging purposes”. Deleted text “This register is for test/ debug purposes” from the following sections: – *Section 89.6.14: FCCU NA Freeze Status Register (N2AF_STATUS)* – *Section 89.6.15: FCCU AF Freeze Status Register (A2FF_STATUS)* – *Section 89.6.16: FCCU NF Freeze Status Register (N2FF_STATUS)* – *Section 89.6.17: FCCU FA Freeze Status Register (F2A_STATUS)*

Table 2499. Document revision history (continued)

Date	Revision	Changes
30-Jul-2020	5 (cont.)	*Chapter 90: Self-Test Control Unit (STCU2)*: *Section 90.4.2: Reset management*: Added a third sentence. *Table 2252: STCU_RUN field descriptions*: Updated MBSPLLEN and LBSPLLEN field descriptions. *Table 2253: STCU_RUNSW field descriptions*: Uupdated MBSWPLLEN and LBSWPLLEN field descriptions. *Table 2255: STCU_CFG field descriptions*: – Updated CLK_CFG field description. – Changed note in LB_DELAY field description. *Table 2257: STCU_WDG field descriptions*: Updated description for WDGEOC. *Section 90.4.6: Interrupt interface*: Added “there are two interrupt sources:” and a bullet “One triggered by MBIST (enabled by programming STCU_RUNSW[MBIE] = 1)”. *Section 90.4.8.1: Initialization sequence*: Updated second bullet with a second sentence. *Section 90.4.9: CRC*: – Added note at the end of the section “The CRC calculated depends on the specific sample and not only the self-test configuration programmed.” – Updated the note “The expected value of the CRC depends on multiple internal critical signals and can differ among different samples.” *Table 2289: STCU_LB_CTRLn field descriptions*: – Updated CWS field description at enumerated value 0x0. – Updated SHS field description. *Table 2301: STCU_MB_CTRLk field descriptions*: Updated PTR field description. *Section 90.5.5: STCU2 PLL Configuration Register (STCU_PLL_CFG)*: Added a note, “f_{RC} is the frequency if the internal RC oscillator (that is 16 MHz). f_{PLLin} must be higher than 8 MHz.” *Section 90.5.6: STCU2 Watchdog Register Granularity (STCU_WDG)*: Updated description of WDGEOC bitfield. *Section 90.5.8: STCU2 CRC Expected Status Register (STCU_CRCE)*: Added the note to “The expected value of the CRC depends on multiple internal critical signals and can differ among different samples.” *Section 90.5.15: STCU2 Online LBIST End Flag Register (STCU_LBESW)*: Decreased number of fields from 16 to 8. *Section 90.5.16: STCU2 Online LBIST Reset Management (STCU_LBRMSW)*: Decreased number of fields from 16 to 8. *Section 90.5.17: STCU2 LBIST Unrecoverable FM Register (STCU_LBUFM)*: Decreased number of fields from 16 to 8. *Section 90.6: Use cases and limitations*: added paragraph: “The STCU setup has to be defined based on the application-specific safety, boot time and consumption requirements. Contact ST representative for support.” For all available MBIST registers: – Added note, “Refer to..... number of NMCUT available in this device.” – Added note to all fields from 31:1, “The availability of this field is based on.....Device configuration chapter”

Table 2499. Document revision history (continued)

Date	Revision	Changes
30-Jul-2020	5 (cont.)	*Chapter 92: Password and Device Security Module (PASS)*: *Table 2315: LOCK3_PGn register field descriptions*: Added note to all RLx field descriptions. *Table 2320: Debug Interface Enable Truth table*: Updated the table. *Table 2321: Flash Memory Read Protection truth table*: Removed the second table. *Section 92.5.6.2.4: **Fuse Bypass Password compare***: – Removed text “of the Flash page contain different data not related to Fuse Bypass” at the end of the first sentence. – Removed “as the most significant 32 bits” in the second sentence. *Chapter 93: Tamper Detection Module (TDM)*: *Figure 2454: Diary implementation*; Changed Diary_TDR0 size from 256x64 to 256x128 bits. *Chapter 95: Real Time Clock / Autonomous Periodic Interrupt (RTC/API)*: *Section 95.4.6: RTC Compare Value Register (RTC_RTCVAL)*: – Added a note “If RTC counter is reset (RTCC[CTNEN] = 0), it is required to reload the RTCVAL via software to re-enable the RTCVAL match interrupt before starting the RTC counter.” – Updated Note. *Chapter 96: Wakeup unit (WKPU)*: *Section 96.1.2: Features*: – Changed “configurable pullup” by “configurable pull”. – Added a note referring to the Device Configuration chapter. *Table 2395: NSR field descriptions*: added missing fields NIF2 and NOVF2. *Table 2403: WIPUER field descriptions*: – Added a table note referring to the Device Configuration chapter on “Description” heading. – Replaced “Pullup” to “Pull” in the complete table. *Section 96.3.2.5: Wakeup Request Enable Register (WRER)*: Added Note “Any wakeup pad...” *Section 96.3.2.8: Wakeup/Interrupt Filter Enable Register (WIFER)*: Added a Note, “Glitch filter must be enabled in case the wake-up sources are used to exit from Standby.” *Section 96.3.2.9: Wakeup/Interrupt Pull Enable Register (WIPUER)* – Updated the note from “This register shall be left at '0' default value” to “WIPUER bits related to those bonded LP pads that are used to wake-up the device from STANDBY Mode, should be left at 0 default value”. – Updated the text from, “This register..... value of 1” to “This register..... SIUL2_MSCR_IO_n.WPDE-WPUE bits.” *Table 2403: WIPUER field descriptions*: – Added a table note referring to the Device Configuration chapter on “Description” heading. – Replaced “Pullup” to “Pull” in the complete table.

IMPORTANT NOTICE – PLEASE READ CAREFULLY

STMicroelectronics NV and its subsidiaries (“ST”) reserve the right to make changes, corrections, enhancements, modifications, and improvements to ST products and/or to this document at any time without notice. Purchasers should obtain the latest relevant information on ST products before placing orders. ST products are sold pursuant to ST’s terms and conditions of sale in place at the time of order acknowledgement.

Purchasers are solely responsible for the choice, selection, and use of ST products and ST assumes no liability for application assistance or the design of Purchasers’ products.

No license, express or implied, to any intellectual property right is granted by ST herein.

Resale of ST products with provisions different from the information set forth herein shall void any warranty granted by ST for such product.

ST and the ST logo are trademarks of ST. For additional information about ST trademarks, please refer to *www.st.com/trademarks*. All other product or service names are the property of their respective owners.

Information in this document supersedes and replaces information previously supplied in any prior versions of this document.

© 2020 STMicroelectronics – All rights reserved